
The CTRW-FEM Package, v3.2: A Practical User's Guide

A Finite Element Method numerical solution package to solve for non-Fickian transport in porous media

Rami Ben-Zvi¹, Shidong Jiang², Harvey Scher¹, Brian Berkowitz¹

1. Department of Earth and Planetary Sciences, Weizmann Institute of Science, Rehovot
7610001, Israel

2. Department of Mathematical Sciences, New Jersey Institute of Technology, Newark, NJ
07102 USA

Email addresses: rami.ben-zvi@weizmann.ac.il; shidong.jiang@njit.edu; harvey.scher@weizmann.ac.il;
brian.berkowitz@weizmann.ac.il

January 2019

-----LEGAL NOTICE-----

The "CTRW-FEM Package Software" and the "CTRW-FEM Package" are produced by Rami Ben-Zvi, Harvey Scher, Brian Berkowitz (Weizmann Institute of Science) and Shidong Jiang (New Jersey Institute of Technology).

Permission to use "CTRW-FEM Package software" is hereby granted to non-profit educational and research institutions, for educational and research purposes only, provided and as long this Notice appears.

Distribution of this package in whole or in part, in its current or in any modified form, is strictly forbidden.

THE CTRW-FEM PACKAGE SOFTWARE IS BEING DEVELOPED AS A TOOL FOR SCIENTIFIC RESEARCH. HENCE IT IS NOT PRESENTED AS ERROR FREE, ACCURATE, COMPLETE, OR USEFUL FOR ANY SPECIFIC APPLICATION, AND THE WEIZMANN INSTITUTE AND NEW JERSEY INSTITUTE OF TECHNOLOGY MAKE NO WARRANTY OR REPRESENTATION THERETO.

THE WEIZMANN INSTITUTE AND NEW JERSEY INSTITUTE OF TECHNOLOGY SHALL NOT BE LIABLE FOR ANY CLAIMS, DEMANDS, LIABILITIES, COSTS, LOSSES, DAMAGES OR EXPENSE OF WHATSOEVER KIND OR NATURE CAUSED TO OR SUFFERED BY ANY PERSON OR ENTITY THAT DIRECTLY OR INDIRECTLY ARISE OUT OR RESULT FROM THE USE OF THE CTRW-FEM SOFTWARE OR IN CONNECTION THEREOF.

Trademarks appear throughout this package and documentation without any trademark symbol; they are the property of their trademark owner. There is no intention of infringement; the usage is to the benefit of the trademark owner.

For usage by and for commercial entities please contact Brian Berkowitz.

If you publish work benefiting from this software package, please cite it as:

Rami Ben-Zvi, Shidong Jiang, Harvey Scher, and Brian Berkowitz
The CTRW-FEM Package v3.2: A practical user's guide,
<http://www.weizmann.ac.il/EPS/People/Brian/CTRW/software>

or using

Rami Ben-Zvi, Shidong Jiang, Harvey Scher, and Brian Berkowitz (2018).
Finite Element Method Solution of Non-Fickian Transport in Porous Media: The CTRW-FEM Package, *Groundwater*, doi:10.1111/gwat.12813

Contents

1.	Introduction.....	5
1.1	Software	5
1.2	Further notes.....	5
1.3	Practical advice	6
1.4	Transport Equation.....	6
1.5	Versions log.....	7
2.	Installation of the packages.....	7
2.1	FEM-1D: 1D solver.....	8
2.2	FEM-2D: 2D solver.....	8
2.3	MeshGenRect: 2D simple rectangular and uniform mesh generator	8
2.4	Darcy-2D: 2D Darcy solver	9
2.5	Optim: CTRW (1D) parameter calibration tool	9
3.	Input/Output (I/O).....	9
3.1	Input files (for both FEM-1D and FEM-2D packages).....	9
3.1.1	The main input file, finp.txt	9
3.1.2	The nodal data file, nodes.txt	10
3.1.3	The elements data file, elements.txt	10
3.1.4	The velocity data file, v.txt	11
3.1.5	The dispersion data file, d.txt	11
3.1.6	The BC data file, bcs.txt	11
3.1.7	The time-independent source data file, source.txt	11
3.1.8	The materials data file, materials.txt	11
3.2	Output files (for both FEM-1D and FEM-2D packages)	11
3.2.1	1D solver output.....	11
3.2.2	2D solver output.....	12
3.3	2D pre-processor for FEM-2D	12
3.3.1	The 2D pre-processor input file, meshinp.txt	12
3.4	Post-processors for FEM-1D and FEM-2D	14
3.4.1	The 1D post-processors.....	14
3.4.2	The 2D post-processor	14
3.4.3	Other tools.....	14
3.5	Examples	14
3.5.1	FEM-1D example.....	14
3.5.2	FEM-2D example.....	15
4.	Darcy equation solver	15
4.1	Theory	15

4.2	Input files.....	17
4.2.1	The main input file, finp.txt	17
4.2.2	The nodal data file, nodes.txt	17
4.2.3	The elements data file, elements.txt	17
4.2.4	The conductivity data file, d.txt	18
4.2.5	The BC data file, bcs.txt	18
4.3	Output files	18
4.4	2D post-processors	18
4.5	DARCY-2D example	18
5.	Optim: Parameter calibration.....	19
5.1	Sum-of-Pole MATLAB Function	19
5.2	Optim example	20
6.	Extensions	21
	References	22

1. Introduction

The Continuous Time Random Walk (CTRW) framework used to model non-Fickian (anomalous) transport in porous media can be implemented by the Finite Element Method (FEM). The theoretical background and development/implementation of the FEM solvers – specifically relevant to the CTRW-FEM package here – are described in [1] (1D conservative transport), [2] (2D conservative transport), and [3] (1D reactive transport). Background on CTRW theory, equations, and solution methods, using both partial differential equation (PDE) solutions and particle tracking (PT) methods, appear elsewhere on the CTRW website, as well as in references [1 – 3] and literature cited therein. *Please contact the authors directly for reprints of these references* (rami.ben-zvi@weizmann.ac.il; brian.berkowitz@weizmann.ac.il).

Here, ***it is assumed that the reader is familiar with the CTRW framework, and so only the practical aspects of these codes are described***: how to use them in terms of input/output (I/O) files and pre- and post-processing, to allow users to apply them to their needs.

The first part of this manual is devoted to these 1D and 2D chemical transport solvers, explaining their I/O file structures and, briefly, also relevant pre- and post- processing tools. A similar (and much simpler) methodology for solution of the 2D equations for (Darcy) fluid flow is then presented, where the velocity is calculated for a given porosity field in a 2D domain, with prescribed boundary conditions (BCs). After the theoretical derivation, the I/O files, pre- and post- processing (similar to the former part) are detailed. This velocity field can be used as input to the 2D transport solver. The last section of this manual deals with a CTRW parameter calibration tool that uses a given experimental breakthrough curve (BTC), runs the 1D transport solver, and modifies the CTRW parameters to provide a best fit to the experimental measurements.

More specifically, Section 1 summarizes the transport equations in a concise manner, including references to more detailed papers that focus specifically on CTRW-FEM applications. Section 2 gives step-by-step instructions for installing all of the packages. Section 2.5 includes the input and output of each package. Section 4 outlines a theoretical development of the FEM formulation for the 2D Darcy equation, followed by its I/O and an example. Finally, Section 5 describes the CTRW parameter calibration tool theoretically and practically (I/O and an example). Some possible extensions are suggested in Section 6.

1.1 Software

The codes are available as FORTRAN (f77 and f90) source codes and as executables (built with Intel Visual FORTRAN Composer XE 2011 Update 7 Integration for Microsoft Visual Studio on a 64-bit Windows 7 machine). Efforts were made to adhere to accepted standards and avoid using extensions, so that ***any*** FORTRAN suite should be appropriate. The source code is commented extensively and straightforward. No special libraries are needed, except the BLAS level 1 function DDOT (double-precision scalar product of two vectors). For completeness, the DDOT source is added to the package. All floating-point variables use 64 bits. Example cases and files for each of these parts are also included.

1.2 Further notes

The software has been tested on a personal computer with installation of Windows 7 – 64 bit, MS visual Studio 2010, Intel FORTRAN 2011 and MATLAB R2017a.

The FORTRAN codes use both f90 and f77 and comply with general standards; it is hoped that the codes will function on any FORTRAN installation.

The MS Visual Studio files (*.sln, *.vfproj, *.suo, *.u2d) have been included to assist with building the executables within MS Visual Studio. Other systems will have to create the relevant building tools.

In the examples presented here, the geometry and grid are rectangular, but there is no such a limitation in the code: it will accept *any* grid that is admissible (i.e., all elements have positive Jacobian determinant and the elements do not overlap each other). This was verified on a small cluster of arbitrarily-shaped quadrilateral elements built manually. Any mesh generator may be used to create grids, but a translating software should be built to convert it to the required input files formats.

The codes are written in both standard FORTRAN77 (*.f files) and standard FORTRAN90 (*.f90 files). If a user wishes to use C/C++ instead, the codes may be converted. Note that some automatic conversion tools are available, e.g.

f2c - a program to convert Fortran 77 to C code (developed by Bell Laboratories):
<http://www.netlib.org/f2c>

fable - Automatic Fortran (fixed-format) to C++ conversion: <http://cci.lbl.gov/fable>

Fortran 90 to C compiler: <http://www.ncsa.illinois.edu/People/mdewing/f90toC>

which may help. Even manual conversion should be straightforward (although laborious).

1.3 Practical advice

Having downloaded and installed the zipped files (as detailed in Section 2), the user is advised to start with the 1D solver, run and modify the example. Then, if desired, proceed to the 2D solver, optionally with its mesh generator, run the example and modify it (and also practice using the freely available general FEM post-processor GMSH [4]). It is then straightforward to do the same with the Darcy 2D solver. Finally, the CTRW parameter calibration tool, Optim (which runs the 1D solver internally), may be studied.

Troubleshooting: If you get a “Runtime Error”, “Missing DLL Error”, or other such error notification immediately upon running any of the codes, you may be missing .dll files in your system. This is resolved easily by using this link to install Intel Fortran Redistributables for Windows on Intel 64:

https://software.intel.com/sites/default/files/managed/01/9c/w_cprof_p_11.1.072_redist_intel_64.zip

For background, you can read about links to the redistributable installation packages for the Intel Compiler Professional Editions for Windows at:

<https://software.intel.com/en-us/articles/redistributable-libraries-of-the-intel-c-and-fortran-compiler-for-windows>

1.4 Transport Equation

Both the 1D and the 2D solvers solve the reactive transport equation

$$\frac{\partial C^s(\mathbf{x}, t)}{\partial t} + \int_0^t M(t - \tau) \nabla \cdot \{ \mathbf{v}(\mathbf{x}) C^s(\mathbf{x}, \tau) - [\mathbf{D}(\mathbf{x}) \nabla C^s(\mathbf{x}, \tau)] \} d\tau = S^s(\mathbf{x}, t) \quad (1)$$

or in tensor notation

$$\dot{C}^s(x_i, t) + M(t) \otimes \{ v_i(x_i) C^s(x_i, t) - [D_{ij}(x_i) C^s_{,j}(x_i, t)]_{,i} \} = S^s(x_i, t) \quad (2)$$

where the bimolecular reaction $A + B \rightarrow C$ is considered, C^s is the concentration of species s (in [mole/m³]), x_i is the coordinate vector (in [m]), t is time (in [min]), M is the memory function (in [1/min]), v_i is the particle (chemical species) velocity vector (in [m/min]), and D_{ij} is the generalized dispersion tensor (in [m²/min]). S^s is the species source term, including a prescribed time-independent (but spatially dependent) volumetric mass source and a reaction source of species s (in [mole/m³-min]), the superscript s denotes the species number and tensor notation

is used, in which lower case subscript indices denote components, a repeated index denotes summation, and comma denotes covariant derivative. The symbol \otimes denotes convolution.

The reaction kinetics are assumed to follow the law of mass action: $S^A = S^B = -S^C = M \otimes \Gamma C^A C^B$, for this second-order reaction where Γ is the reaction rate coefficient. For the advection-dispersion equation (ADE), $M(t) = \delta(t)$, so that $S^A = S^B = -S^C = \Gamma C^A C^B$. This reaction is just a generic one; extending it to more complex reactions (e.g., bidirectional) or to a set of reactions is straightforward and simple (although it may cause stability issues).

An effective model for the memory function is the truncated power law (TPL). The specific form of the memory function used here is a Prony series (or sum of poles) [1]. A new method to fit the TPL model to the Prony series is discussed in Section 5 below, and the software for this is included. The initial condition (IC) and boundary conditions (BCs) and the numerical method used are detailed in [2], and the reactive source term is developed in [3].

The FEM solvers calculate and output the nodal resident and flux-weighted concentrations at preselected times. These can be accessed and manipulated later (e.g., by the available post processing tools) to produce plots of *spatial concentration profiles* and *temporal concentration breakthrough curves*.

1.5 Versions log

11/6/18	version 1.0.
5/8/18	version 2.0: The approximate treatment of the Dirichlet boundary conditions was replaced with an exact one (FEM-1D, FEM-2D and Darcy-2D).
2/9/18	version 3.0: (1) Addition of an independent volumetric mass source term (FEM-1D, FEM-2D and Darcy-2D). (2) Allowance for spatially variable boundary conditions (FEM-1D, FEM-2D and Darcy-2D). (3) The two 2D preprocessors (MeshGenRect and MeshGenRectDarcy) were merged into one (MeshGenRect). (4) Input changed to all of the subpackages. (5) Some MATLAB scripts were updated.
12/9/18	version 3.1: Bugs fixed in MeshGenRect.
16/10/18	version 3.2: Bugs fixed in FEM-1D and FEM-2D for non-constant velocity. Only relevant files were replaced. The example files were not replaced, since differences are tiny.
14/1/19	Section 6 (“Extensions”) added to this manual.

2. Installation of the packages

All of the software resides in a bundle of five zip files freely accessible at: <http://www.weizmann.ac.il/EPS/People/Brian/CTRW/software>. The following zip files are available:

FEM-1D:	1D transport solver, including source files, executable, post-processors (MATLAB) and a 1D example directory.
FEM-2D:	2D transport solver, including source files, executable, and a 2D example directory.
Darcy-2D:	2D Darcy flow solver for a given (user-generated) conductivity field, including source files, executable, and a 2D Darcy example directory. Output from this subpackage can be used as input to FEM-2D.
MeshGenRect:	2D simple rectangular and uniform mesh generator, including source files, executable, and a 2D mesh generation example directory for FEM-2D. Output from this subpackage can be used as input to FEM-2D or Darcy-2D.

Optim: This subpackage solves the inverse problem in 1D: given a measured breakthrough curve (BTC), the domain, and the initial and the boundary conditions, this subpackage calibrates the CTRW parameters (v_ψ , D_ψ , β , t_1 , t_2 as defined in [1]) to best-fit the measurements by iteratively solving the forward 1D problem (solution of the concentration profile using FEM-1D). This subpackage includes MATLAB source files and i/o files. Output from this subpackage can be used as input to FEM-1D and FEM-2D. [Note: In this Optim package, the user should run main.m; the executable of the 1D FEM solver is a copy of that in FEM-1D; the original finp is kept as finp – Copy.txt (because finp.txt is changed through the run).]

2.1 FEM-1D: 1D solver

Unzip all files in FEM-1D.zip into a convenient location (e.g., <top-dir\FEM-1D>). When completed, this folder includes:

1. The 1D solver executable, FEM-1D.exe.
2. MATLAB scripts to present the results graphically (post_c.m for the concentration and post_f.m for $C_f^s = (q^{sc} + q^{sd}) / v$ along x and at $x = L$).
3. An example I/O subdirectory, 1D-example, (1D reactive flow from [3], including a spreadsheet presenting the results graphically).
4. The FORTRAN source items, *.f90 and *.f.
5. The files used by MS Visual-Studio to build the executable (FEM-1D.vfproj, FEM.sln, FEM-1D.u2d, FEM.suo).

Note: Items 1-3 in the above list are sufficient to use the code “as is”. Items 4-5 are intended for those wishing to study the code, build it on different platforms, and/or extend/modify it for their specific needs.

2.2 FEM-2D: 2D solver

Unzip all files in FEM-2D.zip into a convenient location (e.g., <top-dir\FEM-2D>). When completed, this folder includes:

1. The 2D solver executable, FEM-2D.exe.
2. An example I/O subdirectory, 2D-example, (a 20×1 elements comparison to Gramling experiment shown in [1], including a spreadsheet presenting the results graphically compared to the 1D code results).
3. The FORTRAN source items, *.f90 and *.f.
4. The files used by MS Visual-Studio to build the executable (FEM-2D.vfproj, FEM.sln, FEM-2D.u2d, FEM.suo).

Note: Items 1-2 in the above list are sufficient to use the code “as is”. Items 3-4 are intended for those wishing to study the code, build it on different platforms, and/or extend/modify it for their specific needs.

2.3 MeshGenRect: 2D simple rectangular and uniform mesh generator

Unzip all files in MeshGenRect.zip in a convenient location (e.g., <top-dir\MeshGenRect >). When completed, this folder includes:

1. The 2D mesh generator executable, MeshGenRect.exe.
2. An example I/O subdirectory, MeshGenRect-example.
3. The FORTRAN source items, *.f90.
4. The files used by MS Visual-Studio to build the executable (MeshGenRect.vfproj, MeshGenRect.sln, MeshGenRect.u2d, MeshGenRect.suo).

Note: Items 1-2 in the above list are sufficient to use the code as-is. Items 3-4 are intended for those wishing to study the code, build it on different platforms, and/or extend/modify it for their specific needs.

2.4 Darcy-2D: 2D Darcy solver

Unzip all files in Darcy-2D.zip into a convenient location (e.g., <top-dir\Darcy-2D>). When completed, this folder includes:

1. The 2D solver executable, Darcy-2D.exe.
2. An example I/O subdirectory, Darcy-2D-example, (a comparison to a problem with analytical solution described in CarslawJaeger.docx).
3. A MATLAB script, FEM_to_Matrices.m, to convert the results to matrix form (if desired).
4. The FORTRAN source items, *.f90 and *.f.
5. The files used by MS Visual-Studio to build the executable (FEM-2D.vfproj, FEM.sln, FEM-2D.u2d, FEM.suo).

Note: Items 1-3 in the above list are sufficient to use the code “as is”. Items 4-5 are intended for those wishing to study the code, build it on different platforms, and/or extend/modify it for their specific needs.

2.5 Optim: CTRW (1D) parameter calibration tool

Unzip all files in optim.zip into a convenient location (e.g., <top-dir\optim>). When completed, this folder includes:

1. MATLAB scripts, *.m, to run the code (run main.m).
2. The 1D solver executable, FEM-1D.exe (identical to that of Section 2.1).
3. I/O files (parameters fitting for Scheidegger’s experiment described in [1]).
4. A spreadsheet comparing graphically the experiment to the initial and the final numerical solutions).

Note: The MATLAB code runs the 1D solver in an optimization loop and overrides some of its I/O files. Therefore, the original input files are also kept in the *orig* subdirectory.

3. Input/Output (I/O)

The codes may use any consistent units system. Usually SI is preferred here, but occasionally, for convenience, other units (e.g., minutes rather than seconds) may be used. In the latter case, ensure that all input data are consistent. **NOTE:** Changes / additions for FEM-2D inputs appear in *red italics*.

3.1 Input files (for both FEM-1D and FEM-2D packages)

All I/O files are ASCII, and all input files are also free-formatted. The respective 1D and 2D input files appear in the FEM-1D and FEM-2D packages.

3.1.1 The main input file, **finp.txt**

The main input file consists of the following data.

Parameters

Ne	total number of elements
Nn	total number of nodes
Nb	total number of BC faces
Nm	total number of materials
Np	number of Prony terms (not including p=0)

ADE (Advection Dispersion Equation, i.e., for Fickian transport), for $N_p=0$,
 EXP (Prony series model, i.e., for anomalous transport) for $N_p>0$ (the same for
 all materials $m=1,..., N_m$)
 Nd maximum number of elements per node
 Ng number of Gauss points in each coordinate
 Ns total number of species

Properties

Sx cross section area normal to x (*thickness along z in 2D*)
 Kappa reaction kinetic coefficient
 tc characteristic time for the reaction
 por porosity

Time

tmax end time
 dt time step
 dto time step for output

Temporal scheme

theta implicitness factor (0 – explicit, 0.5 – Crank-Nicholson, 1 – fully implicit)

bCGstab input

The input for par(2:3), ipar(6), fpar(1:2), fpar(11) and the suggested values – for a
 description see comments below (from bCGstab.f)

ipar(2) = 0 no preconditioning
 ipar(3) = 0 default stopping criteria
 ipar(6) = -1 maximum [A]{v} operations (unlimited if <0)
 fpar(1) relative tolerance
 fpar(2) absolute tolerance
 fpar(11) = 0 initial number of floating-point operations from matrix-vector
 multiplications and preconditioners.

Implicit BC and source convergence parameters

maxit maximum number of iterations
 Ctol allowable maximum absolute tolerance for C^s

3.1.2 The nodal data file, **nodes.txt**

The nodal data file consists of N_n nodal coordinates and initial conditions (ICs) data, one
 line per node.

n node number (**NOTE** that this number is not used, and is overridden)
 x, y nodal x *and* y coordinates (*only in 2D*)
 $C_s(x,0)$ nodal IC (N_s values, one for each species, s)

3.1.3 The elements data file, **elements.txt**

The elements data file consists of N_e nodes and material data, one line per element.

e element number (**NOTE** that this number is not used, and is overridden)
 n1 element node number 1
 n2 element node number 2
 n3 *element node number 3 (only in 2D)*
 n4 *element node number 4 (only in 2D)*

mat element material number

3.1.4 The velocity data file, **v.txt**

The Darcian velocity (or the advective flux, i.e., the superficial velocity, being the physical velocity times the porosity) and dispersion of Ne elements data, one line per element.

e the element number (**NOTE** that this number is not used, and is overridden)

Vx Darcian velocity x component

Vy *Darcian velocity y component (only in 2D)*

NOTE that immediately after v.txt reading, Vx (*and Vy also in 2D*) are converted to the physical velocity, which is also written to the results. The Darcian velocity is used to simplify importing the velocity field from the Darcy-2D solver (described in Sec. 4), if desired.

3.1.5 The dispersion data file, **d.txt**

The dispersion of Ne elements data, one line per element.

e element number (**NOTE** that this number is not used, and is overridden)

Dxx dispersion xx component

Dxy dispersion xy component (only in 2D)

Dyx dispersion yx component (only in 2D)

Dyy dispersion yy component (only in 2D)

3.1.6 The BC data file, **bcs.txt**

The boundary conditions (BCs) data file consists of Nb BC faces data, one line per face.

f element local boundary face number (**NOTE** that this number is not used, and is overridden)

BCe BC element number

BCtype 'R', 'N' or 'D' for Robin, Neumann or Dirichlet, respectively

BCvalue prescribed \bar{j} , \bar{q} or \bar{C} nodal values used for the for R, N or D BC, respectively
(Ns values, one for each species (*2 sets - one for each element face node - only in 2D*))
for 'R' and 'N' these are NORMAL fluxes, positive for outflow and negative for inflow

3.1.7 The time-independent source data file, **source.txt**

The source data file consists of Ne elements data, one line per element.

e element number (**NOTE** that this number is not used, and is overridden)

s species number (**NOTE** that this number is not used, and is overridden)

Sxy two prescribed time-independent source nodal values (*four values in 2D*), 1 line for each species. (**NOTE** that for cases that do not include such time-independent source, zeros must be given)

3.1.8 The materials data file, **materials.txt**

The materials data file consists of Nm materials data, Np+1 lines per material (unused for ADE).

par(0,1) is $a_0 = n \ t_{char}/t_1$

par(p,1) are a_p and par(p,2) are b_p in $a_p \exp(-b_p t)$ for $1 \leq p \leq N_p$

3.2 **Output files (for both FEM-1D and FEM-2D packages)**

All output files are ASCII files.

3.2.1 1D solver output

The 1D solver produces the following output files that may be read by the 1D post-processors (see Sec. 3.4):

fout.txt is the general output, including the version identification, a partial input echo, the calculated CFL and Peclet numbers, the calculated SUPG parameter, the initialization and the total CPU run time. In case the run aborts prematurely, the error(s) (if identified) are also reported.

fdbg.txt is the debugging output. As-is, it just prints one line for each entry to the main program, MAIN, and to the output routine, OUT. If there are input errors or if the linear equations solver, BCGSTAB, fails to converge it contains useful data that may help to identify the problem.

fx.txt is the nodal coordinates file including a header line followed by Nn lines of the node number, I , and the coordinate, x_I .

fc.txt and **ft.txt** are the nodal concentration and right hand side (RHS) files including (for the IC and each output time) 3 header lines: n , the step number, t , the time, and $C(i,s)$ followed by Nn lines of the nodal value, C_I^s for node I and species s (and similarly for the RHS, $T(i,s)$).

fqc.txt, **fqd.txt** and **j.txt** are the nodal advection, dispersion and total flux files including (for the IC and each output time) 3 header lines: n , the step number, t , the time, and the label qc(i,s) followed by Nn lines of the nodal advective flux value q_I^{sc} for node I and species s , (and similar for q^d and j).

fcount.txt includes with the number of output steps, Nc, used by the post-processor.

3.2.2 2D solver output

The 2D solver produces the following output files that may be read by the 2D post-processor (see Sec. 3.4):

fout.txt and **fdbg.txt** are similar to these described above for the 1D solver.

post.msh is the output file that may be read by the general purpose GMSH post-processor (see Sec. 3.4):

3.3 2D pre-processor for FEM-2D

While it is possible to enter all of the input data manually, it may be easier to use a pre-processor. For 1D, the larger data (the nodes and elements) may be generated by simple tools, i.e., a simple ad hoc EXCEL spreadsheet or MATLAB scripts. For 2D, a simple FORTRAN pre-processor was implemented for a (i) rectangular domain, (ii) uniform velocity and dispersion coefficient, (iii) constant, uniform or $\delta(x,y)$ ICs, (iv) time-independent BCs (assumed to be piecewise-continuous x - or y -dependent functions) for each boundary face for each chemical species, and (v) time-independent volumetric mass source term (assumed to be the product of piecewise-continuous x - and y -dependent functions).

The pre-processor reads an input file, **meshinp.txt**, and produces the following files: **nodes.txt**, **elements.txt**, **v.txt**, **d.txt**, **source.txt**, and **bcs.txt**. In addition, some of the parameters of **finp.txt** are written to **param.txt** (the rest of **finp.inp** should be entered manually). **NOTE:** This preprocessor is used for both Darcy-2D (see Sec. 4) and FEM-2D. Changes / additions for FEM-2D inputs appear in *red italics*.

3.3.1 The 2D pre-processor input file, **meshinp.txt**

The 2D pre-processor input refers to files in the MeshGenRect package.

Parameters

Darcy	T for Darcy-2D or F for FEM-2D
Xmin, Xmax, Ymin, Ymax	rectangle geometry range
Nx, Ny, <i>Ns</i>	number of x and y intervals <i>and number of species (only for FEM-2D ; NS=1 is assumed only in Darcy-2D)</i>
<i>Vx, Vy</i>	<i>velocity vector components (only for FEM-2D)</i>

Properties

Dxx, Dxy, Dyx, Dyy

conductivity (Darcy-2D) / *dispersion coefficient (FEM-2D)* tensor components

Time-independent source function [$S(x,y) = f(x) g(y)$]

Nxx, Nyy

number of points defining $f(x)$ and $g(y)$

(xx(k), k=1,Nxx)

coordinate x for $f(x)$

(fx(s,k), k=1,Nxx)

$f(x)$ values (one line for each $s=1, \dots, N_s$)

(yy(k), k=1,Nyy)

coordinate y for $g(y)$

(gy(s,k), k=1,Nxx)

$g(y)$ values (one line for each $s=1, \dots, N_s$)

IC

delta (TRUE or FALSE) and its Xdelta, Ydelta

(C(s), s=1,Ns)

IC (constant for all nodes)

BCs function [$v(x)$ or $v(y)$]

(BCt(f), f=1,4)

BC type for (south, east, north, west) faces. One of:

‘D’ for Dirichlet BC

‘N’ for Neumann BC

‘R’ for Robin BC (only for FEM-2D)

(Nxy(f), f=1,4)

number of points defining $f(x)$ or $g(y)$ for (south, east, north, west) faces

(xy(1,i), i=1, Nxy(1))

x coordinates for the south face (f=1)

(vl(1,s,i), i=1,Nxy(1))

$v(x)$ values for f=1 (one line for each $s=1, \dots, N_s$)

(xy(2,i), i=1, Nxy(2))

y coordinates for the east face (f=2)

(vl(2,s,i), i=1,Nxy(2))

$v(y)$ values for f=2 (one line for each $s=1, \dots, N_s$)

(xy(3,i), i=1, Nxy(3))

x coordinates for the north face (f=3)

(vl(3,s,i), i=1,Nxy(3))

$v(x)$ values for f=3 (one line for each $s=1, \dots, N_s$)

(xy(4,i), i=1, Nxy(4))

y coordinates for the west face (f=4)

(vl(4,s,i), i=1,Nxy(4))

$v(y)$ values for f=4 (one line for each $s=1, \dots, N_s$)

NOTES:

- At least 2 points should be used to define any of the functions f , g and v .
- The endpoints of the functions f , g and v must coincide with the domain ends.
- For any pair of Dirichlet BCs meeting at a corner, the values at that corner should be identical (but for other types of BCs pair meeting at a corner, the values may be different, because they are prescribed **normal** values, and normals at a corner have two distinct directions).
- If input errors are found in **meshinp.txt**, error messages will appear at the end of **param.txt** and the run will abort.

3.4 Post-processors for FEM-1D and FEM-2D

3.4.1 The 1D post-processors

post_c.m is a MATLAB script that reads the I/O files and processes them to produce plots of the concentration, C^s , vs. x , and of the breakthrough curve (BTC), at each of the output times. The latter is also written to an ASCII file, **BTC_c.txt**. As is, it reads the FEM-1D I/O files and create two plots: (1) the spatial distribution of the resident concentrations of species A, B, C for all of the output times; and (2) the BTCs, i.e. the A, B, C resident concentration variations with time at the outlet $x = L$.

post_f.m is a MATLAB script that does the same for the flux-weighted concentration, $C_f^s = (q^{sc} + q^{sd})/Vx$, and also writes an ASCII file, **BTC_f.txt**. As is, it reads the FEM-1D I/O files and create two plots: (1) the spatial distribution of the flux-weighted concentrations of species A, B, C for all the output times; and (2) the BTCs, i.e., the A, B, C flux-weighted concentration variations with time at the outlet $x = L$.

In both scripts, three species (A, B and C) are assumed present.

3.4.2 The 2D post-processor

The general FEM post-processor GMSH [4] (freely available) is used for graphical display of the 2D results for each species, s : the nodal resident concentration, C_s , the nodal RHS, T_s , and the nodal advection, dispersion and total flux vectors, q_{is}^c , q_{is}^d , and j_{is} , respectively. The input (at elements centers) velocity vector, v_i , and dispersion tensor, D_{ij} , are also available.

3.4.3 Other tools

FEM_to_Matrices.m and **Darcy_to_Matrices.m** are MATLAB scripts that read the I/O files and rearrange the results for a rectangular domain and mesh.

3.5 Examples

3.5.1 FEM-1D example

The example of the 1D solver is a variation of the one explained in detail in [3]. A 0.36 m long domain ($L = 0.36$ m) is considered, divided uniformly into 20 1D linear elements. Three chemical species are considered to participate in the reaction $A + B \rightarrow C$ (where A is the species $s=1$, B is $s=2$ and C is $s=3$). The velocity and the dispersion coefficient are uniform ($Vx=1.21 \cdot 10^{-4}$, $Dxx=2.2 \cdot 10^{-7}$). A Prony series with $Np=20$ terms is used with $t_1 = 6.6$ s, $t_2 = 10^5$ s and $\beta = 1.6$. The reaction kinetic coefficient is 0.05 m³/s-mol. The initial conditions for concentrations of the three species are: $A=0$, $B=1$ mol/m³ and $C=0$ (all uniform). At the inlet ($x=0$) Dirichlet boundary conditions are prescribed ($A=1$ mol/m³, $B=C=0$), and at the outlet ($x=L$) homogeneous Neumann boundary conditions are prescribed for all species (zero derivatives). No time-independent mass source is present in this case. The numerical solution used a time step of 0.1 s, advances until 1500 s, and produces output every 300 s. Results are shown graphically in the results.xlsx spreadsheet of the FEM-2D example.

To replicate these results:

- Create a new folder, and copy the input files from the example folder.
- Create a shortcut to the FEM-1D executable in the new folder. On Windows OS it is important to set the shortcut properties (right-click on the shortcut, choose “Properties”) such that it starts in the current folder (i.e., clear the “Start in” box in the Shortcut tab).
- Run the FEM-1D executable by double-clicking on the shortcut. **NOTE:** An empty DOS window opens and indicates that calculations are in progress. This window is closed automatically when the run ends.

- The resulting output files should be identical (up to machine precision) to those in the example folder.
- You may also want to run the post_c.m and post_f.m MATLAB scripts (see Sec. 3.4) to post-process the results graphically. In this case, copy these files from the FEM-1D folder to the current folder and run them by double-clicking them. **NOTES:** (1) The vertical scales in the created plots are the MATLAB default. To compare results to those in results.xlsx of the FEM-1D example, they should be rescaled to match those of results.xlsx. (2) The MATLAB scripts can be modified to suit user needs. This is relatively simple: just edit the bottom of these scripts, where the graphics are defined.

3.5.2 FEM-2D example

The example of the 2D solver is identical to the FEM-1D example of the subsection 3.5.1, now considering a 2D domain. Here, a 2D domain ($0 \leq x \leq L=0.36$ m, -0.009 m $\leq y \leq 0.009$ m) is used (so as to have the same increments in both x and y) and is divided to 20×1 uniform bilinear 2D elements. Results are shown graphically and compared to the 1D solution in the 2Dvs1D.xlsx spreadsheet. **NOTE:** The differences between these 1D and 2D solutions result from the SUPG (Streamline Upwind Petrov–Galerkin) scheme when the element Peclet number $Pe = \nu h/(2D)$ exceeds 1, i.e., advection-dominated flow. By sufficiently decreasing the element size (say, 250 elements in x , such that $dx=dy=0.36/250$) the scheme becomes identical to a central scheme, the accuracy becomes 2nd order and the 1D and 2D solutions become identical.

To replicate these results:

- Create a new folder, and copy the input files meshinp.txt and finp.txt from the example folder.
- Create shortcuts to the MeshGenRect pre-processor and to FEM-2D executables in the new folder. On Windows OS it is important to set the shortcut properties (right-click on the shortcut, choose “Properties”) such that it starts in the current folder (i.e., clear the “Start in” box in the Shortcut tab).
- Run the MeshGenRect executable by double-clicking on the shortcut. **NOTE:** An empty DOS window opens and indicates that calculations are in progress. This window is closed automatically when the run ends.
- Edit the finp.txt file, and change the values of Ne, Nn, Nb, Nm and Ns according to those in param.txt.
- Run the FEM-2D executable by double-clicking on the shortcut. **NOTE:** An empty DOS window opens and indicates that calculations are in progress. This window is closed automatically when the run ends.
- The output files produced should be identical (up to machine precision) to those in the example folder.
- You may also want to run the GMSH post-processor (should be downloaded and installed first) by double-clicking the output file post.msh to display the results graphically.

4. Darcy equation solver

4.1 Theory

A similar, and much simpler, methodology is also applied to solve the Darcy equation for fluid flow. Because it has not been published to date, it is developed below.

Consider incompressible flow through a porous medium [5]. The continuity equation may be expressed as

$$q_{i,i} = 0 \quad (3)$$

where q_i is the flux vector (in [m/min]), related to the velocity vector, v_i , through

$$q_i = v_i \phi \quad (4)$$

ϕ is the porosity (dimensionless) and tensor notation is used again.

The relation between the flux and the hydraulic head, h (in [m]), is given by Darcy's law

$$q_i = -K_{ij} h_{,j} \quad (5)$$

where K_{ij} is the hydraulic conductivity tensor (in [m/min]).

Combining (3) and (5) yields the governing equation for the head in the domain, V

$$-(K_{ij} h_{,j})_{,i} = 0 \quad (6)$$

The associated boundary conditions (BC) considered here are either of Dirichlet type

$$h - H = 0 \quad \text{on } \Gamma^D \quad (7)$$

or of Neumann type

$$-n_i q_i - Q = 0 \quad \text{on } \Gamma^N \quad (8)$$

where H and Q are the prescribed head and normal flux, respectively and n_i is the outward normal to the domain on the boundary, Γ .

We now apply the standard Bubnov-Galerkin method (see [1], [6]), by multiplying (3) by weights N_I , integrating over the domain V and adding products of the BC terms (7) and (8) multiplied by weights N_I^D and N_I^N integrated over the respective boundaries, Γ^D and Γ^N :

$$\begin{aligned} 0 &= -\int_V N_I q_{i,i} dV + \int_{\Gamma^D} N_I^D (h - H) d\Gamma - \int_{\Gamma^N} N_I^N (n_i q_i + Q) d\Gamma = \\ &= -\int_{\Gamma^D + \Gamma^N} n_i N_I q_{i,i} d\Gamma + \int_V N_{I,i} q_i dV + \int_{\Gamma^D} N_I^D (h - H) d\Gamma - \int_{\Gamma^N} N_I^N (n_i q_i + Q) d\Gamma \end{aligned} \quad (9)$$

where the divergence theorem was used.

Because N_I^D and N_I^N are arbitrary functions, we may choose them as $N_I^D = N_I^N = -N_I$. We may also choose $h=H$ on Γ^D , so that the third integral vanishes, and N_I such that it vanishes on Γ^D . Consequently, (9) becomes

$$0 = \int_V N_{I,i} q_i dV + \int_{\Gamma^N} N_I Q d\Gamma \quad (10)$$

In the finite element method (FEM) the domain is divided into non-overlapping elements of subdomains V^e , where h within each element is approximates by its nodal values, h_I , as

$$h \cong N_J(x_i) h_J \quad (11)$$

where capital indices stand for values at nodes, x_i is the coordinate vector and N_I is the interpolation function (which, in the Bubnov-Galerkin method, is identical to above weights).

Substituting (11) in (10) yields a linear equations system

$$P_{IJ} h_J + Q_I = 0 \quad (12)$$

with

$$P_{IJ} = \int_V N_{I,i} K_{ij} N_{J,j} dV \quad (13)$$

$$Q_I = - \int_{\Gamma^N} N_I Q d\Gamma \quad (14)$$

Specific forms of the element matrix and vector are given explicitly in [1] and [2] for 1D and 2D linear elements, respectively, together with further details of the solution procedures.

Note that these equations are a degenerated form of those for the CTRW transport equations: there is no memory function, a single species is considered and steady-state is assumed here. Note also that because there is no advection term here, there is no need for upwind and use of the more general Petrov-Galerkin method [2], and therefore the simpler and optimal Bubnov-Galerkin was chosen instead [6].

4.2 Input files

4.2.1 The main input file, **finp.txt**

The main input file consists of the following data. **NOTE** that $N_s=1$ is assumed here.

Parameters

Ne	total number of elements
Nn	total number of nodes
Nb	total number of BC faces
Nm	total number of materials
Nd	maximum number of elements per node
Ng	number of Gauss points in each coordinate

Properties

Sx	thickness along z
----	-------------------

bCGstab input

The input for par(2:3), ipar(6), fpar(1:2), fpar(11) and the suggested values – for a description see comments below (from bCGstab.f)

ipar(2) = 0	no preconditioning
ipar(3) = 0	default stopping criteria
ipar(6) = -1	maximum [A]{v} operations (unlimited if <0)
fpar(1)	relative tolerance
fpar(2)	absolute tolerance
fpar(11) = 0	initial number of floating-point operations from matrix-vector multiplications and preconditioners

4.2.2 The nodal data file, **nodes.txt**

The nodal data file consists of Nn nodal coordinates and IC data, one line per node.

n	node number (NOTE that this number is not used, and is overridden)
x, y	nodal x and y coordinates
h ₀	nodal initial guess

4.2.3 The elements data file, **elements.txt**

The elements data file consists of Ne nodes and material data, one line per element.

e	element number (NOTE that this number is not used, and is overridden)
n1	element node number 1

n2	element node number 2
n3	element node number 3
n4	element node number 4
mat	element material number

4.2.4 The conductivity data file, **d.txt**

The conductivity of Ne elements data, one line per element.

e	element number (NOTE that this number is not used, and is overridden)
Dxx	conductivity xx component
Dxy	conductivity xy component
Dyx	conductivity yx component
Dyy	conductivity yy component

4.2.5 The BC data file, **bcs.txt**

The boundary conditions (BCs) data file consists of Nb BC faces data, one line per face.

f	element local boundary face number (NOTE that this number is not used, and is overridden)
BCe	BC element number
BCtype	'N' or 'D' for Neumann or Dirichlet, respectively
BCvalue	prescribed \bar{j} , \bar{q} or \bar{C} <u>nodal</u> values used for the for N or D BC, respectively (2 sets - one for each element face node) for 'N' these are NORMAL fluxes, positive for outflow and negative for inflow

4.3 Output files

The Darcy-2D solver produces the following output files that may be read by the 2D post-processor (see Sec. 4.4):

fout.txt and **fdbg.txt** are similar to these described in Sec. 3.2.1 for the 1D solver.

post.msh is the output file that may be read by the general purpose GMSH post-processor.

4.4 2D post-processors

GMSH [4] is used for graphical display of the 2D results: the nodal head, h , the nodal BC, T , and the nodal flux vector, q_i . The input (at elements centers) conductivity tensor, D_{ij} , is also available. **NOTE:** In the plot heading, D_{ij} is labelled “dispersion tensor” rather than “conductivity tensor”.

4.5 DARCY-2D example

The example of the 2D Darcy solver is described in CarlsJawJaeger.docx (conduction in an orthotropic rectangle compared to textbook analytic solution [7]) in CarlsJawJaeger.xlsx. The errors here are much lower than in the former versions due to the exact treatment of the Dirichlet BCs and their piecewise linear variations along the faces.

To replicate these results:

- Create a new folder, and copy the input files meshinp.txt and finp.txt from the example folder.
- Create shortcuts to the MeshGenRect pre-processor and to Darcy-2D executables in the new folder. On Windows OS it is important to set the shortcut properties (right-click on the shortcut, choose “Properties”) such that it starts in the current folder (i.e., clear the “Start in” box in the Shortcut tab).
- Run the MeshGenRect executable by double-clicking on the shortcut. **NOTE:** An empty DOS window opens and indicates that calculations are in progress. This window is closed automatically when the run ends.

- Edit the `finp.txt` file, and change the values of `Ne`, `Nn`, `Nb` and `Nm` according to those in `param.txt`.
- Run the Darcy-2D executable by double-clicking on the shortcut. **NOTE:** An empty DOS window opens and indicates that calculations are in progress. This window is closed automatically when the run ends.
- The output files produced should be identical (up to machine precision) to those in the example folder.
- You may also want to run the GMSH post-processor (should be downloaded and installed first) by double-clicking the output file `post.msh` to display the results graphically.

5. Optim: Parameter calibration

Calibration (optimization) of the parameters $\{v, D, \beta, t_1, t_2\}$ for transport of an inert tracer in column (1D problem; matching a temporal breakthrough curve) is realized in a MATLAB code. It consists of optimizing these parameters by `my_fminsearch.m`, a variant of the built-in `fminsearch` function, a multidimensional, unconstrained, nonlinear minimization (Nelder-Mead). See also the CTRW Matlab Toolbox [9], which employs a best-fit optimization in a different approach to obtaining a 1D solution of the CTRW PDE.

The working directory should include, in addition to the MATLAB scripts, input files of a 1D inert case to be run by FEM-1D (except `v.txt`, `d.txt` and `materials.txt`, that will be generated). During the optimization process, `v.txt`, `d.txt` and `materials.txt` will be overwritten and `finp.txt` will be modified.

The calculation starts in `main.m`, where these parameters are initialized (initial guess), an error bound, `errbnd` (see Sec. 5.1), the output file name and the reference BTC file are specified. The domain length, L , is read and calculated from `nodes.txt`, and the FEM output times are read from `finp.txt`.

Next, the reference BTC curve is read and interpolated according to the FEM output times, the optimization parameters are set and the initial error is calculated by running FEM-1D and comparing the initial error norm by calling `diff_norm.m`.

The optimization script, `my_fminsearch.m`, is then called and continues running until convergence or stopping criteria are met. Following a final call to `diff_norm.m`, the initial and final parameters are written by calling `gen_output.m` function.

Within `diff_norm.m` the TPL parameters are fitted by a Prony series (or Sum of Poles) by calling `sop2.m`. After the number of poles, N_p , is known, it is modified in `finp.txt`, `v.txt`, `d.txt` and `materials.txt` files are overwritten. The FEM-1D executable is then run with the updated input, followed by a call to `post_f_1.m`, which reads the simulation results, including the numerical BTC. File `BTC_f.txt` includes the latter. The error norm between the numerical and the experimental BTCs is then calculated, and returned to the optimizer.

NOTE: The location of the FEM-1D exe should be verified in `diff_norm.m` according to its actual installed location.

5.1 Sum-of-Pole MATLAB Function

A new method to fit the TPL model to the Prony series is discussed below, and the software for this fit (also in this Optim package) will be detailed.

The MATLAB file `sop2.m` contains a set of MATLAB functions with the function `sop2` as the driver function for computing an efficient sum-of-pole approximation for the Laplace transform $\tilde{M}(u)$ of the memory function for the TPL model.

The user interface of the function `sop2` is as follows:

`[xs,ws,npols] = sop2(param,errbnd)`

Here the first input argument, $param$, is a row vector of length 3 that specifies the parameters for $\tilde{M}(u)$, where $param(1)$ is β , $param(2)$ is $\log_{10}(t_1)$ and $param(3)$ is $\log_{10}(t_2)$. The user is referred to [1] about the physical meaning of these parameters and the specification of $\tilde{M}(u)$. The second input argument, $errbnd$, is the desired precision measured in relative L^2 norm. The output xs is a column vector containing the pole locations; ws is a column vector containing the pole weights; and $npols$ is an integer equal to the number of poles in the sum-of-pole approximation. That is,

$$\left\| \tilde{M}(iy) - n - \sum_{i=1}^{npols} \frac{ws(i)}{iy - xs(i)} \right\|_2 \leq errbnd \left\| \tilde{M} - n \right\|_2 \quad y \in \mathfrak{R}, \quad (15)$$

where n is the normalization factor in $\tilde{M}(u)$, i.e.,

$$\lim_{u \rightarrow \infty} \tilde{M}(u) = n. \quad (16)$$

By the property of the Laplace transform, this is equivalent to a sum-of-exponential approximation for the memory function $\tilde{M}(t)$ itself. That is,

$$M(t) \approx n\delta(t) + \sum_{i=1}^{npols} ws(i) \exp(xs(i)t), \quad t > 0. \quad (17)$$

The function implements a simplified algorithm tailored for $\tilde{M}(u)$ and thus will probably not work well for other functions. However, for $\tilde{M}(u)$ itself, numerical tests indicate that the algorithm works very well. Indeed, the following eight parameter sets have all passed the test with $errbnd = 10^{-6}$ or 10^{-4} :

```
param2 = [1.4393    -2.9943e+00    1.0554e+01];
param3 = [1.3726    -2.4137e+00    1.1541e+01];
param4 = [1.5968    -1.6626e+00    1.0396e+01];
param5 = [1.3605    -9.8386e-01    1.1942e+01];
param6 = [1.9600     8.1954e-01    5.0000e+00];
param7 = [1.9600     8.1954e-01    2.0000e+00];
param8 = [1.6000     8.1954e-01    5.0000e+00];
```

Usage example:

```
>> param8 = [1.6000      8.1954e-01    5.0000e+00];
>> [xs,ws,npols] = sop2(param8,1e-6);
```

Please send any specific questions regarding implementation of this Sum-of-Pole Matlab function to shidong.jiang@njit.edu.

5.2 Optim example

The example of Optim is about fitting the CTRW parameters to the experimental data of Scheidegger [8] discussed in [1]. A 1D domain, $0 \leq x \leq L=1$ filled with is a single (inert) chemical species, is uniformly divided into 20 1D linear elements. The initial concentration is 0. At the inlet ($x=0$) Robin boundary conditions are prescribed ($j_A=1 \text{ mol/m}^2$), and at the outlet ($x=L$) homogeneous Neumann boundary conditions are prescribed for all species (zero derivatives). The initial guess for the velocity and the dispersion coefficient are uniform ($Vx=9 \cdot 10^{-3} \text{ m/min}$, $Dxx=1 \cdot 10^{-5} \text{ m}^2/\text{min}$) and the initial guesses for the TPL parameters (see [1])

are $\beta=1.5$, $t_1=0.02$ min, $t_2=2\cdot 10^{10}$ min. A Prony series with $N_p=24$ terms is used as an initial guess. The numerical solution used a time step of 0.1 min, advances until 340 min, and produces output every 20 min.

The initial and the fitted results are shown graphically and compared to the experimental results in the FEM-NumComparison.xlsx spreadsheet.

To replicate these results:

- Copy the following files from the example folder to a new folder: *.m, *.exe and the *orig* subfolder.
- Remove any *.txt files from the new folder and copy all the *.txt from its *orig* subfolder. **NOTE** that part of these files will change during the run, so that the original inputs are saved in the *orig* subfolder.
- Run Optim by double-clicking main.m. The main output from Optim is named “Scheidegger.txt” in this case (the variable “output” defined in line 72 of main.m), containing v , D , β , t_1 , t_2 , t_{mean} , t_{char} , n [3] and $npols$ (the number of Prony series terms). The flux-weighted concentration at $x=L$ vs. time is written each iteration (by post_f_1.m) to the BTC_f.txt ASCII file. When the run ends, this file will include the converged values, that are also left as the MATLAB arrays t (time) and Cfo (the flux-weighted concentration BTC). These can be copied to the bottom of the FEM-NumComparison.xlsx file to produce the converged results comparison.
- The output files produced should be identical (up to machine precision) to those in the example folder.

For this example, the elapsed time was 414s, the CPU time was 639s on an Intel 4-cores i7 2600 CPU, 3.40GHz, 8Gb RAM, running Windows 7 64 bit PC.

6. Extensions

Most of the extensions and improvements mentioned in [1], [2] and [3] are already implemented in the current version. The remaining ones (and others, according to needs and creativity) may implemented by individual users.

Some extensions require only changes to the input. For example, triangular elements may be included in a mesh and calculated without any change in the code, by merely specifying quadrilaterals with one collapsed edge (e.g., with identical nodes 2 and 3). For other problems, one can easily change the BCs, such that certain regions of a boundary may have different BC types. As an example, if there is one wall surface with small inlets in it, the inlets may be treated as Dirichlet or Robin BCs, and the remainder of the surface as a Robin or Neumann BC. Another possible extension is to use the Darcian velocity field, as calculated with the Darcy solver, to replace the constant field created by the preprocessor; this is achieved simply by copying the Darcian velocity from the post.msh file created by the Darcy solver and pasting it as a new v.inp file. Such extensions may be done manually or by patching the input files created by the existing preprocessors or by creating another preprocessor, either general-purpose or ad hoc, for a specific problem.

More advanced extensions require changes in the source code. For example, using higher order elements is possible if the shape functions and their derivatives are replaced in the appropriate routines (SHAPE, SHAPEB and SHAPEQ, all highly commented), and modifying the Gauss quadrature abscissas and weights in READIN accordingly to make the quadrature accurate. Adding a 3D variant is also relatively simple, but will require more elaborate changes in many parts of the code; however, the existing structure may remain intact, and the theory is as described in [1], [2] and [3].

References

1. Ben-Zvi, R., Scher, H., Jiang S., Berkowitz, B. (2016) One-dimensional finite element method solution of a class of integro-differential equations: Application to non-Fickian transport in disordered media, *Transport in Porous Media*, 115(2), 239-263.
2. Ben-Zvi, R., Scher, H., Berkowitz, B. (2017) Two-dimensional Finite Element Method solution of a class of integro-differential equations: Application to non-Fickian transport in disordered media, *International Journal for Numerical Methods in Engineering*, 112(5), 459-478, doi:10.1002/nme.5524.
3. Ben-Zvi, R., Nissan, A., Scher, H., Berkowitz, B. (2018) A continuous time random walk (CTRW) integro-differential equation with chemical interaction, *European Journal of Physics B*, 91, 15, doi.org/10.1140/epjb/e2017-80417-8.
4. Geuzaine, C., Remacle, J.-F. (2009) Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. *Int. J. Numerical Methods in Engineering*, 79(11), 1309-1331. <http://geuz.org/gmsh>.
5. Guadagnini, A., Neuman, S.P. (1999) Nonlocal and localized analyses of conditional mean steady state flow in bounded, randomly nonuniform domains: 1. Theory and computational approach. *Water Resources Research*, 35(10), 2999-3018.
6. Zienkiewicz, O.C., Taylor, R.L. (2000) *The Finite Element Method, Volumes 1-3: The Basis*, 5th edition, Butterworth Heinemann.
7. Carslaw, H.S., Jaeger, J.C. (1965) *Conduction of Heat in Solids*, 2nd edition, Oxford Clarendon Press.
8. Scheidegger, A.E. (1959) An evaluation of the accuracy of the diffusivity equation for describing miscible displacement in porous media. P. 101-116. In *Proc. Theory of Fluid Flow in Porous Media Conf.*, Univ. of Oklahoma.
9. Ben-Zvi, R., Jiang, S., Scher, H., Berkowitz, B., The CTRW-FEM Package v4.0: A practical user's guide, <http://www.weizmann.ac.il/EPS/Brian/CTRW/software>