MATLAB algorithm to implement soil water data assimilation with the Ensemble Kalman Filter using HYDRUS

Javier Valdes-Abellan\textsuperscript{a,\textsuperscript{*}}, Yakov Pachepsky\textsuperscript{b}, Gonzalo Martinez\textsuperscript{c}

\textsuperscript{a} Department of Civil Engineering, University of Alicante, Alicante, Spain
\textsuperscript{b} USDA-ARS, Environmental Microbial and Food Safety Lab., Beltsville, MD, USA
\textsuperscript{c} Department of Applied Physics, University of Córdoba, Córdoba, Spain

**ABSTRACT**

Data assimilation is becoming a promising technique in hydrologic modelling to update not only model states but also to infer model parameters, specifically to infer soil hydraulic properties in Richard-equation-based soil water models. The Ensemble Kalman Filter method is one of the most widely employed method among the different data assimilation alternatives. In this study the complete Matlab\textsuperscript{©} code used to study soil data assimilation efficiency under different soil and climatic conditions is shown. The code shows the method how data assimilation through EnKF was implemented. Richards equation was solved by the used of Hydrus-1D software which was run from Matlab.

- MATLAB routines are released to be used/modified without restrictions for other researchers

\textsuperscript{*} Corresponding author.
E-mail address: javier.valdes@ua.es (J. Valdes-Abellan).

https://doi.org/10.1016/j.mex.2018.02.008
2215-0161/Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).
• Data assimilation Ensemble Kalman Filter method code.
• Soil water Richard equation flow solved by Hydrus-1D.

Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

ARTICLE INFO
Method name: Climate/soil EnKF efficiency
Keywords: Hydrus, EnKF, Soil water flux modelling
Article history: Received 22 December 2017; Accepted 26 February 2018; Available online 7 March 2018

Specifications Table

| Subject area | Select one of the following subject areas: |
|--------------|------------------------------------------|
|              | • Agricultural and Biological Sciences   |
|              | • Computer Science                       |
|              | • Engineering                            |
|              | • Environmental Science                  |
| More specific subject area | Data assimilation by Ensemble Kalman Filter applied to soil water flux modelling to infer soil hydraulic properties. |
| Method name  | Climate/soil EnKF efficiency.            |

Method details

Data assimilation, DA, methods improve the model performance by integrating observed data (i.e., system states) into the modelling process in order to correct the model predictions and or model parameters [1,2]. Among the different DA alternatives, the Ensemble Kalman Filter (EnKF) is one of the most widely used DA methods [6,8]. Shortly, an ensemble of models is randomly generated, then propagated in time to the to the next update event. For each update event, a state error covariance matrix is calculated from the state values simulated by the different ensemble members before the update (a priori). A covariance matrix of observations is also obtained at the same time. Both covariance matrices are used to obtain a new set of model states and model parameters.

In the present contribution, we share the Matlab code used in Valdes-Abellan et al. [3] to apply the EnKF data assimilation method to soil water flow modelling. The code was employed to infer soil model parameters by updating both states and parameter according to the approach showed in Chen and Zhang [7].

The code was created considering a 1-layer soil profile; nevertheless, it is straightforward modifying and adapting the code to more complex profiles. Additionally, it is prepared to consider different climates and soil types. This feature can also easily modify to be adapted to the new user aims.

All required subroutines and other files required to run the program are included in the present study as Supplementary materials.

Procedure

This lines are not required to run the model but to clearly exposed the variables used in the model.
It calibrates all five parameters of the layer
Made for 4 climates and 2 soil types.

Variable definition

ntobs: number of observation times
n0depth: number of reference depths
nssensor: number of sensors at a given depth
nmat: number of materials
nDTIMES: number of data assimilation times
nnode: number of nodes in soil water simulations with HYDRUS
xref: reference depths, cm; they include measurement depths and
additional interpolation depths
ipick: shows if the reference depth has measurements: 1 = yes, 0 = no
nref: computational node for reference depths, 1 = depth
wobs: array to store observed water contents, cm$^{-3}$
bias: bias corrected deviation of the observed soil water content
for a specific sensor from the average at a given depth, cm$^{-3}$
at a given time
ProfileFile: name of the profile file template
AtmosFile: name of the atmosphere file template
Selector: name of the selector file template

initial: initial time after the warm-up period
inc: period of time between two updates
ult: last updating time;
prim: first updating time after initial
NDTIMES: number of DA times
DATIMES: times when DA takes place
NPTF: number of pdfs

winitl: water initial content and all reference depths
winit: replication of winitl for each of PDFs, winit(NPTF, ndepts)

soil hydraulic properties for each material and pdfs.
thr(iptf, nmat); ths(iptf, nmat); alpha(iptf, nmat);
vgn(iptf, nmat); aks(iptf, nmat); al(iptf, nmat);

immtype: 0 if van Genuchten, 2 if Brooks and Corey

NNODES: number of computational nodes in the profile
dist=constant distance between two computational nodes

x: vector that collects depths in cm for each node
h: vector that collects soil pressure head for each node
mat: vector that collects soil material for each node

aver: average observed water content at a given depth, cm$^{-3}$

cm: correlation matrix of water content observations, cm$^{-6}$
cm: correlation matrix of water content observations, cm$^{-6}$

nodb: number of available observations at a given depth (< number of sensors)

E: covariance matrix covariance matrix of experimental data
C: covariance matrix of simulations
D: random data matrix, for each ensemble unit
H: observation matrix, one/zero according to observed data
F0: augmented param-state vector, before updating
Cxy: covariance parameters-states
K: Kalman update matrix
F: augmented param-state vector, after updating
X: state vector (in present study, states are only wvc), after updating

== FOLDER STRUCTURE ==
INSTRUCTIONS
The current folder has to contain a folder called "Test_Hydrus_GUI".
This folder will include all required HYDRUS templates and all in/out
HYDRUS files.
First lines clean all previous results and identify the folder of the input information.

```matlab
%======================================================================
clear; clc; fclose all;
conerror=0;
%======================================================================

% Initial statements
cDataPath=[pwd '\Test_Hydrus_GUI'];
cFileName=[cDataPath '\Input_dual.dat'];

%======================================================================

% Reading the inputSHP file.
cFileName=[cDataPath '\InputSHP.dat'];
f20=fopen(cFileName,'r');
nsoiltype=3;
for i=1:5; fgetl(f20);end
shpcorrect=zeros(nsoiltype,6); %pre
for insoil=1:nsoiltype
    shpcorrect(insoil,;)=fscanf(f20,'%f',6); fgetl(f20);
end
fgetl(f20);fgetl(f20);
shpini=zeros(nsoiltype,6); %pre
for inmat=1:nsoiltype; stextall(inmat)=fgetl(f20); end %capture soil texture
for inmat=1:nsoiltype;
    shpini(inmat,;)=fscanf(f20,'%f',6); fgetl(f20); %capture shp
end
fgetl(f20);
shborder(1,1:6)=fscanf(f20,'%f%f%f%f%f%n',6);
shborder(2,1:6)=fscanf(f20,'%f%f%f%f%f%n',6);
f10=fopen('LEVEL_01.DIR','w');
fprintf(f10,'%s',cDataPath);
fclose(f10); %closing level_01.dir
fclose(f20); %closing inputSHP
(:clear f10 f20;

Next, time information is introduced. \textit{tinitial} informed when the warm-up period finished and when the updating process begins. \textit{inc} collects the time interval in days between different updatings. Time between \textit{tinitial} and the first updating is collected in \textit{prim} variable. Finally, \textit{ult} collects the last day.

```matlab
% Data assimilation time setup

```matlab
% Data assimilation time setup

tinitial=365; \%-=-=-=-=-=- Initial time after the warm-up period
inc=7; \%-=-=-=-=-=- \textbf{MODIFY IF REQUIRED}
ult=1460; \%-=-=-=-=-=- \textbf{MODIFY IF REQUIRED}
prim=7; \%-=-=-=-=-=- \textbf{MODIFY IF REQUIRED}
DATIMES=[tinitial tinitial+prim:inc:ult];
NDATIMES=length(DATIMES)-1;

In the next step, geometrical information: Number of nodes, location of observation points, and others can be modified here.
As abovementioned, the code is prepared to run different climates and soil types. Here the climate-soil loop starts.

There are four different climate alternatives. Users may create new climate files following the same structure.

```matlab
% Beginning of loop clim-soil
for iclima=4  %------------------------ LOOP OF CLIMATES

% 2. Selecting the correct Atmosphere file
switch iclima
  case 1; cFileNameAtm=[cDataPath '\ATMOSP4YR_1_AZ.IN'];
  case 2; cFileNameAtm=[cDataPath '\ATMOSP4YR_2_ID.IN'];
  case 3; cFileNameAtm=[cDataPath '\ATMOSP4YR_3_IN.IN'];
  case 4; cFileNameAtm=[cDataPath '\ATMOSP4YR_4_TX.IN'];
end
for isoil=1:2  %------------------------ LOOP OF SOIL TYPES

% 1. Read observations
file=['\wsynthetic_C' num2str(iclima) '_S' num2str(isoil) ];
load(file);
wobs=wsin; clear wsin
[ntobs,nsensor,~]=size(wobs);  % m is number of
observation depths
  tobs=1:ntobs;

% Selection of pdat (ALWAYS ROSETA IN THIS CODE)
shpvar=shpini(isoil,:);
```

The ensemble of models is created in the following lines. Different alternatives can be chosen by changing the `generation` variable value. The first option uses the covariance matrix show in Faulkner et al. [4]; the second uses a diagonal covariance matrix (i.e., only standard deviation are considered but not covariance between different parameters).

Additionally, users can choose the number of units in `nunit` variable. The code let the user to decide what soil parameters are going to be upated. `elecshp` is a logical variable: 1 means that the parameter will be considered during the updating process, and 0 means the opposite. `elecshp` length is five, according to $\theta r$, $\phi$, $n$, $\alpha$ and $Ks$. 
The previous lines create the variables where results will be saved. At this time they are equal to zero.

In the following the updating process for a specific soil-climate case begins from the initial time, after the warm-up period, to the end of the updating process.
The Hydrus software is used to solve the Richards-equation-based soil water flux. It requires three input files: ATMOSPH.IN, PROFILE.DAT and SELECTOR.IN. The following code lines are devoted to create those files for each unit of the ensemble for the period ranging from the previous to the next updating time.

The PROFILE.DAT file requires the definition of the initial condition for the complete profile. Criterion to get this is to translate volumetric water content into soil pressure head according to the specific soil hydraulic properties in the observation depths (i.e., those depths were there were data) and to interpolate linearly soil pressure head between observation depths. This is made in the function called \( h = W_{TO_H_Bv3} \). Other options were considered but finally were discarded.
% Loop DA times
for idatetime=1:NATIMES
  cdataPath = ['\', idatetime, '.datPath']
  t1 = DATIMES(idatetime)
  t2 = DATIMES(idatetime+1)
  % Generation bias data
  bias = bias + (bias1 - bias2)
  bias1 = bias2
  % Calculation of the bias
  bias = bias / N
end

% Generation ATMSHPS file, from t1 to t2.
% Same file for all units of the ensemble
period = 0:1;
temp = importdata(cFileName,'t',1); temp = temp.data;
for k = 1:3
  fprintf(f26,'%d',k);
  fprintf(f26,'%d','temp(period));
  fprintf(f26,'
');
end
for k = 1:3
  fprintf(f26,'%d',k);
  fprintf(f26,'%d','temp(period));
  fprintf(f26,'
');
end
% Build the 'SELECTOR.THM' file
for k = 1:N
  fprintf(f27,'%d',k);
  fprintf(f27,'
');
end
for k = 3,4,5
  fprintf(f27,'%d',k);
  fprintf(f27,'
');
end
% Paste the files
for k = 1,N
  fprintf(f28,'%d',k);
  fprintf(f28,'
');
end
for k = 3,4,5
  fprintf(f28,'%d',k);
  fprintf(f28,'
');
end
% Generation PROFILE.DAT file
% A particular subroutine for idatetime=1 and other for the rest
With all required files, HYDRUS is finally called. In the present code, simulations requiring more than 6 s to finished computations were considered uncorrected and discarded. To interrupt a HYDRUS running, a system function called taskkill was used.
After computations finish, results from this run are read. This run implies a specific unit of the ensemble, a specific climate, a specific soil and a specific time.
Once all units of the ensemble have been run and their results collected, the updating process can be undertaken. First, all required matrices are obtained \((R, X, C, D, H, Cyx)\) and second an updated augmented vector of states and parameters (\(FIP\)) is obtained.
%% DA core starts here

Observation data on the assimilation time
temp=wobs(find(tobs==t2),:,:); % find the line
for inmd=1:m; v(inmd,:)=temp(1,:,inmd); end
aver=mean(v,2);
% Correction for bias, JVA: removed the component that
substract the
% average, not required because v is just used to computed
the cov
for mm=1:m; v(mm,:)=v(mm,:)-bias(mm,:); end
% =========== R(m,m), covariance matrix of experimental
data.
R=cov(v');

% =========== C(n,n), covariance matrix of simulations
nunitOK=nunit=sum(nerrors(idatetimes,:));
X=Thetas; %XP(n,N), predictions prior updating
X(:,find(nerrors(idatetimes,:)))=[]; % remove simulations
results with errors
C=cov(X');

% =========== D(m,n), random data matrix, for each ensemble
unit
 temp=chol(R);
 D=repmat(aver',nunitOK,1)+randn(nunitOK,m)*temp;
 D=D'; % now D(m,n)

% =========== H(m,n), observation matrix, one/zero according
to observed data
H(m,NNODES)=0; cont=1;
for i=1:NNODES
 if ipick(i)>0; H(cont,i)=1; cont=cont+1; end
 end

% ========= Cyx(t+n,n) covariance parameters-states
clear temp
t=sum(sum(elecshp));
temp=zeros(t,nunit); % preallocation
cont=1;
for inmat=1:nmat
 for j=1:5 % number of soil hydraulic properties, 1 not
 considered
 if elecshp(inmat,j)==1
 temp(cont,:)=shp(inmat,j,:);
 cont=cont+1;
 end
 end
 temp(:,find(nerrors(idatetimes,:)==1))=[]; % elimino comb que
dieron error
FI=[temp;X]; % augmented param-state vector, before
updating
Cyx=cov(FI');
Cyx(:,1:t)=[];

% ========= K(n,m), Kalman update matrix
K=Cyx*H'/(H*C+H'*R);

% ========= FIP(n,N), corrected predictions after
updating
FIP=FI+K*(D-H*X);
When the updated vector is obtained, averages values from the successful runs are assigned to the runs which reported an error and therefore they had no results.

```matlab
% ========= filling the error runnings
FIP2=zeros(NNODES+1,nunit); cont=1;
for inunit=1:nunit
    if nerror(idatetimes,inunit)==1
        FIP2(:,inunit)=mean(FIP(:,2));
    else
        FIP2(:,inunit)=FIP(:,cont);
        cont=cont+1;
    end
end
FIP=FIP2; clear FIP2
XP=FIP; XP(:,1)=[]; %updated vector of just states
% DA finishes here
```

If updated soil parameters falls out of logical boundaries during the updating process (e.g., residual water content below zero), then they are moved to the closest border of a logical domain. Border values are included in the 'InputSHP.dat' file and have been read in the first stages.

```matlab
% Filtering shp out of borders
for i=1:5
    temp=find(FIP(i,:)<shpborder(1,i)); %minimum border
    if ~isempty(temp)
        FIP(i,temp)=shpborder(1,i);
    end
    temp=find(FIP(i,:)>shpborder(2,i)); %maximum border
    if ~isempty(temp)
        FIP(i,temp)=shpborder(2,i);
    end
end
```

After the previous filter, results from soil hydraulic parameters updating are saved
Soil states (volumetric water content and soil pressure head) are also corrected to avoid illogical values, similarly to the process undertaken with soil hydraulic properties.

After the updating process has finished completely, and results have been saved, a direct run is developed with the last updated set of soil properties to obtain the model performance. The direct model is run from the beginning to the time of the last updating.

As all Hydrus runs, before to run the model, all required files have to be built. After the model has run
Similarly to the direct run for the past period, another direct run is accomplished with the last updated value of soil properties to obtain the model performance in case of future predictions. As abovementioned, first all required files are made, then Hydrus software is called. Finally the root mean square error, RMSE, the coefficient of determination, $R^2$, and the Nash-Sutcliffe efficiency index, NSE [5], statistics are computed.
```c
// Re-running FUTURE PERIOD FOR STATISTICS

fclose(all);
delete([dataPath 'r.out'], [dataPath 'ATMOSP.FHM'], [dataPath 'PROFILE.DAT'], [dataPath 'SELECTOR.IN'])

// Generation atmosphere file, from t2 to ult.
period=ult-t2
if period=0
    rnzfsu(istimes)=nan;
    nsrfsu(istimes)=nan;
else
    temp=importdata(filenameAtm, '\n', 9);
    for k=1:l
        line=fgetl($25);
        fprintf($26, '%s', line);
    end
end

// Build the 'selector.in' file
filename([dataPath 'SELECTOR_Template_direct.IN'])
for k=1:l
    line=fgetl($21);
    fprintf($22, '%s', line);
end

// Build the 'ship.sh' file
for i=1:imaseat
    line=fgetl($21);
    fprintf($22, '%s', line);
end

// Generation profile.dat file. En el re-running
th1=shipvel(3+istimes, 2);
alpha=shipvel(3+istimes, 3);
vnl=shipvel(3+istimes, 4);

for k=1:l
    line=fgetl($22);
    fprintf($22, '%s', line);
end

// Re-running FUTURE PERIOD FOR STATISTICS
```
Last stages of the code are devote to save all results.
% SAVING RESULTS

% Saving observation in file
filename = ['wout_obs.txt'];
cFileName=[cDataPath '\'
    filename];
f23= fopen(cFileName, 'w');
line=' Day';
for mm=1:m; line=[line '  
    num2str(abs(xref(mm)))
    cm']; end
11=find(tobs==DATIMES(1));
12=find(tobs==DATIMES(NDATIMES+1));
wsbsaver(12-11+1,m+1)=0;
wsbsaver(1:12-11+1,1)=tobs(11:12);
for mm=1:m
    wsbsaver(1:12-11+1,mm+1)=mean(wobs(11:12,:,mm),2);
end
fprintf(f23,line); fprintf(f23, 'n');
fprintf(f23, '%8.2f%12.4f%12.4f\n', wsbsaver');
close (f23); clear f23;

% Saving simulation in file
filename = ['wout_sim.txt'];
cFileName=[cDataPath '\'
    filename];
f23= fopen(cFileName, 'w');
line=' Day';
for mm=1:m; line=[line '  
    num2str(abs(xref(mm)))
    cm']; end
wsimuaver=wsbsaver(1,:); % copy the first line from obs to simu. It's the initial time
wsimuaver(DATIMES(DATIMES+1)-DATIMES(1)+1,m+1)=0;
% preallocation
wsimuaver(2:DATIMES(DATIMES+1)-DATIMES(1)+1,1)=wsimu(:,1,1);
% copy times;
for mm=1:m
    logi=wsimu(:,2:end,mm)>0; % to not consider error simulations
    wSIMUaver(2:DATIMES(DATIMES+1)-
    DATIMES(1)+1,mm+1)=sum(wsimu(:,2:end,mm),2)./sum(logi,2);
end
fprintf(f23,line); fprintf(f23, 'n');
fprintf(f23, '%8.2f%12.4f%12.4f\n', wsimuaver');
close (f23); clear f23;

% 8. Saving results
if generation==1
    line=[results\rocave2_C' num2str(iclima) '-S'
    num2str(isoil) '-Inc_' num2str(inc) '-Discr_' num2str(dist)];
elcse
    line=[results\rocave2_gen2_C' num2str(iclima) '-S'
    num2str(isoil)];
end
save(line, 'shpevol2', 'shpstd', 'r^2', 'mse',
    'nse', 'r^2fut', 'msefut', 'nsefut')
close all;
end % end of soil type
end % end of climate type
disp('The end!!!!!!!')
Acknowledgements

This study forms part of the CGL2013-48802-C3-3-R project financed by the Spanish Ministry of Science and Innovation, the FPD1-2013-16742 from the Spanish Ministry of Economics, and GRE15-19 financed by the University of Alicante. A post-doctoral research fellowship (CAS 15/00244) funded by the Spanish Ministry of Science and Innovation was awarded to J. Valdes-Abellan for this project.

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at https://doi.org/10.1016/j.mex.2018.02.008.

List of submitted files and routines:

- `gen_ensemble.m` and `gen_ensemblev2.m`: functions used to generate the ensemble of models.
- `HID_CALC.EXE`: executable to run HYDRUS-1D from MATLAB®.
- `keyinject.m`: function to convert HYDRUS-1D results.
- `LEVEL_01.DIR`: it collects folder location of HYDRUS-1D.
- `shp_cov_faulkner2003.txt`: it collects covariance matrix according to Faulkner et al. [4]
- `SSDA_DUAL_v2_201711_MX.m`: main script
- `W_TO_H_A.m`, `W_TO_H_Av2.m`, and `W_TO_H_Av2.m`: subroutines to get the initial soil head profile
- `wsynthetic_Ci_Sj.mat`: Matlab file collecting observed soil water content for climate i K and soil j.

List of folders:

- `results`: it collects all results
- `Test_Hydrus_GUI`: it collects all required HYDRUS templates and all in/out HYDRUS files.

References

[1] G. Evensen, Data Assimilation: The Ensemble Kalman Filter, Springer Berlin Heidelberg, 2009, pp. 307.
[2] D.A. Plaza, R. De Keyser, C.J.M. De Lannoy, L. Giustarini, P. Matgen, V.R.N. Pauwels, The importance of parameter resampling for soil moisture data assimilation into hydrologic models using the particle filter, Hydrol. Earth Syst. Sci. 16 (2012) 375–390.
[3] J. Valdes-Abellan, Y. Pachepsky, G. Martinez, Obtaining soil hydraulic parameters from soil water content data assimilation under different climatic/soil conditions, Catena 163 (2018) 311–320.
[4] B.R. Faulkner, W.G. Lyon, F.A. Khan, S. Chattopadhyay, Modeling leaching of viruses by the monte carlo method, Water Res. 37 (2003) 4719–4729.
[5] J.E. Nash, J.V. Sutcliffe, River flow forecasting through conceptual models part i — a discussion of principles, J. Hydrol. 10 (1970) 282–290.
[6] H. Moradkhani, S. Sorooshian, H.V. Gupta, P.R. Houser, Dual state-parameter estimation of hydrological models using ensemble Kalman filter, Adv. Water Resour. 28 (2005) 135–147.
[7] Y. Chen, D. Zhang, Data assimilation for transient flow in geologic formations via ensemble Kalman filter, Adv. Water Resour. (2006) 1107–1122.
[8] H. Zhang, H.J. Hendricks Franssen, X. Han, J. Vrugt, H. Vereecken, Joint State and Parameter Estimation of Two Land Surface Models Using the Ensemble Kalman Filter and Particle, Filter. Hydrol. Earth Syst. Sci. Discuss. (2016) 1–39.