Solute Transport in Hydrosystems I

Philipp Selzer, Haibing Shao
Dresden, 11.07.2025

HELMHOLTZ
CENTRE FOR
ENVIRONMENTAL
RESEARCH – UFZ

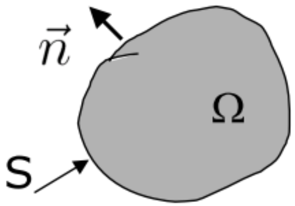
Outline

- Continuity Equation
- Advection, Dispersion, and Diffusion
- Sorption Isotherms and Decay
- Derivation of Advection Dispersion Equation (ADE) for a sorbing compound and decay
- Qualitative exercise on solute transport

Governing Equations

Continuity equation for an extensive quantity with volumetric density ρ:

$$\frac{\partial}{\partial t}\int_{\Omega} \rho d\mathbf{x} + \oint_{\partial\Omega = S} \mathbf{n}\cdot\mathbf{f} ds = \int_{\Omega} q d\mathbf{x} \qquad \text{(Eq. 0)}$$



Or, in ordinary language:

$$\left\{ \begin{array}{l} \text{Rate of} \\ \text{accumulation for} \\ \text{unknown } \rho \text{ within} \\ \text{the volume } \Omega \end{array} \right\} + \left\{ \begin{array}{l} \text{Net influx of } \rho \text{ into} \\ \text{the volume } \Omega \\ \text{through its} \\ \text{surface S} \end{array} \right\} = \left\{ \begin{array}{l} \text{Net rate of } \rho \\ \text{production/} \\ \text{decay within} \\ \text{the volume } \Omega \end{array} \right\}$$

Governing Equations

Recall the Divergence Theorem ("Satz von Gauß"):

$$\oint_{\partial\Omega} \mathbf{n} \cdot \mathbf{f} d\mathbf{s} = \int_{\Omega} \nabla \cdot \mathbf{f} d\mathbf{x}$$

So, our Eq. 0 becomes.

$$\frac{\partial}{\partial t} \int_{\Omega} \rho d\mathbf{x} + \int_{\Omega} \nabla \cdot \mathbf{f} d\mathbf{x} = \int_{\Omega} q d\mathbf{x} \qquad \text{(Eq. 1)}$$

Advection Dispersion Equation (ADE) in 1D

If write Eq. 1 in derivative form,

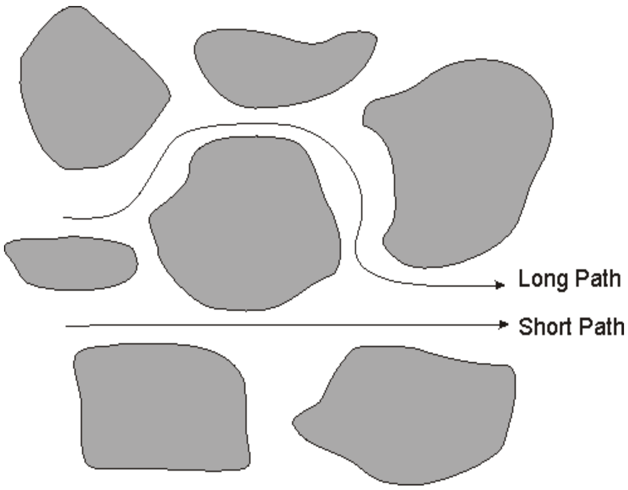
$$\frac{\partial \rho}{\partial t} + \nabla \cdot \mathbf{f} = s \qquad \text{(Eq. 2)}$$

Source / Sink

Our primary unknown is solute mass,
where: volumetric density = concentration

$$\rho = c_w n_e$$

Our flux is composed of:
1) Advective flux
2) Diff./ Disp. flux



Advection (shift of location):

Flux of advection: $J_{adv} = q c_w$

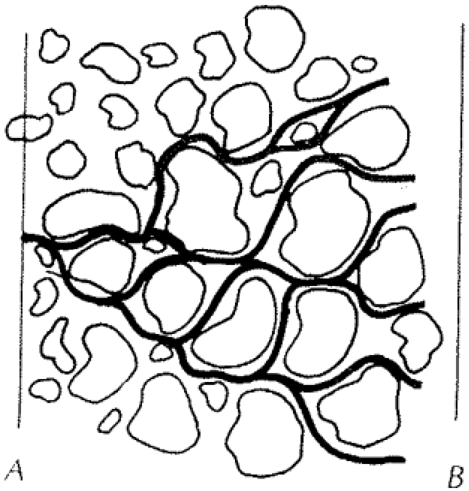
Darcy's law / specific discharge: $q = -K \frac{\partial h}{\partial x}$

Linear average velocity: $v = \frac{q}{n_e}$

Specific discharge (q); Effective porosity (n_e)

Velocity of a conservative tracer.

Advection Dispersion Equation (ADE) in 1D



Dispersion

Diff./ Disp. Flux (Fick's 1st Law)

$$J_{diff} = -n_e D_p \frac{\partial c_w}{\partial x}$$ 1D

$$J_{D_{eff}} = -n_e D_L \frac{\partial c_w}{\partial x}$$ 3D

Think about:

Why negative sign?

Hydrodynamic Dispersion (hydromechanical spreading)

Including diffusion:

$$D_L = \alpha_L v + D_p$$

$$D_T = \alpha_T v + D_p$$

Pore diffusion coefficient

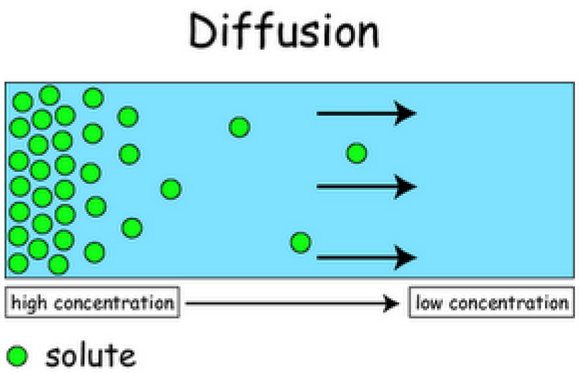
Longitudinal / transverse dispersivity

Note: Dispersivity values are often scale dependent!

$$\alpha_L = 0.83(logL)^{2.414}$$ Xu and Eckstein (1995)

Xu and Eckstein (1995) Use of weighted least-squares method in evaluation of the relationship between dispersivity and field scale. Ground Water 33, no. 6: 905-08

Advection Dispersion Equation (ADE) in 1D



Solute transport is from the left to the right; movement of the solutes is due to the concentration gradient (dC/dx).

Diffusion (Molecular spreading)

- In shallow aquifers, it is typically orders of magnitudes lower than dispersion
- Does not dependent on flow
- It is largely temperature dependent.

Putting advection and dispersion into Eq. (2) then:

The governing equation for non-sorbing solute transport with homogeneous coefficients is:

$$\frac{\partial c}{\partial t} + v\frac{\partial c}{\partial x} - D_{eff}\frac{\partial^2 c}{\partial x^2} = r$$

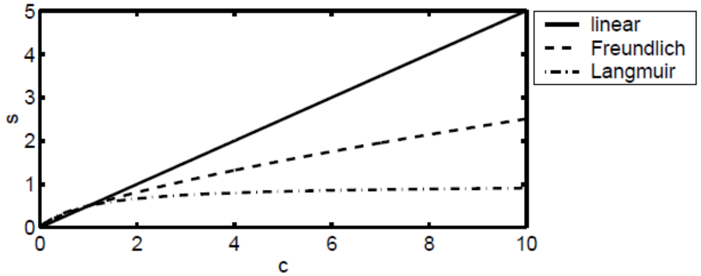
Diffusion of ions in sea water and in deep-sea sediments 705

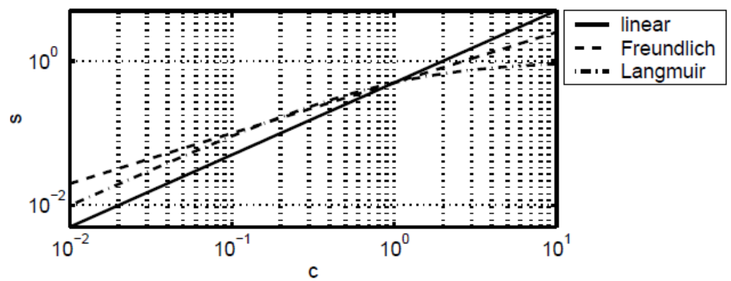
Table 1. Tracer and self-diffusion coefficients of ions at infinite dilution

Cation	D_j^0 (10^{-6} cm²/sec) 0°C	18°C	25°C	Anion	D_j^0 (10^{-6} cm²/sec) 0°C	18°C	25°C
H^+	56·1	81·7	93·1	OH^-	25·6	44·9	52·7
Li^+	4·72	8·69	10·3	F^-	—	12·1	14·6
Na^+	6·27	11·3	13·3	Cl^-	10·1	17·1	20·3
K^+	9·86	16·7	19·6	Br^-	10·5	17·6	20·1
Rb^+	10·6	17·6	20·6	I^-	10·3	17·2	20·0
Cs^+	10·6	17·7	20·7	IO_3^-	5·05	8·79	10·6
NH_4^+	9·80	16·8	19·8	HS^-	9·75	14·8	17·3
Ag^+	8·50	14·0	16·6	S^{2-}	—	6·95	—
Tl^+	10·6	17·0	20·1	HSO_4^-	—	—	13·3
$Cu(OH)^+$	—	—	8·30	SO_4^{2-}	5·00	8·90	10·7
$Zn(OH)^+$	—	—	8·54	SeO_4^{2-}	4·14	8·45	9·46
Be^{2+}	—	3·64	5·85	NO_2^-	—	15·3	19·1
Mg^{2+}	3·56	5·94	7·05	NO_3^-	9·78	16·1	19·0
Ca^{2+}	3·73	6·73	7·93	HCO_3^-	—	—	11·8
Sr^{2+}	3·72	6·70	7·94	CO_3^{2-}	4·39	7·80	9·55
Ba^{2+}	4·04	7·13	8·48	$H_2PO_4^-$	—	7·15	8·46
Ra^{2+}	4·02	7·45	8·89	HPO_4^{2-}	—	—	7·34

Li Yuan-Hui, Sandra Gregory (1974) Diffusion of ions in sea water and in deep-sea sediments. *Geochimica et Cosmochimica Acta, Volume 38, Issue 5, 703-714*

Sorption and Decay





- Soption isotherm describes the distribution of a particular chemical component in the aqueous and solid phase.
- It is called an isotherm because it is measured at a constant temperature.
- Because of different property of the solid phase and sorption component, the isotherm behaves differently and can be described by different mathematical equations.
- Another important behavior of the chemical component is the decay process. Simplest decay process can be described as the first-order decay.
- The radioactive decay exactly follows this decay process.

Henry $$s = K_d c_w$$

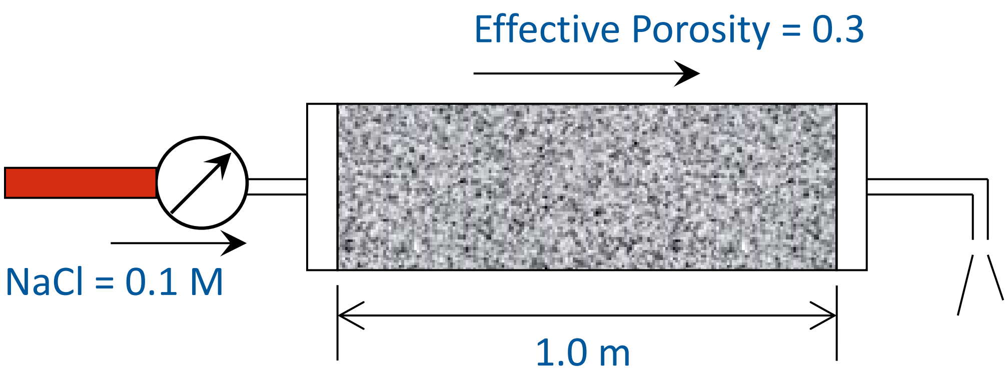
Freundlich $$s = K c_w^n$$

Langmuir $$s = \frac{s_{max} c_w}{c_w + K}$$

1st-order decay $$\frac{\partial c}{\partial t} = -\lambda c_w$$ or $$\frac{\partial c}{\partial t} = -R \lambda c_w$$

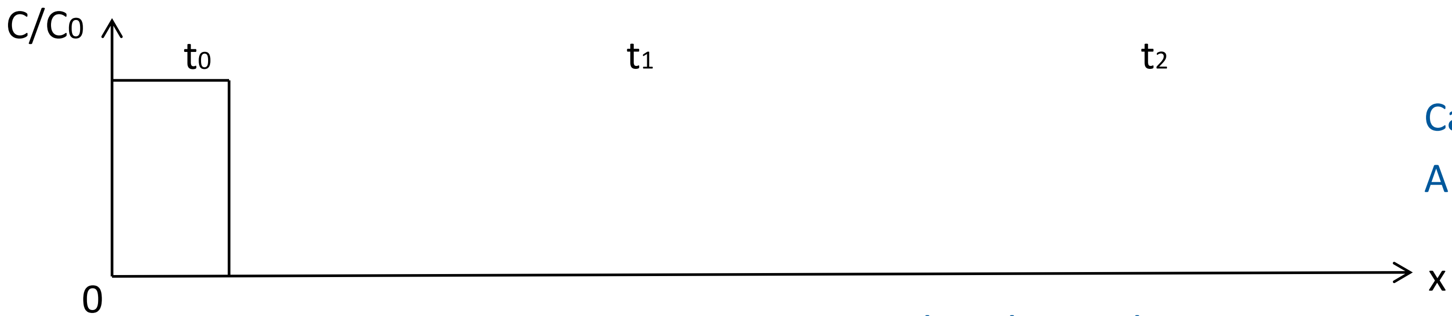
Derivation of the advection dispersion-reaction equation for a sorbing compound and linear decay from first principles

Exercise: Draw qualitatively



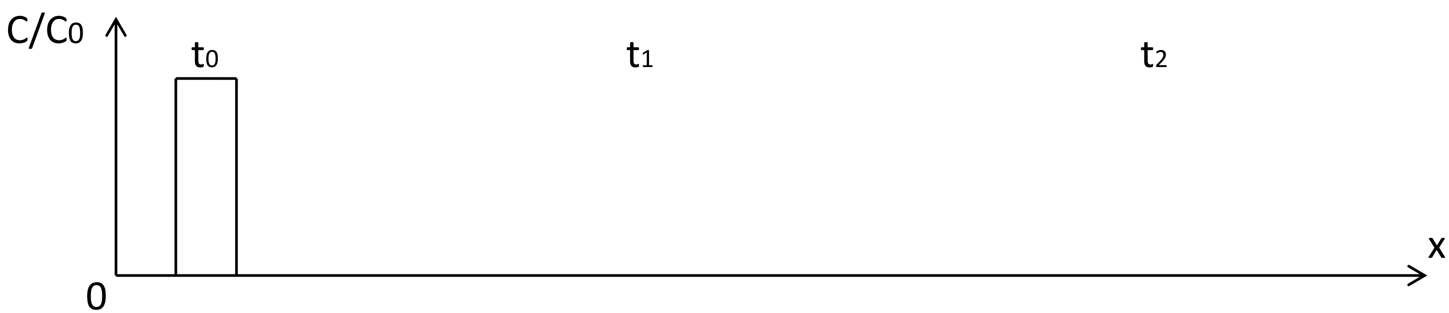
Cross-sectional area:
0.001 m2

Q = 0.003 m3/day



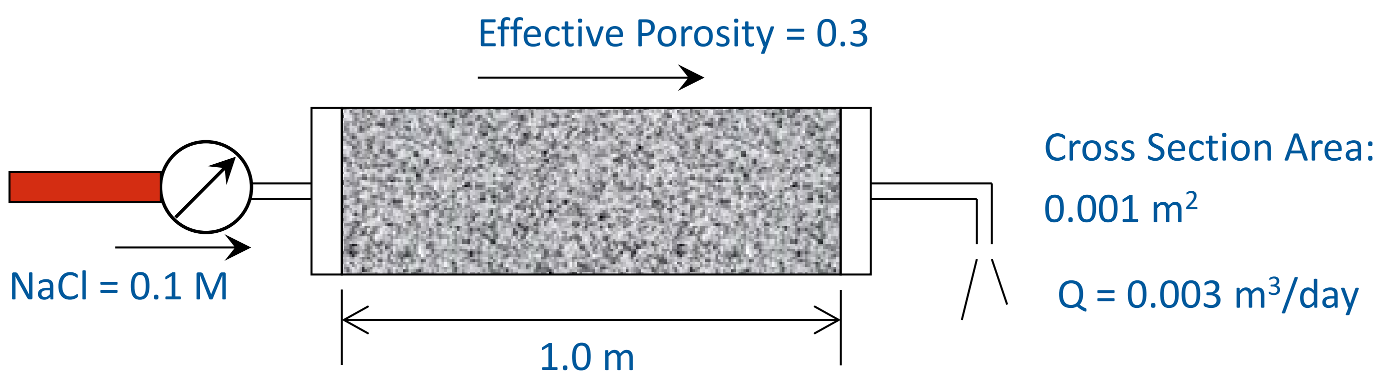
Case (1)
A continuous source

Conc. over space at t1=1d and t2=2d



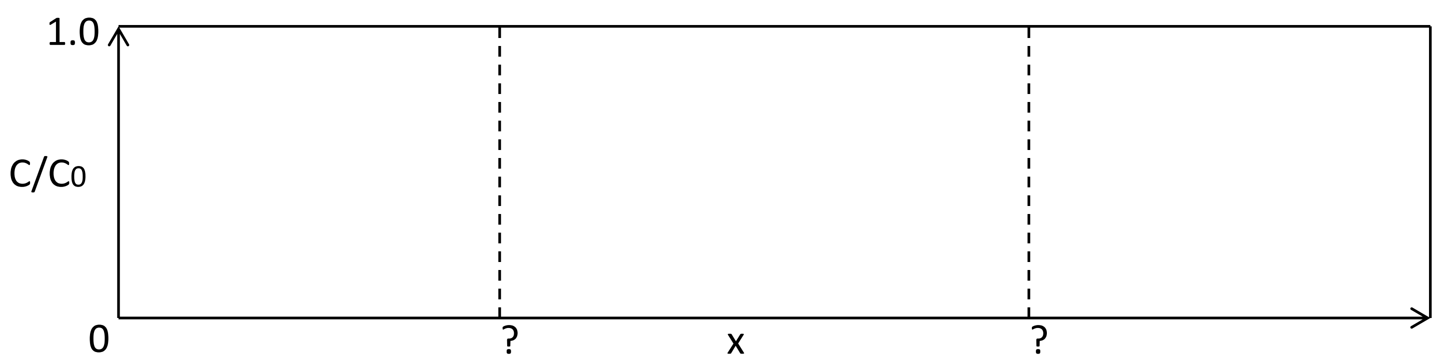
Case (2)
A shock source

Exercise



Case (3): Assuming soil grain density is 2650 kg/m^3 and Kd = 2.0 mL/g, compute the retardation factor R? How does this retardation effect influence the concentration profile?

Case (4): What if there is a 1st order decay on the transported contaminant with ($\lambda = 0.7$ 1/day)



HELMHOLTZ
Centre for Environmental Research

Modelling Reactive Transport Process with OpenGeoSys

with Decay-Chain Benchmark as an Example

Dr. Philipp Selzer / Dr. Haibing Shao

philipp.selzer@ufz.de

Helmholtz Centre for Environmental Research - UFZ, Leipzig, Germany

Prepared for the Lecture "**Modellierung von Hydrosystemen**"

in TU Dresden on 30.06.2023

Overview

- Overview of Reactive Transport Process Features
- The Decay-Chain Benchmark
- Exercise: Set up the Decay-Chain prj File
- Simulation and Visualization

Overview

- Overview of Reactive Transport Process
- The Decay-Chain Benchmark
- Exercise: Set up the Decay-Chain prj File
- Simulation and Visualization

How is Reactive Transport simulated by the Component Transport Process?

The component transport process can handle

- Fully saturated condition
- Variable-density flow
- Component transport with advection, dispersion, sorption and decay
- Reactive transport

There are two sets of process variables

- Pressure
- Concentration of each component

ComponentTransport

$$\frac{\partial(\phi\rho)}{\partial t} + \nabla\cdot(\mathbf{q}\rho) + Q_p = 0$$

$$\mathbf{q} = -\frac{\mathbf{k}}{\mu}(\nabla p - \rho\mathbf{g})$$

$$\frac{\partial(\phi R c_\alpha)}{\partial t} + \nabla\cdot(\mathbf{q}c_\alpha - \mathbf{D}\nabla c_\alpha) + Q_{c_\alpha} + \phi\lambda R c_\alpha = 0$$

$$D = (\phi D_p + \beta_T\|\mathbf{q}\|)\mathbf{I} + (\beta_L - \beta_T)\frac{\mathbf{q}\mathbf{q}^T}{\|\mathbf{q}\|}$$

$$\lambda = ln2/t_{1/2}$$

$$R = 1 + \rho_b K_D/\phi$$

Modelling Reactive Transport Process with different algorithms

OP – Operator Splitting

Transport stage

$$\begin{bmatrix} K_{c1c1} & & & \\ & K_{c2c2} & & \\ & & K_{c3c3} & \\ & & & K_{c4c4} \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}$$

Reaction stage

Call Phreeqc to compute $\mathbf{R}^{min}$

$$\begin{bmatrix} M_{c1c1} & & & \\ & M_{c2c2} & & \\ & & M_{c3c3} & \\ & & & M_{c4c4} \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}$$

GIA - Global Implicit Approach

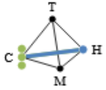
$$\begin{bmatrix} K_{c1c1} & \color{red}{K_{c1c2}} & & \\ & K_{c2c2} & \color{red}{K_{c2c3}} & \\ & & K_{c3c3} & \color{red}{K_{c3c4}} \\ & & & K_{c4c4} \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}$$

```
<chemical_system chemical solver = "SelfContained">
  <chemical_reactions>
    <chemical_reaction>
      <!-- 0 = -1 [Cm-247] + 1 [Am-243] -->
      <stoichiometric_coefficients>-1 1 0 0 0 0</stoichiometric_coefficients>
      <!-- t1_half_life = 1.56e7;
           LOG(2) / t1_half_life / 3.1536e7 -->
      <rate_constant>1.4089456993390242e-15</rate_constant>
    </chemical_reaction>
  </chemical_reactions>
</chemical_system>
```

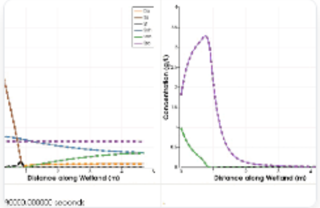
Available benchmarks with Reactive Transport Process

Find all RTP benchmarks here, along with the descriptions: https://www.opengeosys.org/docs/benchmarks/reactive-transport/

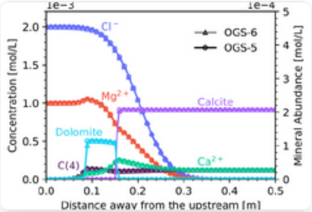
Docs › Processes › Reactive Transport



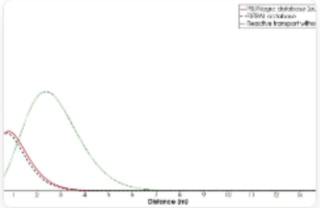
Reactive Transport



Complex kinetic reaction network

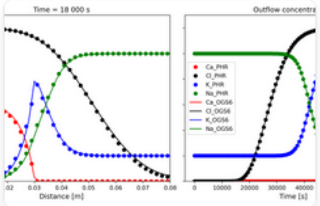


Precipitation/dissolution equilibrium reactions in a saturated column

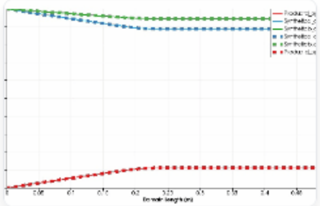


Sorption of U(VI) in porous media

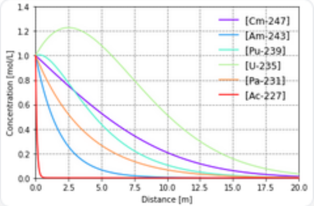
Tracer diffusion in a thermal gradient



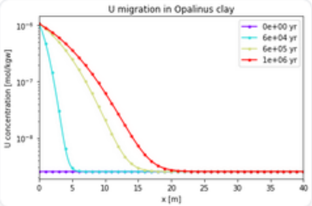
Transport and Cation Exchange



Solute transport including kinetic reaction



Decay-chain problem

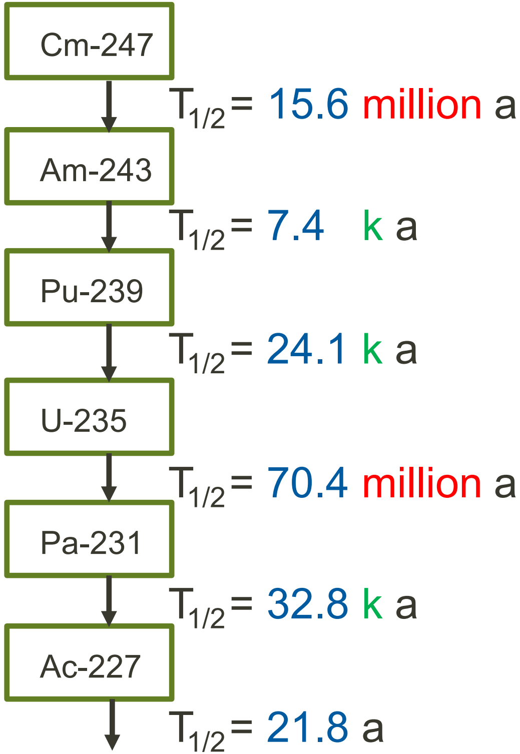


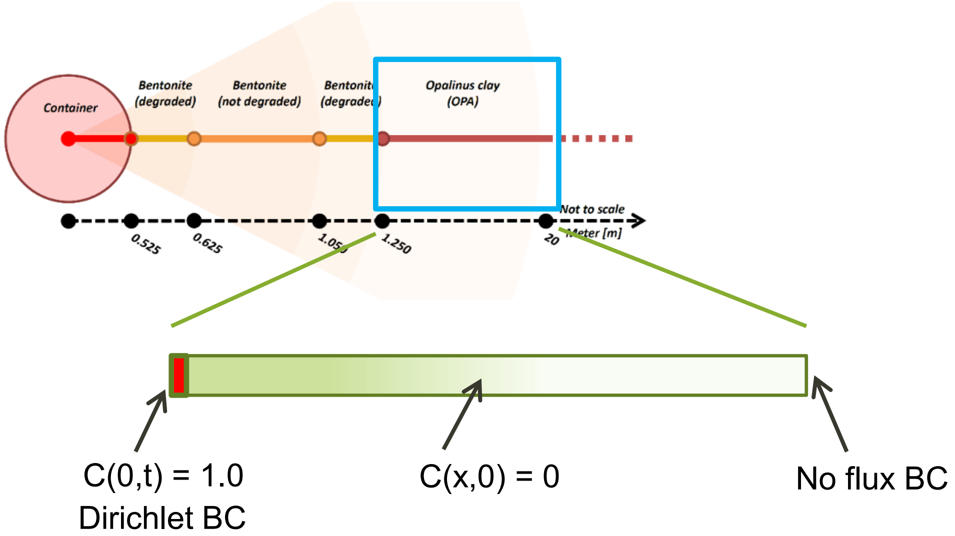
Radionuclides migration in Opalinus clay

Overview

- Overview of Reactive Transport Process Features
- **The Decay-Chain Benchmark**
- Exercise: Set up the Decay-Chain prj File
- Simulation and Visualization

The Decay-Chain Benchmark





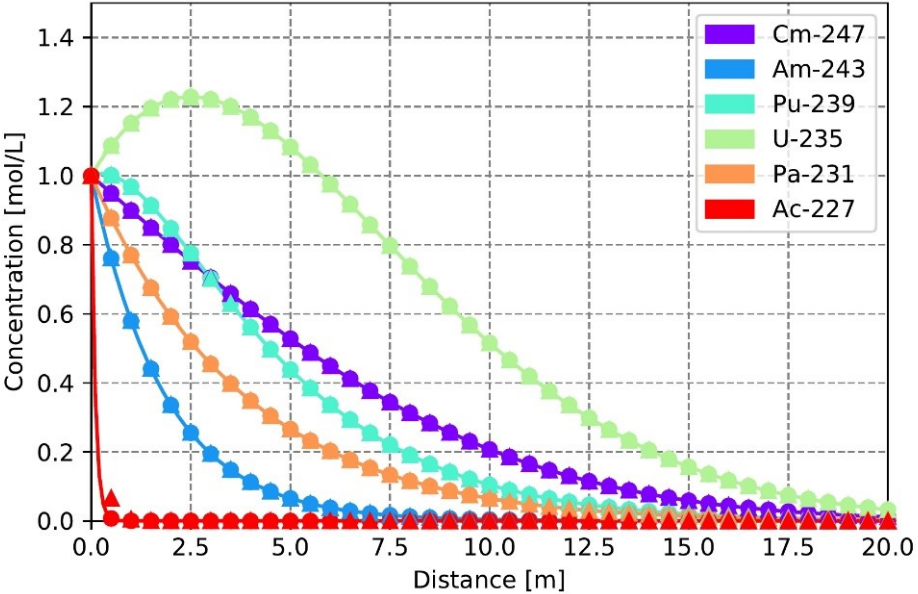
- 1D Domain [200 m × 10^6 y]
- Only diffusion + decay is considerred
- Uniform transport properties of OPA

Porosity	0.12
Pore diffusion coefficient	1e-11

Source: Dr. Christoph Behrens (BGE)

The Decay-Chain Benchmark

After 10k years, the simulated results as follows, also verified against the analytical solution



- In this case each component has the same pore diffusion coefficient, i.e some travels faster than others.
- U-235 is the slowest decaying component in this chain, therefore its concentration will accumulate over time and get more than 1.0
- Decay of Cm-247 is also slow, therefore it diffuses the second far
- Ac-227 is the fastest decaying component (only 23 years of half-life), therefore it can barely travel some distance

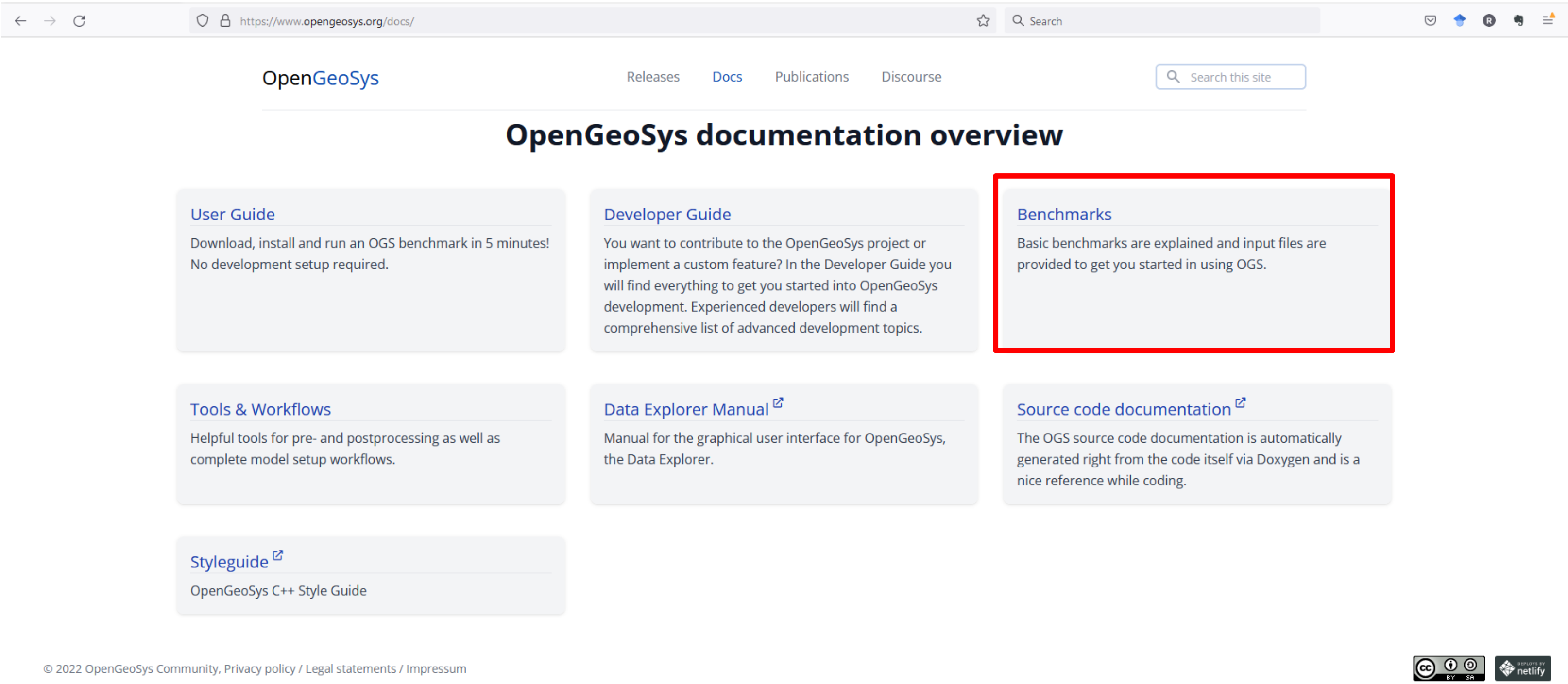
Analytical Solution: Sun, Y., Petersen, J. N., & Clement, T. P. (1999). *Journal of contaminant hydrology*, *35*(4), 429-440.

Overview

- Overview of Reactive Transport Process Features
- The Decay-Chain Benchmark
- **Exercise: Set up the Decay-Chain prj File**
- Simulation and Visualization

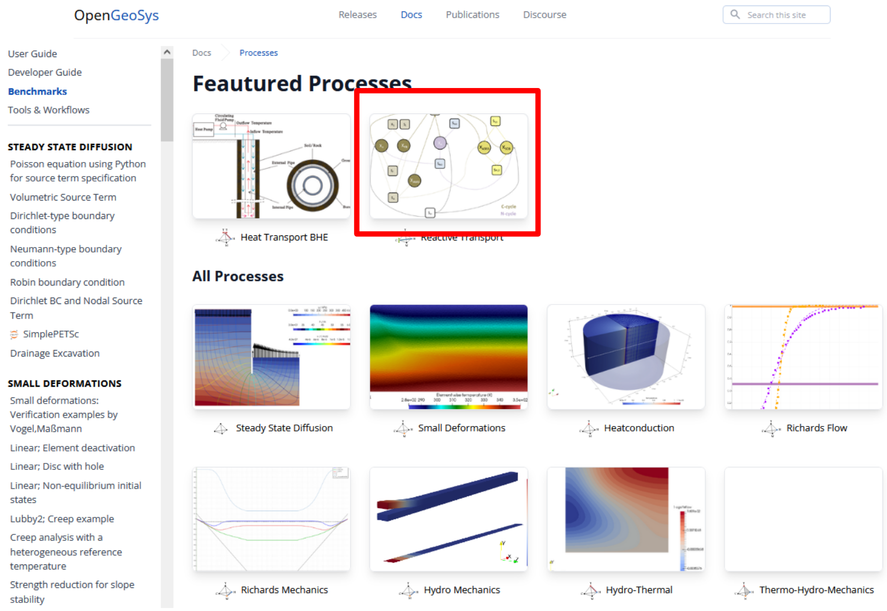
Exercise: Set up the Decay-Chain prj File

https://www.opengeosys.org/docs/



Exercise: Set up the Decay-Chain prj File

https://www.opengeosys.org/docs/benchmarks/reactive-transport/decaychain/

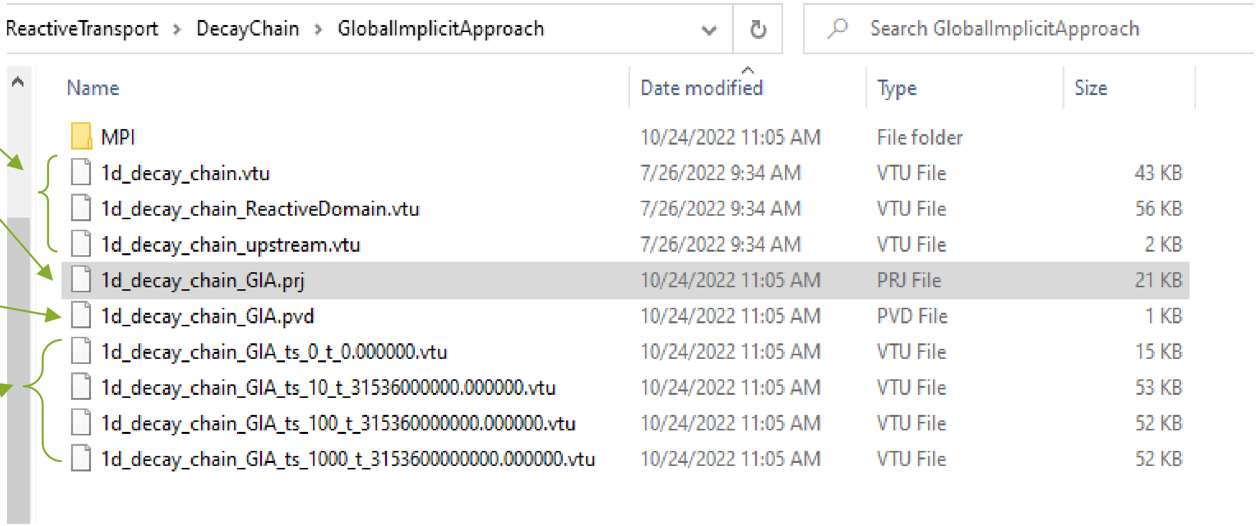


- **This exercise is based on this benchmark**

Exercise: Set up the Decay-Chain prj File

In the source code folder
\Tests\Data\Parabolic\ComponentTransport\ReactiveTransport\DecayChain\GlobalImplicitApproach

- **3 mesh files with *.vtu**
- ***.prj project configuration file**
- ***.PVD is the Paraview index file referring to all result files**
- **4 *.vtu files containing the simulation results from time step #0, #10, #100 and #1000**



Exercise: Set up the Decay-Chain prj File

- The prj file is the model configuration file in XML format
- You can expand / collapse each section by clicking on the + and – symbol
- The order of the section does not matter
- When error pops up while reading the configuration, messages will be recorded in the log file or on screen

1d_decay_chain_GIA_TODO.prj

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<OpenGeoSysProject>
    <meshes>
    <processes>
    <media>
    <time_loop>
    <chemical_system chemical_solver="SelfContained">
    <parameters>
    <process_variables>
    <nonlinear_solvers>
    <linear_solvers>
    <test_definition>
</OpenGeoSysProject>
```

- The key words are always surrounded by the “Klammer” structure
- <!-- … --> means comments, which will not be read into the program

```xml
<property>
    <name>pore_diffusion</name>
    <type>Constant</type>
    <value><!--TODO #4 Set this to 1e-11, unit is m2/s--></value>
</property>
```

- You need to change the content when seeing the TODOs
- There are 10 TODOs in total

Exercise: Set up the Decay-Chain prj File :: Mesh and Processes

1d_decay_chain_GIA_TODO.prj

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<OpenGeoSysProject>
    <meshes>
        <mesh>1d_decay_chain.vtu</mesh>
        <mesh>1d_decay_chain_upstream.vtu</mesh>
        <mesh><!-- TODO #1 Fill in here the file name for reactive domain --></mesh>
    </meshes>
    <processes>
        <process>
            <name>hc</name>
            <type><!-- TODO #2 The process type is ComponentTransport --></type>
            <integration_order>2</integration_order>
            <process_variables>
                <concentration><!-- TODO #3 Type here component name [Cm-247] -->
                </concentration>
                <concentration><!-- TODO #3 Type here component name [Am-243] -->
                </concentration>
                <concentration><!-- TODO #3 Type here component name [Pu-239] -->
                </concentration>
                <concentration><!-- TODO #3 Type here component name [U-235] -->
                </concentration>
                <concentration><!-- TODO #3 Type here component name [Pa-231] -->
                </concentration>
                <concentration><!-- TODO #3 Type here component name [Ac-227] -->
                </concentration>
                <pressure>pressure</pressure>
            </process_variables>
            <specific_body_force>0</specific_body_force>
        </process>
    </processes>
    <media>
```

There are 3 mesh files here

- The entire domain
- Upstream boundary nodes
- The reaction domain (TODO#1)

We have to set the process type to

- ComponentTransport (TODO#2)

The process variables include the concentration of each component (TODO#3)

- [Cm-247]
- [Am-243]
- [Pu-239]
- [U-235]
- [Pa-231]
- [Ac-227]

Exercise: Set up the Decay-Chain prj File :: Media Properties

We devide the media into 3 phases

- AqueousLiquid
  (the only phase in DecayChain)
- Solid
- Gas

For each component in Aq. phase

- Pore diffusion coefficient
- Retardation Factor set to 1 (no sorption)
- Decay Rate (set to 0 in this case)

Please repeat it for all components

1d_decay_chain_GIA_TODO.prj

```
<media>
    <medium id="0">
        <phases>
            <phase>
                <type>AqueousLiquid</type>
                <components>
                    <component>
                        <name>[Cm-247]</name>
                        <properties>
                            <property>
                                <name>pore_diffusion</name>
                                <type>Constant</type>
                                <value><!--TODO #4 Set this to 1e-11, unit is m2/s-->
                                </value>
                            </property>
                            <property>
                                <name>retardation_factor</name>
                                <type>Constant</type>
                                <value><!--TODO #5 Set this to 1, i.e. no retardation-->
                                </value>
                            </property>
                            <property>
                                <name>decay_rate</name>
                                <type>Constant</type>
                                <value>0</value>
                            </property>
                        </properties>
                    </component>
```

Exercise: Set up the Decay-Chain prj File :: Media Properties

1d_decay_chain_GIA_TODO.prj

```xml
                </components>
                <properties>
                    <property>
                        <name>density</name>
                        <type>Constant</type>
                        <value>1e3</value>
                    </property>
                    <property>
                        <name>viscosity</name>
                        <type>Constant</type>
                        <value>1e-3</value>
                    </property>
                </properties>
            </phase>
        </phases>
        <properties>
            <property>
                <name>permeability</name>
                <type>Parameter</type>
                <parameter_name>kappa</parameter_name>
            </property>
            <property>
                <name>porosity</name>
                <type>Parameter</type>
                <parameter_name>porosity</parameter_name>
            </property>
            <property>
                <name>longitudinal_dispersivity</name>
                <type>Constant</type>
                <value>0</value>
            </property>
            <property>
                <name>transversal_dispersivity</name>
                <type>Constant</type>
                <value>0</value>
            </property>
```

This section is the property of water

- Density (1000 kg/m3)
- Viscosity (0.001 Pa sec)

This section is about the porous media properties

- Permeability
  (refer to the kappa in the parameters list)
- Porosity
  (refer to the value in the parameters list)
- Longitudinal and transversal dispersivity
  (0 = no dispersion)

Exercise: Set up the parameters (TODO #6 and #7)

Give the following values in the parameters list (TODO #6 and #7)

- Set permeability to 1.157e-12 m2/s
- Set porosity to 0.12

```
<parameters>
    <parameter>
        <name>kappa</name>
        <type>Constant</type>
        <values><!--TODO #6 Set permeability kappa to 1.157e-12--></values>
    </parameter>
    <parameter>
        <name>porosity</name>
        <type>Constant</type>
        <value><!--TODO #7 Set porosity to 0.12--></value>
    </parameter>
```

```
<parameter>
    <name>TimeDependentDirichlet_right</name>
    <type>TimeDependentHeterogeneousParameter</type>
    <time_series>
        <pair>
            <time>0</time>
            <parameter_name>bc_right_ts1</parameter_name>
        </pair>
        <pair>
            <time>1180</time>
            <parameter_name>bc_right_ts59</parameter_name>
        </pair>
        <pair>
            <time>1200</time>
            <parameter_name>bc_right_ts60</parameter_name>
        </pair>
        <pair>
            <time>2000</time>
            <parameter_name>bc_right_ts100</parameter_name>
        </pair>
    </time_series>
</parameter>
```

A parameter can be defined as

- A constant value
- Time dependent
- Space dependent

https://www.opengeosys.org/docs/benchmarks/liquid-flow/time-dependent-heterogeneous-source-term-and-boundary-conditions/

Exercise: Set up the Decay-Chain prj File :: Types of Boundary Conditions

For solving the elliptic problem $k\,\Delta h = 0 \quad \text{in } \Omega$

- **Dirichlet → Given Value of Primary Variable**

$$h = g_D \quad \text{on } \Gamma_D,$$

https://www.opengeosys.org/docs/benchmarks/elliptic/elliptic-neumann/

```
<boundary_condition>
    <geometrical_set>square_1x1_geometry</geometrical_set>
    <geometry>bottom</geometry>
    <type>Dirichlet</type>
    <parameter>p_Dirichlet</parameter>
</boundary_condition>
```

- **Neumann → Given Flux Value**

$$k\frac{\partial h}{\partial n} = g_N \quad \text{on } \Gamma_N,$$

https://www.opengeosys.org/docs/benchmarks/elliptic/elliptic-neumann/

```
<boundary_condition>
    <geometrical_set>square_1x1_geometry</geometrical_set>
    <geometry>right</geometry>
    <type>Neumann</type>
    <parameter>p_neumann</parameter>
</boundary_condition>
```

- **Robin → Given Flux calculated by Primary Variable**

$$\frac{\partial h}{\partial n} = \alpha(h_0 - h(x)) \quad \text{on } \Gamma_R,$$

https://www.opengeosys.org/docs/benchmarks/elliptic/elliptic-robin/

```
<boundary_condition>
    <geometrical_set>line_1_geometry</geometrical_set>
    <geometry>left</geometry>
    <type>Robin</type>
    <alpha>alpha</alpha>
    <u_0>u_0</u_0>
</boundary_condition>
```

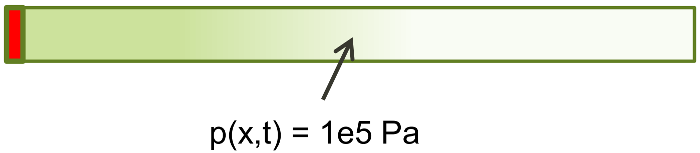
- **More complex BC via Python script**

https://www.opengeosys.org/docs/benchmarks/python-bc/elder/

```
<boundary_condition>
    <geometrical_set>elder</geometrical_set>
    <geometry>whole_domain_boundary</geometry>
    <type>Python</type>
    <bc_object>bc_c</bc_object>
</boundary_condition>
```

Exercise: Set up the Decay-Chain prj File :: Initial and Boundary Conditions

For pressure, We set both initial and boundary conditions to 1 bar (1e5 Pa),

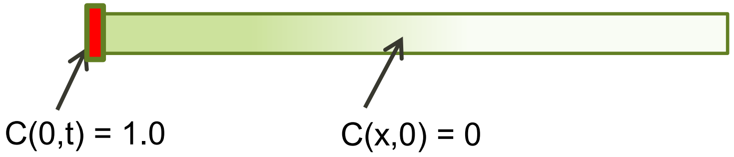



```xml
<process_variables>
    <process_variable>
        <name>pressure</name>
        <components>1</components>
        <order>1</order>
        <initial_condition>p0</initial_condition>
        <boundary_conditions>
            <boundary_condition>
                <mesh>1d_decay_chain_ReactiveDomain</mesh>
                <type>Dirichlet</type>
                <parameter>p0</parameter>
            </boundary_condition>
        </boundary_conditions>
    </process_variable>
    <process_variable>
```

- **Since there is no difference in pressure, there will be no advection. Hence, the transport is only controlled by diffusion and decay.**

Exercise: Set up the Decay-Chain prj File :: Initial and Boundary Conditions

For concentration, we set initial condition to 0 and boundary condition to 1,




```
<process_variable>
    <name>[Cm-247]</name>
    <components>1</components>
    <order>1</order>
    <initial_condition>c0_default</initial_condition>
    <boundary_conditions>
        <boundary_condition>
            <mesh>1d_decay_chain_upstream</mesh>
            <type>Dirichlet</type>
            <parameter>c_default</parameter>
        </boundary_condition>
    </boundary_conditions>
</process_variable>
```

- **Values set for each component**
- **Initial condition set to 0**
- **Boundary condition applied on a subset of the domain (just the boundary node in this case)**
- **Boundary condition set to 1**

Exercise: Set up the Decay-Chain prj File :: Decay Reactions

- The stoichiometry is filling in following each reactions
- The order of participating components are following the order of process variables

		Components					
		Cm-247	Am-243	Pu-239	U-235	Pa-231	Ac-227
Reactions	0 = -1 [Cm-247] + 1 [Am-243]	-1	1	0	0	0	0
	0 = -1 [Am-243] + 1 [Pu-239]	0	-1	1	0	0	0
	0 = -1 [Pu-239] + 1 [U-235]	0	0	-1	1	0	0
	0 = -1 [U-235] + 1 [Pa-231]	0	0	0	-1	1	0
	0 = -1 [Pa-231] + 1 [Ac-227]	0	0	0	0	-1	1
	0 = -1 [Ac-227] + 1 [n]	0	0	0	0	0	-1

```
<chemical_reactions>
    <chemical_reaction>
        <!-- 0 = -1 [Cm-247] + 1 [Am-243] -->
        <stoichiometric_coefficients>-1 1 0 0 0 0
        </stoichiometric_coefficients>
        <reaction_type>FirstOrderReaction</reaction_type>
        <!-- t1_half_life = 1.56e7 years;
             LOG(2) / t1_half_life / 3.1536e7 secs-->
        <first_order_rate_constant>1.4089456993390242e-15
        </first_order_rate_constant>
    </chemical_reaction>
```

- Rate of reaction is first order type

$$Rate = \frac{d[C]}{dt} = k[C]$$

- Rate constant is according to the half-life of this radionuclide

Exercise: Set up the Decay-Chain prj File :: Decay Reactions (TODO #8 and TODO #9)

- **What is the stoichiometric coefficients of the last reaction?**
- **How much is the reaction rate constant for this reaction?**

		Components					
		Cm-247	Am-243	Pu-239	U-235	Pa-231	Ac-227
Reactions	0 = -1 [Cm-247] + 1 [Am-243]	-1	1	0	0	0	0
	0 = -1 [Am-243] + 1 [Pu-239]	0	-1	1	0	0	0
	0 = -1 [Pu-239] + 1 [U-235]	0	0	-1	1	0	0
	0 = -1 [U-235] + 1 [Pa-231]	0	0	0	-1	1	0
	0 = -1 [Pa-231] + 1 [Ac-227]	0	0	0	0	-1	1
	0 = -1 [Ac-227] + 1 [n]	0	0	0	0	0	-1

```
        <chemical_reaction>
            <!-- 0 = -1 [Ac-227] + 1 [n] -->
            <stoichiometric_coefficients><!--TODO #8 Define the stoichiometric vector
            of this reaction--></stoichiometric_coefficients>
            <reaction_type>FirstOrderReaction</reaction_type>
            <!-- t6_half_life = 21.773 years;
                 k6 = LOG(2) / t6_half_life / 3.1536e7 secs-->
            <first_order_rate_constant><!--TODO #9 Define the first-order decay rate
            constant of this reaction--></first_order_rate_constant>
        </chemical_reaction>
    </chemical_reactions>
```

Exercise: Set up the Decay-Chain prj File :: Time Stepping Scheme

```xml
<time_loop>
    <processes>
        <process ref="hc">
            <nonlinear_solver>basic_picard</nonlinear_solver>
            <convergence_criterion>
                <type>PerComponentDeltaX</type>
                <norm_type>NORM2</norm_type>
                <reltols>1e-14 1e-14 1e-14 1e-14 1e-14 1e-14 1e-14</reltols>
            </convergence_criterion>
            <time_discretization>
                <type>BackwardEuler</type>
            </time_discretization>
            <time_stepping>
                <type>FixedTimeStepping</type>
                <t_initial>0.0</t_initial>
                <t_end>3.1536e12</t_end>
                <timesteps>
                    <pair>
                        <repeat>1000</repeat>
                        <delta_t>3.1536e9</delta_t>
                    </pair>
                </timesteps>
            </time_stepping>
        </process>
    </processes>
```

- **A FixedTimeStepping scheme is set here**
- **Other stepping schemes are:**
- **Each step is set to be 100 years**
- **Repeating 1000 times gives 100 k years**

Exercise: Linear Solver

The linear solver is internally calling one of the following external lib:

- LIS solver
- Eigen solve (default)
- PETSC solver

The following sparse linear solvers in Eigen library has been included:

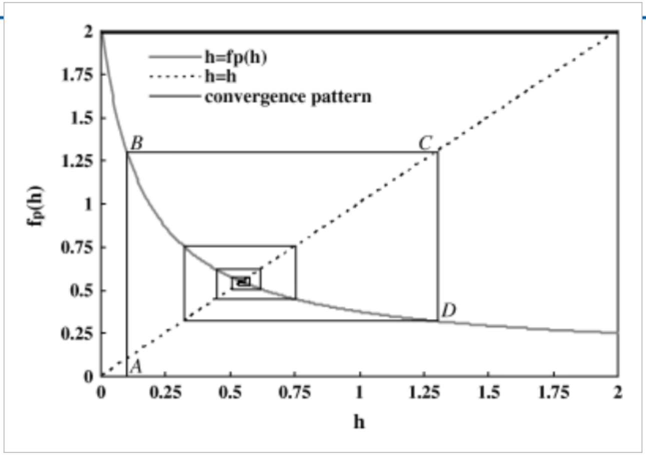
- CG
- BiCGSTAB
- IDRS
- IDRSTABL
- SparseLU
- PardisoLU
- GMRES

```xml
<linear_solvers>
    <linear_solver>
        <name>general_linear_solver</name>
        <lis>-i cg -p jacobi -tol 1e-16 -maxiter 20000</lis>
        <eigen>
            <solver_type>BiCGSTAB</solver_type>
            <precon_type>ILUT</precon_type>
            <max_iteration_step>10000</max_iteration_step>
            <error_tolerance>1e-14</error_tolerance>
        </eigen>
        <petsc>
            <prefix>hc</prefix>
            <parameters>-hc_ksp_type bcgs -hc_pc_type bjacobi -hc_ksp_rtol 1e-8
            -hc_ksp_max_it 20000</parameters>
        </petsc>
    </linear_solver>
</linear_solvers>
```

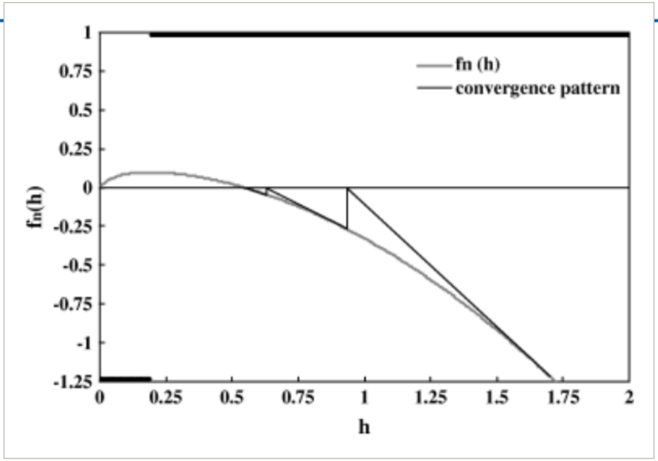
- In Eigen solver, one can choose different solver type and also preconditioners
- PETSC solver is prepared for parallel computing with MPI library (distributed memory type)
- Another choice of linear solve is the PADISO solver provided by Intel MKL lib

Exercise: Non-Linear Solvers

PICARD



NEWTON



Source: Mehl (2006) doi: 10.1111/j.1745-6584.2006.00207.x

```
<nonlinear_solvers>
    <nonlinear_solver>
        <name>basic_picard</name>
        <type>Picard</type>
        <max_iter>10</max_iter>
        <linear_solver>general_linear_solver</linear_solver>
    </nonlinear_solver>
</nonlinear_solvers>
```

- **The maximum num of iterations is set to 10**

```
<nonlinear_solver>basic_picard</nonlinear_solver>
<convergence_criterion>
    <type>PerComponentDeltaX</type>
    <norm_type>NORM2</norm_type>
    <reltols>1e-14 1e-14 1e-14 1e-14 1e-14 1e-14 1e-14</reltols>
</convergence_criterion>
```

- **This is the relative tolerance for each component**

Exercise: Set up the Decay-Chain prj File :: Output Control (TODO #10)

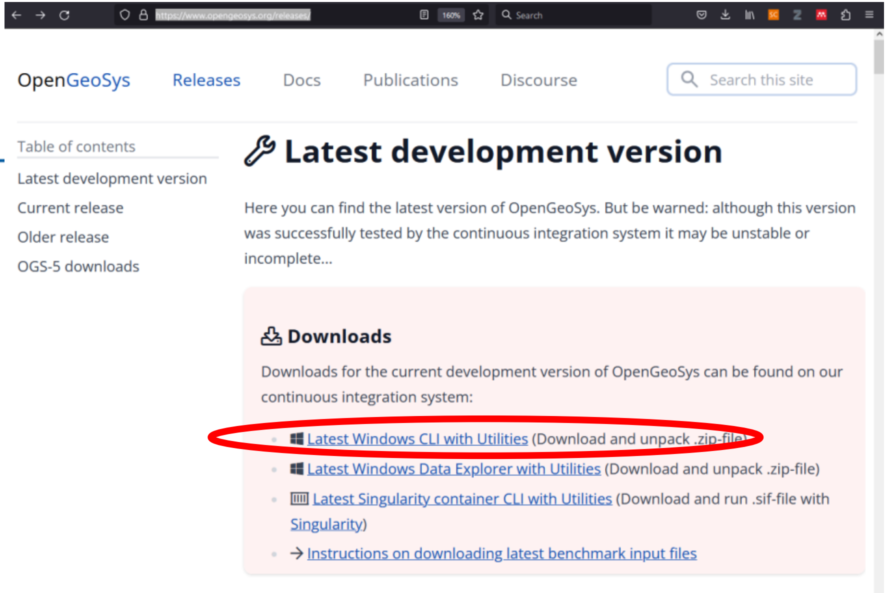
- The output file will always start with this prefix
- Followed by the time step and time info
- Here we define when we want to output
  - T = 0 is always printed out (initial condition, 0^{th} step)
  - Then output after 10 steps (10^{th} step)
  - Then after 90 steps (100^{th} step)
  - Then after 900 steps (1000^{th} step)
- Under the keyword <variables>, we specify all variables that will appear in the vtu result files

```xml
        <output>
            <type>VTK</type>
            <prefix>1d_decay_chain_GIA</prefix>
            <suffix>_ts_{:timestep}_t_{:time}</suffix>
            <timesteps>
                <pair>
                    <repeat>1</repeat>
                    <each_steps>10</each_steps>
                </pair>
                <pair>
                    <repeat>1</repeat>
                    <each_steps><!--TODO #10 We want the
                     100-th step to be printed out--></each_steps>
                </pair>
                <pair>
                    <repeat>1</repeat>
                    <each_steps><!--TODO #10 We want the
                     1000-th step to be printed out--></each_steps>
                </pair>
            </timesteps>
            <variables>
                <variable>[Cm-247]</variable>
                <variable>[Am-243]</variable>
                <variable>[Pu-239]</variable>
                <variable>[U-235]</variable>
                <variable>[Pa-231]</variable>
                <variable>[Ac-227]</variable>
                <variable>pressure</variable>
            </variables>
        </output>
    </time_loop>
```

Overview

- Overview of Reactive Transport Process Features
- The Decay-Chain Benchmark
- Exercise: Set up the Decay-Chain prj File
- **Simulation and Visualization**

Different Approaches of Simulation



- **Download OGS software package from the web**

  https://www.opengeosys.org/releases/

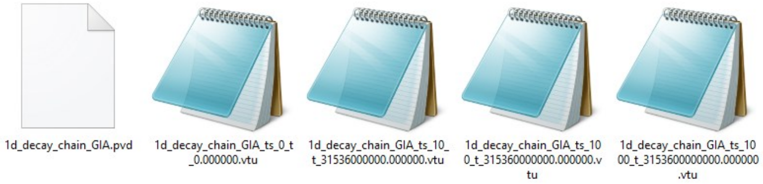
- **Unzip the package into a folder called "ogs"**

- **Run simulation**

  Syntax: ogs.exe   path_to_the_project_file

```
C:\work\CloudStation\2023_conferences_and_trips\20230630_TU_Dresden\DecayChain_DONE>..\ogs\bin\ogs.exe 1d_decay_chain_GIA.prj
```

- **When simulation starts, time-series data files will be generated (.pvd file + .vtu files)**



Different Approaches of Simulation

- Display output messages

```
C:\Windows\System32\cmd.exe
info: [time] Iteration #1 took 0.0438331 s.
info: [time] Assembly took 0.0194844 s.
info: [time] Applying Dirichlet BCs took 0.0014799 s.
info: ------------------------------------------------------------------
info: *** Eigen solver computation
info: -> solve with Eigen iterative linear solver BiCGSTAB (precon ILUT)
info:    iteration: 0/10000
info:    residual: 3.322583e-24

info: ------------------------------------------------------------------
info: [time] Linear solver took 0.0128669 s.
info: Convergence criterion, component 0: |dx|=0.0000e+00, |x|=2.4515e+06, |dx|/|x|=0.0000e+00
info: Convergence criterion, component 1: |dx|=0.0000e+00, |x|=1.3593e+01, |dx|/|x|=0.0000e+00
info: Convergence criterion, component 2: |dx|=0.0000e+00, |x|=9.3455e+00, |dx|/|x|=0.0000e+00
info: Convergence criterion, component 3: |dx|=0.0000e+00, |x|=1.3881e+01, |dx|/|x|=0.0000e+00
info: Convergence criterion, component 4: |dx|=0.0000e+00, |x|=1.9006e+01, |dx|/|x|=0.0000e+00
info: Convergence criterion, component 5: |dx|=0.0000e+00, |x|=1.1422e+01, |dx|/|x|=0.0000e+00
info: Convergence criterion, component 6: |dx|=0.0000e+00, |x|=3.0790e+00, |dx|/|x|=0.0000e+00
info: [time] Iteration #2 took 0.0381688 s.
info: [time] Solving process #0 took 0.0828339 s in time step #1000
info: [time] Time step #1000 took 0.0864828 s.
info: [time] Output of timestep 1000 took 0.0085149 s.
info: The whole computation of the time stepping took 1000 steps, in which
         the accepted steps are 1000, and the rejected steps are 0.

info: [time] Execution took 98.7813 s.
info: OGS terminated on 2023-06-29 14:33:51+0200.
```

Syntax: ogs.exe   path_to_the_project_file > log.txt

```
C:\work\CloudStation\2023_conferences_and_trips\20230630_TU_Dresden\DecayChain_DONE>..\ogs\bin\ogs.exe 1d_decay_chain_GIA.prj > log.txt
```

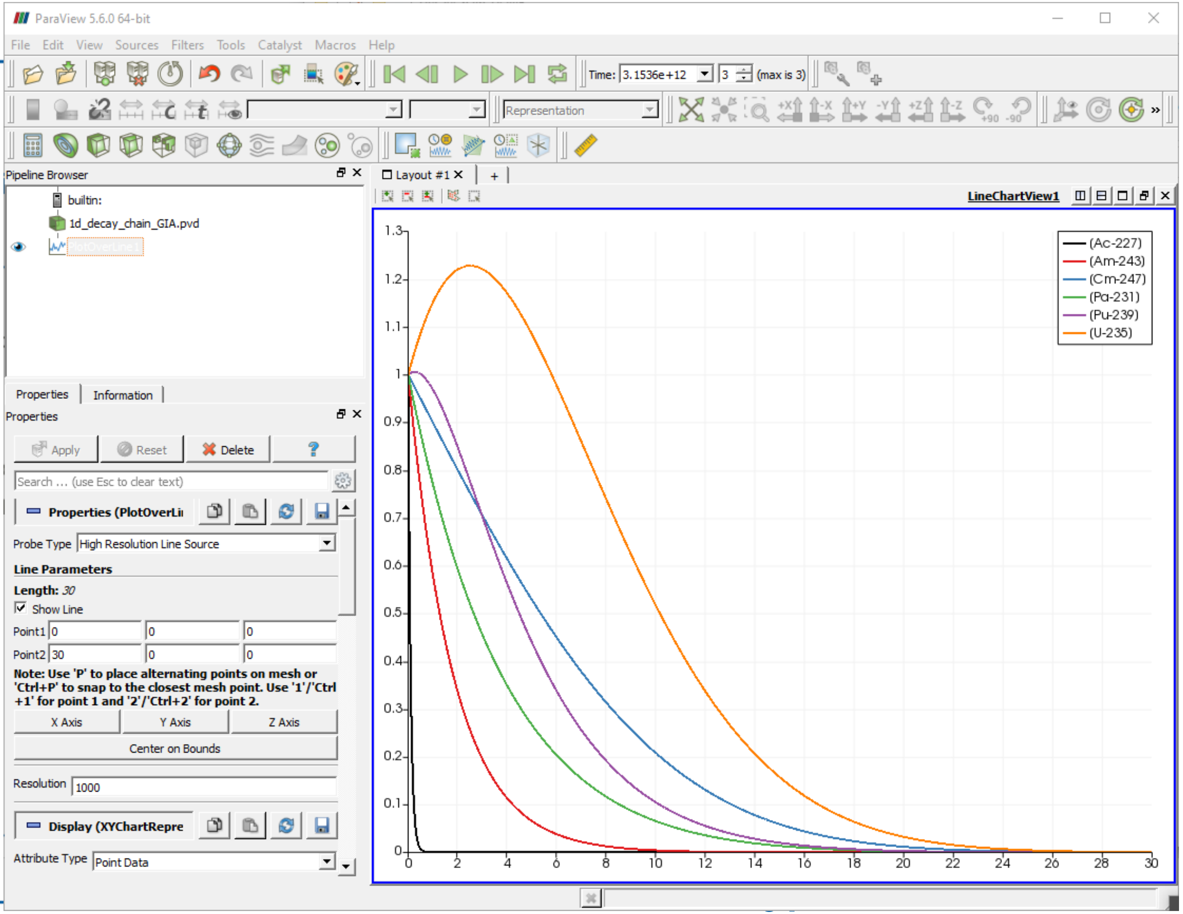
- Visualization

ParaView
Parallel Visualization Application

jupyter

Visualizing the simulation results

- Load the PVD file in Paraview
- Remember to “Apply” the View
- Add a “Plot over Line” Filter
- Just need to see from 0 to 30 m
- The results include from 0 to 100k years
- Choose which components to display (6 of them)
- Try to see the Spreadsheet View
- Try to export the data to CSV file



~The End~

Solute Transport in Hydrosystems II, Dresden, Germany

Some Analytical Solutions for Solute Transport in 1D

Applicant: **Dr. Philipp Selzer**[1]

[1] Department of Environmental Informatics, Helmholtz Centre for Environmental Research – UFZ, Leipzig, Germany

July 2, 2025

Advective-Dispersive Transport for a Point Source

$$\frac{\partial c}{\partial t} + v\frac{\partial c}{\partial x} - D_{eff}\frac{\partial^2 c}{\partial x} = -\lambda c$$

Initial condition:

$$c(t_0, x) = \frac{m}{An_e}\delta(x)$$

with

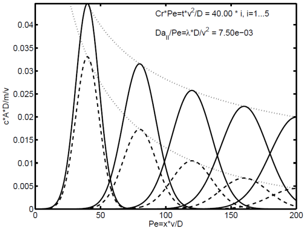
$m =$ "solute mass"

$A =$ "cross-sectional area perpendicular to x"

$\delta(x) =$ "Dirac delta function"

Boundary conditions:

$$\lim_{x\to\pm\infty} c(t,x) = 0$$



Gaussian distribution with mean: $x = vt$ and variance: $\sigma^2 = 2D_{eff}t$

$$c(x,t) = \frac{m}{An_e}\frac{1}{\sqrt{4\pi D_{eff}t}}\exp\left(-\frac{(x-vt)^2}{4D_{eff}t}\right)\exp\left(-\lambda t\right)$$

Characteristic times and dimensionless numbers

Table: Characteristic time scales in advective-dispersive-reactive transport

Process	**Symbol**	Definition
Advection	τ_{adv}	$\frac{x}{v}$
Dispersion	τ_{disp}	$\frac{x^2}{D_{eff}}$
1st-Order-Decay	τ_{reac}	$\frac{1}{\lambda}$

Table: Dimensionless numbers used in transport computations

Name	**Symbol**	**Meaning**	Definition
Peclét number	Pe	$\frac{\text{characteristic time of diffusion/dispersion}}{\text{characteristic time of advection}}$	$\frac{vx}{D_{eff}}$
Damköhler number I	Da_I	$\frac{\text{characteristic time of advection}}{\text{characteristic time of reaction}}$	$\frac{\lambda x}{v}$
Damköhler number II	Da_{II}	$\frac{\text{characteristic time of diffusion/dispersion}}{\text{characteristic time of reaction}}$	$\frac{\lambda x^2}{D_{eff}}$
Courant number II	Cr	$\frac{\text{real time}}{\text{characteristic time of advection}}$	$\frac{tv}{x}$
Neumann number II	Ne	$\frac{\text{real time}}{\text{characteristic time of diffusion/dispersion}}$	$\frac{tD_{eff}}{x^2} = \frac{Cr}{Pe}$

Be aware: If you use rates instead of characteristic times, the "meaning" is flipped, but the "Definition" stays the same

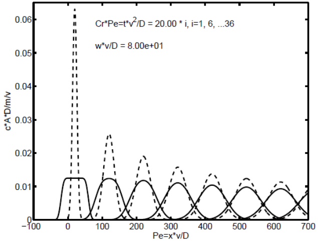
Advective-Dispersive Transport for a Rectangular Source

$$\frac{\partial c}{\partial t} + v\frac{\partial c}{\partial x} - D_{eff}\frac{\partial^2 c}{\partial x} = -\lambda c$$

$$\operatorname{erfc}(x) = 1 - \operatorname{erf}(x) = 1 - \frac{2}{\pi}\int_0^x \exp\left(-\xi^2\right) d\xi$$

Initial condition:

$$c(t_0, x) = \begin{cases} 0 & \text{for } x < -\frac{w}{2} \\ c_{ini} & \text{for } -\frac{w}{2} \leq x \leq \frac{w}{2} \\ 0 & \text{for } x > \frac{w}{2} \end{cases}$$



Boundary conditions:

$$\lim_{x \to \pm\infty} c(t, x) = 0$$

$$c(x, t) = \frac{c_{ini}}{2}\left(\operatorname{erf}\left(\frac{x + \frac{w}{2} - vt}{\sqrt{4D_{eff}t}}\right) - \operatorname{erf}\left(\frac{x - \frac{w}{2} - vt}{\sqrt{4D_{eff}t}}\right)\right)\exp\left(-\lambda t\right)$$

One-Dimensional Transport in a Semi-Infinite Domain I

$$\frac{\partial c}{\partial t} + v\frac{\partial c}{\partial x} - D_{eff}\frac{\partial^2 c}{\partial x} = -\lambda c$$

$\Rightarrow$ Constant-Concentration Boundary Condition (Ogata & Banks, 1961)

Initial condition:

$$c(t_0, x) = 0$$

Boundary conditions:

$$\lim_{x\to\infty} c(t, x) = 0$$

$$c(t, x_0) = c_0$$

$$c(x,t) = \frac{c_0}{2}\exp\left(\frac{xv}{2D_{eff}}\right)\left(\exp\left(-\frac{xv\gamma}{2D_{eff}}\right)\operatorname{erfc}\left(\frac{x - vt\gamma}{\sqrt{4D_{eff}t}}\right) + \exp\left(\frac{xv\gamma}{2D_{eff}}\right)\operatorname{erfc}\left(\frac{x + vt\gamma}{\sqrt{4D_{eff}t}}\right)\right)$$

with $\gamma = \sqrt{1 + 4\frac{\lambda D}{v^2}}$

One-Dimensional Transport in a Semi-Infinite Domain II

$$\frac{\partial c}{\partial t} + v\frac{\partial c}{\partial x} - D_{eff}\frac{\partial^2 c}{\partial x} = -\lambda c$$

$\Rightarrow$ Constant-Flux Boundary Condition (Kinzelbach, 1992)

Initial condition:

$$c(t_0, x) = 0$$

Boundary conditions:

$$\lim_{x\to\infty} c(t, x) = 0$$

$$J(t, x_0) = n_e\left(cv - D_{eff}\frac{\partial c}{\partial x}\right)_{x=0} = J_{in}(t)$$

$$c(x,t) = \frac{c_{in}}{2}\exp\left(\frac{xv}{2D_{eff}}\right)\left(\exp\left(-\frac{xv\gamma}{2D_{eff}}\right)\operatorname{erfc}\left(\frac{x - vt\gamma}{\sqrt{4D_{eff}t}}\right) - \exp\left(\frac{xv\gamma}{2D_{eff}}\right)\operatorname{erfc}\left(\frac{x + vt\gamma}{\sqrt{4D_{eff}t}}\right)\right)$$

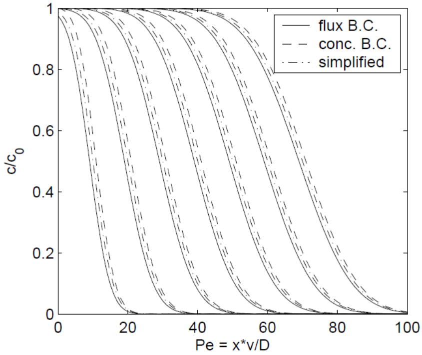
with $\gamma = \sqrt{1 + 4\frac{\lambda D}{v^2}}$

Comparison of Solutions

Solution for the Riemann problem with initial condition:

$$c(t_0, x) == c_{ini} H(-x) = \begin{cases} c_{ini} & \text{for } x < 0 \\ \frac{c_{ini}}{2} & \text{for } x = 0 \\ 0 & \text{for } x > 0 \end{cases}$$

$$c(x,t) = \frac{c_{ini}}{2} \operatorname{erfc}\left(\frac{x - vt}{\sqrt{4D_{eff}\,t}}\right) \exp\left(-\lambda t\right)$$



Exercises