LSMPQS software manual (v. 0.5)

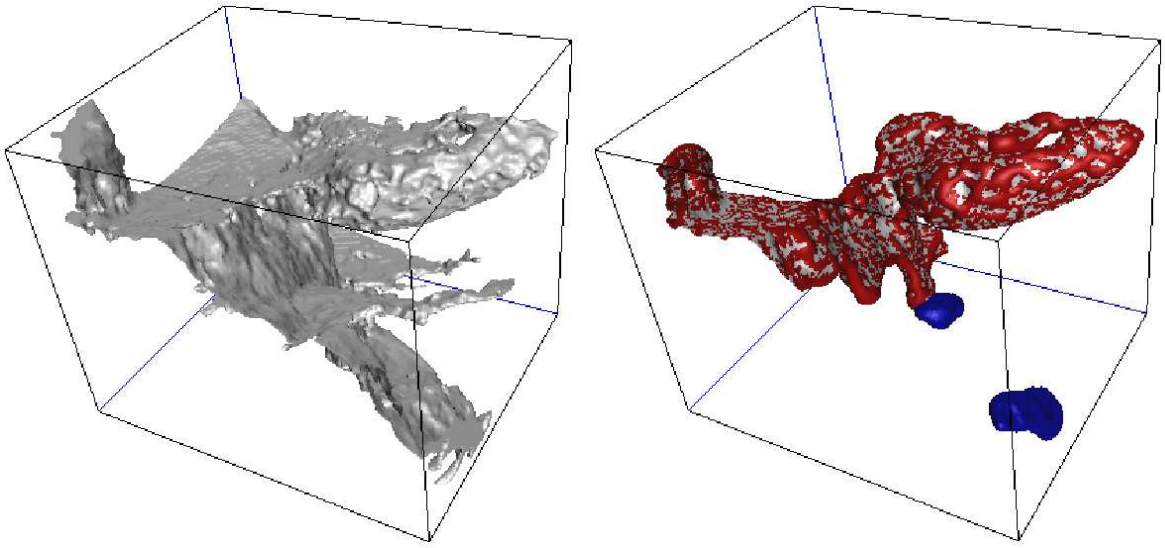
Maša Prodanović

May 3, 2009

Center for Petroleum and Geosystems Engineering
University of Texas at Austin, Austin, TX, USA
masa.prodanovic@gmail.com

1 Introduction

LSMPQS is a software package for simulating capillarity controlled, immiscible fluid displacement (drainage and imbibition) in porous media.



a) b)

Figure 1: a) Pore-grain surface extracted from an x-ray microcomputed tomography image of a naturally fractured carbonate sample (approximately $600\mu m$ on each side). b) Non-wetting (NW) phase configuration at imbibition (curvature $C = 0.09\mu m^{-1}$). Non-wetting/wetting fluid contact is shown in red, NW phase/grain contact is in gray, and snapped-off (trapped) blobs are in blue. For more information see [13].

Features include:

- ◇ Porous medium can be of arbitrary complexity, described analytically (e.g. sphere packs) or based on segmented images of real rocks.
- ◇ No pre-set parameters other than the description of the porous medium geometry is needed.
- ◇ The fluid configurations obtained from LSMPQS simulation are detailed (see Fig. 1) and quantifying saturation, volumes, interfacial areas etc. is straightforward. Having suitable geometry information, you can obtain capillary-pressure curves or study pore scale phenomena in a small ensemble of pores.
- ◇ LSMPQS is developed on Linux, and should compile on any Unix-like system. C/C++ and FORTRAN compilers are required.

The name LSMPQS is derived from Level Set Method based Progressive-Quasistatic algorithm originally published in [12].

Needless to say, LSMPQS and its documentation are works in progress. Please email me (`masa.prodanovic@gmail.com`) if you have any type of feedback: that will motivate future improvements (and possibly move things up the priority list). Further, if you use the code in your work, I would appreciate if you cite [12] along with the LSMPQS webpage as that will help my academic career.

2 Installation

Here we outline the prerequsites and the skeleton of installation procedure. Please refer to Appendix A for detailed instructions.

- ◇ Install LSMLIB (the library of level set methods) [3].
- ◇ (Optional) Install GTS and Geomview and Matlab (or Octave), if you would like to use available visualization and post-processing routines. (Section §7 has more details on visualization).
- ◇ Configuration and compilation, followed by an optional installation (`'configure-make-make install'` standard for Linux based software) should be all you need to do before you can use the software.

3 License

LSMPQS is distributed under the University of Texas at Austin source code research license and is free for academic, research, experimental or personal use. Should you wish to make other use of the software, consult LICENSE.txt or LICENSE.pdf files in the top directory of the LSMPQS distribution, or contact Office of Technology Commercialization, University of Texas at Austin, `info@otc.utexas.edu`.

4 Organization

The top LSMPQS directory contains the following subdirectories:

```
./
|-- bin
|-- doc
|-- examples
|-- include
|-- lib
|-- src
```

which store executables, documentation, examples, code header files, compiled libraries and source code, in that order. Documentation subdirectory `doc/` contains this manual as well as multiple paper manuscripts with LSMPQS results.

Source code `src` subdirectory is organized in several subdirectories as follows:

```
./src/
  |-- analysis
  |-- base
  |   `-- development
  |-- geometry
  |   |-- development
  |   `-- matlab
  |-- obsolete
  |-- tools
  |   `-- development
  `-- visualization
    |-- development
    |   `-- analytic_solution
    |-- geomview_templates
    `-- matlab
```

Every directory can have a subdirectory `matlab` that contains auxiliary and postprocessing Matlab/Octave routines (have not tested in Octave, but in theory they should work). There is no compiled Matlab code at this point so these are independent from the rest of the code.

Only LSMPQS developers have subdirectories named `development`, `obsolete` and you are more than welcome to become one. If you are interested in joining the coding effort or would like to incorporate an application of interest, be sure to email me at `masa.prodanovic@gmail.com`.

Appendix B has documentation for all C-implemented programs. All Matlab routines are documented, and the utility `help` can be used in order to read the documentation.

If you think you might want to modify or add to the code, read on. Otherwise there is no need for you to delve into more details.

4.1 Code dependency

As the name might suggest, `src/base/` contains the base of the LSMPQS code and programs in other subdirectories might use it. The code in `base/` is compiled first and its object files are linked into a library `liblsmpqs.a` that the other programs can be linked with. No code from other directories is added to this library. For instance, programs in visualization can depend on the code in `base/` since they are linked with `liblsmpqs.a`, but not on anything compiled in `tools/`. Bear this in mind when adding new code. As a rule of thumb, add code in `development/` subdirectory at first.

`include/` directory contains links to all of the header (*.h) files from the code. These are soft links: pointers to the original file location. Having such a centralized pool of header files makes it easier to compile the code without remembering where individual headers are. Another simplification are the two header files `lsmlib_header.h` and `lsmpqs_header.h`. They contain the list of all LSMLIB and LSMPQS header files (again, so you do not need to figure out which particular header file has the function you are using).

4.2 Navigating the code

First of all, it is helpful to use 'tags' to navigate the code. You can type 'make ctags' in `src/` directory to create tags file via 'ctags' utility and that should automatically tag both LSMPQS or LSMLIB (see `src/Makefile` for more info). `ctags` are supported by 20+ editors. To give you one example, using `nedit`, you can highlight the function/structure name and type `Ctrl-D` to open the file where it is defined.

4.2.1 What is LSMLIB_REAL?

This is probably the single most confusing (and omnipresent) type that you encounter when you open a source code file. If you configured your code with `--enable-float`, LSMLIB_REAL means `float`, otherwise `double`. LSMLIB_REAL is defined in LSMLIB library and enables configuring the code for single or double precision, without having to code everything twice.

4.3 Adding to the code

I have included some pointers on how to add new functionality within the code.

- ◇ **Adding another geometry**. Follow the "How to add a new geometry" suggestions in `src/base/mask.h`.

- ◇ **Adding a compiled program** that uses merely usede existing LSMLIB/LSMPQS functionality but is otherwise independent is probably the simplest thing to do. Say you want to add program `new.c` in `tools/` directory. You can take a look at any of the files there as an example (say, `seal_data_array.c`). In order to compile, add 'new' to `PROGS, PROGS_LSMPQS` or `PROGS_C` list in `Makefile` and `Makefile.in`, as appropriate. Type `'make'` to compile.

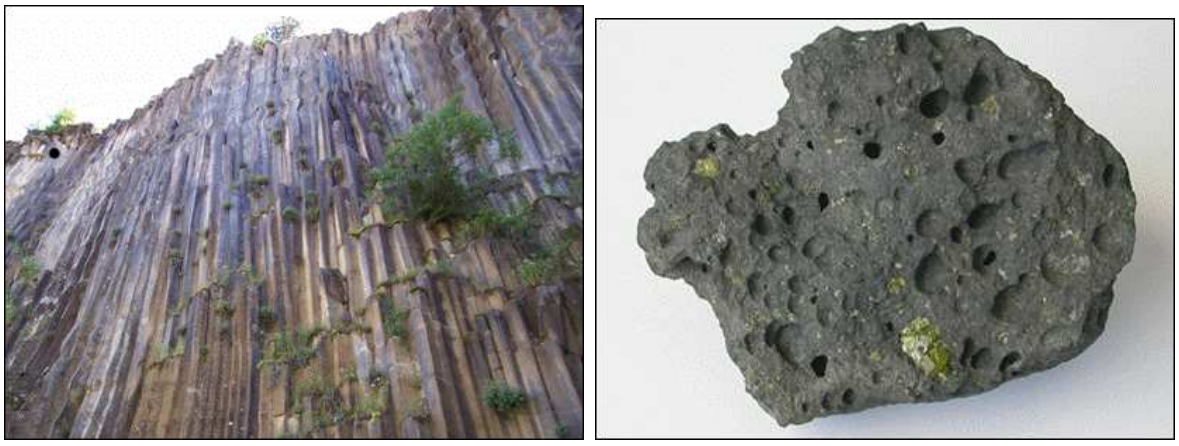
- ◇ **Adding another curvature model** that inherits the structure of `compress_curv_model` or `const_curv_model` in terms of geometry/input options initialization. See instructions "How to add a new curvature model" in `src/base/curv_model_top.h`
- ◇ **Debugging** You can manually change `CFLAGS, FFLAGS` to `-g` option in `src/Makefile.config` in order to recompile for debugging. For more details see file `src/howto-debug-lsmpqs.txt`.

5 Terminology

Most of the terminology comes from petroleum and environmental engineering, the level set methods and the relevant numerical schemes. While the method appplies to any immiscible fluid displacement in porous media, you might not relate to the LSMPQS naming conventions simply because in many instances there exist two (or more) terms that essentially describe the same phenomenon. What is more, the interfaces of interest are constant mean curvature surfaces. As such they can find their place in geometry/mathematics, material science and a number of other fields.

We briefly introduce basic terminology in order to enhance readability of this manual. This introduction, however, cannot replace reading relevant textbooks - [1, 4, 6] all have chapters relevant to capillarity and immiscible flow - or technical papers.

5.1 Motivation: Resolving multiple scales in porous media and relevance of capillarity



a) b)

Figure 2: a) Columnar basalt - field scale. b) A piece of basaltic rock that fits into your hand - note pore structures (openings) of two different sizes (cm and less than mm). Source of images: Wikipedia.

Porous media flow parameters of interest are at the **field (macro)** scale, however **pore (micro)** scale has a strong influence (Fig. 2). Physics of the problem, as well

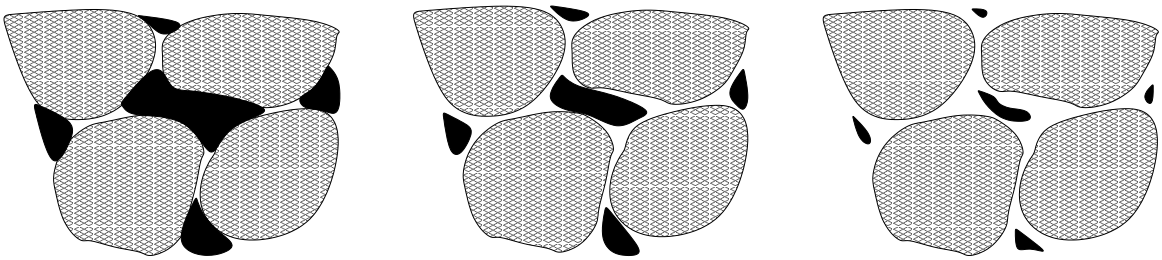


Figure 3: The wetting fluid is white, the non-wetting is black and solid phase is grey. **(left)** At very low wetting fluid saturations, the wetting fluid forms **pendular rings** which do not form a continuous phase. **(center)** With increasing wetting fluid saturation, the rings expand and at some critical saturation value the wetting phase becomes continuous. **(right)** As the wetting fluid saturation increases further, the nonwetting phase becomes disconnected.

as equations describing it, at different scales is different. Modeling difficulties include connecting models from different scales in order to account for heterogeneities, and if the macro-scale equations were obtained by some averaging/upscaling technique, providing closure relations in upscaled equations (before attempting to solve the problem). LSMPQS software focuses on investigating the pore scale phenomena, as well as provides input parameters for the upscaling effort.

If pore spaces are on micrometer scale or less (as is the case in e.g. sedimentary rocks, fine sands, tight gas sands or shale), that has a strong effect on the macroscale flow because of the capillary forces at the fluid-fluid interface are responsible for a range of nonlinear phenomena. The schematic effect of capillarity on fluid configurations is shown in Fig. 3. The ratio between viscous and capillary pressure drop is measured via dimensionless capillary number

$$N_{Ca} = \frac{\mu v}{\sigma},$$

where μ is viscosty of the wetting phase, v characteristic velocity and σ interfacial tension between the fluid phases. For capillary number approximately less than 10^{-4}, the flow is said to be dominated by capillary forces [6]. This is the situation where quasi-static modeling implemented by LSMPQS is valid.

In order to simulate the flow on the pore scale, one can either use model porous media (such as sphere packs, see Fig. 8), or imaged real media (Fig. 1a), typically from X-ray microtomography. Direct simulation using LSMPQS in such media can produce capillary-pressure relationships required for further upscaling on the field scale. In case that you are using network modeling, LSMPQS can help investigate critical curvatures for throat drainage and pore imbibiton required as input in the network flow modeling.

5.2 Basic concepts of immiscible fluid displacement

We are concerned with fluid displacement where the fluids do not mix so there exists a distinct interface between them. In a porous medium that includes two fluid phases in the pore space we recognize **wetting** (W) fluid as the one that preferentially adheres to the solid surface, and the other one is **non-wetting** (NW). For a fluid-fluid interface at

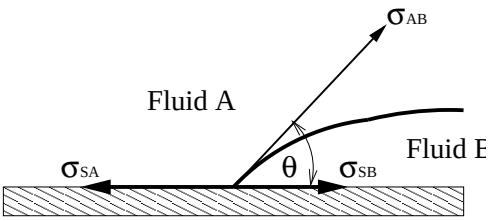


Figure 4: Contact angle θ in the system of two fluids and a solid at equilibrium satisfies $\sigma_{AB}cos(\theta) = \sigma_{SA} - \sigma_{SB}$. This relationship is still valid in the limit of $\theta = 0$.

equilibrium, this relationship is formalized by stating the contact angle (see Fig. 4) which is smaller than 90 degrees for the wetting fluid.

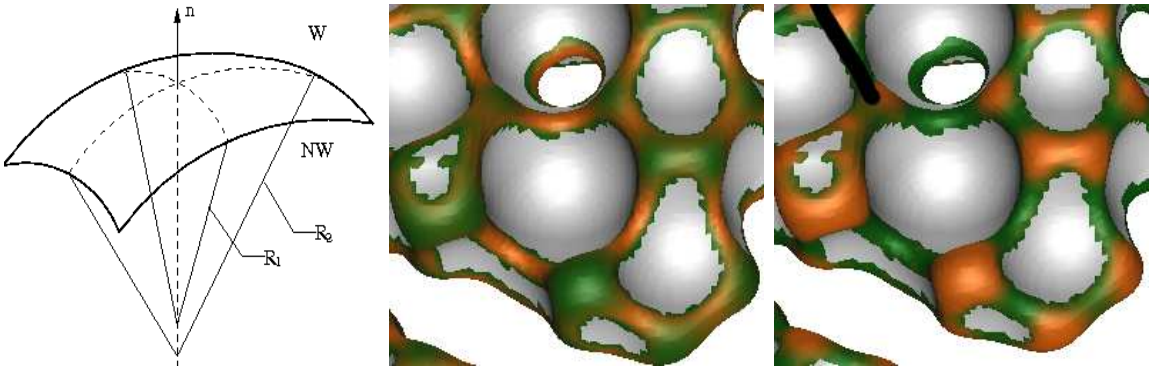


Figure 5: a) Geometry of the interfacial element of area between W and NW fluid phases. $\kappa_i = \frac{1}{R_i}$, $i = 1, 2$ are referred to as principal curvatures, and for interface at equilibrium their arithmetic mean is constant. b) An example NW phase surface: fluid-fluid interface is in colored according to κ_1 and the NW-grain contact is in gray (grains are spherical). The meniscus shown in has constant mean curvature value (approx. 2.1), but very different principal curvatures. κ_1 ranges from green (2.0 and below) to orange (6.0 and above). c) For the same meniscus of constant mean curvature, κ_2 ranges from green (-1 and below) to orange (1 and above). More details on the particular simulation the images were extracted from can be found in [13].

Pore scale fluid-fluid interface (**meniscus**) between fluids at rest supports a const. pressure difference (**capillary pressure**) p_c,

$$p_c = p_{nw} - p_w = \sigma_{AB}\kappa \tag{1}$$

where σ_{AB} is interfacial tension (Fig. 4) and κ is twice the mean curvature κ_M of the interface:

$$\kappa = 2\kappa_M = \frac{1}{R_1} + \frac{1}{R_2}. \tag{2}$$

Radii R_1 and R_2 are principal radii of curvature (the equation is valid for any two radii in two orthogonal directions, see Fig. 5a).

Note that principal radii in the above formula can be negative - saddle-like interfaces are one example. While in two dimensions the (mean) curvature is an inverse of the only possible radius of curvature, in three dimensions there is an infinite number of principal curvature pairs that sum up to a given mean curvature value. This is illustrated in Fig. 5b, c. Assuming the interface is spherical (as common in many modeling approaches) is thus very limiting.

For a slow quasi-static displacement, we model fluid-fluid interfaces as **constant mean curvature surfaces** and assume Young-Laplace equation is valid at each discrete curvature (equivalently, pressure) step. Constant mean curvature surfaces are not easy to calculate, and analytical solutions are known only for very simple geometries. Irregular porous media geometry further complicates the problem and accounting for the interface topological changes such as formation of pendular rings or snaped-off (isolated) blobs of non-wetting phase (see Fig 3.) is not easy. The level set method proves to be a robust tool for this purpose and is briefly introduced in §5.3. Finally, the methodology implemented in LSMQPS is uniquely distinguished by finding pore level menisci of constant curvature without any simplifying assumptions imposed on menisci or the bounding solid walls (such as those of sphericity).

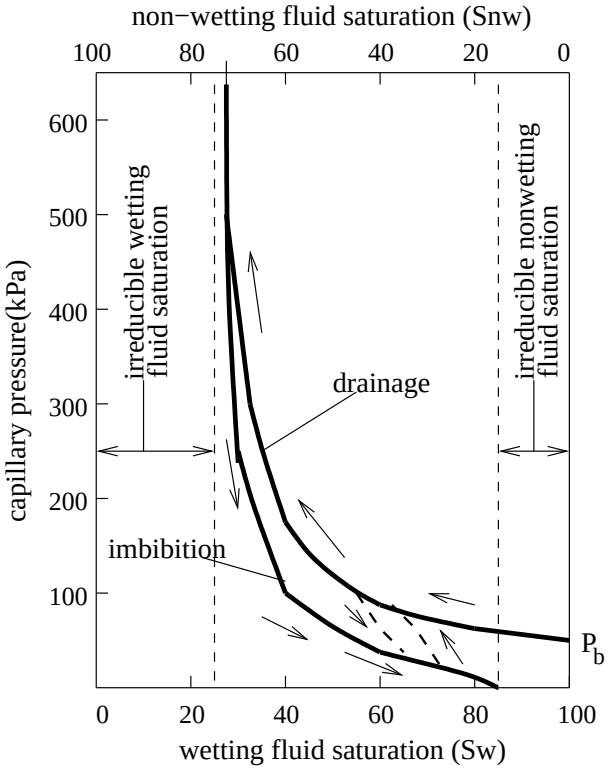


Figure 6: Typical capillary pressure - wetting fluid saturation (S_w) curves illustrating hysteresis.

Drainage is the displacement process when non-wetting fluid displaces the non-wetting fluid. In order to enter the saturated medium, a critical entry value of capillary pressure p_b must exist. Drainage continues through rapid local advances of the fluid inter-

face with local pressure exceeding the critical entry pressure value of the largest accessible throat constriction. **Imbibition** is a process of displacing non-wetting fluid by wetting. If a sample is saturated by a non-wetting fluid, introducing the wetting fluid at its surface is enough to start a spontaneous imbibition process.

The **saturation** of any fluid is defined as the fraction of the pore space it occupies (in the region of interest). Idealized $p_c = p_c(S_w)$ **capillary pressure-saturation relationships (curves)** are very commonly used to describe drainage and imbibition processes (Fig. 6.). The difference between an advancing and receding contact angle as well as the pore geometry (throats and surface irregularities) cause differences (hysteresis) in the capillary pressure curves for drainage and imbibition. Note that imbibition can be started at any point in the drainage process (and vice versa) as indicated by dashed curves between the drainage and imbibition curves in Fig. 6.

5.3 Level Set Methods

The level set method was introduced in 1988. by Osher and Sethian [9], and is by now a mature numerical method for propagating interfaces [8,18]. The moving surface of interest is defined implicitly as the zero level set of function $\phi(\vec{x}, t)$, i.e. it is modeled as the set of points $\vec{x}$ such that $\phi(\vec{x}, t) = 0$ at all times t (see Fig. 7). Such representation allows for elegant calculation of various interface properties. There is no unique way to embed the interface. The zero level set of ϕ is the zero level set of any multiple of the function ϕ, and whether to represent the inside of a closed curve with positive or negative values is a matter of preference.

The (spatial) gradient of the level set function, $\nabla\phi = (\phi_x, \phi_y, \phi_z)$, at a point is perpendicular to the level set passing through that point, that is, it is perpendicular to the surface $\phi = const$ at that point. Thus $\vec{n} = \frac{\nabla\phi}{|\nabla\phi|}$ is the normal of the interface (see Fig. 7a).

It is instructive and quick to derive the governing equation for interface movement. If we want to track the motion of a particle $\vec{x}(t)$ on the interface, then the differentiation with respect to time of the equation

$$\phi(\vec{x}(t), t) = 0 \tag{3}$$

leads to

$$\phi_t + \nabla\phi(\vec{x}(t), t) \cdot \vec{x}'(t) = 0. \tag{4}$$

Therefore, if we assume the interface is moving normal to itself with the speed F, we have $\vec{x}'(t) = F\vec{n}(\vec{x}(t))$.

This yields the governing partial differential equation (PDE)

$$\phi_t + F|\nabla\phi| = 0, \qquad \phi(\vec{x}, t = 0) \; given. \tag{5}$$

This equation is solved in all level set methods, and the physics of the problem depends on how one defines the speed F.

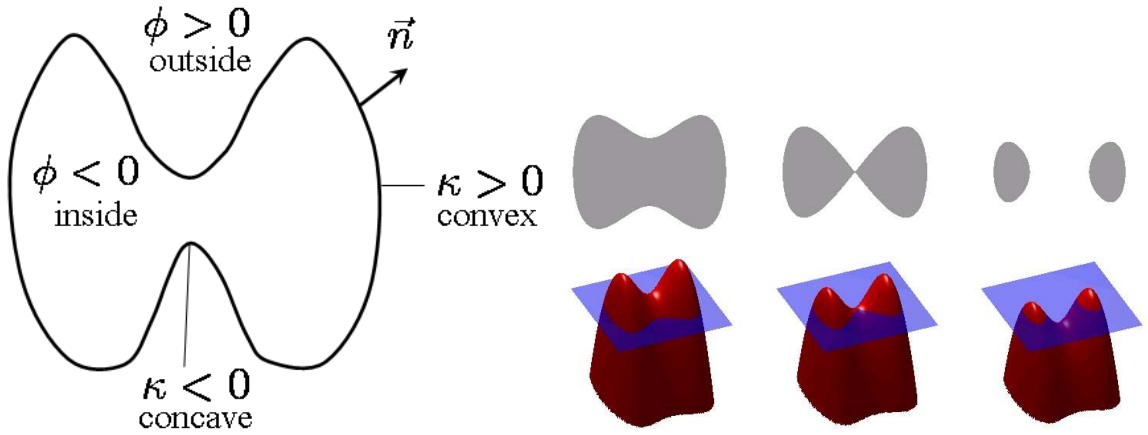


Figure 7: a) Schematic of a one-dimensional interface (closed curve), at an instant of time, described as as level set interface ($\phi = 0$) of a real valued level set function ϕ. The phase ($\phi < 0$) is a two dimensional subset of points in space (say, occupied by a blob of fluid), the phase ($\phi > 0$) represent its surroundings (say, another fluid). b) If you stack one-dimensional interfaces as they involve in time (vertical axis), you schematically obtain the surface colored in red. At different times, the time cross-sections yields different interfaces (in particular, the number of distinct interfaces changes). The apparent ease of changing the topology of the interface (and associated phases) is the main benefit of the embedding the interface into a higher-dimensional function. (Image b. source: Wikipedia.)

5.4 An example speed function

In our application, it is helpful to define F as follows (so called **prescribed curvature model**):

$$F(\vec{x}, t) = a_0 - b_0\kappa(\vec{x}, t). \tag{6}$$

Here, a_0 is a capillary pressure-like term, b_0 is the interfacial tension like term, and κ is twice the mean curvature. We will look for steady state solutions of the level set PDE with the above defines F: its non-trivial solution is the constant curvature interface $\kappa = a_0/b_0 (= p_c/\sigma$, cf. Eqn. (1)).

5.5 Implementation details

We briefly discuss a number of components required for numerical implementation of the Eqns. (5-6), or calculation of basic properties such as areas and volumes that are typically required when processing results. Many basic components are available through the library of level set methods LSMLIB [3] (which I happen to coauthor) since their application is more general than that of the cappilarity controlled fluid displacements. Throughout subsections below, I point to LSMPQS routines that put those LSMLIB components together into a specific application.

5.5.1 Curvature of an implicitly defined interface

For an interface defined as a zero level set of a function ϕ, twice the mean curvature κ is simply the divergence of the normal to the interface.

$$\kappa = \nabla \cdot \frac{\nabla \phi}{|\nabla \phi|}.$$

In three dimensions we can also define Gaussian curvature $\kappa_G = \vec{n} \cdot adj(He(\phi))\vec{n}$ where $adj(He(\phi))$ is transpose of the 3×3 Hessian matrix whose entries are all the second order partial derivatives of ϕ. The relationships of the mean, Gaussian and the principal curvatures of the interface are as follows

$$\kappa_M = (\kappa_1 + \kappa_2)/2$$

,

$$\kappa_G = \kappa_1 \kappa_2,$$

and

$$\kappa_{1,2} = \kappa_G \pm \sqrt{\kappa_M^2 - \kappa_G}.$$

In terms of partial derivatives of ϕ, mean and Gaussian curvatures are given as

$$\kappa_M = \frac{\left\{ \begin{array}{c} \phi_x^2\phi_{yy} - 2\phi_x\phi_y\phi_{xy} + \phi_y^2\phi_{xx} + \phi_x^2\phi_{zz} - 2\phi_x\phi_z\phi_{xz} + \phi_z^2\phi_{xx} \\ +\phi_y^2\phi_{zz} - 2\phi_y\phi_z\phi_{yz} + \phi_z^2\phi_{yy} \end{array} \right\}}{|\nabla\phi|^3} \tag{7}$$

$$\kappa_G = \frac{\left\{ \begin{array}{c} \phi_x^2(\phi_{yy}\phi_{zz} - \phi_{yz}^2) + \phi_y^2(\phi_{xx}\phi_{zz} - \phi_{xz}^2) + \phi_z^2(\phi_{xx}\phi_{yy} - \phi_{xy}^2) \\ +2[\phi_x\phi_y(\phi_{xz}\phi_{yz} - \phi_{xy}\phi_{zz}) + \phi_y\phi_z(\phi_{xy}\phi_{xz} - \phi_{yz}\phi_{xx} \\ +\phi_x\phi_z(\phi_{xy}\phi_{yz} - \phi_{xz}\phi_{yy})] \end{array} \right\}}{|\nabla\phi|^4} \tag{8}$$

Relevant LSMPQS routines are `compute_curvature2d`, `compute_curvature3d` (see Appendix B).

5.5.2 Calculating volumes and areas

The level set function ϕ typically describes a region of interest Ω, say by $\phi < 0$, and its zero level set describes the boundary of the region, $\partial\Omega$. It is convenient to define the Heaviside and the Dirac delta functions H and δ respectively:

$$H(x) = \left\{ \begin{array}{ll} 1, & x \geq 0 \\ 0, & x < 0 \end{array} \right. \tag{9}$$

$$\delta(x) = \frac{d}{dx}H(x) \quad \text{(in the sense of distributions)} \tag{10}$$

They will come in handy for computing other quantities of interest, such as volume occupied by a phase or area of an interface. In numerical computations the following "smoothed" formulas are commonly used for Heaviside and Dirac delta functions:

$$H(x) = \begin{cases} 0 & x < -\epsilon, \\ 0.5(1 + \frac{x}{\epsilon} + \frac{1}{\pi} sin\frac{\pi x}{\epsilon}) & -\epsilon \leq x \leq \epsilon, \\ 1 & x > \epsilon, \end{cases} \tag{11}$$

and

$$\delta(x) = \begin{cases} \frac{1}{2\epsilon}(1 + cos(\frac{\pi x}{\epsilon})) & -\epsilon \leq x \leq \epsilon, \\ 0 & otherwise. \end{cases} \tag{12}$$

where ϵ is a small value (on the order of numerical cell width). Then the surface (line in 2D) integral of a scalar-valued function $f(\vec{x})$ along $\partial\Omega$ is given by

$$A_f(\partial\Omega) = \int \delta(\phi(\vec{x}))|\nabla\phi(\vec{x})|f(\vec{x})d\vec{x} \tag{13}$$

and volume (area in 2D) integral of f in Ω

$$V_f(\Omega) = \int H(-\phi(\vec{x}))f(\vec{x})d\vec{x}. \tag{14}$$

The argument to the Heaviside function in Eqn. (14) assumes that the level set function is negative in the phase of interest. The integrals in Eqns. (13-14) are evaluated over the entire domain. This makes coding straightforward. If f is substituted with a function equal to unity everywhere, Eq. (13) yields the surface area (perimeter in 2D) of the fluid phase boundary $A(\partial\Omega)$, and Eq. (14) yields the volume (area in 2D) of the phase, $V(\Omega)$. Finally, the area average of a quantity f over the boundary is given by

$$\bar{f} = \frac{A_f(\partial\Omega)}{A(\partial\Omega).} \tag{15}$$

Relevant LSMPQS routines are `compute_volume`, `compute_perimeter2d`, `compute_area3d` and `compute_area_simulation` (see Appendix B).

5.5.3 Numerical discretization

The level set method PDE with the interface velocity defined in Eqn. (6) is discretized as follows:

$$\underbrace{\phi_t}_{\substack{TVD \\ Runge-Kutta}} + \underbrace{a_0|\nabla\phi|}_{\substack{hyperbolic \\ HJ\ (W)ENO}} \underbrace{- b_0\nabla\cdot(\frac{\nabla\phi}{|\nabla\phi|})|\nabla\phi|}_{\substack{parabolic \\ central\ diff.}} = 0 \tag{16}$$

Hamilton-Jacobi (W)ENO[1] schemes are suggested for the hyperbolic part of the equation, and central differencing for the parabolic (mean curvature term). TVD[2] Runge-Kutta schemes are suggested for the time discretization. More details on the mentioned schemes can be found in [8] and references therein. The implementation of the schemes is available in LSMLIB.

LSMPQS uses second order methods for both time and spatial discretization.

5.5.4 Reinitialization

Many level set functions can describe the interface (or domain) of your interest, but those whose norm of the gradient is 1.0 (so called signed distance functions, as their value at each point is a signed distance to the closest point on the zero level set surface) behave well in numerical simulations.

Steep or flat (norm of) gradient values can create numerical error to pile up and most of level set method simulations suffer unless the level set function is periodically replaced by a signed distance function (that describes the same zero level set). This process is called reinitialization.

Numerical solution is strongly affected by steepening/flattening of $|\nabla\phi|$, so ϕ occasionally needs to be replaced by the signed distance function.

LSMPQS periodically reinitializes (this is automatically taken care of, unless you specify otherwise). Should you wish to explore the reinitialization yourself, check out LSMPQS routine `reinitialize`, documented in Appendix B.

5.5.5 Localization

We have implemented a localized method which limits computation to a narrow band of voxels around the zero level set [11], which results in a more efficient code when the ratio of the size of the narrow band and the domain size is relatively small. This capability is again available through LSMLIB. An alternative to localization is adaptive meshing (not yet implemented).

You can use LSMPQS routine `visualize_narrow_band` (see Appendix B) if you are curious how narrow band looks like for some simulation output.

5.6 LSMPQS: Progressive Quasi-static Algorithm

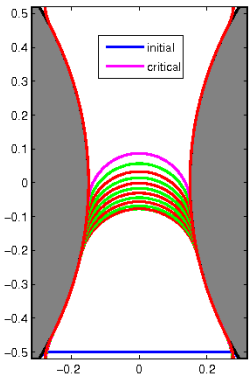
The original level set method describes the motion of interfaces that separate exactly two phases. Our application involves two fluid and one stationary grain phase and therefore triple junctions. In our application, the main level set function ϕ is negative only within the non-wetting fluid phase, i.e. its zero level set is the boundary between the non-wetting phase and all other phases. We also introduce a static level set ψ, also called 'mask', that describes the pore and solid phases. In particular, we adopt convention where $\psi < 0$ in the pore space, and $\psi > 0$ in the solid phase (its zero level set thus defines the pore-grain interface). The advantages of using a mask is the convenience of applying it when

[1] (Weighted) Esentially Non-oscillatory

[2] Total Variation Diminishing

the detailed geometry of the porous medium is known, e.g. from computed tomography images of samples or from geometrically determined model materials (see examples in §8).

5.7 Drainage and imbibition



In order to find curvature sufficiently large for NW phase to enter the medium, a planar (or circular) front at entry (whose interface is shown in blue on the left) is exposed to **slightly compressible curvature model** with normal velocity in the form

$$F(\vec{x},t) = a_0 exp[f(1 - \frac{V(t)}{V_m})] - b_0\kappa(\vec{x},t)$$

until steady state ϕ_I with corresp. pressure a_I is reached (the first red curve on the left).

At every further iteration we increment pressure by Δc (value of 0.1 used on the above figure; pressure is incremented by $\Delta a = b_0\Delta c$) and run **prescribed curvature model** introduced previously

$$F(\vec{x},t) = a_0 - b_0\kappa(\vec{x},t).$$

The NW phase interfaces at these subsequent steps are shown as alternating green and red curves in the above figure. The purple curve identifies the last stable meniscus in this small throat geometry, after which the throat drains completely (red curve lining the pore-grain interface).

Similar algorithm as for drainage is applicable for **imbibition**. We start from a drainage endpoint (with some corresponding curvature), reduce the curvature by Δc and run the prescribed curvature model.

Slightly compressible model is available as routine `compress_const_model` in LSMPQS, prescribed curvature model is available as LSMPQS routine `const_curv_model`, drainage as `drain`, for more examples see §9 and for details on routines see Appendix B. The particular geometry in the figure is `throat2d`, see §8.

5.8 Wall boundary condition

In order to force the fluid phase to stay within the pore space we impose a constraint of the type

$$\phi(x,t) \leq \psi(x)$$

which is very easy to implement ($\phi := max(\phi,\psi)$.

This results in a zero contact angle, corresponding to a surface perfectly wetted by one of the fluids. The disadvantage of this approach is that there is no straightforward way to model a nonzero contact angle, a situation of practical importance. This is not a fatal flaw of the method.

Using the variational formulation that minimizes the total energy due to pressure and interfacial tension allows specification of contact angle [23]. Presently, LSMPQS allows only for zero contact angle and the non-zero contact angle will be part of future distributions.

6 Options

Input parameters are managed via the structure Options, and that structure is the same for most programs regardless of whether the option is actually used or not. User can set options set by providing the program with a text input file that has a number of lines in the format

```
KEYWORD VALUE
```

(in any order, but bear in mind that if you set some option more than once, the last of lines takes precendence).

The program parses the input file for `KEYWORD` and sets the corresponding input option to `VALUE`. All available options have a default value in case the option is not set by the user. Use command

```
> print_input_options
```

to see avaiable input options and their default values.

The list below shows the result from executing `print_input_options`. Most of the options you can safely ignore so do not get overwhelmed by the list. Section §9 overviews the most important options and gives input files' examples. Further, textual output file for drainage and imbibition simulations prints options so you can double-check your simulation settings.

```
Options:

Parameters:
  a              0.1  [ constant; motion in normal direction ]
  a_input          0  [ 1 if normal velocity (a) term nonconstant
                        provide a_fname in that case        ]
  a_fname             [ normal velocity term input data ]
  b             0.05  [ constant; motion by mean curvature   ]
  dc             0.1  [ curvature increment (decrement)      ]
  compute_a_entry  1  [ compute entry pressure (1) or not (0)]
  narrow_band      1  [ apply narrow banding (1) or not (0)]
  ex               1  [ constant; x component of external vel. field ]
  ey               0  [ constant; y component of external vel. field ]
  ez               0  [ constant; z component of  external vel. field ]
  omega            4  [ constant; frequency of  external vel. field ]
  f                1  [ constant in compressible model ]
  vol_frac_target 0.8 [ volume fraction constant in compressible model ]
  k                8  [ scaling constant in sediment mechanics]
  rigid          0.8  [ rigid sphere fraction in sediment mechanics]
  radius_scaling   1  [ radius_scaling for pfc balls input in sediment mechanics]
  conserve_volume  0  [ conserve volume (1) or not (0) in variational method]
  no_penetration   0  [ no_penetration constraint in variational method]
```

```
Geometry:
  dx             0.02  [ x-dir grid spacing ]
  dy             0.02  [ y-dir grid spacing ]
  dz             0.02  [ z-dir grid spacing ]
  do_mask           1  [ use mask  (1) or not (0)]
  num_geom          1  [ number of different mask(solid) geometries]
  geom_type0    duct2d  [ geometry type ]
  geom_fname0           [ geometry input data ]
  do_seal0           1  [ seal volume sides - all (2), orthogonal to flow(1)
                          or none (0)]
  seal_coefficient  1.5  [ sealing plane distance = seal_coefficient*dx]
  reservoir_inlet   0  [ mimic fluid reservoir at inlet (1) or not (0)]
  move              0  [ move grains(1) or not(0) (spherical grains only)
  extrapol_id       0  [ volume boundary extrapolation - 0 (signedLinear)
                         1(linear), 2 (copy), 3 (zeroLevelSet)]

Initial data:
  ic                x  [ initial condition ('x','y','z' or  'i')]
  ic_pos            3  [ ic position (grid (integer) coordinate)       ]
  ic_type              [ initial cond. geom. type (if ic='i')]
  ic_fname             [ initial cond. input data ]

Stopping criteria:
  tmax             20  [ max running time allowed ]
  cmax            100  [ maximum curvature for simulation     ]
  vol_frac_max  0.97  [ max. NW phase volume fraction for drainage  ]
  stop_touch        1  [ stop when opp. bdry is touched (1) or not(0)]
  stop_jump_volume 0  [ stop volume fraction changes by 0.1 (1) or not(0)]

Reinitialization:
  do_reinit         1  [ reinitialize periodically (1) or not (0)]
  subcell_fix       1  [ use subcell fix in reinitialization (1) or not (0)]
  tmax_r          0.2  [ max running time for reinitialization ]
  mask_reinit       0  [ reinitialize mask (1) or not (0)]

Trapping:
  do_trap_nw        0  [ enable NW phase trapping  (1) or not (0)]
  init_trap_nw      0  [ initialize trapped NW phase from file  (1) or not (0)]
  trap_nw_phase_fname  [ trapped NW phase input data ]
  do_trap_w         0  [ enable W phase trapping  (1) or not (0)]
  init_trap_w       0  [ initialize trapped W phase from file  (1) or not (0)]
  trap_w_phase_fname   [ trapped W phase input data ]

Other:
```

```
checkpoint       0  [ output data files every tplot  (1) or not (0)]
debug            0  [ print debug info  (1) or not (0)]
email               [ email address for end-of-simulation notice ]
eps_coefficient  1.5  [ eps = eps_coefficient*dx]
eps_modified     0  [ eps_modified in variational method]
outfile   out_file  [ output file name ]
print_details    1  [ print details (1) or not (0)   ]
save_data        1  [ save data (1) or not (0)       ]
tplot          0.5  [ time interval for evaluation   ]
```

7 Visualization

Limited number of visualization routines are available and reflect software used by the author.

- ◇ **Matlab; 2D, limited 3D** Most two dimensional results and graphs (such as curvature-saturation relationships for drainage and imbibition simulations) are done in Matlab. For larger 3D surface visualization, I prefer Geomview.

  Routine `addPathToMatlab.m` in `src/` directory of the LSMPQS distribution will add its subdirectories to Matlab path.

- ◇ **Raster images (2D)** using 'raster' command will produce pixelized image of the NW phase and solid phase. In other words, the interfaces between the phases are not smooth: there is a staircase effect. Any image viewer should be able to open these files (for instance, 'gimp' [17] is freely available cross-platform software.)

- ◇ **Geomview/GTS ; 3D surfaces** Matlab has limited capability of displaying larger triangulated surfaces, and I created an isosurfacing routines based on GTS Library [7] which output the surfaces in the files loadable in Geomview [5].

- ◇ **Add your own routines.**
  Isosurfacing (plotting a fixed level set of a function) is widely available in many 3D visualization software, some open source examples are Drishti [21], Paraview [10], Visit [20] and Mayavi [19]. You can read in the level set function binary data arrays, and possibly grid data, and manipulate them in your favourite one. Matlab routines for reading data arrays and grid area available within LSMPQS (`readDataArray.m`, `readGridBinaryFile.m` in `src/visualization/matlab/`) and equivalent C routines are available within LSMLIB Serial package. If you would like to resize or rotate a data array, or strip its header, refer to LSMPQS routines `resize_data_array`, `rotate_data_array` and `strip_array_header` (documented in Appendix §B).

8 Pore Space Geometry

Single most important input into a LSMPQS simulation is geometry of the pore space. Masking function that describes the pore space is yet another level set function (ψ) with

negative values where pore space is ($\psi < 0$ and positive values where the solid space is ($\psi > 0$).

You can see the list of available geometries by typing

```
> print_geometry_options
```

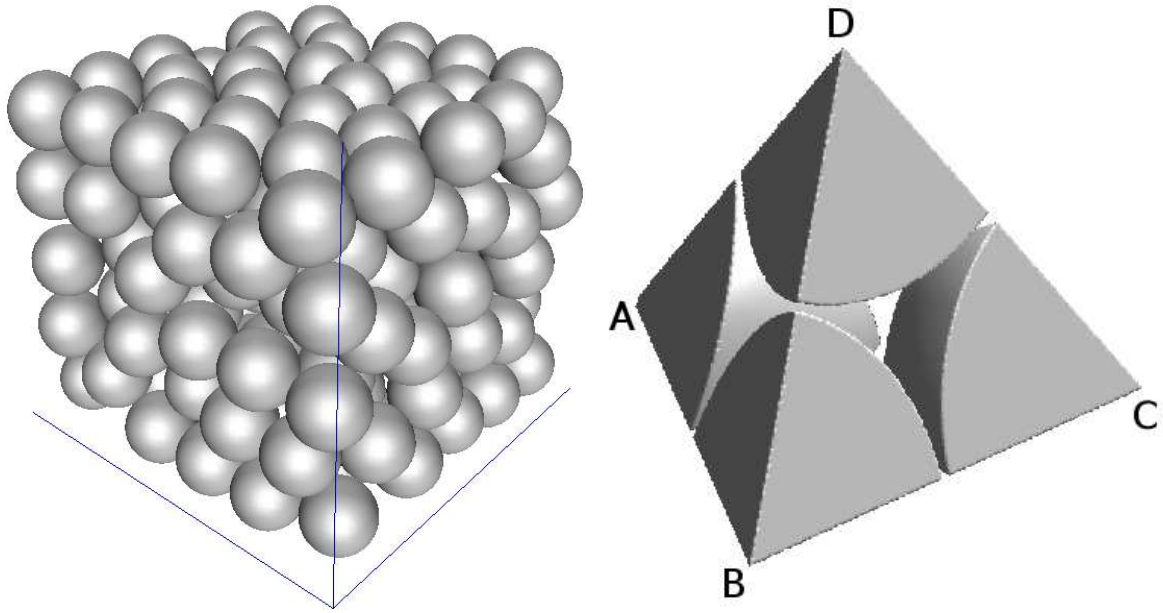
the result in the current LSMPQS version is

```
 ID                       TYPE      FNAME FUNCTION
  0                     duct2d     n create_mask_duct_2d
  1                    throat2d     n create_mask_throat_2d
  2                 throat2dnew     n create_mask_throat_2d_new
  3           parallel_throat2d     n create_mask_parallel_throat_2d
  4            inverse_throat2d     n create_mask_inverse_throat_2d
  5          constricted_tube2d     n create_mask_constricted_tube_2d
  6 doubly_constricted_tube2d     n create_mask_doubly_constricted_tube_2d
  7                   biconic2d     n create_mask_biconic_2d
  8           inverse_biconic2d     n create_mask_inverse_biconic_2d
  9               biconic2dflat     n create_mask_biconic_2d_flat
 10                  granular2d     n create_mask_granular_2d
 11                  fracture2d     n create_mask_fracture_2d
 12                    seg_data     y create_mask_seg_data
 13            LSMLIB_REAL_data     y create_mask_LSMLIB_REAL_data
 14        LSMLIB_REAL_data_pad     y create_mask_LSMLIB_REAL_data_pad
 15                      duct3d     n create_mask_duct_3d
 16                     throat3d     n create_mask_throat_3d
 17                    biconic3d     n create_mask_biconic_3d
 18              delaunay_thr3d     n create_mask_delaunay_thr_3d
 19          new_delaunay_thr3d     n create_mask_delaunay_thr_3d_new
 20             delaunay_pore3d     n create_mask_delaunay_pore_3d
 21 doubly_constricted_tube3d     n create_mask_doubly_constricted_tube_3d
 22           general_spheres2d     y create_mask_general_spheres_2d
 23           general_spheres3d     y create_mask_general_spheres_3d
 24         general_cylinders3d     y create_mask_general_cylinders_3d
 25         general_polyhedra2d     y create_mask_general_polyhedra_2d
 26                 pfc2d_balls     y create_mask_balls_pfc_2d
 27                 pfc3d_balls     y create_mask_balls_pfc_3d

Use TYPE from the above list as an input option, i.e. include
geom_type TYPE
as a separate line in your input file.

In addition, if FNAME is ''y'' you have to provide an appropriate
file input for that geometry and supply file name as follows
geom_fname FILE_NAME
```

The number (and possibly clumsy names) of the geometries has piled up and I left them all in the software hoping they might be helpful. The most flexible (and potentially the most interesting geometry types) will require additional information stored in a binary file that will be supplied by including the following line in the input file:

```
geom_fname FNAME
```

When specifying file names in the input file, make sure that `FNAME` is the name of an existing file (e.g. `'mask.gz'`) and not just its base name (`'mask'`). The code will, however, recognize '.gz' extension and unzip the file accordingly.



a) b)

Figure 8: a) Dense sphere packing - common model of a granular medium. b) Tetrahedral cell resulting from Delaunay tessellation of a dense pack of spheres. Centers of the neighboring spheres are marked A, B, C, D. Each tetrahedral sphere represents a pore opening that is connected to its four neighbors via throats (tetrahedral sides). Use LSMPQS geometry 'delaunay_pore3d'.

To help you create binary input `src/geometry/matlab/` directory contains routines that read and write general_spheres3d (creates input for geometries like in Fig. 8, general_cylinders3d or segmented file types (geometry like in Fig. 1). Names of the matlab functions should be self-explanatory and are commented so you can use Matlab 'help' function in order to learn how to use each of the functions. Here is the list of the functions (you can use 'help' in Matlab to learn more about them.

```
createMicromodelCylinders3D.m
createMicromodelDisks2D.m
createMicromodelSpheres3D.m
```

```
createRhombusSphereArrangement.m
readGeneralSphereBinary.m
readSegmentedFile.m
writeGeneralCylinderBinary.m
writeGeneralPolyhedraBinary.m
writeGeneralSphereBinary.m
writeSegmentedFile.m
```

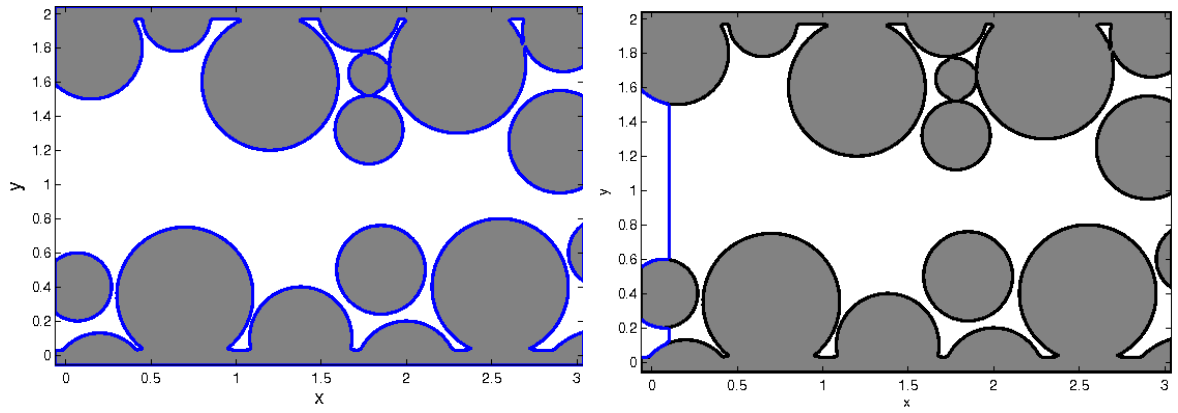
C equivalents are located within `base/geom_util.c`, however not as standalone routines. If neither of these are your favourite programming environments, it should be easy to adapt them to a language of your choice.

To check your input file before simulation (or to plot some desired geometry) you can use `'initialize_geometry'`. For example, create an input file that only contains a line `geom_fname fracture2d`. Note that ALL other input options will be set to default (and that will do for the time being). Now execute

```
> initialize_geometry in_file
```

This creates files `mask.gz` and `data_init.gz` that are LSMLIB data array files storing level set functions that define masking geometry and the initial fluid configuration, respectively. In addition you can find a binary file `grid.gz` storing grid information. If you would like to find out details of the created grid (discretized volume) execute

```
> print_grid grid.gz
```



a) b)

Figure 9: a) plotZeroLevelSet result for a 2D masking function. Positive values of the function are plot in gray, zero level set is outlined in blue and negative values are in white. b) plotZeroLevelSetWithMask will plot zero level sets of provided data arrays in desired colors, and the masking level set function will have its positive values plot in gray.

Let's now visualize the result in Matlab.

Open Matlab and execute function `addToMatlabPath` stored in LSMPQS `src/` directory. This adds all of the LSMPQS subdirectories to matlab path so they are easily

accessible. Note that matlab utility `'help'` works with all of the LSMPQS .m files and should give you information on how to use each function.

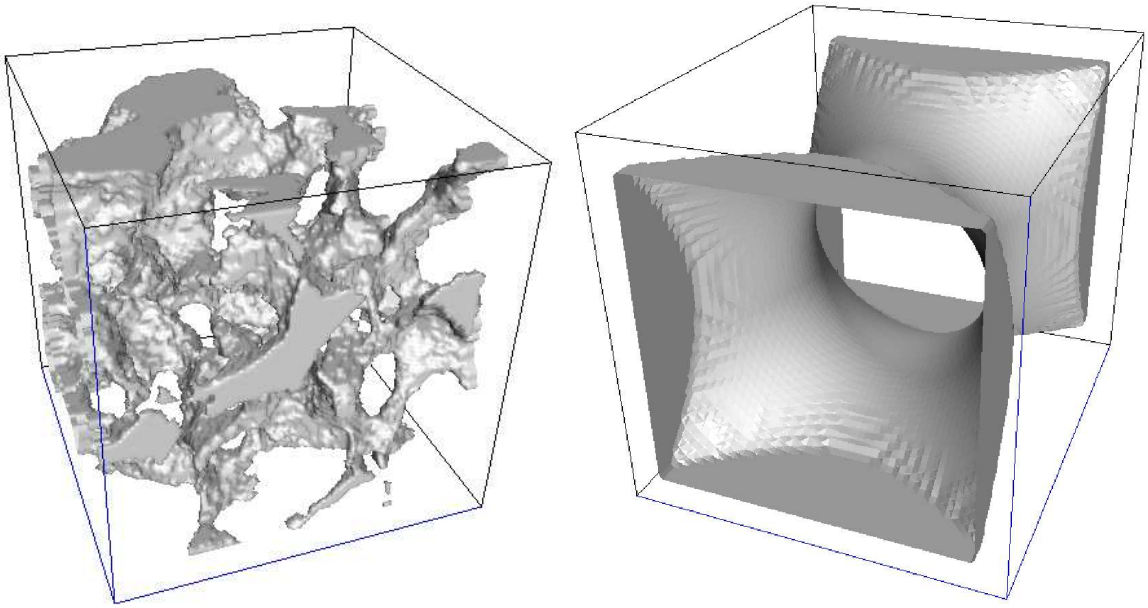
```
> plotZeroLevelSet('mask','grid')
```

If you would like to see both initial fluid-fluid interface and the solid phase (mask),

```
> plotMultipleZeroLevelSetsWithMask('data_init','b','mask','grid')
```

The results are shown in Fig. 9.

Note that the files we are trying to open are zipped (.gz extension). Matlab functions will recognize and uncompress it regardless of whether you include '.gz' extension in the file name. Further, `data_init.gz` stores NW phase initialized, by default, as a plane. For most readily available geometries, inlet and outlet sides of the volume are (conceptually) in the x-direction. The initial plane position is by default in x direction (input option `ic_pos` specifies how many numerical cells away from the inlet boundary.



a) b)

Figure 10: a) Geometry created from segmented data (input file provided in 'examples/berea3d' and visualized in Geomview). b) Predefined geometry named 'throat3d'.

`examples/berea3d` contains a 100^3 segmented file `berea100.gz` of Berea sandstone (subset of the sample available from installation instructions on [22]). To simply visualize the pore-grain interface create an input file with the following lines

```
geom_type seg_data
geom_fname berea100.gz
```

and again execute `initialize_geometry in_file`.

You can use the same commands in Matlab as in two dimensional case in order to visualize the files. Now we show another way of visualizing it. Assuming LSMPQS was configured with GTS enabled, and you have Geomview installed on your system, executing

```
> isosurface mask.gz
```

will create Geomview style `surface.list` file in the current directory (and if you execute the program multiple times, this file will constantly be overwritten). Geomview will open the file and the result should look like Fig. 10a. Another example geometry is shown in Fig. 10b.

The command `'isosurface_peel 3 mask.gz'` will 'peel off' the three numerical cells added to each volume side for boundary conditions (creates neater plot). Feel free to "peel" as many cells as you would like, of course. `isocolor` and `isocolor_peel` do the same, but you can specify different colors for the surface and plot multiple files together (if you type just the name of those routines, you will get suggested usage). Try `'isocolor data_init.gz b mask.gz n'`. (see more information on `isosurface`, `isocolor`, `isosurface_peel`, `isocolor_peel` in Appendix B).

You can also set the option `reservoir_inlet 1` in the input file to create "a reservoir" `ic_pos` numerical cells away from the volume boundary to make it "easier" for fluid to find a stable configuration at an entry of a tight, complicated geometry (set input option `ic_pos` as far away as desired). A suggested input file for a drainage simulation is included in the same directory, but be sure you understand §9 before attempting it.

9 Examples

9.1 2D fracture

In order to simulate drainage, create the file `in_file` with the following information:

```
geom_type fracture2d
dx 0.02
ic x
ic_pos 5
tmax 360
dc 0.4
```

This input file can be found in `examples/fracture2d`. Input files are parsed line by line, so do not specify more than one option in the same line.

Executing

```
> drain in_file
```

on the command prompt (`'>'` stands for Unix shell command prompt) will run a drainage simulation in 'fracture2d' geometry visualized in Fig. 9.

After the simulation completes, you will see the following files created in the directory: binary data arrays `mask.gz` (masking level set function), `data_init.gz` (initial NW

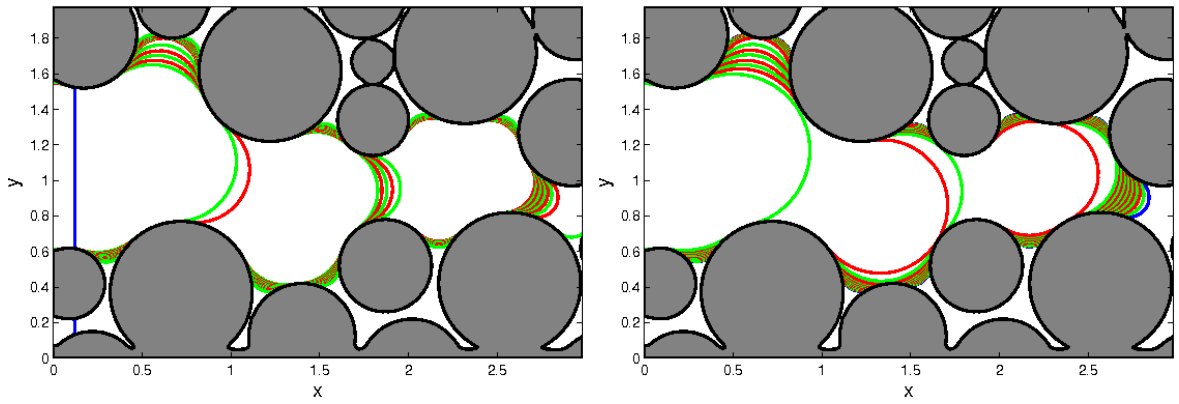


Figure 11: NW phase interface visualization (in alternating green and red colors) for all steps of a) drainage, and b) imbibition simulations. Solid (masked) phase is shown in gray and initial NW phase interface is shown in blue.

phase), and `data_stepSTEP_ID.gz` (NW phase for each step of the simulation - there was 17 steps in my case, though that can slightly change depending on the initialization on different systems), text file `out_file` that has more information that you would like to know about the simulation (options, grid information and the log for each step) and one dimensional data arrays `a.gz,curvature.gz,vol_frac.gz` that contain pressure term (a), curvature (a/b, coefficient b is set via input file and printed in `out_file`) and volume fraction of NW phase (relative to the available pore space) for each step of the simulation.

Matlab visualization of all steps

```
> plotMultipleZeroLevelSetsEasy('./')
```

will produce a plot in Fig. 11a.

To start an imbibition simulation in the same directory you need to pick first the drainage step to start it from. Let's do it from step before last (16 in my case, might slightly differ for you). Imbibition starts from a drainage endpoint and thus the geometry and initial conditions can be set accordingly. We will create a minimal set of input options is

```
tmax 360
dc -0.4
```

and, assuming it is stored in a file `in_file_imb` we can start imbibition simulation as follows:

```
> imbibe in_file_imb 16
```

This creates a subdirectory `imbibe16/` with output files for the imbibition simulation (note that step numbering starts from 2, unlike drainage). As a matter of fact, even if we used the same input file as for drainage, we would still be fine - imbibition simulation automatically converts `dc` to a negative number as curvature is supposed to decrease

and all other options are either irrelevant or superceded by using drainage geometry information.

In Matlab, use command `plotMultipleZeroLevelSetsEasy('./imbibe16/')` to create the plot in Fig. 11b.

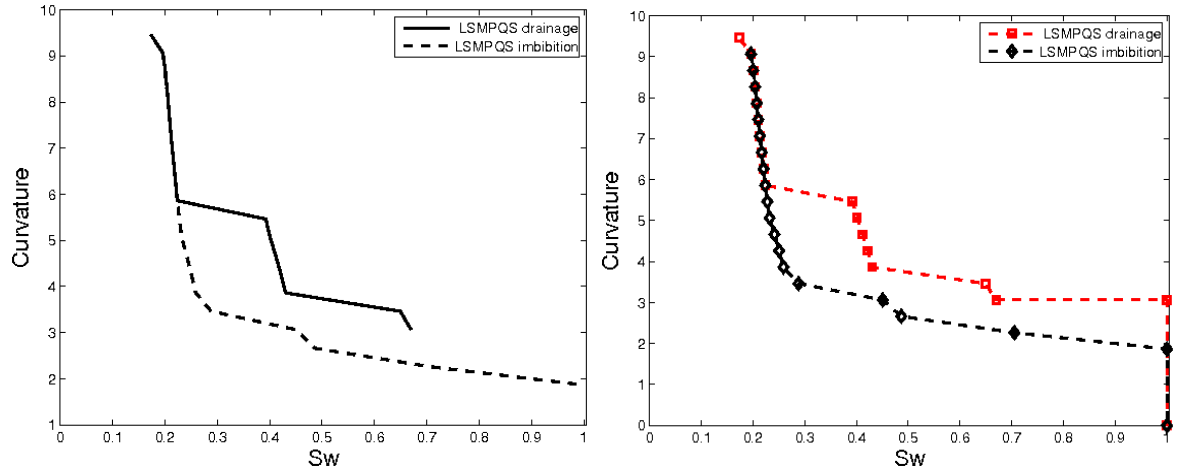


Figure 12: Two different ways to plot curvature - saturation relationship for drainage and imbibition simulations.

```
> plotCurvatureSaturation(1,1,0,0,'./','./imbibe16/')
```

will produce curvature saturation relationship (use `'help plotCurvatureSaturation'` to learn more.

You can easily modify linestyles of course in the original .m file. So if you modify the first two lines in `src/visualization/matlab/plotCurvatureSaturation.m` to

```
linestyle{1} = 'rs--';  %drainage curve
linestyle{2} = 'kd--';  %imbibition curve
```

and replot this time using `'plotCurvatureSaturation(1,1,0,1,'./','./imbibe16/')` with `'connect_to_01` variable is set to 1. This variable has (only) visual effect in graph appearance connecting to the saturation point 1.0. Both graphs can be seen on Fig. 12.

Finally, we will compute meniscus (i.e fluid-fluid) area in the pore space. The program needs to be run in the directory where simulation has taken place and one needs to specify whether the simulation is drainage('d') or imbibition('i') as follows:

```
> compute_area_simulation d
```

This computation will record arrays `area_nw.gz, meniscus_area.gz` that measure area for each step of the simulation as well as a text (ASCII) file `area_solid` with the total pore-grain surface area. In order to plot the meniscus area (normalized by solid area), use the following function in Matlab:

```
> plotAreaSaturation(1,1,0,0,'./')
```

Note that plotAreaSaturation.m, plotCurvatureSaturation.m and plotAreaCurvature.m require same parameters (and are very much similar). `'scaling'` variable can be used to scale the area/curvature values to physical ones, and all of these routines can return the arrays plotted, should you wish to manipulate them further.

9.2 3D pore/segmented data

We describe an example that is both a three-dimensional geometry as well as one described by a segmented file, extracted from an x-ray microtomographic image of a Berea core. Input files can be found in `examples/berea_pore3d/drain/` directory. The input file

```
geom_type seg_data
geom_fname ../pore63rot.gz
dx 0.05
a 0.1
b 0.05
ic_pos 10
vol_frac_target 0.5
do_seal 0
narrow_band 1
dc 0.5
stop_touch 0
tmax 100
```

specifies geometry first (the segmented file `pore63rot.gz` should be in the directory `examples/berea_pore3d/` which is one directory up (that's what `../` stands for, if you're new to Unix/Linux).

Let me point out a couple of more details about the input file. The segmented file does not carry any length information (except for fixed number of cells), and we specify the length of a cell(voxel) to be `dx=0.05` which is somewhat arbitrary (and can always be scaled later to a meaningful physical length, which is in this case $4.93\mu m$. Note that the simulation depends on curvature (twice the mean curvature) which is defined by the ratio $\frac{a}{b}$. If you don't set `b` it is set to `dx` value by default (so setting it as above was not necessary). Further, we do not "seal" the volume with a solid wall boundary condition (`'do_seal 0'`) which means that fluid can freely flow through the open parts of the boundary (as a matter of fact, you will notice that the NW phase interface appears to cut the volume boundary at 90 degree angle due to extrapolating boundary condition). Further, `narrow_band` is set to 1 (which is also a default value, but is mentioned here to stress that you should use so called localized methods that compute in the neighborhood of the interface only, in order to reduce the computation time. Finally, the stopping criterion `'stop_touch 0'` ensures that the simulation does not stop when the interface "breaks through" and touches the opposite volume side (outlet x-direction side). `tmax 100` ensures that there is plenty of time to reach the steady state solution in each step (but puts a

limit in cases where there is no such solution - either due to numerical resolution or simply because an stable interface at desired curvature does not exist).

As before, `'drain in_file'` will perform a drainage simulation. By default, the simulation will complete when 97% of the pore space becomes occupied by NW phase (controlled by option `vol_frac_max`, you can explore some other stopping criteria in §6). If you wish the system to email you when the simulation is done, add the line `email YOUR_EMAIL` in the input file. Drainage simulation took 1 hour on my desktop machine and produced 34 curvature steps in between.

In any simulation, the time needed to complete the simulation depends on a number of paramaters. The spatial and curvature step (dc) are most important, of course, but finding a stable, steady state solution at each curvature step also depends on how complicated the pore space is. For the same number of pore voxels (numerical cells) a single converging-diverging geometry (throat) of course is much easier to tackle than a sandstone-like maze of interconnected channels. Regardless, all the published results for this method were done on a single processor, even if the largest ones [15] took a couple of weeks to complete.

If you would like to visualize individual surfaces, you can use routines mentioned in §8 (for instance, `isocolor_peel 3 data_step13.gz r mask.gz n` will plot NW phase interface for the specified curvature step and pore-solid surface at the same time. Note that you can improve the appearance by choosing Smooth shading in Inspect-¿Appearance menu option in Geomview).

If you would like to produce plots for all (or a specified range) of drainage steps, execute

```
> create_geomview_files 1 34
```

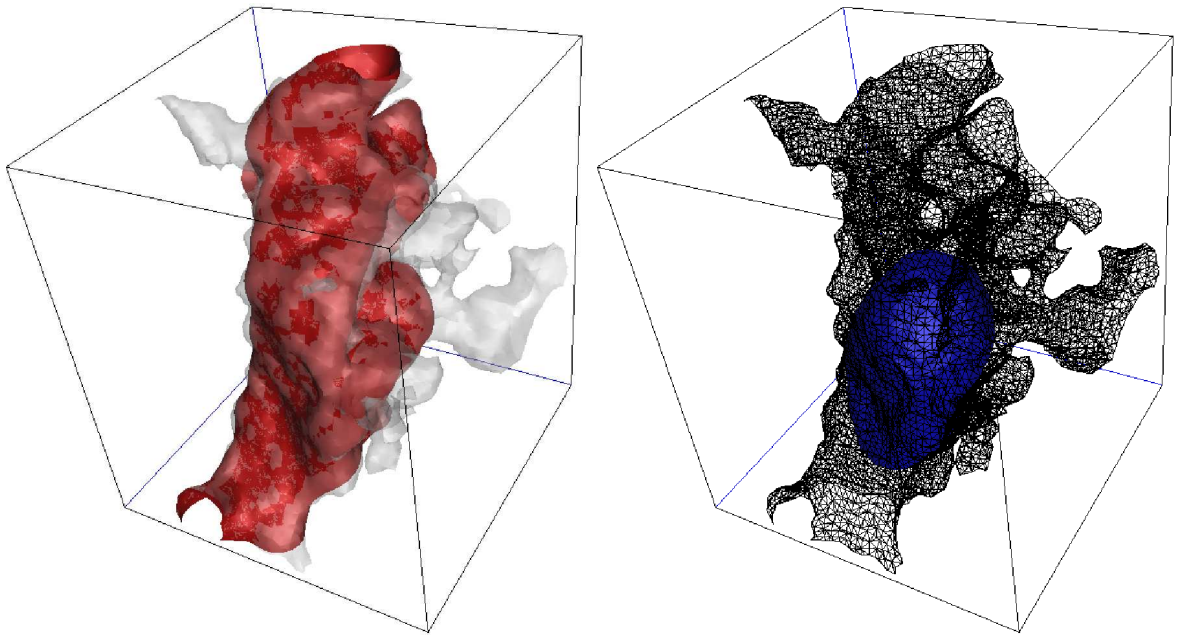
This creates a subdirectory `geomview/` with surface files for each of the steps. Files of the type `surf_dSTEP_ID.list,surf_mask.list'` are "raw surfaces" and the files of the type `view_dSTEP_ID.list` rotate the geometry so that the entry face in x-direction is in the back and add smooth shading. If you open any of the `view*list` files with a text editor, you will see that it is a template that plays with the appearance and position/rotation of `surf*list`. There are more templates available in `src/visualization/geomview_templates`, feel free to experiment. Let's now produce a nice plot:

```
> cd geomview/
> geomview view_d13.list surf_mask.list
```

When geomview opens, select `'surf_mask.list'` on the list of 'Targets'. Then open 'Inspect-¿Material' from the menu, check 'Transparent' and set 'Alpha' to be 0.3. The result is shown in Fig. 13a.

Let's perform the imbibition simulation as well. We have a range of drainage endpoints to pick from, say the above visualized drainage step 13 (`out_file` in my simulation claims step has curvature `c=a/b 10.1557`). Input file similar to drainage can be used (we will call it `in_file_imb`) with the following differences:

```
dc -0.4
do_trap_nw 1
```



a) b)

Figure 13: By convention, the NW phase surface in Geomview plots is shown as red, trapped NW phase as blue and pore-grain surface as gray. a) Drainage NW phase configuration in Berea pore at $C = 10.56$, the grain surface is shown as transparent. b) Imbibition (trapped) NW phase configuration in Berea pore at $C = 3.66$, with only the grain surface edges shown in black.

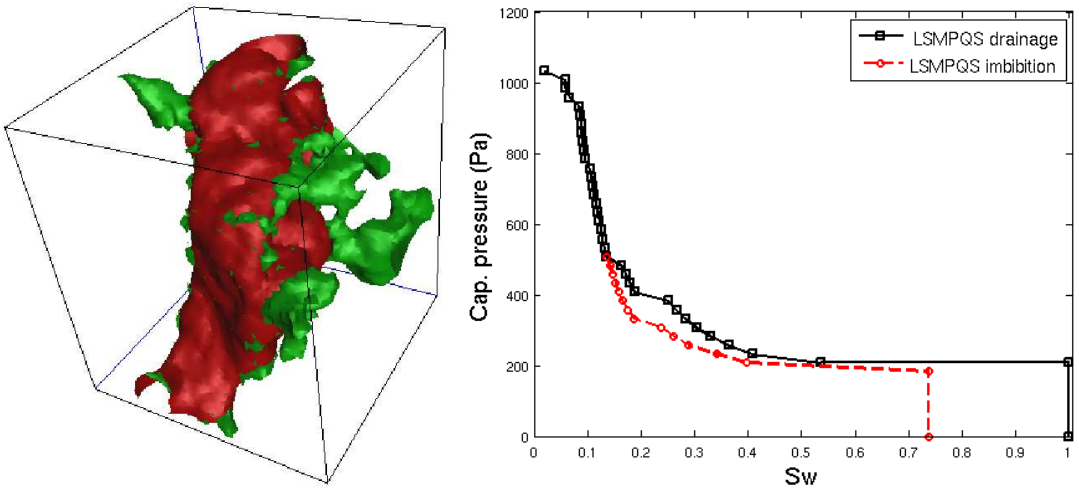
The latter option periodically checks for and records disconnected (trapped) blobs of NW phase that would otherwise disappear due to curvature term growing infinitely. If trapped phase exists, it will show up recorded (as negative phase) in binary data arrays that have `"_trap_nw"` indicator in their name (and can be visualized using isosurfacing routines, as usual). Make note, trapping is available only in three-dimensional localized (narrow band) code and trapping of wetting phase is available as well (`do_trap_w` is the relevant input option).

```
> imbibe in_file_imb 13
```

will run imbibition simulation. It appears the non-wetting phase is trapped in step 14 (curvature 3.66) and at the same time the rest of the NW phase withdraws from the volume through the open boundaries. You can visualize all steps similarly as before

```
> cd imbibe13
> create_geomview_files 1 14
> cd geomview
> geomview view_d14.list surf_mask.list
```

When geomview opens, select `'surf_mask.list'` on the list of 'Targets' and this time around open 'Inspect-¿Appearance' from the menu, check 'Edges' and check off 'Faces' to



a) b)

Figure 14: a) Drainage NW phase interface in Berea pore at $C = 10.56$ is shown in red and wetting phase in blue. b) Capillary pressure - saturation curve for drainage and imbibition in Berea pore assuming interfacial tension $\sigma = 5 \cdot 10^{-3} \frac{N}{m}$.

show grain surface as wiry frame through which to see the trapped NW phase blob shown in Fig. 13b.

9.3 Where is the wetting phase?

We have visualized the interface of NW and grain phases only so far, so this is a legitimate question. NW phase stored in data arrays where the store level set function is negative, $\phi < 0$, and, similarly, pore space is where 'mask' is negative, $\psi < 0$. W phase is then the part of the pore space where NW is not, i.e. $(\phi > 0)$ & $(\psi < 0)$. Use

```
> reverse_phases data_step13.gz data_step13_w mask.gz
> isocolor_peel 3 data_step13_w.gz g
```

to create a file `data_step13_w.gz` in the current directory that corresponds to the wetting phase (at step 13), and then visualize its surface (in green color). Note that the latter visualization does not open Geomview automatically (the reason is `create_geomview_files` routine we used above: it's not a good idea to have Geomview windows pop open for all the steps visualized. Isosuurfacing routines create `surface.list` in the current directory. Thus

```
> geomview surface.list geomeview/view_d13.list
```

will produce the plot in Fig. 14a.

9.4 Scaling to physical values

Just in case you were wondering what is the volume of the wetting phase, you can find out as follows:

```
> compute_volume data_step13_w.gz grid.gz

Warning: second order derivatives storage turned off
in set_array_allocation_curvature_model()
V1 = volume (data < 0) 0.5216
V1 by voxel counting   0.52525
V2 = volume (data > 0) 17.6516
V = brick domain volume (no ghostcells) 18.1688
V1+V2 =(should be equal to brick volume)18.1732 error 0.02%
```

Don't worry about the printed warning (unless you get a segmentation fault). The volume is expressed in whatever the length units we chose for the simulation. The numerical spacing `dx=0.05` is equivalent to $l = 4.93\mu m$, thus the physical value of the volume above is $V_1(\frac{l}{dx})^3 = 5 \cdot 10^{-4} mm^3$...by all means a small quantity.

As a final scaling exercise, we will convert generic curvature-saturation plot to capillary pressure. We will still use `plotCurvatureSaturation`. Assuming interfacial tension of $\sigma = 5 \cdot 10^{-3} \frac{N}{m}$, and approximating $l = 5\mu m$, in Matlab we set scaling to be $\sigma \frac{dx}{l}$ and the following

```
> scaling = 5*10^(-3)*0.05/(5*10^(-6));
> plotCurvatureSaturation(1,scaling,0,1,'./','./imbibe13/')
```

The result is shown in Fig. 14b. Note that when scaling is not 1, the plotting function automatically calls the y-axis 'Cap. pressure' instead of 'Curvature'. You can change that, of course, to fit your scaling.

9.5 Checkpointing, data recovery and continuing simulation

If you expect a longer simulation, and would like to periodically record intermediate data, add the line `'checkpoint_data 1'` to your input file. This will write binary data array file `checkpoint_data.gz` to the current directory every time period `tplot`. Note that the file is constantly overwritten. Further, `tplot` is an input option that can be changed.

Further, if your drainage or imbibition simulation has abruptly stopped (for reasons ranging from accidentally closing your shell window to power outage), you need to extract information (like curvature, volume fractions, etc) before continuing the simulation (or post-processing the results). Use `'recover_data'` routine for the purpose and then follow up with `continue_drain` or `continue_imbibe` as appropriate (all routines are documented in Appendix B).

9.6 Where to find more information?

This focus of this manual was to work through simpler examples step-by-step. Many (and more exciting) examples are available in the original paper [12], as well as follow-ups: [16] focuses on a snap-off study in doubly-constricted pores, [13, 15] on applications to micro-fractures, and [2, 14] on coupling with sediment mechanics with implications to methane hydrate growth.

10 Acknowledgements

Steve L. Bryant has not only been always available for scientific discussions, but also tremendously supportive of the coding effort and making it public. Elena Rodriguez Pin has helped test the examples and is about to join the developers' ranks. Finally, since 2006 the effort has been supported, at various levels by J. T. Oden ICES Postdoctoral Fellowship, Baker Atlas, and US Departments of Energy (DE-FC26-06NT43067) and Agriculture (#2007-35102-18162). My heartfelt thanks to all.

References

[1] J. Bear. *Dynamics of Fluids in Porous Media*. Dover Publications, Inc, New York, 1988.

[2] J. Behseresht, Y. Peng, M. Prodanović, S. L. Bryant, A. Jain, and R. Juanes. Mechanisms by which methane gas and methane hydrate coexist in ocean sediments. otc paper 19332. In *Offshore Technology Conference*, May 4-8 2008.

[3] K. T. Chu and M. Prodanović. Level Set Method Library (LSMLIB). http://ktchu.serendipityresearch.org/software/lsmlib/index.html.

[4] F.A.L. Dullien. *Porous Media: Fluid Transport and Pore Structure*. Academic press, 2 edition, 1992.

[5] Geomview. http://www.geomview.org.

[6] Larry W. Lake. *Enhanced Oil Recovery*. Prentice Hall, 2 edition, 1989.

[7] GNU Triangulated Surface Library. http://gts.sourceforge.net/index.html/.

[8] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag, 2003.

[9] S. Osher and J.A. Sethian. Fronts propagating with curvature dependent speed: algorithms based on Hamilton-Jacobi formulation. *J. Comp. Phys.*, 79:12–49, 1988.

[10] multi-platform data analysis ParaView and visualization application. http://www.paraview.org/.

[11] D. Peng, B. Merriman, S. Osher, H. Zhao, and M. Kang. A PDE-based fast local level set method. *J. Comp. Phys.*, 155:410–438, 1999.

[12] M. Prodanović and S. L. Bryant. A level set method for determining critical curvatures for drainage and imbibition. *J. Colloid Interface Sci.*, 304:442–458, 2006.

[13] M. Prodanović and S. L. Bryant. Physics-driven interface modeling for drainage and imbibition in fractures, spe paper 110448. In *SPE Annual Technical Conference and Exibition*. Society of Petroleum Engineers, 11-14 November 2007.

[14] M. Prodanović and S. L. Bryant. Capillarity controlled displacements in sediments with movable grains: Implications for growth of methane hydrates. spe paper 116663. In *SPE Annual Technical Conference and Exibition*. Society of Petroleum Engineers, 21-24 September 2008.

[15] M. Prodanović, S. L. Bryant, and Z. T. Karpyn. Investigating matrix-fracture transfer via a level set method for drainage and imbibition. spe paper 116110. In *SPE Annual Technical Conference and Exibition*. Society of Petroleum Engineers, 21-24 September 2008.

[16] M. Prodanović and S.L. Bryant. *Focus on Water Resources Research*, chapter Resolving Meniscus Movement Within Rough Confining Surfaces Via the Level Set Method, pages 237–262. Nova Science Publishers, Hauppage, New York, 2008.

[17] Gimp The GNU Image Manipulation Program. http://www.gimp.org.

[18] J. A. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1999.

[19] The MayaVi Data Visualizer. http://mayavi.sourceforge.net/.

[20] Visit Visualization Tool. https://wci.llnl.gov/codes/visit/.

[21] Drishti Volume Exploration/Presentation Tool. http://sf.anu.edu.au/vizlab/drishti/index.shtml.

[22] 3DMA-Rock W. B. Lindquist. http://www.ams.sunysb.edu/∼lindquis/3dma/3dma_rock/3dma_rock.h

[23] H. Zhao, B. Merriman, S. Osher, and L. Wang. Capturing the behavior of bubbles and drops using the variational level set approach. *J. Comp. Phys.*, 143:495–518, 1998.

A Appendix: Detailed Installation Instructions

LSMPQS installation follows "./configure; make; make install" rule from free Linux distributions. If you have never encountered this process, I outline the basics for LSMPQS and its dependencies.

I will assume that you have an access to a Unix/Linux based workstation (note that all of this should work under Cygwin on Windows as well), and that you do not necessarily have the root password so whatever you install has to happen locally.

- ◇ **Prepare....**
  A lot of Linux software is installed in `/usr/local` directory which typically has `bin/`, `include/`, `lib/` and `src/` subdirectories that store executables, header files, libraries and source code to be used by different applications. Creating such a subdirectory in your home directory (e.g., `/home/masha/`) is one way to keep things organized. Thus,

```
> cd /home/masha
> mkdir local local/bin local/include local/lib local/src
```

◇ **Install LSMLIB library [3](version 1.0.1 or higher).**

Download the LSMLIB library from the webpage [3] to /home/masha/local/src/, change to that directory and extract the archive:

```
> cd /home/masha/local/src
> tar xvzf lsmlib-1.0.1.tar.gz
```

That creates lsmlib-1.0.1 directory, and you should remember its location (`LSMLIB_DIR=/home/masha/local/src/lsmlib-1.0.1`) as the main LSMLIB distribution directory. Now change to that directory in order to configure and compile LSMLIB:

```
> cd lsmlib-1.0.1
```

Note that none of the optional/parallel support in LSMLIB is needed for LSMPQS. Further, LSMLIB can be configured both for single (float) and double precision calculations and the same is true for LSMPQS. I have found floating point simulations sufficient - needless to say, they run faster and require less memory. Thus, suggested configuration is:

```
> ./configure --enable-float --without-samrai --without-matlab
--without-mpich
```

`./configure --help` command will give you detailed information on all options. Optionally, specify prefix by adding `--prefix=/home/masha/local` to the above line - that will make sure that headers and libraries get installed in `/home/maska/local/include` and `/home/masha/local/lib` respectivelyafter executing `make install` - see below.

If all goes well, type (as noted, the second line is optional):

```
> make
```

Optionally, execute `make install` as well.

◇ **(OPTIONAL) Install GTS library [7] (version 0.7.6 or higher)**
This library enables isosurfacing routines for interface visualization. LSMPQS currently supports viewing them in Geomview [5]. Geomview webpage should be consulted on how to install it, if you do not already have it on your system. (Note that rpms are available so the installation is possibly simpler than for LSMLIB and GTS).

Unfolding the downloaded archive (for instance into the same `/home/masha/local/src/`)

```
> tar xvzf gts-0.7.6.tar.gz
```

creates gts-0.7.6 subdirectory - remember the full path to this directory as GTS_DIR. As customary, this one also has detailed instructions in INSTALL. Follow the steps:

```
> ./configure
> make
```

As with LSMLIB, 'make install' is not necessary.

◇ **Install LSMPQS**
Download lsmpqs-0.5.tar.gz (say, in the same directory `/home/masha/local/src/`), untar and change to the newly created subdirectory, and excute the following (make sure to replace `LSMLIB_DIR` and `GTS_DIR` with the actual paths!)

```
> tar xvzf lsmpqs-0.5.tar.gz
> cd lsmpqs-0.5/
> ./configure --enable-float --with-lsmlib=LSMLIB_DIR --with-gts=GTS_DIR
> cd src/
> make all
```

As with LSMLIB, specifying installation path using `--prefix` while configuring and executing `make install` is optional.

Note that LSMPQS executables are either in the directory `/home/masha/local/bin` or `/home/masha/local/src/lsmpqs-0.5/bin` depending on whether or not you chose to run `make install`. Set your environmental variable `$PATH` accordingly so you can run the executables from anywhere. (Your system administrator should be able to help you with that).

Congratulations! You are ready to try running some examples from §9.

B Appendix: Documentation for LSMPQS routines implemented in C

The documentation below is (dynamically) extracted from the code. While the appearance could be improved, this ensures that the changes are updated only once thus hopefully avoiding confusion.

Files/routines are documented in alphabetical order (name reflects the original source file name) and separated by a starry line. Should you wish to look at the source code, find the appropriate file in one of the LSMPQS source code directories (see §4).

```
**********************************************************
compare_data_array_segdata.c
```

```
   This driver is "cleans up" LSMLIB_REAL data isolated clusters
   by using supplied segmented file: every time phases in files differ,
   LSMLIB_REAL data is changed to the opposite phase

   Usage: 'compare_data_array_segdata data_array_file seg_data_file'

   Output:
 "cleaned up" segmented file with _cln extension

   Notes:
   - LSMLIB data array phase data < 0 corresponds to 0 in segmented files.


************************************************************
compress_curv_model.c

 Main driver for slightly compressible curvature level set method model.

 For details see
 M. Prodanovic and S. L. Bryant. A level set method for determining critical
 curvatures for drainage and imbibition.
 Journal of Colloid and Interface Science, 304 (2006) 442--458.

 Usage:
 1. 'compress_curv_model'
    Everything runs with default options.
 2. 'compress_curv_model in_file'
    Input (geometry and other options) set according to the provided input
    file.
 3. 'compress_curv_model in_file data_in grid_in mask_in'
     data_in, grid_in and mask_in are input binary files for
     fluid level set function, grid structure and masking level set
     function. If the files are zipped, the names should have .gz
     extension.
 'in_file' sets all other relevant options

 Outputs
  data_init.gz   - data array storing initial level set function
                   (NW phase == data_init < 0)
  data_final.gz - data array storing final NW phase level set function
  grid.gz       - binary data array storing grid information
  mask.gz       - masking level set function (solid phase == mask > 0)
  out_file      - options, Grid data and any other text output

 Notes:
```

```
 - Main input parameters are 'a' and 'b' that determine initial guess for
 curvature (a/b), but the final curvature (that is hopefully large enough
 for a non-trivial and stable fluid configuration) is figured out during
 the simulation. Play with initial position of the zero level set
 ('ic_pos' or target volume fraction 'vol_frac_target') if the final
 result is not satisfactory.

************************************************************
compute_area3d.c

 Testing various implementations of delta function for area
 computation - a sphere of radius 1 will be created (and masked
 if desired) so you can measure accuracy, or you can provide your own input

 Usage:
 1. 'compute_area3d in_file data_file grid_file mask_file'
    Computing area for some LSMPQS simulation result.

 2. 'compute_area3d in_file data_file grid_file'
    In this case masking is ignored.
    (Note that mask_file is ignored if 'do_mask 0' in input file)

 3. 'compute_area3d in_file'
    A test geometry (sphere) is created for you.

 Notes:
 - In this routine you CANNOT set geometry using the input file.
   Only the following input file options are taken into account:
     dx - grid spacing
     eps_coefficient - numerical smearing width for Heaviside/delta functions
                       will be set to eps_coefficient dx
     do_mask (set to 1 if the fluid data/sphere is supposed to be  masked)
     save_data - if 1, level set function (phi), mask, grid, and some
                 of the computed delta function will be output as
                 binary files


************************************************************
compute_area_simulation.c

 Computes area of the fluid-fluid meniscus for entire drainage or
 imbibition simulation in the current directory
 using level set based formulas. Works for both 2D and 3D.

 Usage:
```

```
 1. 'compute_area_simulation'
    OR
    'compute_area_simulation d'  (for drainage simulation)
 2. 'compute_area_simulation i'  (for imbibition)

 The program writes the following binary files  (same type as 'vol_frac',
 'curvature' files)

 'meniscus_area' - contains interfacial (NW-W) area for each step
                   (trapped NW phase is excluded and reported separately)
 'area_nw'       - total NW phase area
 'area_nw_trap' (if exists) - trapped NW phase area
 'area_w'        - total W phase area.

 Notes:
 - In 2D we're measuring perimeter length, not area, but the file name
   is the same for both dimensions.

 - TO-DO: Trapped W (if exists) is ignored. Need to fix that.
   Also, changing solid space(aka moving mask, if applicable) is not
   taken into account

************************************************************
compute_curvature2d.c

 Computing (perimeter) averaged mean curvature for a circle of radius 1.0
 or computing curvatures for arbitrary input data (result of a LSMPQS
 simulation) using different curvature discretizations
 implemented in LSMLIB.

 Usage:
 1. 'compute_curvature3d in_file data_file grid_file mask_file'
     for  LSMPQS input
 2. compute_curvature3d in_file
    Circle input created within the program

 Notes:
 - In input file only the following options matter:
   dx - grid spacing (superceded by grid_file in Usage 1)
   eps_coefficient - numerical smearing width for Heaviside/delta functions
       will be is set to eps_coefficient dx
   do_mask - set this to 1 if the sphere is supposed to be partially masked
   save_data - 0 or 1 if you want curvature arrays to be output
```

```
************************************************************
compute_curvature2d_simulation.c

 Computes perimeter averaged curvature of the fluid meniscus
 for entire drainage or imbibition simulation in the current directory.
 Unlike compute_curvature2d.c, only one, standard 2nd order curvature
 discretization is used (same as the one used in simulation).

 Usage:
 1. 'compute_curvature2d_simulation'
     OR
    'compute_curvature2d_simulation d'  (for drainage)
 2. 'compute_curvature2d_simulation i'  (for imbibition)


************************************************************
compute_curvature3d.c

 Computing (area) averaged mean curvature computations for a
 circle of radius 1.0 or computing curvatures for arbitrary input data
(result of LSMPQS simulation) using different curvature discretizations
 implemented in LSMLIB.

 Usage:
 1. compute_curvature3d in_file data_file grid_file do_mask_file
 2. compute_curvature3d in_file

 Notes:
 - In input file only the following options matter:
   dx - grid spacing (superceded by grid_file in Usage 1)
   eps_coefficient - numerical smearing width for Heaviside/delta
       functions will be set to eps_coefficient dx
   do_mask (set to 1 if the sphere is supposed to be partially masked)
   save_data - 0 or 1 if you want curvature arrays to be output



************************************************************
compute_perimeter2d.c

 Testing various implementations of delta function for perimeter
 computation - a sphere of radius 1 will be created (and masked
 if desired) so you can measure accuracy, or you can provide your own input

 Usage
```

```
 1. 'compute_perimeter2d in_file data_file grid_file mask_file'
 2. 'compute_perimeter2d in_file'
     In input file set
      dx - grid spacing
      eps_coefficient - numerical smearing width for Heaviside/delta functions
      will be set to eps_coefficient dx
      do_mask - set to 1 if you want to use partial masking
      save_data - if 1, level set function (phi), mask, grid, and some
                  of the computed delta function will be output as
                  binary files

************************************************************
************************************************************
compute_volume.c

 Computes volume (in 3D; area in 2D) of the region where
 the level set function 'data' (stored in data_file) is negative.

 Usage:  'compute_volume data_file grid_file'

 Notes:
 - Level set methodology is used for volume and surface area computations.
  Integration is done over entire volume via numerical/smoothed Heaviside
  function (standard sine version), so some resolution effects on accuracy
  are to be expected. Volume computed by counting voxels is provided for
  comparison.

************************************************************
const_curv_model.c

 Main driver for slightly constant curvature level set method model.

 For details see
 M. Prodanovic and S. L. Bryant. A level set method for determining critical
 curvatures for drainage and imbibition.
 Journal of Colloid and Interface Science, 304 (2006) 442--458.

 Usage:
 1. const_curv_model
    Everything runs with default options.
 2. const_curv_model in_file
    Input (geometry and other options) set according to the provided input
    file.
 3. 'const_curv_model in_file data_in grid_in mask_in'
     data_in, grid_in and mask_in are input binary files for
```

```
    fluid level set function, grid structure and masking level set
    function. If the files are zipped, the names should have .gz
    extension.
    in_file sets all other relevant options

 Outputs
  data_init.gz   - data array storing initial level set function
                   (NW phase == data_init < 0)
  data_final.gz - data array storing final NW phase level set function
  grid.gz       - binary data array storing grid information
  mask.gz       - masking level set function (solid phase == mask > 0)
  out_file      - options, Grid data and any other text output

 Notes:
 - Main input parameters are 'a' and 'b' that determine desired
 curvature (a/b). Note that a non-trivial solution might not exist
 for the given geometry.

************************************************************
continue_drain.c

 Continue previously run drainage simulation in current directory
 (see drain.c for more details).

 Usage: 'continue_drain in_file'

 Notes:
 - The program will look for the typical output files  (data_step .gz,
   mask.gz, a.gz,...) to figure out the last step of the previous simulation
   and read in information necessary to continue.

 - If the previous drainage was abruptly stopped (causing some files not to
   be output or be damaged), make sure to run 'recover_data' beforhand.


************************************************************
continue_imbibe.c

 Continue  previously run imbibition simulation in current directory
 (see imbibe.c for more details).

 Usage: 'continue_imbibe in_file'

 Notes:
  - The program will look for the typical output files  (data_step .gz,
```

```
mask.gz, a.gz,...) to figure out the last step of the previous simulation
and read in information necessary to continue.

  - If the previous imbibition was abruptly stopped (causing some files not
to be output or be damaged), make sure to run 'recover_data' beforehand.


************************************************************
convert_to_float_data.c

 Assuming LSMLIB data array is in double precision format, this program
 converts data to float (single precision). Same is true for grid
 binary file.

 Usage:
 1. 'convert_to_float_data data_file'
 2. 'convert_to_float_data data_file grid_file'

 Notes:
 -  THE ORIGINAL FILES WILL BE OVERWRITTEN!


************************************************************
count_blobs.c

   Count disconnected components (blobs) of the negative phase of a level set
   function.

   Usage:
   1. 'count_blobs data_filename grid_filename'
   2. 'count_blobs data_filename grid_filename mask_filename'
  - count components for the specific data step binary file.
   3.  'count_blobs d(rainage)'
   4.  'count_blobs i(mbibition)'
    - analyzes all steps in the current directory and finds critical steps
    - critical steps are those after which the number of (monitored) phase
      components has changed, or after which the volume fraction of
      the phase changed more than the critical fraction allowed
      (CRITICAL_VOLE_FRAC_DIFF - change below/recompile as necessary)

   Notes:
   - typical names for binary files for grid, mask and volume_fraction
 are assumed
```

```
************************************************************
create_bridge_rupture.c

 Creates multiple input geometries for testing at which gap spacing D will
 a liquid bridge between two spheres (of radius R=1.0) rupture.
 All files will be stored in directory ./bridge_rupture/


************************************************************
create_drainage_input.c

 Creates pore space geometry in between a number of spheres that are
 stored in a specific binary file (output by ouputSphereBinary.m).
 Sets input files needed for drainage simulation in such pore space.
 Change path_binary to reflect base filename for the binary files,
 and path_run to reflect where are the input files for LSM simulation
 supposed to be stored.

 Notes:
 - This file is somewhat obsolete - LSMPQS now has a specific
   geometry input option that allows for reading the sphere binary file.
   It is saved as an example of how to create multiple simulation
   directories, however - either drainage in multiple pores or throats.


************************************************************
create_geomview_files.c

   Creates ./geomview in directory that contains Geomview surface visualization
   for desired range of LSMPQS drainage or imbibition steps. All you need to
   specify is a range of steps.

   Usage:
   1. create_geomview_files start_step end_step
   2. create_geomview_files start_step end_step add_mask
  set 'add_mask' to 1 if you wish to visualize grain surface as well
   3. create_geomview_files start_step end_step add_mask geomview_template_file
  Note that provided template file has to fit all data steps.
  See create_geomview_files.c file for more information.

   Outputs: (in ./geomview)

   surf_mask.list - pore-solid surface
   surf_dID.list  - files (ID is step identifier)
   view_dID.list  - a script that rotates the above surface into a specific
```

```
                view (open it with Geomview); you can find more 'view'
templates in src/visualization/geomview_templates/

   Notes:
- assumes LSMPQS files are in CURRENT directory and data arrays are
  gzipped (default behaviour for drainage and imbibition)
- any previous files in ./geomview/ will be overwritten
- assumes 'rpl' installed, needs a different 'replace' routine if not
- assumes isosurfacing routine 'isocolor_peel' in this directory are
  compiled (depends on GTS) and is on $PATH (available from command
  line)
- you need Geomview installed to view files produced


************************************************************
create_imbibition_input.c

 Creates imbibition input for multiple pores from the previously run
 drainage simulations.

 Notes:
 - This file is somewhat obsolete and probably does not run properly.
 It is saved as an example of how to create multiple simulation directories.


************************************************************
create_MasonMorrow87.c

 Contains main routine for creating configurations of plates and rods
 from the paper of Mason and Morrow '87.

 For more details, see Section 3.4 in
 M. Prodanovic and S. L. Bryant. A level set method for determining critical
 curvatures for drainage and imbibition.
 Journal of Colloid and Interface Science, 304 (2006) 442--458.

 Usage: create_MasonMorrow87 MODE
        where MODE is one of the following
        1 - symmetrical plate - gap -rod - rod - gap -plate config
        2 - asymmetrical plate - rod - rod - gap -plate
        3 - added obstacle to Symm1 to achieve different fld-fld config
        4 - added obstacle to Asymm1 to achieve different fld-fld config


 Directory where configurations will be stored is created automatically,
```

```
 see (&modify as desired) 'path_run' below.

 For each gap spacing D, subdirectory 'configD' will be created
 in the 'path_run' directory with input files for a level set
 simulation


************************************************************
create_raster_files.c

 Creates directory ./raster that contains rasterfiles for all steps
 of LSMPQS simulation in desired range.

 Usage: 'create_raster_files magnification start_step end_step'

 Outputs: (in ./raster/)
  Multiple files whose naming follows the rule:
    data_step00STEP_ID.ras.gz (for STEP_ID < 10)
    data_step0STEP_ID.ras.gz  (for STEP_ID < 100)
    data_stepSTEP_ID.ras.gz

 Notes:
 - assumes 'raster' command from src/visualization is compiled


************************************************************
create_triply_constricted_pore3d.c

A triply constricted pore will be created. Throat (bounded by spheres,
id is specified) on one end, same throat will be added with enlarged
sphere radius on two positions on the other two ends.

Usage: create_triply_constricted_pore3d binary_basename THR_ID ratio1 ratio2

Similar to create_doubly_constricted_pore3d.c

Modify binary_baseneme below to reflect throat binaries.


************************************************************
data_array2seg.c

 Converts LSMLIB_REAL data array to segmented file format. Data values
 less than 0 are set to 0, and data values are set to 1 in the segmented
 file.
```

```
 Usage: data_array2seg LSMLIB_REAL_data_fname seg_data_fname

************************************************************
diff_data_arrays.c

 Finds maximal absolute pointwise difference between two data files storing
 level set functions (LSMLIB data arrays).

 Usage: diff_data_files filename1 filename2 grid_filename

************************************************************
do_for_range_of_steps.c

Runs given executables for data arrays output by drainage or imbibition
simulation.

Usage: do_for_range_of_steps executable start_step end_step out_file
  executable - some compiled LSMPQS routine that takes the following form
           'executable data_stepSTEP_ID.gz grid.gz mask.gz'
  start_step, end_step - range of steps to process
  outfile - all output will be redirected to this file


************************************************************
drain.c

 Progresive Quasistatic (PQS) Algorithm for Drainage

 For details see
 M. Prodanovic and S. L. Bryant. A level set method for determining critical
 curvatures for drainage and imbibition.
 Journal of Colloid and Interface Science, 304 (2006) 442--458.

 Usage:
  1. 'drain'  - everything set to default
  2. 'drain in_file' - in_file sets simulation options
  3. 'drain in_file data_in grid_in mask_in' -
          data_in, grid_in and mask_in are input binary files for
          fluid level set function, grid structure and masking level set
          function. If the files are zipped, the names should have .gz
          extension.
 Output:
  a.gz           - 1D data array storing 'a' (pressure like term) for
                   each step
```

```
  curvature.gz   - 1D data array storing curvature for each step
  data_init.gz   - data array storing initial level set function
                   (NW phase == data_init < 0)
  data_stepID.gz - data array storing the NW phase level set function
                   for each simulation step ID
  For imbibition, ID starts from 1.
  grid.gz        - binary data array storing grid information
  mask.gz        - masking level set function (solid phase == mask > 0)
  out_file       - options, Grid data and any other text output
  vol_frac.gz    - 1D data array storing volume fraction occupied by NW phase
                   for each step


************************************************************
imbibe.c

Progresive Quasistatic (PQS) Algorithm for Imbibition
For details see
M. Prodanovic and S. L. Bryant. A level set method for determining critical
curvatures for drainage and imbibition.
Journal of Colloid and Interface Science, 304 (2006) 442--458.

Usage
1. 'imbibe in_file data_in grid_in mask_in'
   where data_in, grid_in, mask_in corresponds to data from a previous
   drain simulation (drainage end-point). Make sure that pressure 'a'
   in input file is set to the pressure corresponding to the end-point.

2. 'imbibe in_file STEP_ID'
   Assumes you start the simulation from the drainage directory, and
   wish to imbibe from step STEP_ID. Pressure 'a' in this case is set
   according to the array 'a' stored in the directory. A subdirectory
   'imbibeSTEP_ID' is created.

Output:
  a.gz           - 1D data array storing 'a' (pressure like term) for
                   each step
  curvature.gz   - 1D data array storing curvature for each step
  data_init.gz   - data array storing initial level set function
                   (NW phase == data_init < 0)
  data_stepID.gz - data array storing the NW phase level set function
                   for each simulation step ID
  For imbibition, ID starts from 1.
  grid.gz        - binary data array storing grid information
  mask.gz        - masking level set function (solid phase == mask > 0)
```

```
  out_file      - options, Grid data and any other text output
  vol_frac.gz   - 1D data array storing volume fraction occupied by NW phase
                  for each step


************************************************************
identify_snapoff.c

 Designed for analysis of imbibition simulations in doubly constricted geometries
 in order to find out if snap-off has occurred.

 Usage: 'identify_snapoff directory_path'

 Notes:
 - Assumes the current directory contains standard LSMPQS simulation output files.
   'vol_frac.gz' is checked for the number of steps and only the step before last
   is checked for disconnected blobs of NW fluid, and if found, a text file
   'snapoff' is written to a directory (this made sends once upon a time, I swear).
 - This routine is somewhat obsolete in the following sense. If imbibition simulation
  was done with option 'do_trap_nw==1', then the snapped off blobs will be recorded
  as trapped, and the data array 'data_step _trap_nw.gz' will appear and make
  it obvious snapoff has occured.

************************************************************
initialize_geometry.c

 Sets input options according to input file 'in_file' and creates/outputs
 files for
 - geometry (data array stored in 'mask.gz')
 - initial interface (data array stored in 'data_init.gz')
 - grid structure (grid binary array stored in 'grid.gz')

 Usage: 'initialize_geometry in_file'

 Notes:
 - Use 'isosurface', 'isocolor' etc. to visualize data arrays in 3D (based
   on Geomview) or 'plotZeroLevelSet' and the like routines to visualize
   2D geometries in Matlab.


************************************************************
isocolor.c

 This program plots triangulated zero level set from a number of
 level set functions which are stored in a binary file (LSMLIB)
```

```
 that contains:
   nx - binary interger
   ny - binary interger
   nz - binary interger
   binary array of nx ny nz elements (double or float, depending on
   how LSMLIB_REAL is set).

 Usage: 'isocolor file1 color1 file2 color2...'
        where color is ONE letter that stands for
       n(one), r(ed), g(reen), y(ellow), (b)lue,
       m(agenta),c(yan),(blac)k, o(range), (b)l(uish)
  All gridpoints from the files are used in producing surfaces.

 Notes:
 - Surface is output to Geomview file GEOMVIEW_FILE (set in
   isosurface_main.h) and deleted after viewing if
   DELETE_GEOMVIEW_FILE (also in isosurface_main.h) is set to 1.


**********************************************************
isocolor_peel.c

 Similar as isocolor.c, but a specified number of cells is peeled off
 (i.e. removed) from each of the volume sides. Note that LSMPQS
 standardly adds 3 cells on each side for numerical differentiation.

 Usage: 'isocolor_peel num_cells file1 color1 file2 color2...'
        color - ONE letter that stands for
               n(one), r(ed), g(reen), y(ellow), (b)lue,
               m(agenta),c(yan),(blac)k, o(range), (b)l(uish)
        num_cells - number of numerical cells to peel on each side

 Notes:
   - Surface is output to Geomview file named GEOMVIEW_FILE
     (set in isosurface_main.h).


**********************************************************
isosurface.c

 This program plots triangulated zero level set from a number of
 level set functions which are stored in a binary file (LSMLIB)
 that contains:
   nx - binary interger
   ny - binary interger
```

```
   nz - binary interger
   binary array of nx ny nz elements (double or float, depending on
   how LSMLIB_REAL is set).

 Usage: 'isosurface file1 file2 ...'
   Surfaces from all files will, however, be of the same Geomview
   default color.
   All gridpoints from the files are used in producing surfaces.

 Notes:
 - Surface is output to Geomview file GEOMVIEW_FILE (set in
   isosurface_main.h) and deleted after viewing if
   DELETE_GEOMVIEW_FILE (also in isosurface_main.h) is set to 1.


************************************************************
isosurface_peel.c

 This program plots triangulated zero level set from a number of
 level set functions which are stored in a binary file (LSMLIB)
 that contains:
   nx - binary interger
   ny - binary interger
   nz - binary interger
   binary array of nx ny nz elements (double or float, depending on
   how LSMLIB_REAL is set).

 Usage: 'isosurface_peel num_ghostcells file1 file2 ...'
   Surfaces from all files will, however, be of the same Geomview
   default color.
   'num_ghostcells' ghostcells will be are peeled off on each side
   of the volume.

 Notes:
 - Surface is output to Geomview file GEOMVIEW_FILE (set in
   isosurface_main.h) and deleted after viewing if
   DELETE_GEOMVIEW_FILE (also in isosurface_main.h) is set to 1.


************************************************************
merge_data_arrays.c

 Merges two data arrays (serial LSMLIB package) into an array that contains
 This was developed so that information can be loaded into Paraview.
```

```
 Usage: 'merge_data_arrays datafile1 datafile2 merged_array_fname'


************************************************************
print_geometry_options.c

 Prints all available geometry options.

 Usage: 'print_geometry_options'

 Notes:
 - mask.[ch] contain functions that create geometries. Look into those files
   for instructions on how to modify or add a geometry.


************************************************************
print_grid.c

 Prints grid information (stored in the binary file 'grid_filename') to the
 screen. File can be zipped, in which case filename should contain ".gz"
 extension.

 Usage: 'print_grid grid_filename'


************************************************************
print_input_options.c

   Prints default Options structure (input options).

   Usage: 'print_input_options'

   Notes:
   - If the user doesn't specify an option, it likely has a default value and
 this is the way to check which one.

   - For code expansion it is convenient to have one centralized structure
 of all possible input options. However, just because you can set an input
 option to a value, does not mean that the options is meaningful or used
 within a program.


************************************************************
raster.c
```

```
Reads in LSMLIB data arrays storing (2D) fluid and mask level set functions
and produces a raster file image viewable by most image viewing software.

Usage: 'raster magnification data_file_1 ... data_file_m mask_data_file'
       magnification - integer (1 if no magnification desired)

Outputs: 'tmp.ras.gz' file in the current directory.

Notes:
  - provide at least two data array files
  - the last provided data file is assumed to be masking function
  - fluid_data negative phases (data < 0) are plot in different colors
    (up to 5 different colors, then they start repeating)
  - in case of two fluid arrays having negative value at the same pixel,
    'first come first served' applies
  - mask positive phase (mask > 0) is always plot in black and assumed
    solid phase
  - the result is pixelized (binarized), smooth information from level set
    function is lost


************************************************************
   recover_data.c

   This driver reads in existing level set function data from the
   current directory, recomputes volume fractions and rewrites curvature
   information (rewrites files). This can be used to rewrite 'a.gz',
   'vol_frac.gz and 'curvature.gz' files from an abruptly stopped drainage
   or imbibition simulation (drain and imbibe respectively).

   Run from the local directory where the simulation was performed.
   Files expected to be found in the current directory:
 in_file
 out_file
 grid.gz
 mask.gz
 data_stepSTEP_ID.gz (one for each step of the simulation)

   If read_outfile is 1, program reads 'out_file' for curvature
   and pressure data information; volume fractions are recomputed from data
   files. Any missing data is replaced by 0.

   Usage: 'recover_data [mode] [read_outfile]'
      mode - d (drainage), i (imbibition), m (drainage with movable grains)
           - if not provided, default value is 'd'
```

```
 read_outfile - 1 if file './out_file' is to be read for values of
  pressure (a), curvature step (dc) and intfc. tension (b)
  coefficients
  - if not provided, default value is 0 and information in
   './in_file' is used instead ('./in_file' presumably exists
            in the current directory
  Notes:
 - if the simulation has been stopped/restarted a number of times
   or has crashed, then reading output file is not a good idea
 - in drainage simulation, './in_file' might not have the correct/final
   value of the pressure term 'a'. You need to set it manually!

************************************************************
resize_data_array.c

 A routine for resizing LSMLIB data arrays.

 Usage: resize_data_array data_filename xs xe ys ye zs ze
     [xyz]s,[xyz]e are starting and ending voxel in [xyz] direction.

 Notes:
 - Voxel numbering starts from 0 (just like C arrays).
  (i.e., if there are 5 slices they're numbered 0..4).

 Example:
   Assume mask.gz is 106 x 106 x 106 array.");
   'resize_data_array mask.gz 3 102 3 102 3 102 mask_rsz'
   will store the 100 x 100 x 100 mid section of the volume
   into data array 'mask_rsz.gz' (zipped by default)



************************************************************
resize_grid.c

 A routine for resizing grid structure.

 Usage: resize_grid grid_filename xs xe ys ye zs ze [grid_resize_filename]

 Output: resized grid binary file, if name is not provided
         it will be named grid_rsz.gz

 Notes:
 - [xyz]s,[xyz]e are starting and ending voxel in [xyz] direction.
  (i.e., if there are 10 slices they're numbered 0..9.)
```

```
************************************************************
resize_simulation.c

 Applies 'resize_data_array' to all desired data steps of the simulation

 Usage: resize_simulation start_step end_step directory xs xe ys ye zs ze
      start_step, end_step - range of data steps to process
      directory - directory where the resized data should be stored
      xs,xe,ys,ye,zs,ze - input for 'resize_data_array'

 Notes:
 - data_init.gz, mask.gz, grid.gz (typically output during simulation)
  are assumed to exist in the current directory and are resize as well.


************************************************************
reverse_phases.c

 Program flips positive and negative phases of the level set function
 stored in the LSMLIB data array.

 Usage: reverse_phases data_fname new_data_fname (mask_fname)
    data_fname -     LSMLIB data array (phi)
    new_data_fname - name of the reversed LSMLIB data array (new_phi=-phi)
    mask(optional) - masking level set function
              if provided, mask will be imposed as maximum
 new_phi = max(-phi,mask)


************************************************************
reinitialize.c

   Reinitialize input data. Reinitialization replaces level set function
   (whose zero level set describes an interface of interest) with
   a signed distance function (and the zero level set doesn't move in the
   process).

   Usage:
1. reinitialize
   All options are set to default and the main level set function
   is reinitialized (default but not useful behavior)
2. reinitialize in_file
   This will reinitialize masking function (that determines geometry)
```

```
   Use input file options to set the masking function.
3. reinitialize in_file data_file grid_file mask_file
   Reinitializes level set function provided in data_file.

   Output: data_init.gz  (before reinitialization)
       data_final.gz (after reinitialization)
   out_file - options, Grid data and any other text output
   Notes:
   - Input option 'tmax_r' limits the running time/how far away from the
 zero level set do we reach. The speed is 1.0, so say if tmax_r==2 dx,
 points within 2 grid cells of the interface will get properly updated.

   - If input option 'subcell_fix' is set to 1 (default behavior), then the
 reinitialization is done based on "subcell fix" approach in
 A Remark on Computing Distance Functions. G. Russo and P. Smereka.
 J. Comp. Phys., 163, 51-67, 2000.

   - If input option 'subcell_fix' is set to 0, then the standard 2nd order
 discretization of the reinitalization function is done (See Osher/Fedkiw
 book) and as a result the zero level set is slightly "smoothed" and moved.


************************************************************
rotate_data_array.c

Rotates LSMLIB_REAL data array or segmented file (volume file).

Usage:
1. 'rotate_data_array data_in data_out' for REAl data array
2. 'rotate_data_array s data_in data_out' for segmented data

Notes:
 - There is a number of hard-coded types of rotations (see below).
 You will need to change them in the file manually and recompile the code.
 It should be straightforward to define more rotations.

- Rotation types:
 ROT0 -- Rotate x-y-z to z-x-y
 ROT1 -- Rotate x-y-z to z'-y-x where ' indicates flipping/reversal
                                 within the specified coordinate
 ROT2 -- Rotate x-y-z to z-y-x
 ROT3 -- Rotate x-y-z to y-x-z (flipping x and y coord directions)
 ROT4 -- Rotate x-y-z to x'-y-z (flipping x direction coordinates)
```

```
************************************************************
seal_data_array.c

 Program acts on a level set function (which presumably defines porous
 medium geometry such that solid is where the function is positive, and
 pore space is where the function is negative. It "seals" volume sides
 with positive values and thus acts as a planar solid  wall.

 Usage: 'seal_data_array data_file grid_file'
   Arguments:
   data file - binary file that contains LMSLIB data array
   grid file - grid binary file

 Output: The new, sealed level set function data array with '_seal' ending
         (in the current directory).


************************************************************
strip_data_array_header.c

 Removes information on the number of cells in each direction from the LSMLIB
 data array (thus leaving only packed level set function information).

 Usage: 'strip_data_array_header LSMLIB_REAL_data_fname strip_data_fname'

************************************************************
tar_drain_data_files.c

   Create tar archives of very specific drainage simulation directories.
   See the file for more information.
   This is old & rather specific, kept only as a template.
   Stand-alone tool.


************************************************************
tar_imbibe_data_files.c

   Create tar archives of very specific imbibition simulation directories.
   See the file for more information.
   This is old & rather specific, kept only as a template.
   Stand-alone tool.

************************************************************
visualize_blobs.c
```

```
 Usage: 'visualize_blobs data grid mask phase output_blob_files min_size
                          connectivity'
 data, mask - LSMLIB_REAL arrays storing level set functions
 grid - binary Grid file
 phase - 1 or -1 depending on which phase of data should be investigated
 output_blob_files - integer indicating if separate blob files should be
             output, if 1 then the subdirectory blobs/ will be created
each blob will be stored in a separate LSMLIB data
array file and will have negative values for blob voxels
and positive otherwise (can be visualized using
'isosurface')
 min_size - (optional) input integer that indicates the minimum size (in
     voxels) of blobs to be considered.
 connectivity - 6 or 26 (digitial connectivity of voxels)


************************************************************
visualize_data_range.c

 Debugging tool. Analyzes LSMLIB data array file and
 a) if min_value, max_value not provided
    Finds max. data value and outputs an array that has value 0.5 where
    data >= 0.5 max, and -0.5 otherwise.
 b) if 'min_value, max_value' provided
    Points with values in that range will be set to negative and positive
otherwise.

 In both cases isososurfacing routines can then be used for visualization.

 Usage: 'visualize_data_range data_file grid_file [min_value max_value]'


************************************************************
visualize_narrow_band.c

 (Debugging tool). Program uses LSMLIB localization routines to create narrow
 band around zero level set of the function stored in data_file and outputs
 separate data arrays that can be visualized.

 Usage: 'visualize_narrow_band data_file grid_file'

 Output:
 narrow_band.gz          - narrow band points (all levels!)
 index_space0.gz         - the inner narrow band tube
 index_space_outer_plus.gz   - outer narrow band points where phi > 0
```

```
 index_space_outer_minus.gz  - outer narrow_band points where phi < 0

 All of the above data arrays have points of interest set to value -0.5,
 all others to 0.5 so standard isosurfacing routines can be used to
 visualize them.


************************************************************
write_load_movie.c

 Writes a simple script for loading a list of Geomview files
 (produced by create_geomview_files.c) into Geomview's module StageManager
 for creating a movie.

 Usage: 'write_load_movie step_indices critical_step_indices'
       step_indices, critical_step_indices are ASCII files with list
       of integer IDs corresponding to LSMPQS simulation steps.


************************************************************
write_load_movie_simple.c

 Writes a simple script for loading a list of Geomview files
 (produced by create_geomview_files.c) into Geomview's module StageManager
 for creating a movie.

 Usage: 'write_load_movie step_start step_end'
        step_start, step_end - a continuous range of steps to be loaded
```