

The CTRW Matlab toolbox v4.0: A practical user's guide

A. Cortis, S. Emmanuel, S. Rubin, K. Willbrand, R. Ben-Zvi, A. Nissan
(in chronological order of updates)
and B. Berkowitz
Department of Earth and Planetary Sciences,
Weizmann Institute of Science, 7610001 Rehovot, Israel
Correspondence: brian.berkowitz@weizmann.ac.il

September 15, 2020

Contents

| | | |
|----------|--|-----------|
| 1 | The CTRW Matlab Toolbox | 2 |
| 1.1 | What is the CTRW toolbox? | 2 |
| 1.2 | Introduction | 2 |
| 1.3 | Installing the CTRW toolbox | 2 |
| 1.4 | Legal notice | 3 |
| 2 | The Equations | 3 |
| 3 | 1D Forward Modeling | 6 |
| 3.1 | Generating a temporal profile | 6 |
| 3.2 | Generating a spatial profile | 8 |
| 3.3 | The options variable | 8 |
| 3.4 | Adsorbing tracer transport | 10 |
| 4 | 1D Inverse Problem: Fitting CTRW solutions to data sets | 11 |
| 4.1 | Fitting single breakthrough curves | 12 |
| 4.2 | Comparing fits for two different models | 14 |
| 4.3 | Fitting multiple breakthrough curves | 15 |
| 4.4 | Fitting for selected parameters | 15 |
| 4.5 | Fitting for a square input of duration ϵ | 15 |
| 4.6 | Fitting for reactive tracers | 16 |
| 5 | 2D Forward Modeling: A sample case | 16 |
| 6 | Appendix I: The Laplace-Domain Analytical Solutions | 17 |
| 6.1 | Inlet-type: 'R' (Robin), Inlet-temporal-function: 'step' | 18 |
| 6.2 | Inlet-type: 'D' (Dirichlet), Inlet-temporal-function: 'step' | 19 |
| 7 | Appendix II: CTRW Toolbox File Descriptions | 20 |

1 The CTRW Matlab Toolbox

1.1 What is the CTRW toolbox?

The CTRW toolbox is a collection of MATLAB scripts and functions for quantifying solute transport in porous media. The toolbox solves the partial differential equation (pde) form of the transport equation developed from Continuous Time Random Walk (CTRW) theory. The reader is referred to relevant literature on CTRW theory and its application, listed in the “Literature” section on the web page containing this toolbox. The CTRW toolbox v4.0 runs on MATLAB versions v6.1, v6.5, v7.0, v7.4 and higher; it has not been tested on earlier versions of MATLAB.

1.2 Introduction

In this User’s Guide we provide detailed instructions on how to generate both spatial and temporal concentration profiles for one-dimensional (1D) and two-dimensional (2D) stationary systems using MATLAB subroutines. In addition, we demonstrate how to fit the breakthrough curves to actual data. Throughout this guide, we assume that the user has a basic working knowledge of MATLAB.

We first provide a short introduction to the equations being solved. Detailed description of the CTRW theory and all solutions considered here are discussed in a series of papers, a list of which can be found at <http://www.weizmann.ac.il/EPS/People/Brian/CTRW> at the link Literature. An outline is given of the governing transport equation, the definition of input and output boundary conditions, and the various functional forms of the probability density function $\psi(t)$ defined in conjunction with the transport equation. We then describe how the toolbox functions can be used to solve both “forward” and “inverse” problems. To obtain solutions for forward problems, the user inputs specific parameter values, the inlet boundary condition, and the choice of $\tilde{\psi}(u)$ (i.e., the Laplace transform of $\psi(t)$), and the toolbox calculates either a 1D temporal or 1D/2D spatial concentration profile of the migrating solute. For inverse problems, in which the user is interested in fitting the transport equation to laboratory and field data to extract equation parameter values, the user inputs all of the above information, as well as measurements of concentration vs. time. The toolbox can then be used to calculate best fit estimates of the parameters in $\tilde{\psi}(u)$, v_ψ , and D_ψ , or any combination thereof. Specific examples of running the toolbox in both forward and inverse modes are given in Sections 3 and 4, respectively.

A complete listing of the files is given at the end of this document. The advanced user may wish to modify these files to change input/output requirements, and/or to consider additional aspects of the CTRW theory.

1.3 Installing the CTRW toolbox

Unzip all the files contained in the CTRW_v4.0.zip. The directory named CTRW_v4.0 has now been created on your disk. Run MATLAB and change the working directory to CTRW_v4.0 or any other name of your choice. Note that data files for fitting and analysis should be located in the same working directory. Happy computing!

1.4 Legal notice

A legal notice on usage, distribution, disclaimer and trademarks is contained. To access the contents of the `Legal_notice.m` file type in the MATLAB window

```
>> type Legal_notice

%
% ----- L E G A L   N O T I C E -----
%
% The "CTRW software" and the "CTRW MATLAB toolbox" are produced by Brian Berkowitz,
% Weizmann Institute of Science.
% Permission to use "CTRW 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 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 MAKES NO WARRANTY OR
% REPRESENTATION THERETO.
%
% THE WEIZMANN INSTITUTE 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 SOFTWARE OR IN CONNECTION THEREOF.
%
% Trademarks appear throughout this toolbox 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 toolbox, please cite it as:
%
% Rami Ben-Zvi, Andrea Cortis, Simon Emmanuel, Alon Nissan, Shira Rubin, Karen Willbrand
% and Brian Berkowitz
% The CTRW Matlab toolbox v4.0: a practical user's guide
% http://www.weizmann.ac.il/EPS/People/Brian/CTRW
% or using [Cortis, A. and B. Berkowitz (2005). Computing "anomalous" contaminant
% transport in porous media: The CTRW MATLAB toolbox, Ground Water, 43(6), 947-950.]
%
```

2 The Equations

In this section, we consider the numerical solution of the 1D, pde form of the CTRW transport equation. Details of the equations and theory are given in the literature listed at the end of this section.

We usually formulate the CTRW transport equation in Laplace space, to represent the time derivative in an algebraic expression. In one dimension, the Laplace transformed concentration $\tilde{c}(x, u)$ is given by

$$u\tilde{c}(x, u) - c_0(x) = -\tilde{M}(u) \left[v_\psi \frac{\partial}{\partial x} \tilde{c}(x, u) - D_\psi \frac{\partial^2}{\partial x^2} \tilde{c}(x, u) \right] \quad (1)$$

where

$$\tilde{M}(u) \equiv t_{char} u \frac{\tilde{\psi}(u)}{1 - \tilde{\psi}(u)} \quad (2)$$

is a memory function which accounts for the unknown, small-scale heterogeneities. In the above equations, u is the (dimensional) Laplace variable (with units of $[t^{-1}]$), the $\tilde{\cdot}$ symbol represents the Laplace transformed variable, t_{char} is a characteristic time, and v_ψ and D_ψ are the transport velocity and generalized dispersion coefficient respectively. The “dispersivity” can then be defined as $\alpha_\psi \equiv D_\psi/v_\psi$. It is important to recognize that the “transport velocity”, v_ψ , is distinct from the “average pore velocity”, v , whereas in the classical advection-dispersion picture, these velocities are identical. We stress that from here on, the terms “velocity”, “dispersion”, and “dispersivity” refer to the CTRW interpretation, and may or may not be indicated by a subscript ψ . We note also that the average mass flux of the solute, j , is defined in Laplace space through

$$\tilde{j} \equiv \tilde{M} v_\psi (\tilde{c} - \alpha_\psi \partial \tilde{c} / \partial x). \quad (3)$$

The function $\tilde{\psi}(u)$ is the “heart” of the CTRW formulation, and characterizes the nature of the solute movement. The function $\tilde{M}(u)$ can take on several expressions, depending on the functional form of $\tilde{\psi}(u)$. General forms that have been described in the literature are:

- the **truncated power law** model,

$$\tilde{\psi}(u) \equiv (1 + \tau_2 u t_1)^\beta \exp(t_1 u) \Gamma(-\beta, \tau_2^{-1} + t_1 u) / \Gamma(-\beta, \tau_2^{-1}), \quad 0 < \beta < 2, \quad (\text{TPL})$$

which requires 3 input parameters β , t_1 , t_2 , where $\tau_2 = t_2/t_1$ (where t_1 and t_2 have dimensions of time). [Note: As discussed in section 4.1, when t_2 is large relative to t_1 , then t_1 can be determined directly from v_ψ and D_ψ .] The memory function is determined by substitution of the expression above into (2) with $t_{char} = t_1$. The time t_1 represents an approximate transition time and sets the lower limit from which the power law behavior begins. Note that setting the cut-off time t_2 to a very large value allows consideration of a pure power law model. To account for realistic anomalous transport in real porous media, we generally find that $\sim 0.8 < \beta < 2$.

- the **modified exponential** or “eta” model,

$$\tilde{\psi}(u) \equiv \mathcal{L} \left\{ 2\eta {}_3F_3 \left[\begin{matrix} 1, 1, 1 \\ 2, 2, 2 \end{matrix} ; -\tau \right] e^{-2\eta\tau {}_4F_4 \left[\begin{matrix} 1, 1, 1, 1 \\ 2, 2, 2, 2 \end{matrix} ; -\tau \right]} ; \tau \rightarrow u \right\}, \quad \eta > 0, \quad (\text{ETA})$$

which requires 1 input parameter, η , the ‘disorder’ parameter (where \mathcal{L} denotes the Laplace transform). The η value can be as small as one likes. When the value of η is close to unity the $\tilde{\psi}(u)$ function is essentially an exponential in time. In the toolbox, the choice of η is limited to the range $0.05 < \eta < 2$.

- Eq. (1) reduces to the classical **advection-dispersion equation** (ADE) when $\tilde{\psi}(u)$ is the Laplace transform of an exponential in time,

$$\tilde{\psi}(u) \equiv \frac{1}{1 + \bar{t}u} \quad (\text{ADE})$$

where \bar{t} is the mean transition time, which does not require any additional input parameter. In the toolbox, this corresponds to the case of $\tilde{M}(u) = 1$, $v = v_\psi$ and $D = D_\psi$ in Eq. (1).

Once $\tilde{\psi}(u)$ is fully defined, the Laplace transformed analytical expression for the resident concentration, $\tilde{c}(u, x)$, or the average mass flux, $\tilde{j}(u, x)$, can be obtained using the method

described in Section 6 (Appendix I). To convert the Laplace-space solution into a time-domain solution, a numerical inverse Laplace transform operation is carried out. The algorithm in `ilap.m` is based on the classical de Hoog et al. algorithm for inverse Laplace transforms (de Hoog, F. R., Knight, J. H., and A. N. Stokes, An improved method for numerical inversion of Laplace transforms. *S.I.A.M. J. Sci. and Stat. Comput.*, 3, 357-366, 1982).

For more details on the 1D/2D solutions and the ψ functions given above, see

- Cortis, A. and B. Berkowitz, Anomalous transport in “classical” soil and sand columns, *Soil Science Society of America Journal*, 68, 1539-1548, 2004. (Erratum: 69, 285, 2005.)
- Dentz, M., A. Cortis, H. Scher and B. Berkowitz, Time behavior of solute transport in heterogeneous media: Transition from anomalous to normal transport, *Advances in Water Resources*, 27(2), 155-173, 2004.
- Cortis A., Y. Chen, H. Scher and B. Berkowitz, Quantitative characterization of pore-scale disorder effects on transport in “homogeneous” granular media, *Physical Review E*, 70, 041108, doi: 10.1103/PhysRevE.70.041108, 2004.

For a comprehensive review and explanation of CTRW and the choice of ψ , see

- Berkowitz, B., A. Cortis, M. Dentz and H. Scher, Modeling non-Fickian transport in geological formations as a continuous time random walk, *Reviews of Geophysics*, 44, RG2003, doi:10.1029/2005RG000178, 2006.
- Edery, Y., A. Guadagnini, H. Scher and B. Berkowitz, Origins of anomalous transport in disordered media: Structural and dynamic controls, *Water Resources Research*, 50, 1490-1505, doi:10.1002/2013WR015111, 2014.
- Nissan, A., I. Dror and B. Berkowitz, Time-dependent velocity-field controls on anomalous chemical transport in porous media, *Water Resources Research*, 53, doi:10.1002/2016WR020143, 2017.

IMPORTANT NOTES:

Note 1: As discussed above, a numerical inversion algorithm for the Laplace transform of the solution is used to obtain a time-domain solution. This algorithm (`ilap.m`) (based on de Hoog, F. R., Knight, J. H., and A. N. Stokes, An improved method for numerical inversion of Laplace transforms. *S.I.A.M. J. Sci. and Stat. Comput.*, 3, 357-366, 1982) must be applied carefully. A crucial fine-tuning parameter is the “tolerance” in the algorithm. For some choices of parameter values (and depending on the shape of the inlet boundary condition, e.g., pulse input or step input), setting the variable `tol=1e-9`, for example, may prevent the `ilap.m` algorithm from converging, while setting the tolerance to `tol=1e-10` gives the correct solution. (This means that γ , the real part of the “ u ” values, is being shifted ($\gamma = \alpha - \log(\text{tol})/(2 \cdot T)$). Also, the value of M in `ilap.m` is critical for the correct solution of the inverse problem. A large value for M is not necessarily the best choice. We recommend working with a value around $M = 10$, but again, this choice depends on the specific of the problem. Thus, while `ilap.m` works well when the fine-tuning parameters are set manually according to a specific problem, it may not be the best choice when a large number of possibilities need to be explored in an “automatic” mode.

Note 2: Further to Note 1 above, and to assist in assessing `ilap.m`, we provide the advanced user with a script to help determine the effectiveness of the numerical Laplace inversion for any particular $\tilde{\psi}(u)$ (not the complete solution). The file `check_ilap.m` is included in the Toolbox package, and tests how well `ilap.m` recovers the $\psi(t)$ in t -space for the ADE, ETA or TPL models. Specifically, this script plots the $\psi(t)$ function in red, and the inverted $\tilde{\psi}(u)$ in blue, as an overlay. If only blue appears on the plot, the match is perfect. Usage instructions are given in the script. For example, type

```
>> check_ilap([1.5 -1 4], linspace(0.1,500,100) ', 'TPL')
```

to examine the TPL with $\beta = 1.5, t_1 = 10^{-1}, t_2 = 10^4$.

Note 3: The normalization scheme (elaborated in section 6) requires knowledge of the mean and median transition times. The mean transition time is calculated in the file `'t_conc.m'`. The median transition time is given as a parameter by the user both for the TPL model and for any other user-defined function, while for the ETA model the median value is given automatically from the file `'median.N.mat'` in the toolbox package for the range $0.05 < \eta < 2$.

3 1D Forward Modeling

The code solves for non-dimensional spatial scale, i.e. in the x range $[0,1]$. The result of converting x in Eq. (1) to dimensionless units is the division of v_ψ and D_ψ by a characteristic length L , and by L^2 , respectively.

3.1 Generating a temporal profile

The easiest way to demonstrate the capabilities of the toolbox is by example. We start first by generating a simple breakthrough curve (temporal profile) for the classical ADE equation. This corresponds to the case of $\tilde{M}(u) = 1$, $v = v_\psi$ and $D = D_\psi$ in Eq. (1). A full list of the toolbox functions and their usage is given in Section 7 (Appendix II).

Assume we have a porous column of length $L = 20$ cm, cross-sectional area $A = 4$ cm² and porosity $n = 0.3$. In general, to simplify the calculations, we wish to work with a column of unit length. It is therefore necessary to normalize the v_ψ and D_ψ parameters so that they both have units of $[t^{-1}]$. These quantities correspond to the MATLAB variables `v` and `D`, respectively. As noted above, recall that the transport velocity v_ψ and the pore fluid velocity v do not, in general, coincide; the same is true for D_ψ and D . The parameters assume identical values only for $\tilde{M} = 1$ (ADE case).

For the ADE solution considered here, we assume a volumetric flux $Q = 2.4$ cm³ min⁻¹, so that the normalized pore velocity is $v = Q/(A L n) = 0.1$ min⁻¹. We choose a value for the dispersion coefficient equal to 2 cm² min⁻¹, and dividing by L^2 we get a normalized value of 0.005 min⁻¹ for D_ψ . To obtain fully dimensionalized units of v_ψ and D_ψ (i.e., cm min⁻¹ and cm² min⁻¹), the values are simply multiplied by L and L^2 respectively.

To see the resulting concentration profile, type the following commands in the MATLAB window:

```
>> v = 0.1;
>> D = 0.005;
```

It is important to remember that MATLAB is case sensitive to variable names. In this particular example (ADE), no parameters are passed to the memory function $\tilde{M}(u)$, so define an empty variable **p**:

```
>> p = [];
```

Create a vector of times **t** [min] at which you want the solution. For example, to calculate 10 concentration values from $t = 1$ to $t = 30$, type

```
>> t = transpose(linspace(1,30,10));
```

The time vector **t** can be a single number and does not have to be linearly spaced, but must be a column vector (note the **transpose** command). Obviously, increasing the density of the points will produce a smoother curve. **Important: t MUST be greater than 0.**

We create the **options** variable (see Section 3.3 for a full description) according to the following structure:

```
>> options = {'ADE','N','step','c.f.norm',1,1,'D'};
```

To obtain the breakthrough curve we finally call the subroutine **t_conc**

```
>> y = t_conc(t,[v D p],options);
```

During the calculation, a “work in progress” message will appear.

The breakthrough curve is stored in the vector **y**, which of course has the same size as the time vector **t**; to view the result, type

```
>> [t y]
ans =
1      0.0000
4.2222    0.0051
7.4444    0.2603
10.6667    0.7023
13.8889    0.9183
17.1111    0.9815
20.3333    0.9962
23.5556    0.9993
26.7778    0.9999
30      1.0000
```

To graphically visualize the breakthrough curve, type the command

```
>> plot(t,y)
```

If the user wishes to save variables created in the workspace, such as **t** and **y**, they can be saved to a separate file by typing

```
save filename t y
```

This command creates a file with a **.mat** extension containing the specified variables in the working folder. For more information concerning the **save** command, the user should refer to the MATLAB documentation by typing

```
help save
```

Figures can easily be saved by using the **save as** or **export** options in the figure window.

3.2 Generating a spatial profile

Resident concentration profiles are straightforward to obtain. For instance, let us assume that we want the resident concentration profile at time $t = 1$, for a step input, with a Dirichlet boundary condition (BC) at the inlet, and a Neumann BC at the outlet. Note that here we are calculating the **resident concentration** so we are using the solution given in Eq. (17) (Section 6, Appendix I).

```
>> clc
>> clear all
>> x = linspace(0,1,8);
>> t = 1;
>> v = 1;
>> D = 0.05;
>> p = [0.075];
>> y = zeros(size(x));
>> for k = 1:length(x);
    options = {'ETA','N','step','c_r_norm',x(k),1,'D'};
    y(k) = t_conc(t,[v D p],options);
end
```

Again, the “work in progress” message will appear. The spatial concentration distribution is stored in the vector **y**, which of course has the same size as the time vector **x**; to view the result, type

```
>> [x ; y]
```

```
ans =
```

| | | | | | | | |
|--------|--------|--------|--------|--------|--------|--------|--------|
| 0 | 0.1429 | 0.2857 | 0.4286 | 0.5714 | 0.7143 | 0.8571 | 1.0000 |
| 1.0000 | 0.9869 | 0.9714 | 0.9500 | 0.9148 | 0.8539 | 0.7590 | 0.6746 |

Though not necessary, it is a good idea to keep spatial profiles as **row** vectors so as not to confuse them with the breakthrough curves (**column** vectors). To graphically visualize the concentration profile, type the command

```
>> plot(x,y)
```

As with the previous example, the **save** command can be used to write the results of a calculation to a separate file.

3.3 The options variable

We describe here the details of the **options** variable, which allows the user to specify the choice of $\tilde{\psi}(u)$ and the associated parameter values, the outlet and inlet boundary conditions, resident concentration (for a spatial distribution) or flux-averaged concentration (for a temporal breakthrough curve) solution, the (relative) length along the column at which the breakthrough curve is to be calculated and the parameters associated with the transport in the case of a reactive tracer.

The next table summarizes the possible values for the **options** variable. Detailed explanation follows below.

| VARIABLE | CONTENT | POSSIBLE VALUES |
|------------|------------------------|--|
| options{1} | model | 'ADE', 'TPL', 'ETA', ' $\tilde{\psi}(u)$ ' |
| options{2} | outlet - type | 'D', 'N' |
| options{3} | inlet - $\tilde{f}(u)$ | 'step', 'pulse', 'square', user defined |
| options{4} | output | 'c.f_norm', 'c.r_norm' |
| options{5} | position - x | [0, 1] |
| options{6} | inlet - f_0 | 1 |
| options{7} | inlet - type | 'D', 'R' |

options{1} The first item of the list in the `options` variable is a string that specifies what kind of memory function is required. Legal values for this value, with the corresponding input parameters \mathbf{p} are:

| MODEL | PARAMETERS OF THE MODEL |
|--------------|--|
| 'ADE' | $\mathbf{p}=[]$ |
| 'TPL' | $\mathbf{p}=[\beta, \log_{10}(t_1), \log_{10}(t_2)]$ |
| 'ETA' | $\mathbf{p}=[\eta]$ |
| user defined | $\mathbf{p}=[t]$ |

A detailed discussion of each of these choices is given in the previous section. To see a plot of these functions, refer to the `t_psi` function documentation. It is possible to implement different $\tilde{\psi}(u)$ functions in problems of forward modeling, by typing the desired expression (in Laplace space, i.e. a function of the Laplace variable u) between quotation marks, e.g., ' $1/(1+u)$ ', with its parameter t_{char} , which is a characteristic time of $\psi(t)$. For any different $\tilde{\psi}(u)$, see directions of implementation in section 7.

options{2} The second item is a string that specifies the kind of boundary condition (BC) for the resident concentration that is imposed at the outlet $x = 1$. It can take the two values: 'N' for the Neumann boundary condition $\partial \tilde{c}_r / \partial x = 0$, and 'D' for the Dirichlet boundary condition $\tilde{c}_r = 0$.

options{3} The third item in the list is a string that specifies the time-dependent $f(t)$ at the inlet $x = 0$. Depending on whether the user specifies the flux (options{7}='R') or the resident concentration (options{7}='D'), the complete function at $x = 0$ reads either $j(x=0, t) = f(t)$ or $c_r(x=0, t) = f(t)$. Possible values for options{3} are: 'step' for a step input, 'pulse' for a pulse, 'square' for a unit square input; these values can be used conveniently instead of entering the functional form. It is also possible to provide a user-defined string input function directly into the field: for instance, the string ' $1/(1+u)$ ' will define an exponentially decaying input. Note that the functional dependence has to be given in Laplace space! Examples of some input functions are given below.

| TIME-DOMAIN $f(t)$ | LAPLACE-DOMAIN $\tilde{f}(u)$ | options{3} |
|-------------------------------|-------------------------------|----------------------------------|
| $\delta(t)$ | 1 | 'pulse' |
| 1 | $1/u$ | 'step' |
| 1 for $(1 > t \geq 0)$ | $(1 - e^{-u})/u$ | 'square' |
| 1 for $(\epsilon > t \geq 0)$ | $(1 - e^{-\epsilon u})/u$ | ' $(1 - e^{-\epsilon u})/u$ ' |
| e^{-at} | $1/(a + u)$ | ' $1/(a + u)$ ' |
| $1 - e^{-at}$ | $1/[u(u/a + 1)]$ | ' $1/[u(u/a + 1)]$ ' |
| $t^n e^{-at}$ | $n!/(a + u)^{n+1}$ | e.g. $n = 2$: ' $2/(a + u)^2$ ' |

options{4} The fourth item in the list is a string that specifies solution for either a flux-averaged concentration ('c.f_norm') or a resident concentration ('c.r_norm'). We use the term “norm” to indicate that we reach a concentration value of 1 at long time in the case of a step input (see section 6). The flux-averaged concentration, c_f , is related to the resident concentration, c_r by the expression

$$\tilde{c}_f = \tilde{c}_r - \alpha \frac{\partial \tilde{c}_r}{\partial \tilde{x}} = \frac{\tilde{j}}{\tilde{M}v_\psi}$$

options{5} The fifth item is a number between 0 and 1, defining the distance x along the column (relative to the length of the column) at which c_f or c_r is being evaluated.

options{6} The sixth item should be set to 1. In previous versions of the toolbox this entry served as a scaling factor. In the current version, this is not needed.

options{7} The seventh item in the list specifies the kind of boundary condition at the inlet. 'R' denotes a Robin boundary condition, meaning the flux j at $x = 0$ is known. 'D' denotes a Dirichlet boundary condition, meaning the resident concentration at $x = 0$ is known.

*** options{8} - options{10} are optional, and should be provided only when modeling sorption with the tracer transport (see section 3.4).

3.4 Adsorbing tracer transport

To model transport that involves adsorption/desorption, changes can be implemented into the single transition pdf, $\tilde{\psi}(x, u)$ only. Equations (1) and (2) are unchanged, and therefore the analytical solutions given in Appendix I apply here as well. One approach to treat sorption is given by Margolin et al. (Continuous time random walk and multirate mass transfer modeling of sorption, *Chemical Physics* 295, 71-80, doi: 10.1016/j.chemphys.2003.08.007, 2003) the single transition pdf of a sorbing tracer can be based on a known function of a passive tracer, $\tilde{\psi}_0$, in the same domain. In Laplace space the new transition pdf takes the form

$$\tilde{\psi}(x, u) = \tilde{\psi}_0(x, u + \Lambda - \Lambda \tilde{\varphi}(u)) \quad (4)$$

where Λ is the average “sticking” rate and $\tilde{\varphi}(u)$ is the Laplace transform of a single “sticking time” pdf, i.e. $u + \Lambda - \Lambda \tilde{\varphi}(u)$ replaces u in the known pdf for a passive tracer. Following Cortis et al. (Transport of *Cryptosporidium parvum* in porous media: Long-term elution experiments and continuous time random walk filtration modeling, *Water Resources Research*, 42, doi: 10.1029/2006WR004897, 2006) we use a sticking time pdf $\tilde{\varphi}(u)$ as a combination of two expressions: a user-defined function, and uniform behavior, weighted by W and $(1 - W)$ respectively, where W is provided by the user. For example, for a power law user-defined function

$$\tilde{\varphi}(u) = W \frac{1}{1 + u^n} + (1 - W) \frac{1}{Tu} \quad (5)$$

where n is a parameter of the user-defined function, and T represents the truncation time for the distribution. To implement this in the CTRW toolbox, three additional items should

be added to the ‘options’ variable, as elaborated below. The item `options{1}` in the sorbing tracer model represents the known form of $\tilde{\psi}_0$. Cortis et al. (2006), for example, solved for the case of ADE, i.e., $\tilde{\psi}_0(u) = \frac{1}{1+u}$.

The following table summarizes the additional items to the `options` variable.

| <code>options{8}</code> | <code>options{9}</code> | <code>options{10}</code> |
|-------------------------|--|---|
| sorption parameters | sorbing function in $\tilde{\varphi}(u)$ | parameters of $\tilde{\varphi}(u)$ |
| $[\Lambda, T, W]$ | 'ADE' | $[\]$ |
| | 'TPL' | $[\beta, \log_{10}(t_1), \log_{10}(t_2)]$ |
| | 'ETA' | $[\eta]$ |
| | user defined | $[\]$ |

Please note that the choice of the parameters in `options{10}` has to match the type of sorbing function in `options{9}` as shown in the table.

`options{8}` The eighth item is a vector specifying the parameters for the new sticking time pdf: $[\Lambda, T, W]$, e.g. $[0.5, 100, 1]$.

`options{9}` The ninth item is a string specifying the user-defined function to be inserted into $\tilde{\varphi}(u)$ and weighted by W . The value can be either an explicit string (a user-defined function of u , e.g. $'1/(1+u^{0.4})'$) or a string specifying one of the built-in function types 'ADE', 'TPL' and 'ETA'.

`options{10}` The last item is a vector of the parameters of the $\tilde{\varphi}(u)$ function specified in the 9th item. If it is a user-defined function, this item should be an empty vector.

4 1D Inverse Problem: Fitting CTRW solutions to data sets

In this section we describe a set of MATLAB m-files which can be used to obtain best-fit parameters, given a set of concentration measurements from an experiment, for the CTRW models described in the previous sections.

It is necessary to provide normalized data sets, i.e. normalize the data by the input concentration. In this way, the concentration will reach a value of 1 at long times in the case of a step input.

Although the basic functions can be compiled into alternative m-file scripts, the toolbox includes examples of how to fit single, as well as multiple, breakthrough curves. A full list of the toolbox functions and their usage is given in (Section 7, Appendix II).

To demonstrate how the functions from the toolbox can be incorporated into an m-file script that solves inverse problems, the toolbox contains a number of example files. With only slight modification, these files can be adapted to suit the specific requirements of the user. This approach is a compromise between simplicity of use and flexibility in meeting specific user needs. The experienced MATLAB programmer can use the examples as a guide to creating codes that can solve more complex situations.

IMPORTANT NOTES:

Note 1: Running a minimization procedure can be a lengthy task, depending on how close the initial values of the parameters are to the optimized ones. For instance, on a PC with a 2.6

GHz Intel processor, the optimization in the example given in Section 4.1 should take a few minutes. Clearly, the better the initial guess for the parameters, the faster the convergence of the minimization algorithm. Also, inaccurate estimation of the parameters may result in non-convergence or convergence to a local minimum. In general, this leads to poor fits which can be seen easily in the resulting plots. In order to rectify the problem, a new choice of initial parameters should be implemented. Given these limitations, it is therefore strongly recommended that the user obtain a reasonable fit manually by trial and error before attempting automatic model fitting.

Note 2: For smoother model results it may be convenient to interpolate the data to a larger number of points. This can be done simply by changing the following variable in the input m-file from its default value (1000) to a larger number:

```
>> points=1000;
```

4.1 Fitting single breakthrough curves

To solve the inverse problem for the TPL model and a single breakthrough curve, the toolbox contains two m-files that require editing. One file contains the experimental data to be fitted, named in this example `data_1.m`. The file consists of a two-column matrix; the first column corresponds to the time at which measurements were made while the second represents normalized concentration data. To edit this file, type

```
>> edit data_1.m
```

in the MATLAB command window. A new window will open and the experimental data can be modified. Note that experimental time can begin at time $t = 0$.

The second m-file `input_1.m`, contains initial guesses of the parameter values (v D p), input for the `options` variable, as well as a list of commands that perform the error minimization algorithm. Before running the file, nine compulsory fields must be defined at the beginning of the `input_1.m` file; to edit this file type

```
>> edit input_1
```

The beginning of the file should appear as follows

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FIELD 1
% Choice of model: ADE, TPL, ETA
% (see 'options{1}' in the user guide)
mod_type = 'TPL';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FIELD 2
% Initial parameter guess
% Input of v and D must be in units of 1/TIME: i.e., divide the dimensional
% v and D by L (column length) and L^2, respectively.
v      = 0.01 ; % v_psi [T^-1]
D      = 2.2e-005 ; % D_psi [T^-1]
beta   = 1.8; % beta
t2     = 10^6; % cut-off time t2

%CHOOSE HERE: t1 specified or t1 calculated automatically
t1      = 10^-1.9; % t1 as an independent variable (specified)
% [t1, t2] = t1_calc(t2,v,D,beta); % calculate t1

p = [beta, log10(t1), log10(t2)]; % parameters must be consistent with model in Field 1
```

```

% NOTE: Do not forget to reconvert to real dimensional units of v and D;
% multiply v and D by L (column length) and L^2, respectively.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FIELD 3
% Relative location (normalized) of measurement along the column
col_pos = 1;
% measurement position, should be in the range [0,1]
% (see 'options{5}' in the user guide)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FIELD 4
% Boundary condition at the outlet
% String may be 'N' (Neumann) or 'D' (Dirichlet).
% (see 'options{2}' in the user guide)
outlet_BC = 'N';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FIELD 5
% Boundary condition at the inlet
% String may be 'R' (Robin) or 'D' (Dirichlet).
% (see 'options{7}' in the user guide)
inlet_BC = 'R';
% Time dependence f(t) at the inlet.
% String options include 'step', 'pulse', or 'square'.
% Or user defined functional dependence (in Laplace space).
% (see 'options{3}' in the user guide)
inlet_ft = 'step';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FIELD 6
% Solve for:
% - either the resident concentration
% - or the flux-averaged concentration
% String may be 'c_f_norm' (flux-averaged) or 'c_r_norm' (resident).
% (see 'options{4}' in the user guide)
out_type = 'c_f_norm';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FIELD 7
% Name of input data file; The .m suffix must be omitted
data_1

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FIELD 8
% Name of output data file
output = 'output.txt';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FIELD 9
% Generate the result for the first guess only [0]
% or initiate minimization [1]
minimize = 0;
optimization_type = 0; % [0] Unconstrained Optimization [1] Constrained Optimization

% Note that use of the Constrained Optimization function type ('fminsearchbnd.m')
% requires lower and upper bounds (see input_1.m file, lines 154-155, parameters LB, UB)

%-----
% MAIN SCRIPT

```

Below the line marked MAIN SCRIPT is a series of subroutines that calculate the normalized error and perform the minimization procedure. Changes to these lines of code are not required and should be performed only if the user is certain that modifications to the code are necessary.

In this example, the initial guess for the input parameters for v and D in Field 2 must be

given dimensions of 1/time, i.e., divide the dimensional v and D by L (column length) and L^2 , respectively. Furthermore, care must be taken to ensure that the time units in the data file, as well as the units of all the parameters in `input_1.m`, are consistent.

Once the data and input files have been edited to contain the desired information, the user can see how the initial guess compares to the experimental data by setting Field 9 to [0]. To run the code, type in the command window

```
>> input_1
```

As well as generating a series of variables in the workspace, the code also generates a figure which can be used to compare the model with the experimental data. An additional figure of $1 - c$ is plotted in log-log scale, in order to highlight the long tailing. The figures can be modified easily by editing the code at the end of the m-file.

From the sample figure created by the `input_1.m` file, it can be seen that the initial guess values do not yield a good fit to the experimental data. The “best fit” process can be initiated by setting Field 9 to [1] and running the code again. In essence, the best fit process tries to identify the model parameters (i.e., v , D , β , t_1 , and t_2 for the TPL model) that will minimize the normalized error between the CTRW model and experimental data. The conditions required for termination of the process can be found in the documentation of the `my_fminsearch` function. When the code has finished running, a good fit should be obtained. On completion of the error minimization process, a text file containing a summary of the initial and final parameters is created, the name of which is defined in Field 8. A vector containing the model parameters can be viewed by typing `vDp` in the command window.

If the user wishes to examine the breakthrough curve at earlier or later times to those generated automatically at the end of the minimization procedure, the parameters from the `vDp` variable can be used as input for the `t_conc` function discussed in sections 3.1 and 3.2.

IMPORTANT NOTES:

Note 1: t_1 , as a characteristic time, can be determined from the transport parameters (i.e., v_ψ , D_ψ , β and t_2). Thus, t_1 **should not be considered a fitting parameter**. For full details of the equations and development, see Nissan, A., I. Dror and B. Berkowitz, Time-dependent velocity-field controls on anomalous chemical transport in porous media, *Water Resources Research*, 53, doi:10.1002/2016WR020143, 2017 (and *Supplementary Material* file therein). For direct calculation of t_1 , activate `t1_calc` function in Field 2. **IMPORTANT:** Direct calculation of t_1 requires that $t_2 \gg t_1$, which is the usual case for realistic situations of transport in porous media.

Note 2: The optimization process can be performed using a constrained multivariable function `fminsearchbnd` (in contrast to the unconstrained method of `my_fminsearch`), when the user can apply bounds to each of the CTRW input parameters (i.e., `vDp`). To use this optimization algorithm, type in Field 9 `optimization_type = 1`, and set the bounds in `input_1.m` file, lines 159-160, parameters LB, UB. Note also that to constrain a specific parameter as a constant value (so that it is not optimized or modified in any way), simply multiply the parameter value by 1.0 in each of LB and UB.

4.2 Comparing fits for two different models

At some point, it is likely that the user will wish to compare the results of different models for the same set of concentration data. In this example, we carry out the minimization

simultaneously for two models (TPL and ADE). The input file for this example can be viewed by typing

```
>> edit input_2
```

As in the previous example, 9 fields must be defined. In this case, Field 1 now requires two inputs: TPL and ADE. By comparing the lines of code that appear below the MAIN SCRIPT section, additional changes to the code can be observed. Although the same result could be achieved by running the `input_1.m` file for both the TPL and ADE models, the main advantage of the `input_2.m` file is that different variables for the two solutions are created which can then be manipulated in the workspace. Because two models are given, there exist as well two parameter vectors which can be viewed by typing `vDp1` and `vDp2` in the command window.

4.3 Fitting multiple breakthrough curves

In some instances, concentration breakthrough curves may be obtained from different locations simultaneously within a column. The minimization can be carried out separately on each data set, as in the previous example, or by using the combined data sets. To show how multiple data sets can be employed, the toolbox also contains an example of how the ADE model is fitted to two curves.

The sample data sets used in this demonstration can be found in the files `data_2.m` and `data_3.m`. The input file for this example can be viewed by typing

```
>> edit input_3
```

in the command window. As with the previous example, nine compulsory fields are listed, although, in Field 7 two data files are now given instead of one. Furthermore, in the MAIN SCRIPT section of the code, a number of changes have been made. These can be seen by comparing the code to the equivalent section in the `input_1.m` file. Note that the subroutine which calculates the error is called `diff_norm_multi`, a function that can take two input data sets. If the user wishes to compare three or more data sets, this subroutine must also be altered, along with the appropriate lines of code in `input_3.m`.

4.4 Fitting for selected parameters

Rather than performing the minimization procedure by varying all the parameter values, the user may wish to constrain selected parameters (for example v and D) and obtain an optimal fit by varying the remaining ones.

As explained at the end of section 4.1, Note 2, the optimization process can be performed using a constrained multivariable function, wherein the user can apply bounds to each of the CTRW input parameters (i.e., `vDp`). To constrain a specific parameter as a constant value (so that it is not optimized or modified in any way), simply multiply the parameter value by 1.0 in each of LB and UB.

4.5 Fitting for a square input of duration ϵ

In the previous examples, step concentration inputs to the systems have been considered; however, experiments often involve a pulse input or step input of limited duration. In this example, we demonstrate how such a boundary condition can be implemented, and the appropriate input file can be viewed by typing

```
>> edit input_4
```

In Field 5, the input has been changed to `bound_input = ['1/u*(1-exp(-5*u))']`, which corresponds to a step input with a duration of $\epsilon = 5$ time units.

4.6 Fitting for reactive tracers

An example for fitting a BTC for the 1D transport of a tracer which undergoes sorption/desorption is provided in 'input_5.m' file. It is the same method used in `input_1`, only for a sorbing tracer (see section 3.4). The minimization process (which is executed when FIELD 9 is set to [1]) changes the parameters v, D, p as in the non-sorbing example of `input_1`, while the sorption parameters are assumed as known parameters, and remain constant. The sample data used in this demonstration can be found in the file `data_5`. To edit the input file, type

```
>> edit input_5
```

The same rules of subsection 4.1 apply here, with the three following changes: Provide the sorption function that comprises the part of $\tilde{\varphi}(u)$ that is multiplied by W (equivalent to `options{9}`) under 'sor_type' in FIELD 1: e.g. `sor_type=['1/(1 + u0.3)']`; Set the parameters of the sorbing tracer model (equivalent to `options{8}`) to 'p_sor' in FIELD 2: e.g.

```
p_sor=[0.5,100,1]; % [Lambda,T,W].
```

Finally provide the parameters of $\tilde{\varphi}(u)$ (equivalent to `options{10}`) in the variable 'pa' in the same field: e.g., `pa=[]`; these parameters must be consistent with 'sor_type'.

5 2D Forward Modeling: A sample case

Two-dimensional resident concentration profiles can be computed analytically for specific cases of boundary conditions. One example is given in “Dentz, M., A. Cortis, H. Scher and B. Berkowitz, Time behavior of solute transport in heterogeneous media: Transition from anomalous to normal transport, *Advances in Water Resources*, 27(2), 155-173, 2004.” In one case, the resident concentration of a solute evolving from a point-like injection at $\vec{s} = (0, 0)$ and at time $t = 0$ is investigated. The function `t_resid_2d`, included in the CTRW toolbox, computes these solutions. The parameters which can be played with despite of flow velocity and dispersion are the type of model, including its parameters, and the height of the injection peak at the inlet.

To use the function, first define the velocity and longitudinal and transverse dispersion coefficients

```
>> clc
>> clear all
>> v = 0.1;
>> DL = 5e-3;
>> DT = 5e-4;
```

Then, define the parameters \mathbf{p} of the $\tilde{\psi}(u)$, for instance the truncated power law model (TPL) for which we assume the following parameters $\mathbf{p} = [\beta \log_{10}(t_1) \log_{10}(t_2)]$

```
>> p = [0.75 log10(1) log10(500)];
```

Create a vector of times \mathbf{t}

```
>> t = transpose( linspace(500,1000,3));
```


Create two vectors `x1` and `x2` which represent the grid of points in the principal direction of the fluid flow (1) and in the orthogonal direction to the flow (2), respectively. Set them equal to

```
>> x1 = linspace(0.1,20,10);
>> x2 = linspace(-1,1,10);
```

For the sake of clarity, the lengths of the time and space vectors were chosen here to lessen the calculation time; set them to larger values in real computations.

Create a `options` variable, which differs from the one in the 1D case, since not all options are analytically tractable.

```
>> options = {'TPL',x1,x2,1};
```

The one in `options{4}` represents the height of the injection peak. To calculate the concentration in the domain at the three different times, we call the subroutine `t_resid_2d`

```
>> y = t_resid_2d(t,[v DL DT p], options);
```

As before, the “work in progress” message will appear.

The matrix containing the values of the concentrations at, for example, the second time step can be viewed by typing

```
>> y(:, :, 1)
```

To visualize the three contour plots corresponding to the three times in `t`, type in the MATLAB window the following commands

```
>> for k = 1:length(t);
>> figure(k);
>> contour(x1,x2,y(:, :, k));
>> colorbar, grid on;
>> end;
```

For the meaning of the parameters in the MATLAB `contour` function, please refer to its online documentation. As before, any of the workspace variables can be saved using the `save` command.

6 Appendix I: The Laplace-Domain Analytical Solutions

The code in the CTRW toolbox is based on the numerical inversion of the analytical solution to Eq. (1) from the Laplace-domain into the time-domain. Here, we demonstrate how such an analytical solution can be obtained for a defined set of boundary conditions.

In the following, we consider a finite domain of unit length $x = [0, 1]$ with two possible boundary conditions (BCs) at the outlet ($x = 1$): a Neumann ('N') condition $\partial \tilde{c} / \partial x = 0$ at $x = 1$, and a Dirichlet ('D') condition $\tilde{c} = 0$ at $x = 1$. We can then develop exact 1D analytical solutions of Eq. (1).

The toolbox code is based on the solution for a pulse input function. Using the convolution property of Laplace transforms, the code accepts any user-defined flux or resident concentration input function and multiplies it by the Laplace transform solution for a pulse input.

In the following we choose a step input function for demonstration which allows us to visualize the (non-zero) long time limit ($t \rightarrow \infty$) for this input function, used for normalization.

To simplify the typesetting of the solutions we define the variables:

$$w \equiv \sqrt{1 + \frac{4u\alpha}{\tilde{M}v}} \quad , \quad (6)$$

$$z \equiv \frac{(1+w)x}{2\alpha} \quad , \quad (7)$$

$$k \equiv \frac{2w - xw + x}{2\alpha} \quad , \quad (8)$$

The long time limit is obtained by the identity: $\lim_{t \rightarrow \infty} f(t) = \lim_{u \rightarrow 0} u \tilde{f}(u)$. The long time limit of the memory function, $\tilde{M}(0)$, is calculated as follows:

First note that $\tilde{\psi}(u) = \int_0^\infty e^{-ut} \psi(t) dt$ and $\tilde{\psi}(0) = \int_0^\infty \psi(t) dt = 1$. Using the expansion $e^{-ut} = 1 - ut + \dots$, we obtain from the Laplace definition $\tilde{\psi}(u) = \int_0^\infty (1 - ut + \dots) \psi(t) dt = 1 - ut_{mean}$, where t_{mean} is the first moment of $\psi(t)$, i.e., $t_{mean} = \int_0^\infty t \psi(t) dt$. Substituting this into eq. 2 yields $\lim_{u \rightarrow 0} \tilde{M}(u) = t_{char}/t_{mean}$ where t_{char} is a characteristic time and t_{mean} is the first moment of $\psi(t)$, calculated in the m-file `t_conc.m`. For example, for the TPL type $\psi(t)$, $t_{mean} = \frac{\exp(-\tau)\tau^{-\beta}[-t_1 + \exp(\tau)\tau^\beta(t_1 + \beta t_2)\Gamma(1-\beta, \tau)]}{\beta\Gamma(-\beta, \tau)}$ where $\tau = t_1/t_2$.

The normalized solutions 'c_f_norm' and 'c_r_norm' (see 'options{4}' variable in section 3.3) are obtained by dividing the solution by the long time saturation value for a step input. For example, for a step input of steady flux (Robin inlet) and Neumann boundary condition at the outlet, the plateau of the flux-averaged concentration in the BTC, $\lim_{t \rightarrow \infty} c_f = \lim_{t \rightarrow \infty} j/Mv$, should reach the value of t_{mean}/t_{char} (see eq. 12). Therefore to correct for this factor and achieve a plateau of the value 1, we divide the analytical solution by the factor t_{mean}/t_{char} (in the m-file `t_conc.m`).

6.1 Inlet-type: 'R' (Robin), Inlet-temporal-function: 'step'

In the first example we assume the flux at the inlet ($x = 0$) to be given. We consider a steady state flux $\tilde{f}(u) = \tilde{j}(x = 0, u) = v/u$, corresponding to a step input of height v in the time-domain, and a Neumann ('N') boundary condition at the outlet. The exact 1D analytical solution of Eq. (1) for the Laplace transformed resident concentration $\tilde{c}_r(u, x)$ can be obtained by using, for example, Mathematica (6.0):

```
> DSolve[{u*c[x] + M*v*(c'[x] - alpha*c''[x]) == 0, M*v*(c[0] - alpha*c'[0]) == v/u,
c'[1] == 0}, c[x], x]
```

The expressions for the Laplace transformed resident concentration $\tilde{c}_r(u, x)$, and the Laplace transformed average mass flux $\tilde{j}(u, x) = \tilde{M}v\tilde{c}_f = \tilde{M}v(\tilde{c}_r - \alpha \frac{\partial \tilde{c}_r}{\partial x})$, take the form

$$\tilde{c}_r(u, x) = \frac{1}{u} \frac{2}{\tilde{M}} \frac{(w+1)e^k + (w-1)e^z}{(w+1)^2 e^{w/\alpha} - (w-1)^2} \quad , \quad (9)$$

$$\tilde{j}(u, x) = \frac{v}{u} \frac{(w+1)^2 e^k - (w-1)^2 e^z}{(w+1)^2 e^{w/\alpha} - (w-1)^2} \quad , \quad (10)$$

For the long time of their Laplace Inverse we get

$$\lim_{t \rightarrow \infty} c_r(t, x) = \lim_{u \rightarrow 0} u \tilde{c}_r(u, x) = \frac{t_{mean}}{t_{char}} \quad (11)$$

$$\lim_{t \rightarrow \infty} j(t, x) = \lim_{u \rightarrow 0} u \tilde{j}(u, x) = v \quad (12)$$

Similar expressions are obtained for a Dirichlet ('D') boundary condition $\tilde{c} = 0$ at $x = 1$

$$\tilde{c}_r(u, x) = \frac{1}{u} \frac{2}{\tilde{M}} \frac{e^k - e^z}{(w+1)e^{w/\alpha} + (w-1)} \quad , \quad (13)$$

$$\tilde{j}(u, x) = \frac{v}{u} \frac{(w+1)e^k + (w-1)e^z}{(w+1)e^{w/\alpha} + (w-1)} \quad . \quad (14)$$

For the long time of their Laplace Inverse we obtain

$$\lim_{t \rightarrow \infty} c_r(t, x) = \lim_{u \rightarrow 0} u \tilde{c}_r(u, x) = \frac{t_{mean}(1 - e^{\frac{x-1}{\alpha}})}{t_{char}} \quad (15)$$

$$\lim_{t \rightarrow \infty} j(t, x) = \lim_{u \rightarrow 0} u \tilde{j}(u, x) = v \quad (16)$$

6.2 Inlet-type: 'D' (Dirichlet), Inlet-temporal-function: 'step'

In the second example we assume the resident concentration at the inlet ($x = 0$) to be given. We consider a unit resident concentration $\tilde{c}_r = u^{-1}$ at $x = 0$, corresponding, as before, to a step input in the time domain. Again we solve Eq. (1).

The Mathematica (6.0) code for $\tilde{c}_r(u, x)$ with the Neumann ('N') boundary condition $\partial \tilde{c}_r / \partial x = 0$ at the outlet reads now:

```
> DSolve[{u*c[x] + M*v*(c'[x] - alpha*c''[x]) == 0, c[0] == 1/u, c'[1] == 0}, c[x], x]
```

The expressions for the Laplace-transformed resident concentration $\tilde{c}_r(u, x)$, and the Laplace-transformed average mass flux $\tilde{j}(u, x)$ with the Neumann boundary condition at the outlet then become

$$\tilde{c}_r(u, x) = \frac{e^k}{u} \frac{(w+1) + (w-1)e^{\frac{w(x-1)}{\alpha}}}{(w+1)e^{w/\alpha} + (w-1)} \quad , \quad (17)$$

$$\tilde{j}(u, x) = \tilde{M}v \frac{e^k}{2u} \frac{(w+1)^2 - (w-1)^2 e^{\frac{w(x-1)}{\alpha}}}{(w+1)e^{w/\alpha} + (w-1)} \quad , \quad (18)$$

For the long time of their Laplace Inverse we obtain

$$\lim_{t \rightarrow \infty} c_r(t, x) = \lim_{u \rightarrow 0} u \tilde{c}_r(u, x) = 1 \quad (19)$$

$$\lim_{t \rightarrow \infty} j(t, x) = \lim_{u \rightarrow 0} u \tilde{j}(u, x) = \frac{t_{char}v}{t_{mean}} \quad , \quad (20)$$

and for a Dirichlet ('D') boundary condition $\tilde{c}_r = 0$ at the outlet $x = 1$ we obtain

$$\tilde{c}_r(u, x) = \frac{e^k}{u} \frac{1 - e^{\frac{w(x-1)}{\alpha}}}{e^{w/\alpha} - 1} \quad , \quad (21)$$

$$\tilde{j}(u, x) = \tilde{M}v \frac{e^k}{2u} \frac{(w+1) + (w-1)e^{\frac{w(x-1)}{\alpha}}}{e^{w/\alpha} - 1} \quad . \quad (22)$$

For the long time of their Laplace Inverse we obtain

$$\lim_{t \rightarrow \infty} c_r(t, x) = \lim_{u \rightarrow 0} u \tilde{c}_r(u, x) = \frac{e^{1/\alpha}(1 - e^{-\frac{x-1}{\alpha}})}{e^{1/\alpha} - 1} \quad , \quad (23)$$

$$\lim_{t \rightarrow \infty} j(t, x) = \lim_{u \rightarrow 0} u \tilde{j}(u, x) = \frac{t_{char} v e^{1/\alpha}}{t_{mean}(e^{1/\alpha} - 1)} \quad . \quad (24)$$

7 Appendix II: CTRW Toolbox File Descriptions

We describe here the files which compose the CTRW toolbox. To get a list of the files type

```
>> ls
```

in the MATLAB command window. For the use of the individual files type in the MATLAB window **help** followed by the name of the file, for instance

```
>> help L_psi
```

According to the MATLAB specifications, in the online help, keywords are capitalized to make them stand out. Always type commands in lowercase since all command and function names are actually in lowercase.

If the user wishes to implement any different functional form of $\tilde{\psi}(u)$, the following additions should be made for forward modeling:

1. Tag the new model with a new name, which will be used in **options{1}**. In the following steps add a new case for the new model, as is done for the cases of 'ADE', 'TPL' etc.
2. The median and mean transition times should be calculated in the file **t_conc.m**. This is used for normalizing the breakthrough curves for a step input.
3. The memory function should be calculated in the file **mem.m**.
4. Define the function $\tilde{\psi}(u)$ in Laplace space and its parameters in the file **L_psi.m**.
5. Define the function $\psi(t)$ in the file **t_psi.m**.

| file name | DESCRIPTION |
|--------------|---|
| L_resid_2D.m | <hr/> <p>L.RESID.2D Solution for the Laplace transformed resident concentration for a 2D stationary domain, at a given position 'x1,x2' with constant velocity (v), longitudinal dispersivity (DL/v) and transverse dispersivity (DL/v).</p> <p>Y = L.RESID.2D(U_VEC,VARPAR,OPTIONS)</p> <p>u_vec := vector of Laplace variables varpar := vector containing the velocity , the longitudinal and transverse dispersion and the psi parameters options:= cell array containing information on the model, the outlet boundary condition, the inlet input, the kind of concentration, and coordinates of the point at which the BTC is taken. y := vector of Laplace transformed concentrations</p> <p>Example y = L_resid_2D([1+i 1-i],[1 0.05 0.005 [0.75 log(1) ... log(100)]],['TPL',0.2,0.1,1})</p> <p>See also t_resid_2D</p> <hr/> |
| L_conc.m | <hr/> <p>L.CONC Solution for the Laplace transformed total flux averaged (or resident concentration) for a 1D stationary domain, at a given position 'x' with constant velocity (v) and dispersivity (D/v) .</p> <p>Y = L.CONC(U_VEC,VARPAR,OPTIONS);</p> <p>u_vec := vector of Laplace variables varpar := vector containing the velocity , the dispersion and the psi parameters options:= cell array containing information on the model, the outlet boundary condition, the inlet input, the kind of concentration, and the section. y := vector of Laplace transformed concentrations</p> <p>Example: y = L_conc([1+i 1+0.1*i],[1 0.05 [0.75 log10(1) ... log10(100)]],['TPL', 'N', 'step', 'c.f.norm',1,1, 'R' })</p> <p>See also t_conc</p> <hr/> |

| | |
|---------------------|--|
| L_psi.m | <hr/> <pre> L_PSI Calculates the Laplace transform of the $\psi(t)$ function for the different models ADE, TPL, ETA. RES = L_PSI(U,PARAM,FUNPAR) u := Laplace variable param := parameters of the psi function funpar:= string specifying the model res := values of the function $\psi(t)$ Usage example: y = L_psi(1+i,[0.75 1 0.1],{'TPL'}) See also T_PSI </pre> <hr/> |
| calcola_coeff_eta.m | <hr/> <pre> CALCOLA_COEFF_ETA Computes the first 28 coefficients of the Taylor expansion for the ETA psi-function. The coefficients are stored in the global variable c_mat </pre> <hr/> |
| check_ilap.m | <hr/> <pre> CHECK_ILAP Plots the ADE/TPL/ETA psi function, to test how well the inverse Laplace script (ilap.m) recovers the psi in t-space. psi(t) is plotted in red; the inverted psi(u) is plotted in blue, as \Rightarrow an overlay. If only blue appears on the plot, the match is perfect. CHECK_ILAP(P,T,MOD) t := time vector p := parameters of the psi function mod:= model type ('ADE', 'TPL' or 'ETA') Usage example: check_ilap([1.5 -1 4],linspace(0.1,500,100)', 'TPL') </pre> <hr/> |

| | |
|------------------|--|
| clencurt.m | <hr/> <p> CLENCURT Computes nodes x (Chebyshev points) and weights w for Clenshaw–Curtis quadrature. Used to evaluate numerically the Laplace transform of the 'ETA' $\psi(t)$ function. </p> <p> $[X,W] = \text{CLENCURT}(N)$ </p> <p> N := number of Chebyshev points x := Chebyshev points w := Clenshaw–Curtis quadrature weights </p> <p> Usage example: $[x,w] = \text{clencurt}(64)$ </p> <p> See also QIS, CLENCURT.COEFF </p> <hr/> |
| clencurt_coeff.m | <hr/> <p> CLENCURT.COEFF Computes nodes x (Chebyshev points) and weights w for Clenshaw–Curtis quadrature. Used to evaluate numerically the Laplace transform of the 'ETA' $\psi(t)$ function. </p> <p> This function calls upon the CLENCURT function, and evaluates Chebyshev points and Clenshaw–Curtis quadrature weights for a number of Chebyshev points equal to 2^i, $i=1:10$. </p> <p> The results are stored in the global variables x_mat, w_mat </p> <p> Usage example: <code>clencurt_coeff</code> </p> <p> See also QIS, CLENCURT </p> <hr/> |

| | |
|-------------------|---|
| diff_norm.m | <hr/> <p>DIFF_NORM Computes the norm of the difference between two vectors Given tha data : <code>t = data(:,1)</code> , <code>y = data(:,2)</code> . and the model function : <code>function(t,param,funpar)</code></p> <p><code>[ERR,MODEL] = DIFF_NORM(VARPAR, TIPO)</code></p> <p><code>varpar</code> := first argument of the model function <code>tipo</code> := second argument of the model function <code>err</code> := norm of the difference between the vectors (or the log of the vectors) <code>model</code> := value of the model function at the data points</p> <p>Example: <code>data = [</code> <code> 0.1000 -0.0064</code> <code> 0.2154 0.0124</code> <code> 0.4642 0.1593</code> <code> 1.0000 0.5104</code> <code> 2.1544 0.7817</code> <code> 4.6416 0.8576</code> <code> 10.0000 0.9281</code> <code> 100.0000 0.9846</code> <code>];</code> <code>options = { 'ADE', 'N', 'step', 'c-f.norm', 1, 1, 'R' };</code> <code>s = [1 0.05 0.75];</code> <code>[err,model] = diff_norm(s,{data, 't.conc', options})</code></p> <hr/> |
| diff_norm_multi.m | <hr/> <p>DIFF_NORM_MULTI Computes the norm of the difference between two sets of vectors. Given tha data : <code>t1 = data1(:,1)</code> , <code>y1 = data1(:,2)</code> <code>t2 = data2(:,1)</code> , <code>y2 = data2(:,2)</code> . and the model function : <code>function(t,param,funpar)</code></p> <p><code>[ERR,MODEL1, MODEL2] = DIFF_NORM_LIM(VARPAR, TIPO1, TIPO2)</code></p> <p><code>varpar</code> := first argument of the model function <code>tipo1</code> := second argument of the model function <code>tipo2</code> := second argument of the model function <code>err</code> := norm of the difference between the vectors (or the log of the vectors) <code>model1</code> := value of the model function at the data points of dataset 1 <code>model2</code> := value of the model function at the data points of dataset 2</p> <p>See DIFF_NORM and DIFF_NORM_LIM</p> <hr/> |

| | |
|----------------|---|
| <p>gammq.m</p> | <hr/> <p>GAMMQ Calculates the incomplete Gamma function for non-integer values of Z, by calling the file 'gammainc07.m'</p> <p>RES = GAMMQ(A,Z)</p> <p>A := parameter of the incomplete Gamma function Z := point of evaluation of the incomplete Gamma function RES := value of the incomplete Gamma function</p> <p>Usage example: y = gammq(2,2.3)</p> <hr/> |
| <p>ilap.m</p> | <hr/> <p>ILAP t := column vector of times F := User defined laplace-space function (string refering to an m-file) P := optional parameters to be passed on to F f := vector of real-space values f(t)</p> <p>F = ILAP(t, F, P);</p> <p>Example:</p> <p>function y = test(u,p) <i>% save these three lines</i> a = p(1); b = p(2); <i>% ... in the file</i> y = a./(u+b) ; <i>% ... 'test.m'</i></p> <p>a = 2 ; b=3; t = transpose(linspace(0.01,1,50)); f = ilap(t, 'test',[a b]); y = a*exp(-b*t); plot(t,y,'b',t,f,'r.')</p> <p>Reference: de Hoog, F. R., Knight, J. H., and Stokes, A. N. (1982). An improved method for numerical inversion of Laplace transforms. S.I.A.M. J. Sci. and Stat. Comput., 3, 357–366.</p> <hr/> |

| | |
|-----------------|---|
| mem.m | <hr/> <p>MEM Laplace transformed memory function $\tilde{M}(u)$ for the different models ADE, TPL, ETA.</p> <p>RES = MEM(U,PARAM,FUNPAR)</p> <p>u := Laplace variable param := parameters of the psi function funpar:= string specifying the model res := values of the function $\tilde{M}(u)$</p> <p>Usage example: y = MEM(1+i,[1 0.05 0.75],{'TPL', 'N', 'step', 'c_f_norm', 1, 1, 'R'})</p> <p>See also L_PSI, QIS</p> <hr/> |
| my_fminsearch.m | <hr/> <p>MY.FMINSEARCH This is a slight modification of the FMINSEARCH Multidimensional unconstrained nonlinear minimization (\Rightarrow Nelder–Mead) to display the output of the nonlinear iterations.</p> <p>Type 'help fminsearch' for the usage of this function</p> <hr/> |
| fminsearchbnd.m | <hr/> <p>FMINSEARCHBND: FMINSEARCH, but with bound constraints by \Rightarrow transformation</p> <p>usage: x=FMINSEARCHBND(fun,x0) usage: x=FMINSEARCHBND(fun,x0,LB) usage: x=FMINSEARCHBND(fun,x0,LB,UB) usage: x=FMINSEARCHBND(fun,x0,LB,UB,options) usage: x=FMINSEARCHBND(fun,x0,LB,UB,options,p1,p2,...) usage: [x,fval,exitflag,output]=FMINSEARCHBND(fun,x0,...)</p> <hr/> |

| | |
|----------|---|
| qis.m | <hr/> <p>QIS Laplace transformed memory function $\tilde{M}(u)$ for the different ETA $\psi(t)$ model.</p> <p>SOMMA = QIS(PARAM,U,FUNPAR)</p> <p>u := Laplace variable param := parameters of the psi function funpar:= string specifying the model res := values of the function $\psi(t)$</p> <p>Usage example: y = qis([0.75 1 0.1],1+i,{ 'TPL' })</p> <p>See also T_PSI</p> <hr/> |
| t_conc.m | <hr/> <p>T.CONC Solution for the total flux averaged (or resident concentration) for a 1D statiionary domain, at a given position 'x' with constant velocity (v) and dispersivity (D/v) .</p> <p>Y = T.CONC(T,VARPAR,OPTIONS);</p> <p>t := time vector varpar := vector containing the velocity, the dispersion and the psi parameters options:= cell array containing information on the model, the outlet boundary condition, the inlet input, the kind of concentration, and the section. y := vector of concentrations</p> <p>(see the User's Guide for a detailed description)</p> <p>Example: y = t_conc(transpose(logspace(-1,2,10)), [1 0.05 [0.75 ... log(1) log(100)]], { 'TPL', 'N', 'step', 'c_f_norm', 1, 1, 'R' })</p> <p>See also L_conc</p> <hr/> |

| | |
|--------------|--|
| t_psi.m | <hr/> <p>T_PSI Calculates the inverse Laplace transform of the \$\tilde{\psi}(u)\$ function</p> <p>RES = L_PSI(U,PARAM,FUNPAR)</p> <p>t := time vector param := parameters of the psi function funpar:= string specifying the model res := Laplace transformed psi function \$\tilde{\psi}(u)\$</p> <p>Usage example: clear all eta = 0.075; calcola_coeff(eta); t = transpose(logspace(-4,4,100)); y = t_psi(t,[eta],{ 'ETA' }); loglog(t,y)</p> <p>See also L_PSI</p> <hr/> |
| t_resid_2d.m | <hr/> <p>T_RESID_2D</p> <p>t := time vector varpar := vector containing the velocity, the dispersion and the psi parameters options:= cell array containing information on the model, the outlet boundary condition, the inlet input, the kind of concentration, and the coordinates of the section at which the concentration is computed. y := vector of concentrations</p> <p>Y = T_RESID_2D(t);</p> <p>(see the User's Guide for a detailed description)</p> <p>Example y=t_resid_2d(transpose(linspace(7,20,3)), ... [0.1 5e-3 5e-4 [0.75 log(1) log(100)]] , ... { 'TPL' , [], linspace(0.1,20,5) , linspace(-1,1,4) , 1 });</p> <p>See also L_resid_2d</p> <hr/> |