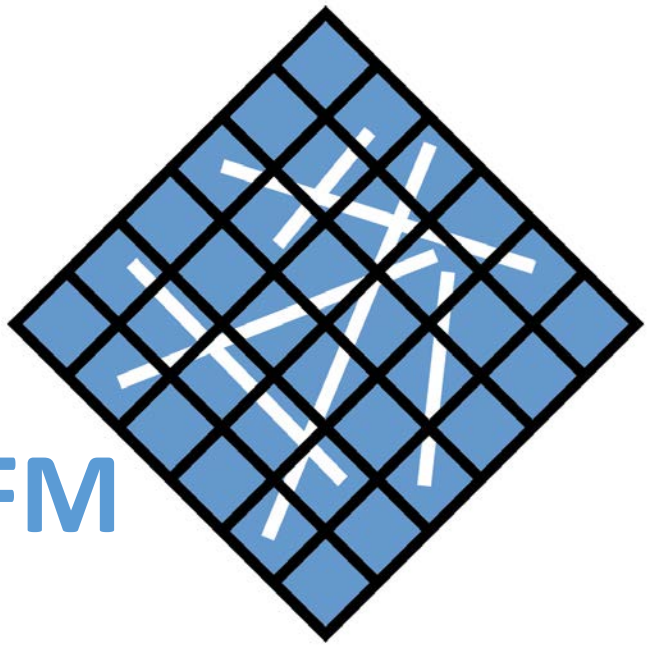
dwarftools.org

LearnEDFM



User Manual v.1.1

Gunnar Jansen
University of Neuchâtel
Rue Emile-Argand 11
2000 Neuchâtel
gunnar.jansen@unine.ch

TABLE OF CONTENTS

Acknowledgements 3

Copyright 3

License 3

Introduction 4

Quick Start 6

Running the Code 6

Elements of the Code 7

Input File 9

Examples 9

Boundary Conditions 10

Theory of the embedded discrete fracture model 11

Governing equations 11

Fracture Matrix coupling 13

Fracture Intersections 14

Discretised EDFM equations by the Finite Volume method 15

Elliptic pressure equation 15

Transport equation 16

System of equations and solution strategy 16

Model validation 18

Example 1: Crossed Fractures 18

Example 2: Effect of matrix-fracture permeability contrast 21

Example 3: Field scale fracture network & gravity effects 23

Concluding remarks 26

Bibliography 27

Appendix : InputFile 28

ACKNOWLEDGEMENTS

LearnEDFM is based on Maflot, developed by Rouven Künze and Ivan Lunati at the University of Lausanne, 2012. Much of the initial code basis was provided through their code available at www.maflot.com.

I thank Prof. Stephen. A. Miller for his support and for sharing his pearls of wisdom with me during the creation of the LearnEDFM code. I would also like to show my gratitude to Reza Sohrabi without whom the code and documentation would not exist in this form.

This project was supported by the Swiss National Fond (SNF) through grant 'NFP70:Energy Turnaround (153971)'.

COPYRIGHT

LICENSE

INTRODUCTION

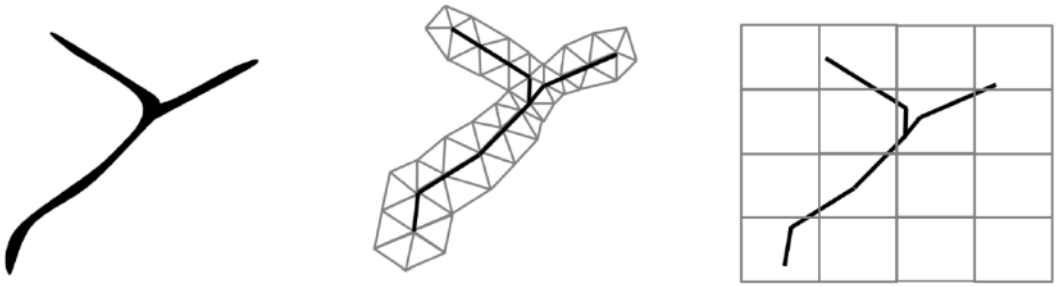
A large fraction of the world's water and energy resources are located in naturally fractured reservoirs within the earth's crust. Depending on the lithology and tectonic history of a formation, fracture networks can range from dense and homogeneous highly fractured networks to single large scale fractures dominating the flow behaviour. Understanding the dynamics of such reservoirs in terms of flow and transport is crucial to successful application of engineered geothermal systems (also known as enhanced geothermal systems) for geothermal energy production in the future.

Fractured reservoirs are considered to consist of two distinct separate media, namely the fracture and matrix space respectively. Fractures are generally thin, highly conductive containing only small amounts of fluid, whereas the matrix rock provides high fluid storage but typically has much smaller permeability.

Simulation of flow and transport through fractured porous media is challenging due to the high permeability contrast between the fractures and the surrounding rock matrix. However, accurate and efficient simulation of flow through a fracture network is crucial in order to understand, optimize and engineer reservoirs. It has been a research topic for several decades and is still under active research. Accurate fluid flow simulations through field-scale fractured reservoirs are still limited by the power of current computer processing units (CPU).

Discrete fracture models (DFM) have been developed to address the problem. However, traditional conforming DFM suffer from computationally expensive pre-processing in the numerical grid generation and can encounter severe time step restrictions during the simulation if small cells around the fractures are used. Another approach are the embedded discrete fracture models (EDFM), which treat fracture and matrix in two separate computational domains. The embedded model was first introduced by Lee et al. (Lee, Lough, & Jensen, 2001) for single phase problems and later extended to two-phase flow (Li & Lee, 2008). An example of the difference in the resulting computational grid from the DFM and EDFM is shown in Figure 1.

The embedded discrete fracture model is a promising new technique in modelling the behaviour of enhanced geothermal systems. Karvounis uses it in order to better understand and possibly forecast the induced seismicity by fluid injection during the stimulation phase of an EGS within a statistical approach (Karvounis, 2013). Norbeck et al. additionally model fracture deformation by linear fracture mechanics (Norbeck, McClure, Lo, & Horne, 2015). However, a general coupling to theory of poro-elasticity and the stress induced by fracture slip has not yet been performed.



(a) fracture (b) conforming unstructured (c) embedded structured

Figure 1: Different Grids resulting from fracture a). b) conforming unstructured as used in many DFM simulators. c) Embedded structured representation as used with the EDFM. Modified from (Karvounis, 2013).

A simple single phase flow and transport implementation to evaluate the strength of the model will be delivered. A MATLAB implementation of an embedded discrete fracture model will be developed and the governing equations of single phase flow and transport (including diffusion and advection) will be discretized using a two-point flux approximation scheme in a finite volume method. Separate computational domains are employed for fracture and matrix space of the model which leads to a coupled system of equations to be solved. An efficient coupling strategy has to be determined for numerical performance of the model.

The successful implementation of the EDFM model will be evaluated based on a reference test case, comparing the implemented model to an existing method earlier developed (Pluimers, 2015). The overall performance of the implemented model will further be evaluated by a field scale simulation through a fracture network. The modelling framework implemented here, can be used as a basis for future simulation tools using the EDFM.

QUICK START

You can get started running the LearnEDFM code right away. Open MATLAB, change into the LearnEDFM directory and run an example script:

```
>> cd PATH_TO_/LearnEDFM
>> cd examples/
>> ex1
```

If your MATLAB version gives you an error message at this point, please try to change back into the main directory and run the example from there:

```
>> cd ..
>> ex1
```

Beware, that this code is meant to be a tool to learn the embedded discrete fracture model (EDFM) and not a complete software that should be used for mindless simulations. Always look at your simulations results critically and evaluate if the computed results are physically making sense.

We encourage you to take the time to carefully read through this manual after you played around a little with the examples, so that you can understand more in depth what how the model works. It will be a great help understanding the method and its implementation and hopefully help you to extend it to your personal needs.

With this being said, let's get started. LearnEDFM is a simple MATLAB implementation of the embedded discrete fracture model. In the following sections the basic elements of the code are outlined. The usage of the input file and the conventions of the boundary conditions are also explained.

RUNNING THE CODE

LearnEDFM can be run from its directory by entering the command

```
>> EDFM('InputFile')
```

In the command window of MATLAB. An input file must be given. You can edit the existing input file, or copy it and edit it according to your needs and calling it by

```
>> EDFM('My_InputFile')
```

The input file is more or less self-explanatory and can be modified according to your needs. Additionally, the graphical output of the results can be suppressed. In order to do this the call has to be modified to

This might be useful, if timed simulations are desired as the graphical output takes a big part

```
>> EDFM('My_InputFile',0)
```

of the simulation time. You can still plot the final results afterwards as the data is written to the workspace.

ELEMENTS OF THE CODE

LearnEDFM consists of multiple functions. EDFM.m itself is the main function that controls the simulation and calls most of the sub functions. The sub functions deal with specific parts of the program, such as initialization, discretization and solution of the flow and transport equations. The individual functions are introduced briefly here while the underlying theory is further explained throughout the user manual.

EDFM.m

The main function. All sub functions are called out of this main function. It contains the time loop and is responsible for general functionalities of the code. More insight to the main functions structure is given in section *System of equations and solution strategy.*

InputFile.m

This is the user's control file for the simulation. It describes to discrete fracture network as well as the problems parameters and boundary conditions. The input file is self-explanatory but a short explanation is given in section *Input File.*

initialize.m

Initializes the interface based values such as permeability and porosity as well the gravity contributions to the pressure system. Read section *Discretised EDFM equations by the Finite Volume method* on the implemented approach.

calc_density.m

The density is temperature and pressure dependent. An equation of state is implemented in this function to calculate the correct value.

initializeDFN.m

Initializes the discrete fracture network.

intersectionsGrid.m

Finds the intersections of the discrete fracture network with the matrix grid and calculates the connectivity index. See section *Fracture Matrix coupling* for more detail.

intersectionsSegments.m

Finds intersections between individual fractures and computes the corresponding transmissivities. See section *Fracture Intersections* for more detail.

pressureSystem.m

Constructs the mass balance matrix and right hand side vector for the pressure system. The linear algebraic system is solver outside in the main function. See section *Elliptic pressure equation* for more detail.

transport_mass_System.m

Solves the transport problem based on the mass continuity equation of the solvent. The system is solved within the function. See section *Transport equation* for more detail.

transport_mass_Advection.m

Constructs the advection matrix using a first order upwind approximation.

transport_mass_Diffusion.m

Constructs the diffusion operator for the transport equation.

transport_mass_Dispersion.m

Constructs the dispersion operator for the transport equation.

transport_heat_System.m

Solves the heat transport problem based on the heat conservation equation. The system is solved within the function. See section *Transport equation* for more detail.

transport_heat_Advection.m

Constructs the advection matrix using a first order upwind approximation.

transport_heat_Diffusion.m

Constructs the diffusion operator for the transport equation.

INPUT FILE

The input file is more or less self-explanatory and can be modified according to your needs. Units are always given in brackets. The form of the Input file respectively all variables shown here must be preserved to avoid errors.

Boundary vectors, initial values, source terms, etc. always have to be initialized in the dimensions shown above. To change the boundary conditions, please read the section on boundary conditions in this manual carefully.

Viscosity and density are defined as a two entry vector with the first value referring to concentration zero and the second to max concentration. Density effects can be neglected by setting gravity to 0.

An example input file that solves a transient, coupled flow and transport problem is given in *Appendix : InputFile.*

EXAMPLES

LearnEDFM is distributed with three examples that can directly be used to understand the method and used as a basis for new simulations. The following examples are included:

- *Example 1: Crossed Fractures*
  A simple test case of two perpendicular fractures interacting to an external pressure field
- *Example 2: Effect of matrix-fracture permeability contrast*
  Short series of simulations evaluating the effect of permeability contrast
- *Example 3: Field scale fracture network : Tracer transport*
  A complex fracture network test case on a larger scale including gravity
- *Example 4: Field scale fracture network : Heat transport*
  A complex fracture network test case on a larger scale including gravity

The examples and their results are presented in the section *Model validation* in greater detail.

BOUNDARY CONDITIONS

The boundary conditions follow a convention that is explained here in detail. Note, that for the pressure equation two types of boundary conditions (Dirichlet and Neumann) can be used. The transport equation can only be constrained by Dirichlet boundary conditions as the flux is prescribed by the pressure equation.

> **Dirichlet**: The value at the boundary should fixed. I.e. on the left side boundary 1MPa pressure is applied. Another example would be the tracer amount at the top is 1kg/m^3.
>
> **Neumann**: The flux at the boundary is fixed. I.e a constant flux of 10m^3/s enters at the bottom of the domain.

The input file uses a vector formulation to assign the boundary condition type and value. Thus, two vectors $ibcs$ and Fix with dimensions $2 \cdot [Nf(2) + Nf(1)]$ where $Nf(1)$ and $Nf(2)$ are the cells in x and y directions respectively, are used.

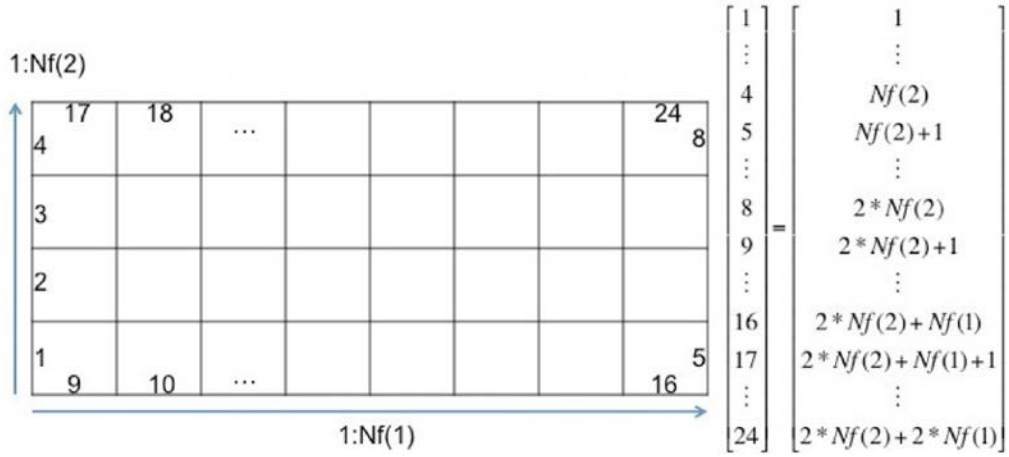


Figure 2: Visualization of the boundary condition conventions used in LearnEDFM. This image is directly taken from the Maflot manual.

The ordering of the vector entries is shown in the example in Figure 2. For convenience the index sets for the four boundaries are:

```
Left:        1 : Nf(2)
Right:       Nf(2)+1 : 2*Nf(2)
Bottom:      2*Nf(2)+1 : 2*Nf(2)+Nf(1)+1
Top:         2*Nf(2)+Nf(1)+1 : 2*Nf(2)+2*Nf(1)
```

The vector $ibcs$ defines the type of the boundary:

```
Neumann:     0
Dirichlet:   1
```

The vector Fix (or $FixC$ for the transport solution) defines the value to the defined type.

THEORY OF THE EMBEDDED DISCRETE FRACTURE MODEL

In this section the embedded discrete fracture model (EDFM) is presented as previously published by (Hajibeygi, Karvounis, & Jenny, 2011; Karvounis, 2013; Pluimers, 2015). As already mentioned in the introduction, the conceptual idea of the EDFM is the distinct separation of a fractured reservoir into a fracture and a damaged matrix domain. In order to account for coupling effects between the two domains, a transfer function is introduced (Figure 2). Fracture and matrix domains are thus computationally independent except for the transfer function. As the fractures are generally very thin and highly permeable compared to the surrounding matrix rock, the gradient of fracture pressure normal to the fracture is negligible. This allows for a lower dimensional representation of fractures (i.e. 1D objects within a 2D reservoir).

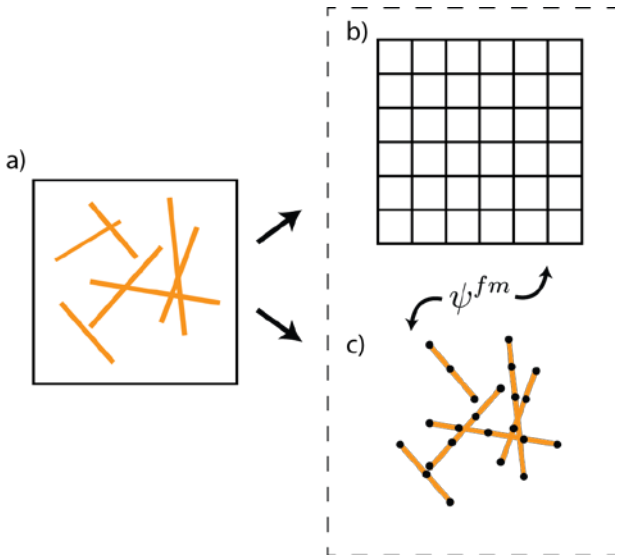


Figure 3: A fractured domain a) is separated in a uniform grid b) and a fracture grid c). The two resulting domains are coupled using the transfer function ψ^{fm}.

GOVERNING EQUATIONS

The flow in naturally fractured reservoirs is often described by the equations for incompressible single-phase flow. As the gravity can play an important role in the flow field evolution of a reservoir fluid with variable densities we also include the gravity effect in the model. It is noted, that the following equations are assumed valid in both the damaged matrix as well as the fractures.

The pressure equation of a single-phase fluid flow can be obtained from the mass balance equation

$$\frac{\partial(\phi\rho)}{\partial t} - \nabla \cdot (\rho\lambda \cdot \nabla(p - \rho g)\,) = \rho q, \quad \text{with } \lambda = \frac{K}{\mu}$$

Here, ϕ[-] is the porosity, ρ [kg/m^3] the fluid density and p [Pa] is the fluid pressure. Moreover λ [m^2/s] is the total mobility of the fluid, defined as the fraction of the permeability K [m^2] and the fluid viscosity μ [s^{-1}]. In general the permeability K is an anisotropic 2nd order tensor. However, only a diagonalised permeability tensor K is used hereafter and in the implementation. In case of an incompressible fluid the porosity ϕ and the density ρ can be assumed independent of the fluid pressure, thus leading to the elliptic pressure equation

$$-\nabla(\lambda \cdot \nabla(p - \rho g)) = q \quad \text{or} \quad -\nabla\left(\frac{K}{\mu} \cdot \nabla(p - \rho g)\right) = q$$

From the fluid pressure p, the fluid velocity can be calculated using Darcy's law, i.e.

$$v = -\frac{K}{\mu}\nabla p$$

The velocities can subsequently be used in the transport equations. The transport equation is derived similarly to the continuity equation by the balancing the components of the transported mass and can be described as

$$\phi\frac{\partial c}{\partial t} + v \cdot \nabla c - \nabla(\kappa\nabla c) = 0,$$

where c [kg/m^3] is the concentration of the transported medium, v [m/s] the previously introduced Darcy velocity and κ [m^2/s] the diffusivity.

The heat transport equation is derived based on the conservation of energy. It can be described as

$$c_p\rho\frac{\partial T}{\partial t} + c_p^f\rho_f v \cdot \nabla T - \nabla(\lambda\nabla T) = 0,$$

where T [°C] is the Temperature, v [m/s] the previously introduced Darcy velocity, $\rho\left[\frac{\mathrm{kg}}{\mathrm{m}^3}\right]$ the density, $c_p[\mathrm{J}/(\mathrm{kg\,K})]$ the specific heat capacity and λ [m^2/s] the effective thermal diffusivity of the porous medium.

The total mass balance equation derived above is separated into for the matrix and the fracture domains, i.e.

$$-\nabla\left(\frac{K^m}{\mu} \cdot \nabla(p - \rho g)^m\right) + \psi^{mf} = q^m \text{ on } \Omega^m \in \mathbb{R}^n$$

and

$$-\nabla\left(\frac{K^f}{\mu}\cdot\nabla(p-\rho g)^f\right)+\psi^{fm}=q^f \text{ on } \Omega^f\in\mathbb{R}^{n-1}$$

where ψ^{mf} and ψ^{fm} is the flux transfer function between the damaged matrix and the fractures, and superscripts m and f denote matrix and fracture quantities respectively. The transport equation is separated accordingly, the derivation omitted here for clarity.

FRACTURE MATRIX COUPLING

As the flow in the damaged matrix and the fractures is treated separately, a transfer function, governing the mass exchange between the two domains is used. The transfer function is treated as a source/sink term in the pressure equations for damaged matrix and fracture respectively which is similar to classical well models (Peaceman, 1978) , i.e.

$$\psi_{f,m}=CI\,\lambda_m(p_f-p_m).$$

CI is the 'connectivity index' which is grid dependent and defined based on the linear pressure distribution assumed within a grid cell intersected by a fracture (Hajibeygi et al., 2011). From the separated mass balance equations it becomes immediately clear that the total flux between matrix and fracture has to be conserved:

$$\int\psi^{mf}dV=-\int\psi^{fm}dA.$$

The connectivity index is defined as the length fraction $A_{ij,k}$ of fracture segment k inside matrix cell ij divided by the average distance $\langle d\rangle_{ij,k}$ between matrix cell ij and fracture segment k.

$$CI_{ij,k}=\frac{A_{ij,k}}{\langle d\rangle_{ij,k}}$$

Obviously, $\psi_{f,m}$ is only non-zero in matrix cells which are actually intersected by at least one fracture segment. The average distance $\langle d\rangle_{ij,k}$ can be calculated as

$$\langle d\rangle_{ij,k}=\frac{\int x_k(x')dx'}{V_{ij}}$$

which in many cases has to be done by numerical integration. For rectangular grids however, there exists an analytical solution derived by (Hajibeygi et al., 2011) for fractures intersections horizontally, vertically or on the diagonal to a grid cell and later extended to general intersection on rectangular grid by (Pluimers, 2015). For enhanced efficiency, the analytical expression are used in the present implementation.

FRACTURE INTERSECTIONS

In naturally fractured reservoirs fractures often intersect other fractures which potentially has a big impact on the flow dynamics in the reservoir. The effect of fracture-fracture coupling has therefore also to be considered in the model. The additional transmissivity at a fracture intersection can be obtained similar to the approach used in electrical engineering for known as the star-delta transformation in circuits (Karimi-Fard, Durlofsky, & Aziz, 2004). For the fracture intersection shown in Figure 3 the additional fracture-fracture transmissivity can be calculated as

$$T_{i,j} = \frac{\alpha_i \cdot \alpha_j}{\alpha_i + \alpha_j} \text{ with } \alpha_i = \frac{A_i^f \lambda_i}{0.5 \cdot dx_f}$$

where A_i^f denotes the fracture aperture, λ_i the total mobility, and dx_f the numerical discretization spacing in the fracture. This approach can be generalized to more than two fractures intersecting which is omitted here due to the rare occasion in reality.

Figure 4: Two intersecting fracture segments i and j

DISCRETISED EDFM EQUATIONS BY THE FINITE VOLUME METHOD

The implementation of the EDFM in two-dimensions by the Finite Volume method is described in this section. Both, the elliptic pressure equation and the transport equation are discretized by a two-point flux approximation scheme with the Finite Volume Method (FVM). Later in this section, the solution of the resulting system of equations as well as the solution strategy for the coupled flow and transport problem are discussed.

ELLIPTIC PRESSURE EQUATION

Using a finite volume approach we can discretise the domain Ω for the elliptic pressure equation as the integration over finite control volumes Ω_{ij} with $\Omega = \sum_{ij=1}^{N} \Omega_{ij}$. Using Gauss's theorem the divergence integral over the volume can be rewritten as the surface integral normal to the boundary of the volume, i.e. for a matrix grid cell

$$-\int_{\Omega_{ij}} \nabla \cdot (\lambda^m \cdot \nabla p)^m + \psi^{mf}\, dV = -\int_{\Omega_{ij}} \mathrm{q}^{\mathrm{m}}\, dV$$

$$\Rightarrow -\int_{\partial\Omega_{ij}} ((\lambda^m \cdot \nabla p)^m + \psi^{mf}) \cdot n\ ds = -\int_{\Omega_{ij}} \mathrm{q}^{\mathrm{m}}\, dV$$

The pressure gradient over the cell boundary $\partial\Omega_{ij}$ is approximated by a two-point flux approximation which is similar to the central difference scheme of the finite difference method and is second-order accurate in space. As the domain is generally heterogeneous in terms of rock properties, a harmonic averaging technique is used to calculate the appropriate values at the cell boundaries. This leads to the discretised expression for the matrix pressure

$$\frac{\Delta A_x \lambda_x^{i-\frac{1}{2},j}}{\Delta x}(p_{i,j} - p_{i-1,j}) + \frac{\Delta A_x \lambda_x^{i+\frac{1}{2},j}}{\Delta x}(p_{i,j} - p_{i+1,j}) +$$
$$\frac{\Delta A_y \lambda_y^{i,j-\frac{1}{2}}}{\Delta y}(p_{i,j} - p_{i,j-1}) + \frac{\Delta A_y \lambda_y^{i,j+\frac{1}{2}}}{\Delta y}(p_{i,j} - p_{i,j+1}) = q_{i,j} + \sum_{\Omega_{ij}\cap\Omega_k} CI_k \lambda_{ij,k}(p_k - p_{i,j})$$

where ΔA_x and ΔA_y represent the surface areas of the grid cells in x and y direction respectively. All other notations are adopted from the elemental literature of the Finite Volume Method. Note that the matrix-fracture interaction term has already been moved to the right-hand side using $\int \psi^{mf} = -\int \psi^{fm}$.

The fracture pressure can be expressed using the same approach as

$$\frac{\Delta A_f \lambda^{k-\frac{1}{2}}}{\Delta x_f}(p_k - p_{k-1}) + \frac{\Delta A_f \lambda^{k+\frac{1}{2}}}{\Delta x_f}(p_k - p_{k+1}) = q_k - \sum_{\Omega_{ij}\cap\Omega_k} CI_k \lambda_{ij,k}(p_k - p_{i,j})$$

Here, Δx_f is the discretisation lengths of the fracture segments and $\Delta A_f$the aperture of the fracture which can be modelled by the cubic law as $\Delta A = \sqrt[2]{12 \cdot K}$.

TRANSPORT EQUATIONS

In order to discretise the transport equations the same approach as for the pressure equation can be followed. Due to the similarity the explicit derivation is omitted here. It is however worth noting, that the advection term has to be treated with special care. This is not a limitation of the EDFM but a coherent issue in a wide range of numerical methods. In contrast to the pressure equation the transport equations involves a time derivative. Here an implicit time-discretization is used. In the current EDFM implementation a simple upwind strategy to evaluate the velocities at the interfaces is used to treat the arising numerical difficulties posed by the advection term of the transport equation. The upwind scheme is an often used elemental technique to treat advective transport problems but suffers from a large amount of numerical diffusion. A Taylor series expansion shows that the additional numerical diffusion, called 'artificial viscosity ν_e', is given by

$$\nu_e = \frac{v \cdot \Delta x}{2}$$

The artificial viscosity indicates an explicit dependence on the numerical grid size as well as the fluid velocity. These effects have to be kept in mind when using the implemented model. A wide range of alternative methods for the advection term exist but are not covered in the current version of the model for simplicity.

SYSTEM OF EQUATIONS AND SOLUTION STRATEGY

The discretized equations for pressure and transport can separately be assembled as a linear systems of equations of the form $A \cdot x = b$. Since the matrix and fracture domains are strongly coupled, the solution for both domains has to be found implicitly at the same time. This can be achieved through the assembly of a block matrix of the pressure and the transport systems respectively, i.e.

$$\begin{pmatrix} A_{mm} & A_{mf} \\ A_{fm} & A_{ff} \end{pmatrix} \begin{pmatrix} p_m \\ p_f \end{pmatrix} = \begin{pmatrix} q_m \\ q_f \end{pmatrix} \quad \text{(Pressure)}$$

$$\begin{pmatrix} T_{mm} & T_{mf} \\ T_{fm} & T_{ff} \end{pmatrix} \begin{pmatrix} c_m \\ c_f \end{pmatrix} = \begin{pmatrix} d_m \\ d_f \end{pmatrix} \quad \text{(Transport)}$$

The submatrix blocks A_{mm} and T_{mm} contain the matrix transmissivities, A_{ff} and T_{ff} contain the fracture transmissivities including any fracture-fracture intersections, the off-diagonal submatrix blocks contain the fracture-matrix and matrix-fracture transmissivities $(A/T)_{mf}$ and $(A/T)_{fm}$. The solution of these matrices is performed by the direct solver of MATLAB in the current implementation.

Although an explicit coupling also exists between the pressure and transport equations themselves, the current model uses a serial scheme to solve the coupled problem. Instead of assembling and solving one very large system for pressure and transport, the problem is divided in two parts. In a first step, the pressure system is assembled and solved. Using Darcy's law, the fluid velocities can be calculated in an intermediate step. Once the fluid velocities are found, the transport system can be assembled and subsequently be solved. In strongly coupled flow and transport problems, an iterative scheme has to be used to capture any arising nonlinearities. In most cases, the flow and transport exhibit rather loose coupling in which only one or two iterations are needed to converge to the solution. Figure 4 shows a flowchart of the implemented solution strategy. It is re-stated that the transport equation is discretised by an unconditionally stable implicit time-discretisation scheme. In case the problem shows non-linear behaviour, issues with non-convergence might appear and place an indirect restriction for the time-step. Nonetheless, compared to explicit schemes much larger time steps are allowed in the implemented approach.

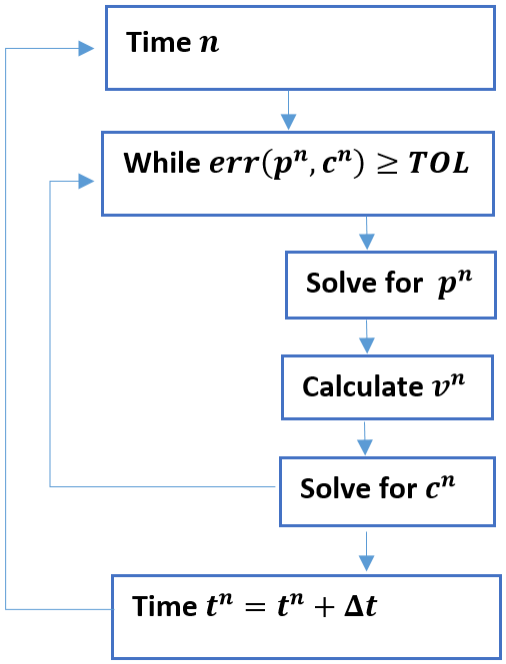


Figure 5: The simplified flowchart of the implemented solution strategy for coupled flow and transport problems.

MODEL VALIDATION

In the following the implementation of the EDFM is validated. Three examples of how the provided code can be used are presented. The scripts to run all examples are provided with the LearnEDFM source code. They can be found in the /examples subdirectory and can be used directly from within MATLAB by the following commands

```
>> cd examples/
>> ex1
```

to run the first example. The other examples can be called in the same fashion.

EXAMPLE 1: CROSSED FRACTURES

To validate the model a test case first presented by (Hajibeygi et al., 2011) is used. The same test case is also used by (Pluimers, 2015) to study the sensitivity of the EDFM with respect to numerical resolution and fracture position within the intersecting grid cells. The test case is shown in Figure 5 and consists of two perpendicular intersecting fractures in the middle of a square domain. The aperture of the fractures is modelled by the cubic's law depending on the

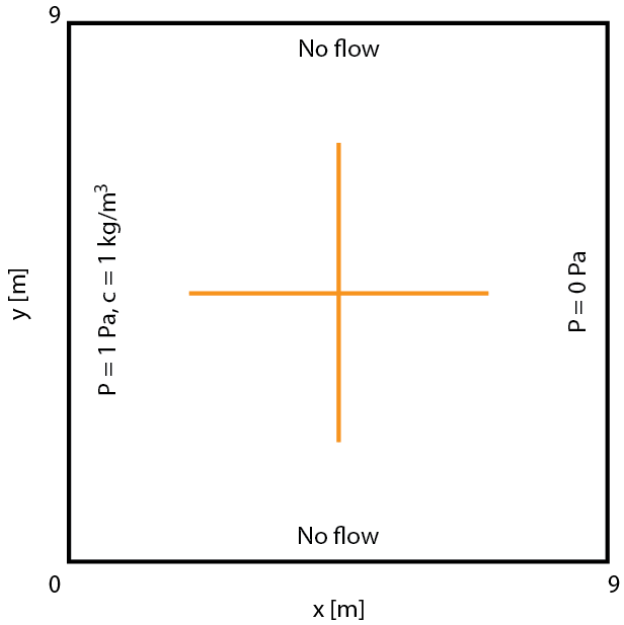


Figure 6: Sketch of the domain and boundary condition for the crossed shaped fracture network test case.

fracture's permeability. This is in contrast to the former studies of the crossed fracture test case where constant fracture apertures are used. The domain is 9m by 9m square domain with Dirichlet boundary condition on the left and right sides. On the left a constant pressure of 1Pa is applied, whereas the right side is fixed to 0Pa. On the top and bottom ides a no-flow Neumann boundary is applied. In this test the resolution of the damaged matrix and the fracture is fixed. The matrix domain is discretised by 100x100 cells whereas the fractures are

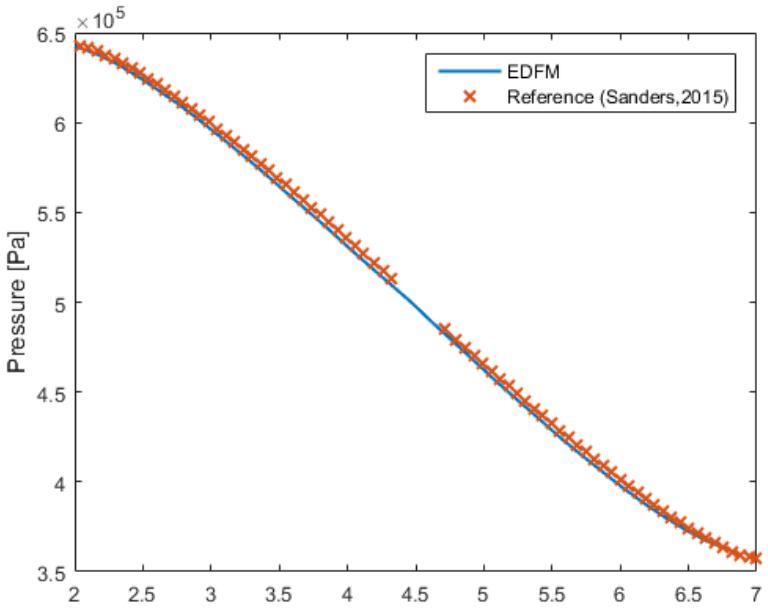


Figure 7: Fracture pressure through the horizontal fracture. EDFM and reference solution show excellent agreement.

modelled by 66 fracture segments (33 each). The solution obtained by the EDFM is compared to a reference solution computed by a conforming discrete fracture network were the matrix elements are aligned exactly on the grid. The reference solution contains 1404225 degrees of freedom (*dof*) and is presented in (Pluimers, 2015). The fracture pressure is shown in Figure 6. A very good match is found between the reference and the implemented model. However, if the results are compared in detail a slight offset between the reference solution and the current implementation becomes visible. This offset is most likely due to the difference in fracture aperture. The reference solution is based on a fixed aperture of $1/250$ whereas the EDFM implementation uses cubic's law which corresponds to an aperture of $\approx 1/285$. Figure 7 shows the matrix pressure at y = 4.5m which is the position of the horizontal fracture. Again, reference and EDFM solutions are in good agreement. Slight deviations from the reference solution are again visible which are likely due to the difference in aperture described above. It is worth noting that the EDFM model contains about two magnitudes less *dof* compared to the reference solution. This clearly shows the computational advantage of the implemented model over traditional approaches. Based on the good agreement of the implemented EDFM and the reference solution the method is validated for this test case.

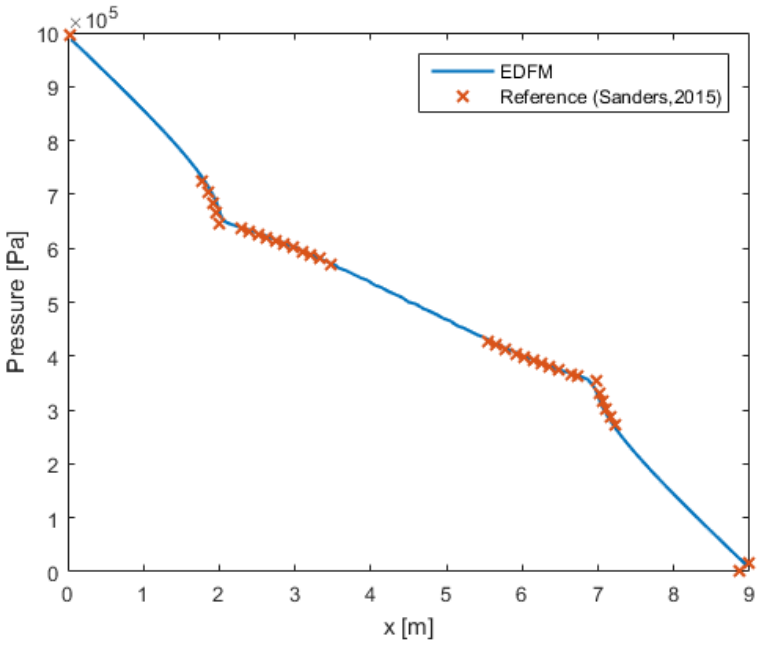


Figure 8: Matrix pressure at y=4.5m - the position of the horizontal fracture.

EXAMPLE 2: EFFECT OF MATRIX-FRACTURE PERMEABILITY CONTRAST

The changes in the pressure field due to the fractures are dominated by the permeability contrast between the damaged matrix and the fractures. The sensitivity of the EDFM on the permeability contrast is of importance as the contrast can be quite large in nature. Thus, three different contrast factors of $\frac{K_f}{K_m} = 10^2$, $\frac{K_f}{K_m} = 10^3$ and $\frac{K_f}{K_m} = 10^4$ are studied here. These different contrast-factors can be considered realistic depending on the geologic setting

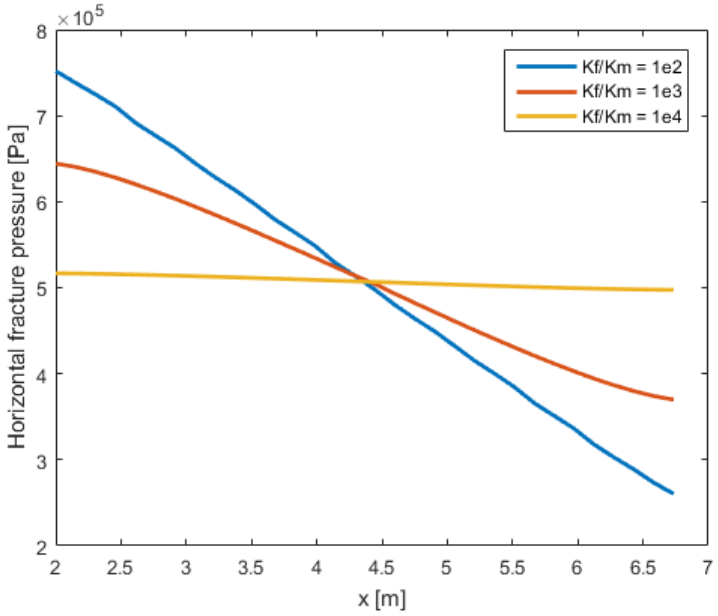
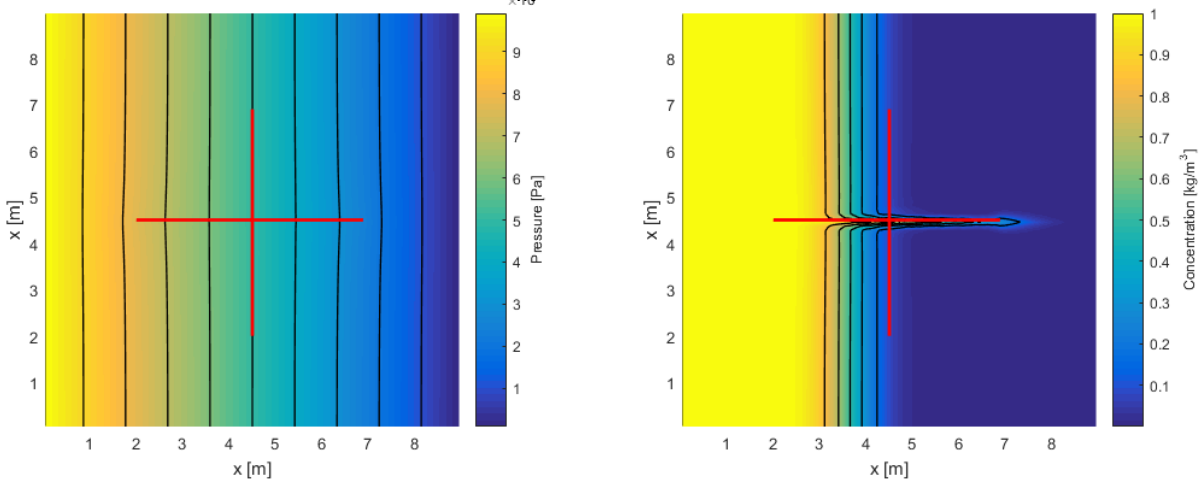


Figure 9: Fracture pressure of the horizontal fracture for different matrix-fracture permeability contrast factors.

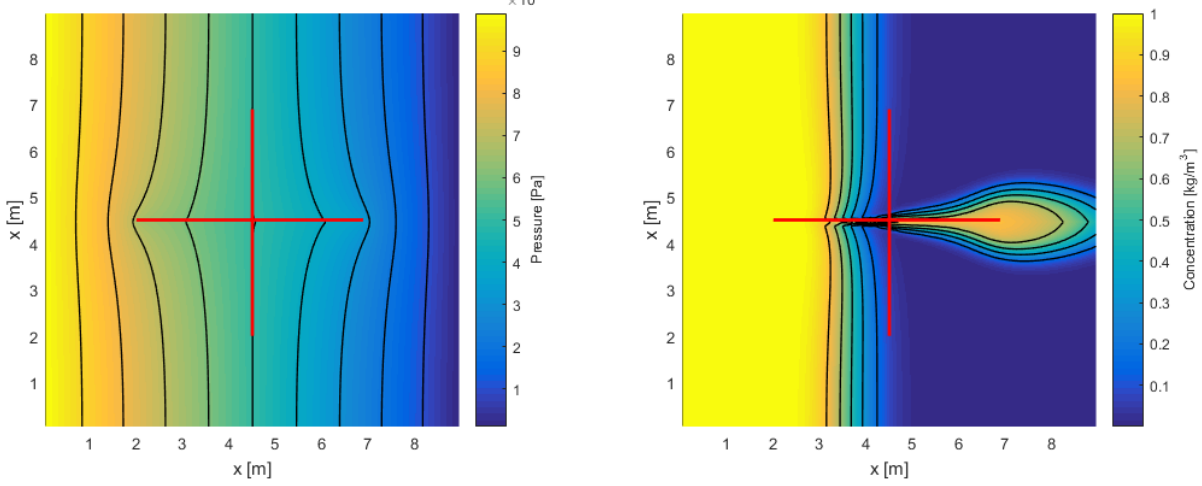
. In the original test case only the pressure solution is considered. Here, also a quantitative analysis of the transport through the crossed fractures is performed. To this end at the left side of the domain a concentration boundary of $c = 1\frac{kg}{m^3}$ is applied. The domain itself has an initially zero concentration of the tracer. Figure 9 shows the solutions for pressure and concentration of the test case after 100 seconds of simulation. The effect of the increasing contrast between matrix and fracture permeability is strongly pronounced in the matrix solutions. The left series of figures shows the pressure solution for each contrast factor. It is evident that the transport problem is heavily dependent on the permeability contrast-factor. For a small permeability contrast the amount of tracer that is transported by the fracture network is relatively small. This is indicated by the rather small plume at the right end of the horizontal fracture. With increasing contrast in the permeability the amount of tracer transported by the horizontal fracture is also increased. The plume at the right side of the fracture becomes more dominant with increasing contrast.

Figure 10: Pressure and transport solutions for the cross shaped fracture test case. The effect of matrix-fracture permeability contrast is clearly visible. Left: Pressure distribution. Right: Concentration after 100s.

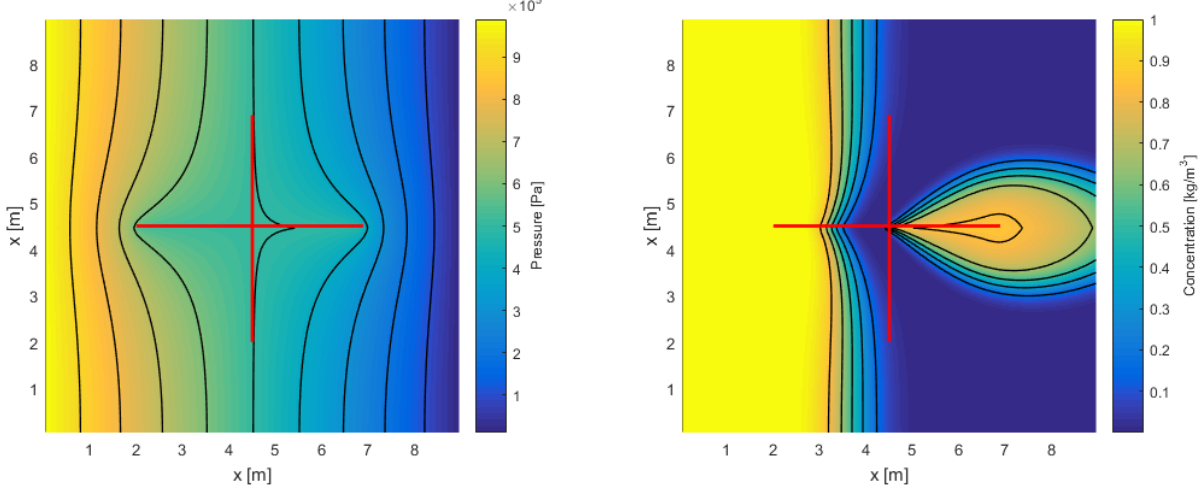
$$\frac{K_f}{K_m} = 10^2$$



$$\frac{K_f}{K_m} = 10^3$$



$$\frac{K_f}{K_m} = 10^4$$



EXAMPLE 3: FIELD SCALE FRACTURE NETWORK & GRAVITY EFFECTS

In order to evaluate the performance of the implemented model on a more realistic environment, a complex fracture network is generated. Note, that the fracture network used here is based solely on artistic sense, than scientific procedures that exist to generate such networks. Figure 9 shows a sketch of the fracture network used. The fracture network consists of 13 large scale fractures in a 500 by 500 m domain. The fractures are discretised in a total of 384 fracture segments with a matrix grid containing 9801 cells. The fracture network features fracture intersections as well as multiple fracture intersection the same matrix cell. The fracture segments intersect the matrix grid at arbitrary orientation and have different intersection lengths. To show the importance of gravity on the transport solution a simple equation of state relationship between concentration and the fluid density is established

$$\rho = \begin{cases} 935\ {}^{\mathrm{kg}}/_{\mathrm{m}^3} \text{ if } = 0 \\ 992\ {}^{\mathrm{kg}}/_{\mathrm{m}^3} \text{ if } = 1 \end{cases}$$

The densities are chosen to represent water at 130°C and 40°C respectively. These values can be considered realistic estimates of geothermal extraction and rejection temperatures for a geothermal power plant in Switzerland. For intermediate concentrations this leads to a linear interpolation of the density with respect to the tracer concentration:

$$\rho = \rho_1 \cdot (1 - c) + \rho_2 \cdot c$$

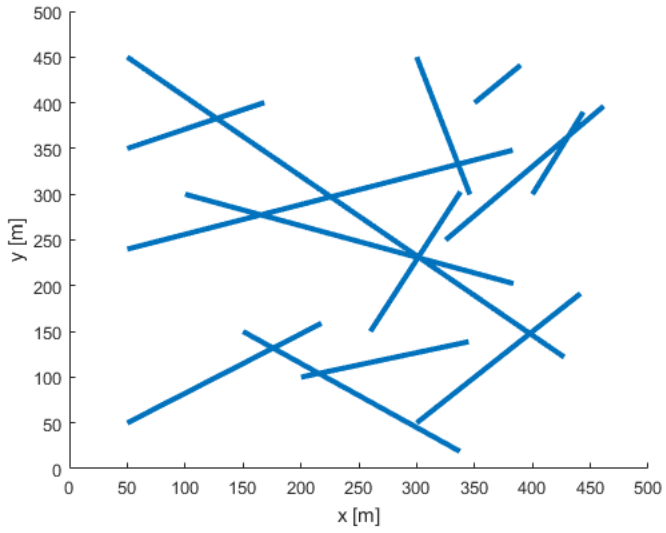


Figure 11: Sketch of the field size fracture network used in the second test case.

Note, that the current implementation does not actually treat the temperature exchange between the hot and cold fluids. A relatively high matrix-fracture permeability contrast of $\frac{K_f}{K_m} = 10^4$ is chosen for this test case. The boundary conditions are chosen as in the first test

case shown in Figure 10. Note, that due to the larger size of the model a larger pressure gradient is applied to the left side with 5 MPa.

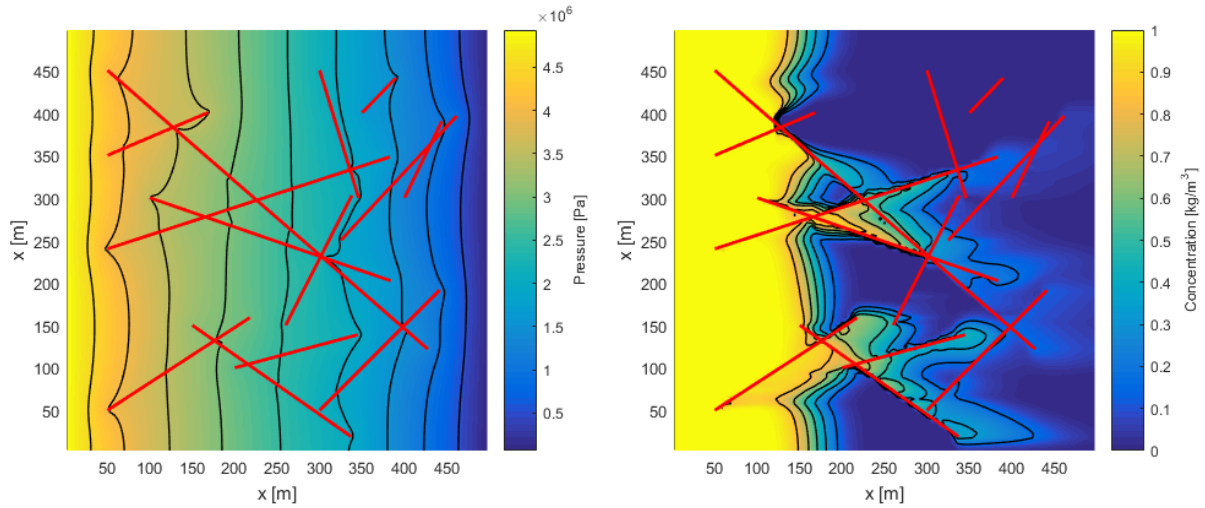


Figure 13: Complex fracture network test case. Gravity effects are neglected. Left: Pressure field; Right: Tracer concentration after 14h simulation.

Figure 11 shows the results of the coupled pressure and transport problem after about 14 hours neglecting gravity effects. The pressure field is almost homogeneous except for the small fracture induced perturbations. In the transport solution the influence on the tracer distribution is clearly visible. Fracture intersections allow the tracer to reach the right side of the domain much earlier than by the advection front through the matrix. An interesting observation can be made for the fracture in the upper left part of the domain (@ $x = 50, y = 350$). The fracture is intersected by a long fracture at $x = 130, y = 380$ which acts as a pressure sink. Another interesting feature is visible at the fracture in the bottom left corner of

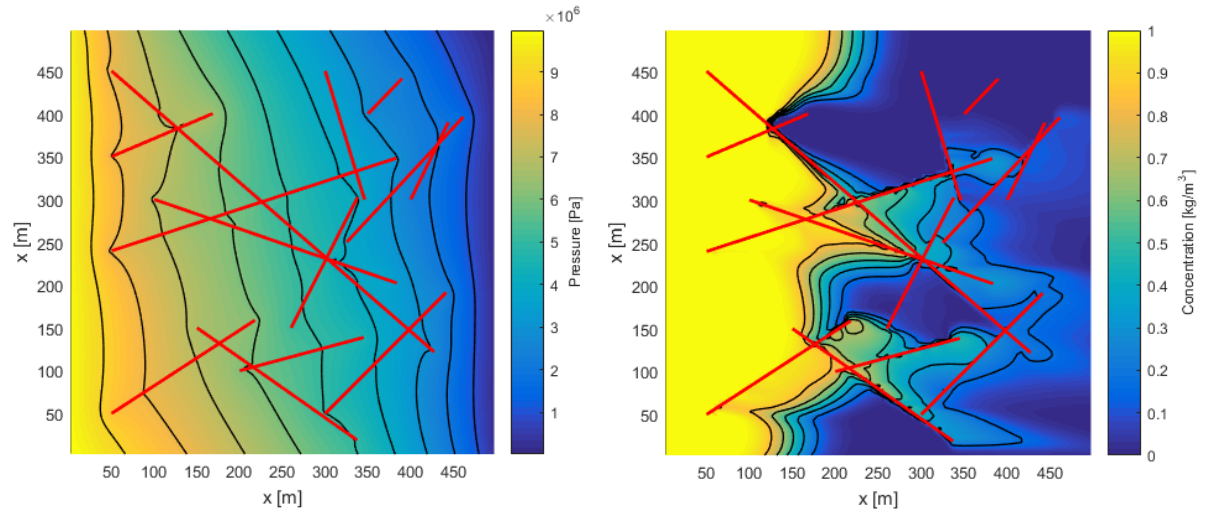


Figure 12: Complex fracture network test case including gravity effects. Left: Pressure field; Right: Tracer concentration after 14h simulation.

the domain (@ $x = 50, y = 50$). Here, a tracer shadow (a zone which is of slightly darker colour than the surrounding cells) following the fracture becomes visible. It is caused by the tracer intake of the fracture. Since advection is dominating in this test case the *missing* tracer is not balanced behind the fracture.

Including the effect of gravity changes the pressure field, as well as the transport solution. Figure 12 shows the results including gravity obtained with the same boundary conditions and after the same simulation time. The pressure contours are slightly rotated by the additional effect of gravity. Comparing the scale on the pressure plots including and neglecting gravity the additional pressure induced by the hydrostatic pressure of the water is visible. Due to the higher pressure gradient the tracer transport is more advanced towards the right side of the domain when gravity is included (Figure 12). In addition to the higher transport rate, a downwards trend in the tracer transport is visible. This trend can be seen nicely in the downwards pointing fracture around $x = 225, y = 250$ as well as in the tracer shadow behind the fracture at $x = 50, y = 50$.

CONCLUDING REMARKS

The EDFM excels in its simplicity. The complexity associated with numerical grid generation of traditional approaches does not pose a problem, as any grid geometry can be used. Fractures are represented as lower dimensional manifolds in the model which gives the EDFM a computational advantage which is especially of importance when field scale problems are considered. To fully understand and prospect reservoir conditions accurate and efficient numerical methods are needed. The EDFM is extremely well-suited for this task and further research in this direction is recommended. This EDFM implementation was applied to some test cases. These test cases showed the numerical accuracy as well as a computational advantage of the method compared to traditional approaches. The EDFM solution converges well to the reference solution obtained with high accuracy.

The current implementation of the EDFM uses a simple first order upwind approximation for the advection term. It is well known, that this scheme introduces a large amount of artificial numerical diffusion. This is problematic for applied simulations because it alters the transport solutions. In order to accurately characterize the heat and fluid transport a different approach should be used to treat the advection term of the transport equation. Multiple approaches have been developed in the past and can be applied to the EDFM in the future.

The EDFM model developed within the scope of this project and its future extensions can provide new insight and better performance for the application to practical settings in order to help the design of new geothermal reservoirs, optimize drilling and exploration strategies with respect to induced seismicity and aid in research settings to better understand the fundamental reservoir processes.

BIBLIOGRAPHY

Hajibeygi, H., Karvounis, D., & Jenny, P. (2011). A hierarchical fracture model for the iterative multiscale finite volume method. *Journal of Computational Physics*, *230*(24), 8729–8743. http://doi.org/10.1016/j.jcp.2011.08.021

Karimi-Fard, M., Durlofsky, L. J., & Aziz, K. (2004). An Efficient Discrete Fracture Model Applicable for General Purpose Reservoir Simulators. *SPE Journal, SPE-88812-PA*, *9*(02), 227 – 236. http://doi.org/10.2118/79699-MS

Karvounis, D. (2013). Simulations of enhanced geothermal systems with an adaptive hierarchical fracture representation, (21222). Retrieved from http://e-collection.library.ethz.ch/view/eth:7369

Lee, S. H., Lough, M. F., & Jensen, C. L. (2001). Hierarchical modeling of flow in naturally fractured formations with multiple length scales. *Water Resources Research*, *37*(3), 443–455. http://doi.org/10.1029/2000WR900340

Li, L., & Lee, S. (2008). Efficient Field-Scale Simulation of Black Oil in a Naturally Fractured Reservoir Through Discrete Fracture Networks and Homogenized Media. *SPE Reservoir Evaluation & Engineering*. http://doi.org/10.2118/103901-PA

Norbeck, J. H., McClure, M. W., Lo, J. W., & Horne, R. N. (2015). An embedded fracture modeling framework for simulation of hydraulic fracturing and shear stimulation. *Computational Geosciences*. http://doi.org/10.1007/s10596-015-9543-2

Peaceman, D. W. (1978). Interpretation of Well-Block Pressures in Numerical Reservoir Simulation. *Society of Petroleum Engineers Journal*, *18*(3), 183 – 194. http://doi.org/10.2118/6893-PA

Pluimers, S. (2015). *Hierarchical Fracture Modeling Approach*. TU Delft.

APPENDIX : INPUTFILE

```
%% GRID PARAMETERS ----------------------------------------------------------------------------%
global Nf Nf_f len dx
len     = [500 500];                    % physical length of the domain in x and y direction [m]
Nf      = [51 51];                       % number of cells in x and y direction
dx      = len./Nf;                      % cell length [m]

%% SIMULATION PARAMETER FOR TRANSPORT ---------------------------------------------------------%
global dt
timeSim  = 7200;%1e4;                          % total simulation time [s]
dt       = 10;                           % time step length [s]
tol    = 1.e-4;                        % saturation tolerance on pressure-concentration-heat loop [-]
maxit    = 100;                          % maximum number of pressure concentration-heat loops to converge

%% FRACTURE NETWORK
% Crossed fracture test case
%frac_cross
% Random fractures
%frac_rand
% Single fracture
%frac_single
% Crossed fractures (rotated 45°)
%frac_cross_rot45
% Thirteen 'random' fractures
frac_complex_n13;

if (dxf < min(dx))
    error('dxf < dx')
end

%% INITIAL CONDITIONS-------------------------------------------------------------------------%
global cmax
c0   = zeros(Nf(1),Nf(2));              % Initial saturation (normalized concentration) [-]
c0f  = zeros(Nf_f,1);
cmax = 1;                               % maximum concentration [kg/m3] for normalization

T0   = zeros(Nf(1),Nf(2));              % Initial matrix temperature [°C]
T0f  = zeros(Nf_f,1);                   % Initial fracture temperature [°C]
tmax = 0;                              % maximum temperature [°C] for plotting

p0   = zeros(Nf(1),Nf(2));              % Initial matrix pressure [Pa]
p0f  = zeros(Nf_f,1);                   % Initial fracture pressure [Pa]

%% BC FLUID -----------------------------------------------------------------------------------%
global Fix ibcs
ibcs = zeros(2*sum(Nf),1);              % type 0:Neumann(N); 1:Dirichlet(D)
Fix  = zeros(2*sum(Nf),1);              % value N [m2/s] (inflow>0); D [Pa]

ibcs(1:Nf(2)) = 1;
ibcs(Nf(2)+1:2*Nf(2))=1;
Fix(1:Nf(2)) = 5e6;%5e6;
Fix(Nf(2)+1:2*Nf(2))=0;

%% BC TRANSPORT -------------------------------------------------------------------------------%
flagTracerTransport = 1;
flagHeatTransport = 0;

global FixT FixC
FixT     = zeros(2*sum(Nf),1);           % normalized concentration of boundary flow [-]
FixC     = zeros(2*sum(Nf),1);           % normalized concentration of boundary flow [-]

FixC(1:Nf(2)) = 0.5;

%% SOURCE TERMS -------------------------------------------------------------------------------%
Q       = zeros(Nf);                    % source term [m2/s]; inflow positive
QC      = zeros(Nf);                    % normalized concentration for source term [-]
QT      = zeros(Nf);                    % normalized concentration for source term [-]

%% GRAVITY------------------------------------------------------------------------------------%
global gravity
gravity   = 9.81;                          % gravity acceleration in y [m/s2]

%% PERMEABILITY -------------------------------------------------------------------------------%
K       = ones(Nf(1),Nf(2))*1e-9;               % permeability field [m2]
K_f     = ones(Nf_f,1)*1e-5;             % fracture permeability field [m2]

%% Porosity -----------------------------------------------------------------------------------%
phi     = ones(Nf(1),Nf(2))*0.3;      % porosity field
phi_f   = ones(Nf_f,1)*0.3;             % fracture porosity field

%% ROCK DENSITY -------------------------------------------------------------------------------%
density_s = 2000*ones(Nf);       % density of the rock [kg/m3]
density_sf = 2000*ones(Nf_f,1); % density of the rock [kg/m3]
```

```
%% IN SITU STRESS ------------------------------------------------------------------------%
SH_max = 52e6*ones(Nf_f,1);                     % Maximum principal stress [Pa]
SH_min = 20e6*ones(Nf_f,1);                     % Minimum principal stress [Pa]

%% THERMAL DIFFUSION ---------------------------------------------------------------------%
global lambda_l lambda_s cp_l cp_s ibcD
lambda_l = 0;                           % Thermal conductivity of the fluid [W/(m*K)]
lambda_s = 0;                           % Thermal conductivity of the rock [W/(m*K)]
cp_l = 0;                               % Specific heat capacity of the fluid [J/(kg*K)]
cp_s = 0;                               % Specific heat capacity of the rock [J/(kg*K)]
ibcD    = zeros(2*sum(Nf),1);           % 1 -> Diffusion on boundary cells

%% MOLECULAR DIFFUSION -------------------------------------------------------------------%
global DifC ibcDC
DifC     = 0;                  % [m2/s] molecular diffusion
ibcDC    = zeros(2*sum(Nf),1);           % 1 -> Diffusion on boundary cells

%% MASS DISPERSION -----------------------------------------------------------------------%
global alphal alphat
alphal   = 0.0;                         % longitudinal dispersivity [m]
alphat = 0.0; % transversal dispersivity [m]%% GRID PARAMETERS ------------------------------------------
```