MEDIALAB (v 1.0) – User manual

Version 1.0

March 1st, 2014

Babak Shafei (babak.shafei@eawag.ch)

Updates, examples and walkthrough are posted on our website at:

1. Overview

MEDIALAB (***M***odeling ***E***arly ***DIA***genesis using MAT***LAB***) is an early diagenesis model, which calculates concentrations and fluxes of chemical species as well as rates of all the biogeochemical pathways at each depth of aquatic sediments for a specific time period.

A system of partial differential equations corresponding to early diageneis equation are automatically generated through MATLAB's symbolic programming capabilities and solved using MATLAB's built-in solver *pdepe* to evaluate temporal and spatial distribution of chemical species.

MEDIALAB is executed through the MATLAB home screen. The execution of MEDIALAB requires an active installation of MATLAB (Version 7.6 release R2008a or later[1]). It is recommended to allow at least 1GB of space on the hard drive for the model output and 1GB of contiguous random-access memory for the initialization routine.

2. Theoretical model

Diagenesis equation

The general, one-dimensional continuum representation of coupled mass transport and biogeochemical reaction in aquatic sediments is expressed by a set of PDEs of the form (Aguilera et al., 2005):

$$\frac{\partial(\varepsilon C_i)}{\partial t} = \left[\frac{\partial}{\partial x}\left(D\varepsilon\frac{\partial(C_i)}{\partial x}\right) - \frac{\partial}{\partial x}(\vartheta\varepsilon C_i)\right] + \sum \varepsilon r_i(x,t,C_i,\dots) \qquad i=1,2,..,N_s \tag{1}$$

where C_i is the concentration of constituent *i* (solid-bound and solute concentrations are expressed in mass per g solid and mass per L porewater, respectively), *x* is the position along the 1-D vertical domain, with *x* = 0 corresponding to the sediment-water interface (SWI), and *t* denotes time. In Eq. (1), $\sum r_i(x,t,C_i,\dots)$ is the sum of the sources and sinks of species *i*; it includes the rates of all the (bio)geochemical reactions producing or

[1] MATSEDLAB was developed in and tested up to release R2011b.

Table 1. Meaning of the generalized variables in Eq. (1) for solids and solutes in aquatic sediments.

	Solids	Solutes
ε	1-φ	φ
ϑ	ω	$\omega + V$
D	D_b	$D_b + Dmol/(1 - \ln(\varphi^2))$

where : φ [-], porosity; V [LT^{-1}], externally imposed flow velocity; ω[LT^{-1}], burial velocity defined with respect to the sediment water interface (SWI); D_b[L^2T^{-1}], bioturbation coefficient and $Dmol$ molecular diffusion coefficient.

consuming species *i*, as well as the non-local transport processes that add or remove the dissolved species, most notably through the irrigation of macrofaunal burrows. In a multicomponent reaction network containing N_s species and N_r reactions, Eq. (1) is solved for each species while various chemical species are coupled to one another via N_r reaction rate expressions appearing in $\sum r_i(x, t, C_i, \dots)$.

The variables ε, D and v take different meanings depending on whether the given constituent is solid-bound or dissolved (Table 1).

One of major capabilities of MATLAB is to operate on matrix and arrays. Hence, to implement Eq. (1) in MATLAB, it is practical to convert it to a vector form as follows:

$$\frac{\partial(\varepsilon \boldsymbol{C})}{\partial t} = \frac{\partial(\varepsilon \boldsymbol{F})}{\partial x} + \varepsilon \boldsymbol{S} \tag{2}$$

where $\boldsymbol{C}$ [N_s×1] is the vector of concentrations of species, $\boldsymbol{F}$ [N_s×1] is the transport vector encompassing burial and bioturbation terms with molecular diffusion only applied for the solutes:

$$\boldsymbol{F} = D\frac{\partial \boldsymbol{C}}{\partial x} - \vartheta \boldsymbol{C} \tag{3}$$

Sum of the all biogeochemical reaction rates are embedded in vector **s** [N_s×1] where rate of the production and consumption of each species is a function of reaction network stochiometrics and associated reaction rates. It is defined in a matrix notation as below:

$$\boldsymbol{S}_{Ns\times 1} = \boldsymbol{\mu}_{Ns\times Nr}.\boldsymbol{R}_{Nr\times 1} \tag{4}$$

In this equation $\boldsymbol{S}_{Ns\times 1}$ is the vector of reaction rates with the components of $\sum r_i(x,t,C_i,\dots)$, $\boldsymbol{\mu}_{Ns\times Nr}$ is the stochiometrics matrix and $\boldsymbol{R}_{Nr\times 1}$ is the vector of corresponding rate of each reaction. In a fully kinetic description of the set of the biogeochemical reaction rates, equilibrium reactions are treated by the fast kinetic rate constants.

Boundaries conditions

For solutes, the upper boundary condition at *x* = 0 is the bottom water concentration. In non-steady state simulations the concentration of sediment water interface (SWI) can be time-dependant reflecting e.g. annual or seasonal variations. In a vector form we have:

$$\boldsymbol{C}(0,t) = \boldsymbol{B}(t) \tag{5}$$

where $\boldsymbol{B}(t)$ is vector of solutes concentrations at SWI. For solid-bound species, the flux continuity condition is used at *x* = 0:

$$D_b\,\partial\boldsymbol{C}/\partial x - w\boldsymbol{C} = \frac{\boldsymbol{J}(t)}{\rho_b(1-\varphi)} \tag{6}$$

where ***J(t)*** is the vector of depositional flux of the given solid-bound species (time-dependent in transient simulations), while w and D_b are the sedimentation rate and bioturbation coefficient. The dominator, $\rho_b(1-\varphi)$ ensures consistency among the units of *J(t)* and *C*. In the scripts provided with this manuscript the units used are µmol cm^{-2} yr^{-1} (*J*), µmol g^{-1} (*C*) for solid species, cm yr^{-1} (*w*) and cm^2 yr^{-1} (D_b); ρ_b is the dry sediment density (g cm^{-3}) and φ is the sediment porosity. As lower boundary condition, zero gradients are imposed for all solute and solid-bound species:

$$\partial\boldsymbol{C}(x_L,t)/\partial x = 0 \tag{7}$$

3. Implementing diagenesis equation in MATLAB

pdepe Equation

The general form of PDE that is solved by *pdepe* in MATLAB within temporal (t_0<t<t_f) and spatial (a<x<b) domains is as following:

$$\boldsymbol{c}\left(x,t,u,\frac{\partial u}{\partial x}\right)\frac{\partial \boldsymbol{u}}{\partial t}=x^{-m}\frac{\partial}{\partial x}\left(x^{m}\boldsymbol{f}\left(x,t,u,\frac{\partial u}{\partial x}\right)\right)+\boldsymbol{s}\left(x,t,u,\frac{\partial u}{\partial x}\right) \quad (8)$$

in which vector ***u*** contains all the unknown variables (here the solid and solute concentrations, i.e. ***C***). Coupling of the partial derivatives with respect to time is restricted to multiplication by matrix *c*. On the right hand side of Eq. (8), *m* is a parameter corresponding to the symmetry of the problem and can be 0 for slab, 1 for cylindrical, or 2 for spherical.

Note: In all 1-D early diagenesis problems *m* will be always assigned zero.

Functions ***f*** and ***s*** – the flux and sink/source terms, respectively – are vector functions, which depend on depth (x), time (t) , concentrations (***u***) and concentration gradients ($\partial \boldsymbol{u}/\partial x$). Comparing diagenesis Eq. (1) with *pdepe* Eq. (8), the term ***f*** corresponds to vector function ***F*** as:

$$\boldsymbol{f}=\varepsilon\boldsymbol{F} \quad (9)$$

and represents transport of chemical constituents; it encompasses advective flux (advection rate ϑ), diffusive fluxes of solutes by molecular diffusion (diffusion coefficient D_m) and sediment mixing by benthic organisms (bioturbation coefficient D_b). The term ***s*** and vector ***S*** are balanced as

$$\boldsymbol{s}=\varepsilon\boldsymbol{S} \quad (10)$$

They account for the net production or consumption of all the chemical species by (bio)geochemical reactions and, for solutes, also bioirrigation.

In order to solve parabolic PDE of Eq. (8), necessary initial and boundary conditions must be imposed.

The initial condition at t=t_0 and all depths x has the form of:

$$\boldsymbol{u}(x, t_0) = \boldsymbol{u_0}(x) \tag{11}$$

Eq. (11) returns initial values of all the chemical species at depth *x* in the column vector ***u***. The user may thus specify any set of initial depth profiles, including spatially heterogeneous distribution. General boundary conditions at *x*=*a* and *x*=*b* are defined using the equation:

$$p(x, t, u) + q(x, t)\boldsymbol{f}(x, t, u, \partial u/\partial x) = 0 \tag{12}$$

where ***f*** is the transport vector from Eq. (8).

At first sight Eq. (12) may not resemble common diagenesis boundary conditions of Eq. (5)-Eq. (7). However, commonly-used formulations for boundary conditions, such as Dirichlet, Neumann and Cauchy/Robin are embedded in Eq. (12) and for each type, the coefficients $p(x, t, u)$ and $q(x, t)$ take different values. In order to transform boundary conditions of Eq. (5)-Eq. (7) to Eq. (12), we have:

1.Eq.(5) → Eq.(12)

$$\boldsymbol{f}(x, t, u, \partial \boldsymbol{u}/\partial x) = -\varphi\left(D_b + \frac{D_m}{1 - \ln(\varphi^2)}\right)\frac{\partial \boldsymbol{C}}{\partial x} + \varphi w \boldsymbol{C}$$

$$p(x, t, u) + q(x, t)\left[-\varphi\left(D_b + \frac{D_m}{1 - \ln(\varphi^2)}\right)\frac{\partial \boldsymbol{C}}{\partial x} + \varphi w \boldsymbol{C}\right] = 0$$

$$\Rightarrow p(x, t, u) = \boldsymbol{ul} - \boldsymbol{B}(t) \qquad \& \qquad q(x, t) = 0 \tag{13}$$

ul [N_s×1] is the approximate solution of *pdepe* solver at *x*=0.

2.Eq.(6) → Eq.(12)

$$\boldsymbol{f}(x, t, u, \partial \boldsymbol{u}/\partial x) = -(1 - \varphi)\rho_b D_b \frac{\partial \boldsymbol{C}}{\partial x} + (1 - \varphi)\rho_b w \boldsymbol{C}$$

$$p(x, t, u) + q(x, t)\left[-(1 - \varphi)\rho_b D_b \frac{\partial \boldsymbol{C}}{\partial x} + (1 - \varphi)\rho_b w \boldsymbol{C}\right] = 0$$

$$\Rightarrow p(x, t, u) = \boldsymbol{J}(t) \qquad \& \qquad q(x, t) = 1 \tag{14}$$

3.Eq.(7) → Eq.(12) for solutes

$$\boldsymbol{f}(x,t,u,\partial \boldsymbol{u}/\partial x) = -\varphi\left(D_b + \frac{D_m}{1-\ln(\varphi^2)}\right)\frac{\partial \boldsymbol{C}}{\partial x} + \varphi w \boldsymbol{C}$$

$$p(x,t,u) + q(x,t)\left[-\varphi\left(D_b + \frac{D_m}{1-\ln(\varphi^2)}\right)\frac{\partial \boldsymbol{C}}{\partial x} + \varphi w \boldsymbol{C}\right] = 0$$

$$\Rightarrow p(x,t,u) = -\varphi w \boldsymbol{ur} \qquad \& \qquad q(x,t) = 1 \tag{15}$$

$\boldsymbol{ur}$ [N_s×1] is the approximate solution of *pdepe* solver at $x=L$.

1.Eq.(7) → Eq.(12) for solids

$$\boldsymbol{f}(x,t,u,\partial \boldsymbol{u}/\partial x) = -(1-\varphi)\rho_b D_b \frac{\partial \boldsymbol{C}}{\partial x} + (1-\varphi)\rho_b w \boldsymbol{C}$$

$$p(x,t,u) + q(x,t)\left[-(1-\varphi)\rho_b D_b \frac{\partial \boldsymbol{C}}{\partial x} + (1-\varphi)\rho_b w \boldsymbol{C}\right] = 0$$

$$\Rightarrow p(x,t,u) = -(1-\varphi)\rho_b w \boldsymbol{ur} \qquad \& \qquad q(x,t) = 1 \tag{16}$$

As it is seen from Eq. (12), the boundary conditions can depend on *t*, which means that **MEDIALAB** evaluates the boundary values at each time step. This gives the user the possibility to impose transient boundary conditions. The latter is particularly useful when simulating the fate of compounds whose inputs are changing due to, for example, anthropogenic activity, or when dealing with systems where the bottom-water chemistry varies over time.

Structure of *pdepe*

The *pdepe* function is designed to solve initial-boundary value problems (IBVPs) consisting of systems of parabolic and elliptic PDEs in one space variable and time. The numerical method is based on a piecewise nonlinear Petrov–Garlekin method with second-order accuracy. The method solves the ordinary differential equations (ODEs) resulting from the spatial discretization of the PDEs, using a built-in MATLAB ODE

solver to obtain approximate solutions at specified times within a defined time interval. It is run through following command:

```
sol = pdepe(m,pdeFun,icFun,bcFun,x,t)
```

There are 6 main input arguments that must be parsed to *pdepe* function. They are :

1) Parameter ***m*** which equals to zero in current 1-D diagenesis problem
2) ***pdeFun*** is a function that computes the components of the PDE i.e. *c*,*f*,*s*. It has the following format

   ```
   [c,f,s] = pdeFun(x,t,u,dudx)
   ```

   The input arguments are :

   - *x*: vector of spatial domain
   - *t*: vector of temporal domain
   - *u*: vector of unknown concentrations
   - *dudx*: vector *of* approximate partial derivative with respect to *x* or concentration gradients.

3) ***icFun*** is the initial condition function which evaluates spatial-dependent initial concentration of species and returns the values in vector *u*. It has the form of:

   ```
   u = icFun(x)
   ```

   where vector *x* of spatial domain is the only input argument.

4) ***bcFun*** is the function that evaluates the terms *p* and *q* of the boundary conditions in Eq. (12). It has the form

   ```
   [pl,ql,pr,qr] = bcFun(xl,ul,xr,ur,t)
   ```

   ul is the approximate solution at the upper boundary *xl* = *a* and *ur* is the approximate solution at the lower boundary *xr* = *b*. *pl* and *ql* are column vectors corresponding to *p* and *q* evaluated at *xl*, similarly *pr* and *qr* correspond to *xr*.

Therefore, 4 vectors of *pl, ql, pr* and *qr* with the dimension of [N_s×1] are computed by this function.

The ultimate output of *pdepe* function returns values of the concentrations as a multidimensional matrix, *sol*, on a mesh provided in *x* and *t* vectors. To extract the concentration of i^{th} species in spatial and temporal steps *j* and *k* we can write:

```
u_jki = sol(tspan(j), xmesh(k), i)
```

Symbolic programming

One of the advantages of MATLAB application is its symbolic programming capability which is enhanced in the MEDIALAB model to automatically construct the reaction vector, *s*. It is done through extracting the coefficients of the chemical species in each reaction to build the stoichiometric matrix, $\boldsymbol{\mu}_{Ns\times Nr}$, that is used to generate sum of the reaction rates matrix, i.e. $\boldsymbol{S}_{Ns\times 1}$, that control production and consumption of each species. For this purpose, each biogeochemical pathway is assumed to be an algebraic polynomial equation with the products of the reaction all gathered with the reactants on the left side of the reaction with negative sign. Analogously chemical species are defined as symbols in MATLAB language which facilitates algebraic operations that are required to set up the stoichiometric matrix.

To clarify the process, let's assume nitrification as one of the reactions in the proposed conceptual model in reaction network:

$$NH_4^+ + 2O_2 + 2HCO_3^- \rightarrow NO_3^- + 2CO_2 + 3H_2O \qquad (17)$$

There are 6 species involved in this reaction, all required to be defined as symbols. The corresponding command in MATLAB is as follows:

```
syms nh4 o2 hco3 no3 co2 h2o
```

Symbolic names associated with each species is arbitrary. However, it is handy if they are chosen in a way that can be recognizable in the script. Then, the reaction is

converted to a polynomial by rewriting the reaction as below using the symbols defined above:

```
R = nh4 + 2*o2 + 2*hco3 - no3 - 2*co2 - 3*h2o
```
(18)

To extract number of moles of each species that are produced or consumed in this reaction, following MATLAB command is used to obtain the coefficients of `R` with respect to every species. For example, 1 mole of ammonium in this reaction is consumed, thus it has a coefficient of 1 which is attained through following command:

```
c = coeffs(R,nh4)
```

By applying two consecutive 'for' loops over total number of reactions and species stoichiometric matrix, $\boldsymbol{\mu}_{Ns\times Nr}$ is obtained which is saved as `stochiometrix` variable. `stochiometrix (i,j)` corresponds with number of moles of species `i` that are produced or consumed in reaction `j`. Positive and negative signs are associated with production and consumption, respectively. If reaction `j` doesn't invoke species `i`, then `stochiometrix(i,j)` equals to zero.

4. Structure of MEDIALAB

MEDIALAB encompasses five .m files:

- *mainMEDIALAB.m*
- *userMEDIALAB.m*
- *autoMEDIALAB.m*
- *plotMEDIALAB.m*
- *postprocessMEDIALAB.m*

In addition there is a text file, *inputMEDIALAB.txt*, that includes all the required parameters incorporated in the conceptual model. In case of availability of measured field data, most notably sediment profile data, there will be an Excel spreadsheet with measured field data and depth that measurements have taken place. There must be a separate sheet for each species with the exact name of chemical species.

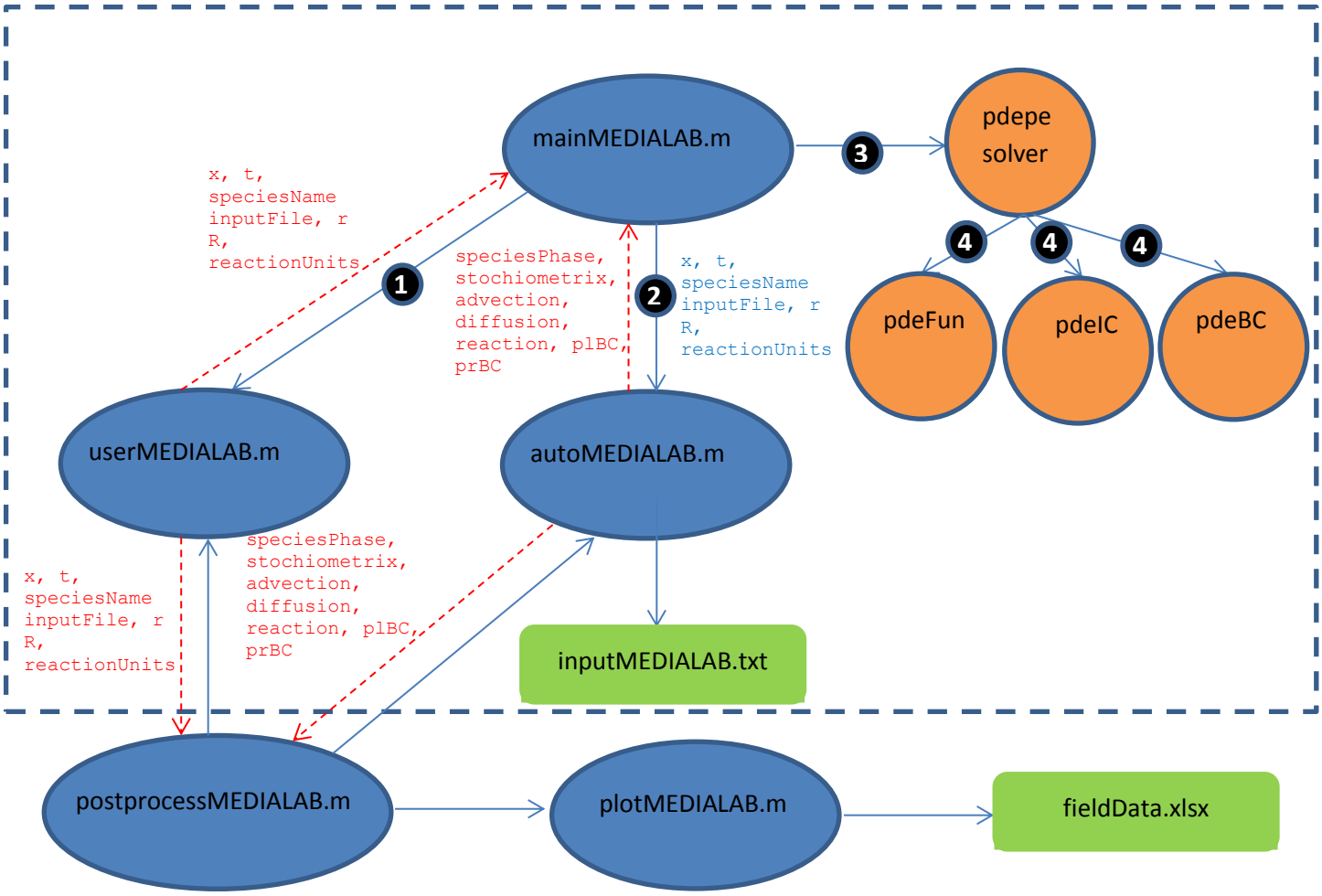

Figure (1). This figure shows schematic structure of MEDIALAB. It consists of 5 .m scripts (shown by blue ovals), 1 input text file and 1 excel file of field data (shown by green rectangular), and 3 functions as components of pdepe solver (shown with orange circles). Blue arrows are in the direction of functions or files that are called from other script. The input arguments in the calling process are denoted by blue text next to the blue arrow. The output arguments resulting from calling each function is depicted by red arrows and red text next to them. Numbers on blue arrows are the sequence of functions that are called through *mainMEDIALAB.m*. The blue dotted box contains all the required files during the simulation while *postprocessMEDIALAB.m* and *plotMEDIALAB.m* are utilities for post processing and plotting.

Table 2 is the summary of the files in the MEDIALAB package:

Table 2. Files included in MEDIALAB package.

FILE	TYPE	User Modified	Input Arguments	Output Arguments
mainMEDIALAB.m	MATLAB file	No	-	`simValues, depth, time`
userMEDIALAB.m	MATLAB file	Yes	-	`x, t, w, speciesName, inputFile, r, R, reactionUnits`
autoMEDIALAB.m	MATLAB file	No	`speciesName,inputFile, r, R, reactionUnits`	`speciesPhase, stochiometrix, advection, diffusion, reaction, plBC`
plotMEDIALAB.m	MATLAB file	No	-	
postprocessMEDIALAB.m	MATLAB file	No	-	
inputMEDIALAB.txt	Text file	Yes	-	-
fieldDATA.xlsx	Excel file	Yes	-	-

Starting with MEDIALAB

userMEDIALAB.m is the first script that has to be modified by the user to provide all the necessary information of spatial and temporal domains as well as biogeochemical reaction network. There are some other parameters such as list of species names, organic matter composition coefficients and input file name that have to be included in the script. Species are characterized by string variables and their assigned names contain either '(s)' or '(aq)' representing solid or aqueous species, respectively. This is

an approach to characterize phase of the species in the script when building the transport matrix (as molecular diffusion is excluded in case of solid species) and reaction matrix (to ensure consistency between reactions and species units). As described above, variable names used for the chemical species are defined as symbols to set up the reaction network. In general for an irreversible reaction:

`aA + bB → cC + dD` (19)

species `A, B, C` and `D` have to be defined as symbols and reaction is rewritten as:

`aA + bB - cC - dD` (20)

If the reaction is reversible, there will be two algebraic polynomials representing each direction of the reaction:

`aA + bB - cC - dD` (21)

and

`cC + dD - aA - bB` (22)

There are two attributes that have to be determined along with every reaction, `R`: reaction rate, `r`, and unit of the reaction, `reactionUnits`. Reaction rate is a string variable containing reaction rate constants and concentrations of species upon which rates are dependent on. For example if a bimolecular reaction rate is used to define the kinetics of the reaction (19), then `r` equals to:

```
r = 'k_AB * A * B '
```
(23)

`k_AB` is the reaction rate constant and provided through *inputMEDIALAB.txt* as an input parameter. In a similar manner, all other parameters appeared in kinetic terms of reaction network must be included in the input file with exact same name. `reactionUnits` is a vector of binary values 1 and 0. If the unit of reaction rate is same as solid phase unit, i.e. µmol/g, then `reactionUnits` equals to 1 otherwise it is 0. This is a flag variable which ensures the consistency between species and reaction

units. It means if, for instance, `A` and `B` in above reaction are aqueous and solid species respectively and `k_AB` has the units of [1/(mM.yr)], then `r` will have a unit of [µmol/g] which is solid species unit. Therefore, `reactionUnits` equals to 1.

inputMEDIALAB.txt is the input parameters text file modified by the user and consists of parameters of transport, reaction rate constants, diffusion and bioturbation coefficients as well as boundary conditions. The input file has two columns: parameters names and their values separated by space. Reaction rate constants must have exact name as they are used in the *userMEDIALAB.m* script. Molecular diffusion coefficients and boundary conditions are named as '`Dmol_`' and '`BC_`' added with species name, respectively. For example molecular diffusion coefficients and boundary condition of ammonium with species name of `nh4` are `Dmol_nh4` and `BC_nh4`. Number of molecular diffusion coefficients and boundary conditions must match with number of dissolved species and total number of species, respectively.
Other parameters that are included in the *inputMEDIALAB.txt* are: bioturbation coefficient, `D_bio`, sedimentation rate, `w`, sediment dry density, `w`, and `porosity`. Since concentration of H^+ is needed for calculation of the saturation index of iron sulfide and vivianite, it is also imposed under variable name `ph` and has the value of 10^{-pH} stated in mM units.

mainMEDIALAB.m is the core script file of MEDIALAB which is executed from MATLAB's home screen. Home directory must be the same address as the folder containing the contents of MEDIALAB files unless it has been saved in the similar directory. It is in this script that *userMEDIALAB.m*, *autoMEDIALAB.m* and *pdepe* solvers are called and the final solution, `sol`, is saved in the *resultMEDIALAB.mat* which involves all the simulated concentrations, depth and time values in a MATLAB matrix format. After termination of simulation, variables of `depth`, `time` and `simValues` are saved in the workspace. They are used for plotting and reaction rate calculations through *postprocessMEDIALAB.m*.

autoMEDIALAB.m

Transport, reaction and boundary condition vectors are automatically generated via *autoMEDIALAB.m* in form of MATLAB function handles and are passed to *mainMEDIALAB.m*. A function handle is a callable association to a MATLAB® function. It enables the user to pass a function to another function.

In the first block of the script input parameters are read from *inputMEDIALAB.txt* file and stored as `paraNames` (first column of the input file) and `paraValues` (second column of the input file). `paraNames` elements will be substituted with corresponding `paraValues` in advection, diffusion, reaction rate and boundary condition terms. Phase of the species is also extracted from `speciesName` vector and saved as variable `speciesPhase` in the workspace. Transport and boundary condition functions of solid species and solutes are computed differently as mentioned in previous sections.

Stoichiometric matrix is built in the 2nd block following the symbolic programming capability of MATLAB described above. The unit consistency between species and reaction units is enforced by applying conversion factor, `F`. From coding point of view it is implemented through comparing `speciesPhase` and `reactionUnits` inside an 'if' clause:

```
for i=1:numSpecies
     for j=1:length(R)
         c=coeffs(R{j},speciesName{i});
            if (length(c)>1)
            if(((speciesPhase(i)==1)&&(reactionUnits(j)==1))||((speciesPhase(
            i)==0)&&(reactionUnits(j)==0)))
                stochiometrix(i,j) = -c(2);
             elseif ((speciesPhase(i)==1)&&(reactionUnits(j)==0))
                stochiometrix(i,j) = -c(2)/F;
             else
                 stochiometrix(i,j) = -c(2)*F;
             end;
         else
             stochiometrix(i,j) = 0;
         end

     end
 end
```

Function handle of reaction rates is computed in the 3rd block. Reaction rates described in the string form in the *userMEDILAB.m* are not recognizable by *pdepe* and their numeric value has to be computed instead. Rate constants values and corresponding `u` vector elements are substituted in reaction rate string. For example, in reaction rate Eq. (23), if k_AB has the numeric value of 0.2 provided through *inputMEDIALAB.txt* and `A` and `B` are 5th and 8th species in the `speciesName` vector, then Eq. (23) will be transformed to:

```
r = '0.2 * u(5) * u(8)'
```
(24)

which will be ultimately converted to a function handle using MATLAB built-in function, str2func. `str2func('str')` constructs a function handle for the function named in the string `'str'`.

In a similar manner, function handles of transport and boundary conditions are computed in the 4th block. It means advection, diffusion (including molecular diffusion for solutes) and boundary conditions are formed in a string format and then are converted to function handles.

Function handles of advection, diffusion, boundary conditions along with `stochimerix` matrix are passed to *mainMEDIALAB.m* to be used as components of *pdepe* solver.

postprocessMEDIALAB.m

After termination of simulation, using the modeling results of spatial and temporal concentrations, `simValues`, are used to integrate reaction rate versus time and space to obtain spatial and temporal variations. Fluxes of reduced species through SWI such as Fe(II), S(II), NH_4^+ are calculated and plotted versus time. Graphical presentation of modeling results is done through calling *plotMEDIALAB.m*. Depth profiles can be compared with field data of *fieldData.xlsx* if measurements were available.

5. Potential problems and troubleshooting

Time integration failed: This is one of the most frequent problems concerning this model. It basically means that the solver crashed and failed to solve the system of the partial differential equations. There are many probabilities why this happened, but the detailed discussion has to do with numerical stabilities and is not included here. Most likely, this might be a result of a wide spacing of the spatial domain, or the effect of certain parameters. Try increasing the space resolution, e.g. from 500 points to 1000 points (at the expanse of longer running time). Alternatively, the user is encouraged to go back to the set of parameters, for which the solver worked. Change the parameter one by one, to figure out the one that failed the solver.

Segmentation violation: This error is not frequently seen. Most of the time this is a result of parallel running of the model, e.g. seven to eight at the same time, which might be needed when fitting the parameters. To resolve this problem, try not to run a higher number of instances of the model, than the number of cores of the CPU. Thus, if the CPU has 4 cores, try not to run 5 or more models at the same time.

Long running time: Normally, the model should finish running within half an hour. If the running time is too long, it might be a result of certain values being 'too small'. Try changing the parameters, or even adding the 'if' conditions, so that those values will not be 'too small'. Also, one might find alternative formulation for certain rates of reaction, since experience shows that the solver always works faster with rate laws that are first order with respect to the species.

Specific worksheet not found: This error might occur when using the 'result_plot' functionality. If an excel file name is provided (to plot the field data on top of the model simulation), there must be a worksheet for every species specified by 'VarNames', even if there is no data for that species. In that case, the sheet will be empty. A template of the excel file is provided, with the name '*FIELD_DATA.xls'*. Note that the names of the sheet have to match the names of the species exactly.

DAE of index greater than 1: This error occurs when the partial differential equation is not parabolic, e.g. when the bioturbation rate is set to be zero. The pdepe is only capable for solving parabolic-elliptic problems. Therefore, for that case, set the bioturbation rate to be something very small e.g. 0.001, to allow the solver to work. Most of the time, the theoretical model can be well expressed as a system of parabolic equations.

Memory - out of bounds: This error mostly occurs on system with less than 4 GB of random-access memory installed because MATSEDLAB required ~ 1 GB of contiguous RAM available for the initialisation of the matrices. Disabling start up programs and performing a clean reboot usually solves the issue.

6. References cited

Aguilera, D.R., Jourabchi, P., Spiteri, C., Regnier, P., 2005. A knowledge-based reactive transport approach for the simulation of biogeochemical dynamics in earth systems. Geochemistry Geophysics Geosystems 6(Q07012). doi: 10.1029/2004GC000899

Boudreau, B.P., 1999. Metals and models: Diagenetic modelling in freshwater lacustrine sediments. J. Paleolimnol. 22, 227-251.

Couture, R.M., Gobeil, C., Tessier, A., 2008. Chronology of atmospheric deposition of arsenic inferred from reconstructed sedimentary records. Environ. Sci. Technol. 42, 6508-6513.

Couture, R.M., Shafei, B., Van Cappellen, P., Tessier, A., Gobeil, C., 2010. Non-Steady State Modeling of Arsenic Diagenesis in Lake Sediments. Environ. Sci. Technol. 44, 197–203.

Couture, R.M., Shafei, B., Van Cappellen, P., 2012. A Multi-Component, Non-Steady State Biogeochemical Simulation Module of Early Diagenesis in MATLAB

Matisoff, G., Holdren, G.R., 1995. A model for sulfur accumulation in soft water lake sediments. Water Resour. Res. 31, 1751-1760.

Shannon, J.D., 1999. Regional trends in wet deposition of sulfate in the United States and SO2 emissions from 1980 through 1995. Atmospheric Environment 33, 807-816.

Aguilera, D.R., Jourabchi, P., Spiteri, C., Regnier, P., 2005. A knowledge-based reactive transport approach for the simulation of biogeochemical dynamics in earth systems. Geochemistry Geophysics Geosystems 6(Q07012). doi: 10.1029/2004GC000899