1.0 General Remark

In the directory `DUNE/dune-funcep` type

```
$ git stash
$ git pull
$ git stash pop
```

to obtain the latest software version. Then change to the directory `DUNE/release-build/dune-funcep/src/exc01` and type `make`. The code for the excercise is provided in the directory `DUNE/dune-funcep/src/exc01`.
The deadline for submitting the solutions is Thursday, Nov 10, 2016, 12:00pm.

1.1 Horizontal Groundwater Flow

Verify that

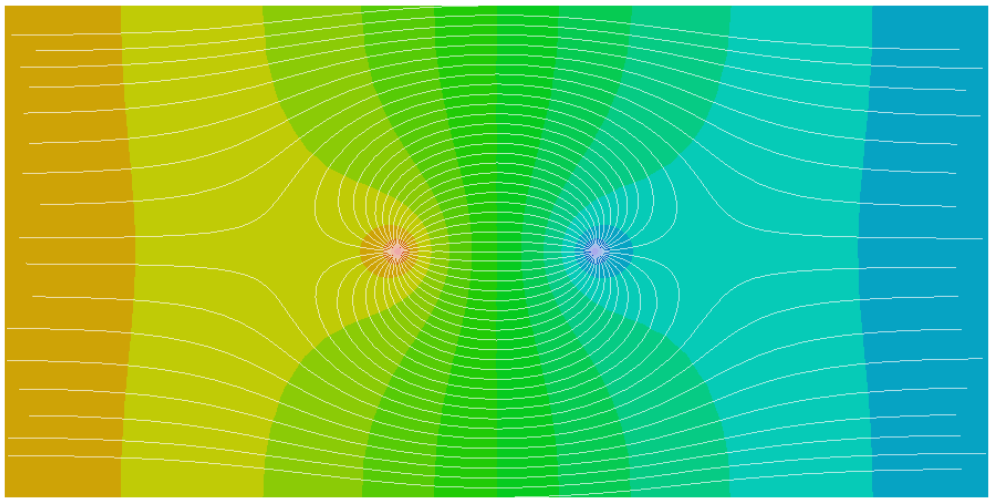
$$u(x) = \begin{pmatrix} a & 0 \end{pmatrix} \cdot x + u_0 - \frac{1}{2\pi} \ln \|x - x_{source}\| + \frac{1}{2\pi} \ln \|x - x_{sink}\|,$$

solves

$$-\Delta u = \delta_{x_{source}} - \delta_{x_{sink}} \quad \text{in } \mathbb{R}^2, \qquad \lim_{x \to \infty} -\nabla u = \begin{pmatrix} a & 0 \end{pmatrix}^T,$$

where δ_z is the delta-distribution located at z given by $\int_\Omega \delta_z(x) v(x)\, dx = v(z)$, $\begin{pmatrix} -a & 0 \end{pmatrix}^T$ is a given regional (hydraulic) gradient and u_0 is a reference pressure.

The application behind this problem is to have a point source at x_{source} where water is pumped in and a point sink at x_{sink} where water is pumped out in a horizontal aquifer with (undisturbed) velocity a. (*Notice that Problem 1.2.2 below is essentially the same, just with physical units added*). The purpose of this excercise is to explore how point sources and sinks are realized in the numerical simulation and how well the analytical solution can be approximated. An illustration of the result to be expected (showing pressure in color and streamlines in white) is given in the following figure:



Problems:

1. Set up a simulation solving this problem in a domain $\Omega = (0, LX) \times (0, LY)$ with $x_{source} = (LX/2 - D, LY/2)$ and $x_{sink} = (LX/2 + D, LY/2)$. Approximate the delta distribution by a function that has value H^2 in a square of side length H centered at x_{source} and x_{sink} respectively. Choose u_0 such that $u(LX/2, y) = 0$ and use the exact solution given above as Dirichlet boundary value. A corresponding geometry file for `gmsh` is provided in `horizontal_point_sources.geo` where the parameters given above can be prescribed.

2. Visualize the result using `paraview` and explore the `Warp By Scalar` and `Stream Tracer` filters. Note that the velocity field $v_h = -\nabla u_h$ is provided in the paraview output file.

3. Check the accuracy of the numerical solution by employing different mesh sizes *and* side lengths H for approximation of the delta distribution. Choose points close to the source/sink and away from it.

1.2 Groundwater Flow in Uniform Aquifer

Consider a uniform, isotropic, horizontal, and confined freshwater aquifer that is horizontally unbounded (fig. 1). Let the natural flow in this aquifer be stationary and driven by two rivers that run parallel and that both are connected to the aquifer.

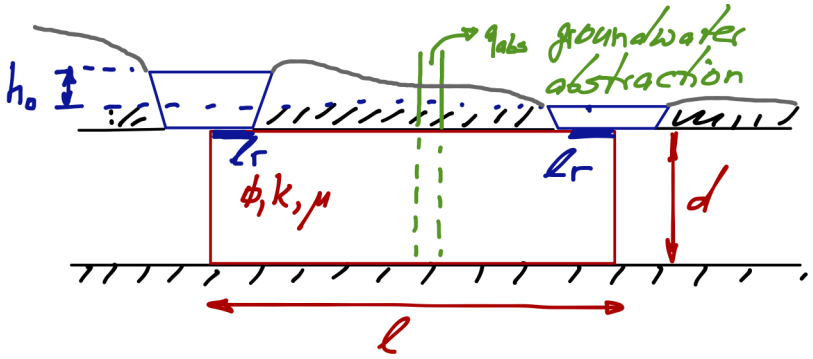


Figure 1: Sketch of the cross-section through the aquifer, orthogonal to the rivers that drive the flow. This cross-section is the same at any position along the direction of the rivers.

The architecture, material properties, and forcing of the system are described by the parameters given in Table 1.

Problems

1. Set up and run a two-dimensional numerical simulation of the vertical cross-section. You may imagine a two dimensional simulation as a three-dimensional one with unit extent in the omitted direction and no dependence of the solution on the omitted dimension. Neglect all pumping and assume the vertical boundaries to be impermeable, e.g., because the rivers are repeated with the same distances and height differences (symmetry). How does the flow field depend on h_0?

Table 1: Parameters for uniform groundwater flow.

property	symbol	value	unit
distance between rivers	ℓ	10^4	m
aquifer depth	d	100	m
connected length	ℓ_r	20	m
porosity	ϕ	0.1	
permeability	k	10^{-11}	m^2
dynamic viscosity	μ	$1.3 \cdot 10^{-3}$	Pa s
height difference rivers	h_0	0.3	m
abstraction flow	q_{abs}	s^{-1}	

2. Consider a vertical, fully penetrating pumping well in the middle between the two rivers with a fixed extraction rate q_{abs}. Realize two different implementations of the well:

   (a) Model the well as a volumetric sink term with a correspnding rate such that extraction rate q_{abs} is realized.

   (b) Model the well as a flux boundary condition comnined with a region of high permeability.

1.3 Reentrant Corner Problem

Consider the Poisson equation in polar coordinates r, ϕ,

$$\Delta u(r,\phi) = \frac{\partial^2 u}{\partial r^2} + \frac{1}{r}\frac{\partial u}{\partial r} + \frac{1}{r^2}\frac{\partial^2 u}{\partial \phi} = f$$

and verify that $u(r,\phi) = r^k \sin(k\phi)$ solves the Poisson equation in polar coordinates.

Now consider the Poisson equation in the semi-infinite domain $\Omega_\Theta = \{(r,\phi) : r > 0, 0 < \phi < \Theta\}$ with the following Dirichlet boundary data: $u(r,0) = u(r,\Theta) = 0$ for $r > 0$. Here $0 < \Theta < 2\pi$ is a parameter. Verify that $u(r,\phi) = r^{\pi/\Theta}\sin(\phi\pi/\Theta)$ solves this problem. Observe that this solution has an infinite derivative at $(0,0)$ if $\Theta > \pi$.

Problems:

1. Implement a C++ class along the one given in `app01/problem.hh` that solves the problem in the L-domain $\Omega_\Theta \cap (-1,1)^2$. Use the exact solution as Dirichlet boundary data. In order to use your new `problem.hh` with the main program `exc01.cc` you have to change the `include` statement in line 10 accordingly.

2. Construct a mesh with `gmsh` for the ldomain. Start form the geometry file in `app01/ldomain.geo`.

3. Visualize the solution using the `Warp By Scalar` filter in `paraview`.

4. For several values of Θ in the range $(\pi, \frac{7}{4}\pi)$ check the convergence of the numerical scheme by listing the pointwise error $|u(x) - u_h(x)|$ for different mesh sizes obtained through uniform mesh refinement (`refinement` parameter in the `.ini` file). Use a position x_1 close to the origin and a position x_2 not close to the origin and not close to the boundary. (Yes, these are lot of runs since domain Ω_Θ, mesh size h and probe positon x_i need to be varied).

5. Assuming that the error has the form $|u(x) - u_h(x)| = Ch^\alpha$ determine the exponent α from your tests. Note that uniform refinement reduces the mesh size by a factor 1/2 for each refinement.

Hint: The implementation requires the conversion of (x,y) coordinates to polar coordinates (r,ϕ) which can be done as follows:

```
typename Traits::RangeFieldType
g (const typename Traits::ElementType& e,
const typename Traits::DomainType& xlocal) const {
  typename Traits::DomainType x = e.geometry().global(xlocal);
  double phi = std::atan2(x[1], x[0]);
  if(phi < 0.0) phi += 2*M_PI;
  double r = x.two_norm();
  return ....
}
```