Todaro, V.; D'oria, M.; Tanda, MG.; Gómez-Hernández, JJ. (2022). genES-MDA: a generic open-source software package to solve inverse problems via the Ensemble Smoother with Multiple Data Assimilation. Computers & Geosciences. 167:1-11. https://doi.org/10.1016/j.cageo.2022.105210

genES-MDA: a generic open-source software package to solve inverse problems via the Ensemble Smoother with Multiple Data Assimilation

Valeria Todaro[a], Marco D'Oria[a], Maria Giovanna Tanda[a] and J. Jaime Gómez-Hernández[b]

[a]*Department of Engineering and Architecture, University of Parma, 43124 Parma, Italy*
[b]*Institute for Water and Environmental Engineering, Universitat Politècnica de València, 46022 Valencia, Spain*

ARTICLE INFO

Keywords:
Stochastic inverse modeling
Uncertainty characterization
Covariance localization and inflation
Python

ABSTRACT

Ensemble Kalman filter methods have been successfully applied for data assimilation and parameters estimation through inverse modeling in various scientific fields. We have developed a new generic software package for the solution of inverse problems implementing the Ensemble Smoother with Multiple Data Assimilation (genES-MDA). It is an open-source, platform-independent Python-based program. Its aim is to facilitate the management and configuration of the ES-MDA through several programming tools that help in the preparation of the different steps of ES-MDA. genES-MDA has a flexible workflow that can be easily adapted for the implementation of different variants of the ensemble Kalman filter and for the solution of generic inverse problems. This paper presents a description of the package and some application examples. genES-MDA has been tested in three synthetic case studies: the solution of the reverse flow routing for the estimation of the inflow hydrograph to a river reach using observed water levels and a calibrated forward model of the river system, the identification of the hydraulic conductivity field using piezometric observations and a known forward flow model, and the estimation of the release history of a contaminant spill in an aquifer from measured concentration data and a known flow and transport model. The results of all these tests have demonstrated the flexibility of genES-MDA and its capabilities to efficiently solve different types of inverse problems.

CRediT authorship contribution statement

Valeria Todaro: Conceptualization, Methodology, Investigation, Software, Writing - original draft. **Marco D'Oria:** Conceptualization, Methodology, Validation, Writing - review & editing, Supervision. **Maria Giovanna Tanda:** Conceptualization, Methodology, Writing - review & editing, Supervision. **J. Jaime Gómez-Hernández:** Conceptualization, Methodology, Writing - review & editing, Supervision.

1. Introduction

Inverse problems are of great interest in several research and application areas as they allow to infer unknown parameters from a set of measurements (i.e., Capilla et al., 1999; Zhou et al., 2012; Franssen and Gómez-Hernández, 2002; Capilla et al., 1998; Feyen et al., 2003; Wen et al., 1999; Li et al., 2012; Gómez-Hernández et al., 2003). Inverse problems are ill-posed, so their solution is challenging and more difficult than the corresponding forward problem. Many approaches have been proposed in the literature to solve inverse problems; good overviews can be found in Sun (1994); Tarantola (2005); Zhou et al. (2014). Applications of inverse modeling can be found in different fields such as surface and subsurface hydrology, geophysics, communication theory, optics, radar, acoustic, petroleum engineering, meteorology, oceanography, astronomy, among others.

ORCID(s): 0000-0002-9313-6999 (V. Todaro); 0000-0002-5154-7052 (M. D'Oria); 0000-0002-8357-1348 (M.G. Tanda); 0000-0002-0720-2196 (J.J. Gómez-Hernández)

This work presents a generic (in the sense that it does not depend on a specific forward model or state-transition equation) open-source software for the solution of inverse problems based on ensemble Kalman Filter (EnKF) techniques, more precisely on the Ensemble Smoother with Multiple Data Assimilation (ES-MDA) introduced by Emerick and Reynolds (2012, 2013). The EnKF, initially proposed by Evensen (1994), is a Monte Carlo-based implementation of the Kalman filter (Kalman, 1960) able to handle estimation problems for high-dimensional, nonlinear and non-Gaussian systems. Since the introduction of the EnKF, many variants have been developed and applied for data assimilation and estimations of system states and parameters, one of which is the ES-MDA.

There are some open source data assimilation libraries that use Kalman filter methods, such as the Data Assimilation Research Testbed (DART; Anderson et al., 2009), PEST++ (White et al., 2020), the Ensemble-based Reservoir Tool (ERT, available from https://github.com/equinor/ert), the Parallel Data Assimilation Framework (PDAF; Nerger and Hiller, 2013), and the open Data Assimilation library (openDA; Ridler et al., 2014). DART consists of tools that implement the EnKF and the Ensemble Adjustment Kalman Filter (EAKF; Anderson, 2001) techniques. PEST++ includes PESTPP-IES, a localized iterative ensemble smoother tool for history matching and uncertainty quantification. ERT is a tool designed to perform model updating and data assimilation of reservoir models using ensemble-based methods. PDAF is a software environment that provides optimized data assimilation algorithms based on the EnKF and the Local Ensemble Transform Kalman Filter (LETKF; Hunt et al., 2007). Similarly, OpenDa is an open-source toolbox for the development and application of data assimilation and calibration algorithms such as the Ensemble Square-Root filter (EnSR; Tippett et al., 2003).

In this paper, we present genES-MDA, a new generic software package for the solution of inverse problems by means of an iterative ensemble smoother. Compared to available software packages, the key novelty of genES-MDA is its flexibility to handle very different problems and the provision of several tools that allow optimizing the ES-MDA settings improving the algorithm performance. Although genES-MDA specifically implements the ES-MDA, it would be quite simple to adapt the code to other ensemble-based methods. The code is written in Python, is open source and provides many functionalities to make it as flexible and efficient as possible. It is designed to solve problems in three dimensions, one and two-dimensional problems must be cast as three-dimensional ones with fixed values for one or two of their coordinates. An important aspect of the software is that it is designed with the capability of estimating continuously varying parameters in time, as in the works by Todaro et al. (2019, 2021). Ensemble Kalman filter techniques are usually applied in the context of inverse problems for model calibration and the identification of time-independent parameters (see e.g., Butera et al., 2021; Godoy et al., 2021; Silva et al., 2021; Xu and Gómez-Hernández, 2016; Xu et al., 2020; Gómez-Hernández and Xu, 2021).

A full description of the package and how to use it is included in the following, together with three synthetic case studies related to surface and groundwater hydrology. The examples show the applicability of genES-MDA to

different types of inverse problems. The first example solves the reverse flow routing problem in a river reach, the second concerns the estimation of a hydraulic conductivity field, and the last one deals with the identification of the release history of point-source groundwater pollutants.

The paper is organized as follows: Section 2 presents an overview of the ES-MDA methodology; in Section 3, the software package is described in detail; Section 4 outlines the results of three genES-MDA applications; and conclusions are drawn in Section 5.

2. Ensemble smoother with multiple data assimilation

The ensemble smoother with multiple data assimilation (ES-MDA) (Emerick and Reynolds, 2013) is an iterative data assimilation method that estimates model parameters (such as hydraulic conductivities or pumping rates) from a set of measurements (such as piezometric heads or flow rates). The process of estimating components of a model from observations of the state of the system is referred to as solving an inverse problem. Every inverse problem has associated a forward model, which is the analytical or numerical model that provides the state of the system given that all parameters and forcing terms are known. Parameters are seldom known, particularly in earth sciences applications, and for this reason the solution of the inverse problem is so relevant.

The ES-MDA workflow starts with an initialization phase followed by an iterative phase consisting of two steps: a forecast step and an update step. The initialization phase consists in the generation of an initial ensemble of parameter realizations of size N_e, an ensemble of observation errors, and the establishment of the number of iterations (multiple data assimilations) N to perform together with the set of inflation coefficients $\{\alpha_i, i = 1, \dots, N\}$ required by the ES-MDA. The observation errors are assumed to follow a Gaussian distribution with zero mean and given variance. The coefficients α_i must satisfy

$$\sum_{i=1}^{N} \frac{1}{\alpha_i} = 1, \tag{1}$$

which ensures coherence in the treatment of the observation errors between the smoother with a single assimilation and the smoother with multiple data assimilation. Here, the scheme by Evensen (2018) is adopted for the definition of these coefficients

$$\alpha_i = \alpha'_i \left(\sum_{i=1}^{N} \frac{1}{\alpha'_i} \right), \tag{2}$$

where α_i' are computed as

$$\alpha'_{i+1} = \frac{\alpha'_i}{\alpha_{geo}}. \tag{3}$$

The coefficient α'_1 must be a non-zero value and α_{geo} is a constant factor that defines the extent of the change of α_i from one iteration to the next. A gradual decrease of α_i, obtained with $\alpha_{geo} > 1$, can improve the performance of the procedure, since it helps to avoid overcorrections in the initial updates when the misfit between model predictions and observations is usually large.

After the initialization phase, the method proceeds iteratively with the forecast and the update steps.

1. Forecast step

   At each iteration i, for each realization j of the ensemble of parameters $\mathbf{X}_{j,i} \in \mathfrak{R}^{N_p}$, the forward model is run to obtain the model predictions, of which a subset $\mathbf{Y}_{j,i} \in \mathfrak{R}^m$ is extracted coinciding with the same locations and times as the observations $\mathbf{D} \in \mathfrak{R}^m$

$$\mathbf{Y}_{j,i} = g\left(\mathbf{X}_{j,i}\right), \tag{4}$$

   where $g(\cdot)$ is an operator that includes the forward model plus an observation function to extract the predictions at the m spacetime locations where observation have been taken.

2. Update step

   The ensemble of parameters is updated according to the equation

$$\mathbf{X}_{j,i+1} = \mathbf{X}_{j,i} + \mathbf{C}^i_{\mathbf{XY}}\left(\mathbf{C}^i_{\mathbf{YY}} + \boldsymbol{\alpha}_i\mathbf{R}\right)^{-1}\left(\mathbf{D} + \sqrt{\boldsymbol{\alpha}_i}\boldsymbol{\epsilon}_j - \mathbf{Y}_{j,i}\right), \tag{5}$$

   where $\mathbf{C}^i_{\mathbf{XY}} \in \mathfrak{R}^{N_p \times m}$ is the cross-covariance matrix between parameters and predictions, $\mathbf{C}^i_{\mathbf{YY}} \in \mathfrak{R}^{m \times m}$ is the auto-covariance matrix of predictions and $\mathbf{R} \in \mathfrak{R}^{m \times m}$ is the auto-covariance matrix of the measurement errors, which are assumed to be uncorrelated. $\boldsymbol{\epsilon}_j \in \mathfrak{R}^m$ is the vector of measurement errors for realization j. $\mathbf{C}^i_{\mathbf{XY}}$ and $\mathbf{C}^i_{\mathbf{YY}}$ are computed at each iteration i as

$$\mathbf{C}^i_{\mathbf{XY}} = \frac{1}{N_e - 1}\sum_{j=1}^{N_e}\left(\mathbf{X}_{j,i} - \overline{\mathbf{X}}_i\right)\left(\mathbf{Y}_{j,i} - \overline{\mathbf{Y}}_i\right)^T, \tag{6}$$

$$\mathbf{C}_{\mathbf{YY}}^{i} = \frac{1}{N_e - 1} \sum_{j=1}^{N_e} \left(\mathbf{Y}_{j,i} - \overline{\mathbf{Y}}_i \right) \left(\mathbf{Y}_{j,i} - \overline{\mathbf{Y}}_i \right)^T , \tag{7}$$

where $\overline{\mathbf{X}}_i$ and $\overline{\mathbf{Y}}_i$ are the ensemble means, at iteration i, of $\mathbf{X}$ and $\mathbf{Y}$, respectively.

Then, the iteration index advances and the procedure goes back to the forecast step until the last iteration.

Some modifications to the original algorithm can be applied to improve its performance and overcome some problems that may limit its applicability. For instance, the update step can be performed in a transformed space that avoids the appearance of negative parameter values. In such case, the ensemble of parameters is transformed before performing the update step (Eq. (5)) and back-transformed after the updating prior to the next forecast. The logarithmic and square root transformations are commonly used to handle non-negative data; other type of transformations can be useful to limit the parameter space to a suitable range. A different transformation can be used for each parameter; particularly when different types of parameters are estimated simultaneously.

To avoid overshooting for strongly non-linear problems, a linear relaxation can be performed on the ensemble of parameters at the end of each update step, modifying the values resulting from (5) as follows

$$\widetilde{\mathbf{X}}_{j,i+1} = (1 - w)\, \mathbf{X}_{j,i+1} + w \mathbf{X}_{j,i}, \tag{8}$$

where w is a relaxation coefficient between 0 and 1.

Another issue of ensemble-based methods is the undersampling problem, which may occur when the ensemble size is small and therefore not statistically representative. This can lead to filter divergence and the appearance of long-range spurious correlations. Since the computational time depends on the ensemble size, as the procedure requires N_e runs of the forward model for each iteration, it is convenient to limit the number of ensemble realizations. Covariance localization and covariance inflation are two techniques developed to handle the undersampling problem. Several covariance localization (CL) approaches are presented in the literature to mitigate the long-range spurious correlation problem (Houtekamer and Mitchell, 1998; Hamill et al., 2001; Anderson, 2007; Chen and Oliver, 2009). CL is usually applied based on spatial distances among the variables. In genES-MDA, we implemented a localization approach that takes into account both spatial and temporal distances, so that it can also be applied for the estimation of time and spacetime dependent parameters. CL is applied by an elementwise multiplication of the covariance matrices obtained from Eqs. (6) and (7) by distance dependent correlation matrices

$$\widetilde{\mathbf{C}}^i_{\mathbf{XY}} = \rho^i_{\mathbf{XY}} \circ \mathbf{C}^i_{\mathbf{XY}}, \tag{9}$$

$$\widetilde{\mathbf{C}}^i_{\mathbf{YY}} = \rho_{\mathbf{YY}} \circ \mathbf{C}^i_{\mathbf{YY}}, \tag{10}$$

where $\circ$ represents the Schur product and $\rho^i_{\mathbf{XY}}$ and $\rho_{\mathbf{YY}}$ are distance correlation matrices between parameters and observations and between observations and observations, respectively. $\rho^i_{\mathbf{XY}}$ and $\rho_{\mathbf{YY}}$ may depend on space, time or spacetime distances; for this last case, the correlations in space ($\rho^i_{\mathbf{XY},s}$ and $\rho_{\mathbf{YY},s}$) and time ($\rho^i_{\mathbf{XY},t}$ and $\rho_{\mathbf{YY},t}$) are computed separately and then coupled via a Schur product:

$$\rho^i_{\mathbf{XY}} = \rho^i_{\mathbf{XY},s} \circ \rho^i_{\mathbf{XY},t}, \tag{11}$$

$$\rho_{\mathbf{YY}} = \rho_{\mathbf{YY},s} \circ \rho_{\mathbf{YY},t}. \tag{12}$$

An interesting inverse problem is the identification of the spatial coordinates of some feature, for instance the release location of a contaminant in an aquifer. In these cases, the parameters themselves (the spatial coordinates) change in space after each iteration and so do the distances between the parameter being estimated and the observation locations. For these applications, the distance matrices $\rho^i_{\mathbf{XY},s}$ and $\rho^i_{\mathbf{XY},t}$ must be recomputed after each iteration.

The elements of the correlation matrices are computed using the fifth-order correlation function by Gaspari and Cohn (1999), which smoothly reduces the correlations between points for increasing distances and cuts off long-range correlations above a specific distance

$$\rho = \begin{cases} -\frac{1}{4}\left(\frac{\delta}{b}\right)^5 + \frac{1}{2}\left(\frac{\delta}{b}\right)^4 + \frac{5}{8}\left(\frac{\delta}{b}\right)^3 - \frac{5}{3}\left(\frac{\delta}{b}\right)^2 + 1, & 0 \le \delta \le b; \\ \frac{1}{12}\left(\frac{\delta}{b}\right)^5 - \frac{1}{2}\left(\frac{\delta}{b}\right)^4 + \frac{5}{8}\left(\frac{\delta}{b}\right)^3 + \frac{5}{3}\left(\frac{\delta}{b}\right)^2 - 5\left(\frac{\delta}{b}\right) + 4 - \frac{2}{3}\left(\frac{\delta}{b}\right)^{-1}, & b \le \delta \le 2b; \\ 0, & \delta \ge 2b; \end{cases} \tag{13}$$

where δ is the parameter-observation or observation-observation distance in space ($\delta^i_{XY,s}$ and $\delta_{YY,s}$) or time ($\delta_{XY,t}$

and $\delta_{YY,t}$). The coefficient b defines the spatial (b_s) or temporal (b_t) distance at which correlation is lost.

Undersampling can also cause filter divergence. Covariance inflation is a technique developed to overcome this problem (Anderson and Anderson, 1999; Li et al., 2009; Liang et al., 2011; Zheng, 2009). Here, we follow the scheme proposed by Anderson and Anderson (1999). At the end of each update step, after the linear relaxation is applied (Eq. (8)), each realization of the updated and relaxed ensemble of parameters, $\widetilde{\mathbf{X}}_{j,i+1}$, suffers a linear inflation around its mean, $\overline{\mathbf{X}}_{i+1}$

$$\widetilde{\widetilde{\mathbf{X}}}_{j,i+1} = r\left(\widetilde{\mathbf{X}}_{j,i+1} - \overline{\mathbf{X}}_{i+1}\right) + \overline{\mathbf{X}}_{i+1}. \tag{14}$$

with r being an inflation coefficient slightly larger than 1.

3. Software package description

genES-MDA can be used with any Python-based platform and does not require an installation. The software consists of a main module, two subordinate modules, a tools package and some text input files (Fig. 1). All files need to be downloaded to the same folder preserving the directory hierarchy. The ESMDA.py is the main module containing the script implementing the ES-MDA method. The code is generic and does not depend on the specific case study. Mod.py is a subordinate module used to manage the forward model; it contains the functions that write the model input files, run the forward model and read the model outputs. InputSettings.py is the subordinate module to configure the algorithm. Mod.py and InputSettings.py depend on the specific problem at hand and they must be modified by the user. The Tools package includes different modules with the instruments to build the initial ensemble of parameters, generate the observation errors, apply localization, perform parameter transformations and calculate some evaluation metrics. Files Obs.txt and Par.txt are mandatory input files with information about observations and parameters. Furthermore, three optional external input files can be defined: Ens.txt file with an initial ensemble of parameters and Errors.txt and R.txt files that contain the measurement error matrix and its covariance matrix, respectively. Alternatively, the initial ensemble, the measurement errors and its covariance matrix can be generated within the code package. Finally, folder called Model contains all input and output files used to run the forward model.

All components of the software package are presented in the following subsections.

3.1. Input files

The input files provide information about the parameters to estimate and the available observations. They are free-format text files that can be edited with any text editor.

- Obs.txt (mandatory): This file contains the observation data and their locations and sampling times. Each row

pertains to an observation. The spatial location is given in terms of x, y and z coordinates. It must be written as follows:

1st column: x-coordinate

2nd column: y-coordinate

3nd column: z-coordinate

4rd column: sampling time

5th column: observed value

If spatial coordinates or sampling times are not needed or observations are time/space independent, they must be replaced with NaN. One- and two-dimensional problems must be cast as three-dimensional ones by using a fixed dummy value for the coordinates that are in the second and/or third dimensions as needed.

- Par.txt (mandatory): This file contains information about the parameters that are to be estimated. Its structure is similar to the previous one with the spacetime coordinates of the parameter location in the first four columns plus a reference value in the 5th column. The reference value is of interest for comparison purposes in synthetic cases where the solution of the inverse problem is known. If a parameter does not depend on space or time its coordinates or time must be replaced with NaN. As with Obs.txt, one- and two-dimensional problems must be cast at three-dimensional ones with dummy coordinate values as needed. If the reference solution is not available, its value should be set to NaN.

- Ens.txt (optional): This file contains the initial ensemble of parameters. Each column is a realization of the parameters in the same order as they are input to the Par.txt file. There are as many columns as realizations.

- Errors.txt (optional): This file contains the observation errors ϵ. Each column is a realization of the observation errors in the same order as observations are input to the Obs.txt. There are as many columns as realizations.

- R.txt (optional): This file contains the observation error covariance matrix $\mathbf{R}$.

3.2. Tools package

The tools package consists of five modules designed to build the initial ensemble of parameters, generate the observation errors, specify when and how to apply covariance localization, perform parameter transformations and back-transformations and compute evaluation metrics. The modules are described in the following subsections.

EnsembleGenerator.py

The EnsembleGenerator.py module contains the functions to build the initial ensemble of parameters. Three options are available: to assume that all parameters are homogeneous, in which case, each realization will have a different

constant value drawn from a chosen probability distribution; to assume that they are heterogeneous and independent, in which case, each parameter is drawn independently of each other from a chosen probability distribution; to assume that they have some spatial continuity, in which case, the parameters are generated using smooth functions; in the latter case, the parameters take the values of discretized probability distribution functions that differ among the realizations because the hyperparameters that define each function are generated randomly.

The available functions are:

- `Random`: Initial parameter realizations are random values selected from a uniform distribution. The required input arguments are: the extreme values of the range, the number of parameters and the ensemble size.

- `PdfGamma`: Initial parameter realizations are the values of discretized gamma functions defined as

$$f(t) = A + B \cdot \frac{1}{k^n \Gamma(n)} t^{n-1} e^{-t/k}, \tag{15}$$

where t is time or spatial coordinate, A represents a base amount of the considered variable, B is the volume under the Gamma function, n is the shape coefficient, k is the scale coefficient and $\Gamma(n)$ is the gamma function. The hyperparameters A, B, n and k are drawn from uniform distributions. The required input arguments are: the minimum and maximum values for the four hyperparameters, a discretized vector of variable t, the number of parameters and the ensemble size.

- `PdfGammaNpeaks`: Initial parameter realizations are the values of discretized functions given by a sum of M gamma functions

$$f(t) = A + \sum_{r=1}^{M} B_r \cdot \frac{1}{k_r^{n_r} \Gamma\left(n_r\right)} t^{n_r-1} e^{-t/k_r}. \tag{16}$$

For all M Gamma functions, each hyperparameter is drawn from the same uniform distribution. Input arguments are the same as in the previous function, plus the value of M.

- `PdfNormal`: Initial parameter realizations are the values of discretized Gaussian functions

$$f(t) = A + B \cdot \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{t-\mu}{\sigma}\right)}, \tag{17}$$

where t is time or spatial coordinate, A is a base amount of the considered variable, B is the volume under the Gaussian function and μ and σ are the mean and variance of the distribution. The hyperparameters A, B, μ and σ are drawn from uniform distributions. The required input arguments are: the minimum and maximum values for hyperparameters, a discretized vector of variable t, the number of parameters and the ensemble size.

- `ConstantRandom`: Initial parameter realizations are constant values drawn from a uniform distribution. The required input arguments are the minimum and maximum values of the parameter range, the number of parameters and the ensemble size.

- `ConstantNormal`: Initial parameter realizations are constant values drawn from a known Gaussian distributions. The required input arguments are the mean and the variance of the Gaussian function, the number of parameters and the ensemble size.

ErrorGenerator.py

ErrorGenerator.py is the module used to generate the observation errors. It consists of two functions that return the ensemble of measurement errors and its covariance matrix. The options are:

- `NormalError`: It generates random errors normally distributed with zero mean and fixed variance. The required input arguments are the variance, the number of observations and the ensemble size.

- `PercNormalError`: It generates errors based on a user-defined percentage p that serves to set the standard deviation of a Gaussian distribution according to the relationship $3\sigma = p/100 \cdot obs$, where obs is the observed value. The error of each observation is selected from a normal distribution with zero mean and variance σ^2; in this way, in 99.7% of the times the error is between $\pm p/100 \cdot obs$. The user can specify a minimum variance value to avoid too small errors. The required input arguments are the observation vector, the value p, the lower limit of the variance, the number of observations and the ensemble size.

Localization.py

Localization.py is the module used to perform covariance localization. It contains two functions that return the correlation matrices (Eq. (13)) based on spatial or temporal distances. The options are:

- `TimeLocal`: It generates the temporal correlation matrices. The required input arguments are the time correlation distance (coefficient b in Eq. (13)), the vectors of parameter times, and the observation times.

- `SpaceLocal`: It generates the spatial correlation matrices. The required input arguments are the spatial correlation distance (coefficient b in Eq. (13)), the location of parameters, and the location of observations.

Transformation.py

Transformation.py is the module to perform parameter transformations and back-transformations. It uses the vector of parameters in their physical space as input and returns them in the transformed space as output, and vice versa for the back-transformations. In case of bounded transformations, the bounded space interval must be specified. The functions available are listed below:

- `Log_forward`: Parameters are log transformed.

- `Log_backward`: Parameters are back-transformed from log space.

- `LogLim_forward`: Parameters are log transformed into the bounded space [a,b] following the modified log-transformation:

$$y = f(x) = \log\left(\frac{x-a}{b-x}\right). \tag{18}$$

- `LogLim_backward`: Parameters are back-transformed from the modified log space:

$$x = f^{-1}(y) = \frac{(b-a)\,e^{y}}{1+e^{y}} + a. \tag{19}$$

- `SquareRoot_forward`: Parameters are transformed using the square root transformation.

- `SquareRoot_backward`: Parameters are back-transformed from square root space.

- `SquareRootLim_forward`: Parameters are transformed using the square root transformation in the bounded space [a,b]:

$$y = f(x) = \left(\frac{x-a}{b-x}\right)^{\frac{1}{2}}. \tag{20}$$

- `SquareRootLim_backward`: Parameters are back-transformed from the modified square root space:

$$x = f^{-1}(y) = \frac{(b-a)\,y^{2}}{1+y^{2}} + a. \tag{21}$$

Metrics.py

Metrics.py provides the following functions to compute performance metrics:

- `RMSE` calculates the root mean square error between actual and predicted values

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}\left(\overline{S}_i - O_i\right)^2}{n}}, \tag{22}$$

where n is the sample size (number of parameters or observations), $\overline{S}_i$ is the ensemble mean of the i-th estimated value and O_i is the i-th actual value.

- `AES` calculates the average ensemble spread

$$AES = \sqrt{\frac{\sum_{i=1}^{n} \sigma_i^2}{n}}, \tag{23}$$

where n is the sample size and σ_i^2 is the ensemble variance of the i-th parameter.

- `NSE` calculates the Nash-Sutcliffe efficiency criterion

$$NSE = \left(1 - \frac{\sum_{i=1}^{n} \left(\overline{S}_i - O_i\right)^2}{\sum_{i=1}^{n} \left(O_i - \overline{O}\right)^2}\right) \cdot 100, \tag{24}$$

where n is the sample size, $\overline{S}_i$ is the ensemble mean of the i-th estimated parameter, O_i is the i-th actual value and $\overline{O}$ is the average of the actual values.

- `RSS` calculates the residual sum of squares between actual and estimated data

$$RSS = \sum_{i=1}^{n} \left(\overline{S}_i - O_i\right)^2, \tag{25}$$

where n is the sample size, $\overline{S}_i$ is the ensemble mean of the i-th estimated parameter and O_i is the i-th actual value.

- `spatial_distance` calculates the spatial distance between actual and estimated locations. It is used when parameters to be estimated are spatial coordinates.

$$L = \sqrt{\left(\overline{x}_s - x_0\right)^2 + \left(\overline{y}_s - y_0\right)^2 + \left(\overline{z}_s - z_0\right)^2} \tag{26}$$

where $\overline{x}_s$, $\overline{y}_s$ and $\overline{z}_s$ are the ensemble means of the estimated coordinates and $\left(x_0, y_0, z_0\right)$ is the actual location.

3.3. Modules

In the following, we describe the three modules of genES-MDA with their tasks and functionalities.

ESMDA.py

ESMDA.py is the main module of genES-MDA. It depends on the numPy and OS Python libraries and does not require modification by the user. This module contains the main codes to perform the ES-MDA and calls all the subordinate modules. At the beginning of the run, the user is prompted to choose whether to use the default settings

or not: if the answer is yes, the input information is read from the InputSettings module; otherwise, the code interacts with the user prompting on screen for the general settings.

Mod.py

Mod.py is the module that links genES-MDA with the forward model. The forward model must have the ability of being run using a system command through an executable file or batch file, and read input and produce output ASCII files. All model files must be contained in the Model subfolder. Mod.py must be adapted according to each specific application. It is up to the user to handle the link between genES-MDA and the forward model by modifying the functions that allow to write the input files, run the model and read the outputs. It should be noted that there are several Python packages that provide useful tools to interact with widely used software, which could be integrated into this module. The Examples folder in the software distribution contains three examples on how to write this module, two of which interact with a Python package for the creation of the input files and reading the output files. Mod.py includes three functions:

- `write_input` writes the parameters to be estimated in the input files of the forward model. More than one input file may need to be modified depending on the model to be used and the parameter types to be estimated. To facilitate the writing of the input files, it is suggested to create template files that map the parameters into their proper locations in the forward model input files; and then edit them making use of appropriate keywords. The function argument is a vector containing a parameter realization $\mathbf{X}_{j,i}$.

- `run` contains the command to execute the forward model;

- `read_output` reads the results after the model run from the output files of the forward model. It contains the instructions on how to extract the values of interest by defining their specific location or searching for specific keywords in the output files. The function returns the vector $\mathbf{Y}_{j,i}$ containing the model predictions corresponding to the parameter realization $\mathbf{X}_{j,i}$ and coinciding with the same locations and times of the observations.

To enable reading of output files, instruction files are created that contain a set of instructions (including locating specific line numbers or searching for specific text) that enable extraction of output values to be compared with site observation data.

InputSettings.py

InputSettings.py is the module containing the input information required by genES-MDA. It depends on the specific problem at hand. The user needs to define the number of iterations N, the α_{geo} coefficient (Eq. (3)) and the relaxation coefficient w (Eq. (8)). Then, the user indicates whether to generate ensembles of parameters and observation errors,

whether to apply inflation and localization corrections, and whether to perform parameter space transformation. If the answer to any of the previous questions is yes, the user must define the corresponding parameters: ensemble size N_e, inflation coefficient r (Eq. (14)), type of covariance localization (spatial and/or temporal, standard, or iterative). Then, the user must modify the following functions:

- `Func_ens` is used to set up the generation of the initial ensemble of parameters when this ensemble is not read from the external file Ens.txt. The EnsembleGenerator.py module is used to select how to generate the ensemble. The function arguments are the matrix obtained after reading the Par.txt file and N_e. `Func_ens` returns the initial ensemble of parameters $\mathbf{X}$.

  For instance, if the user needs to build an ensemble made of random values in the range [5,150], the function is

```
def Func_ens(par, ens):
    from Tools import EnsembleGenerator
    N_par=par.shape[0]
    Ensemble=np.zeros((N_par,ens))
    (Min,Max)=(5,150)
    Ensemble=EnsembleGenerator.Random(Min,Max,N_par,ens)
    return Ensemble
```

- `Func_err` is used to set up the generation of the ensemble of observation errors and its covariance matrix when they are not read from the external files Errors.txt and R.txt. The ErrorGenerator.py module is used to select how to generate them. The function arguments are the vector of observations, and the ensemble size (N_e). `Func_err` returns the ensemble of measurements errors $\boldsymbol{\varepsilon}$ and its covariance matrix $\mathbf{R}$.

  For instance, if the user needs to generate an ensemble of observation errors normally distributed with zero mean and variance $1{\cdot}10^{-4}$, the function is

```
def Func_err(Obs, ens):
    from Tools import ErrorGenerator
    N_obs=Obs.shape[0]
    var_y=1e-4
    eps,R=ErrorGenerator.NormalError(var_y, N_obs, ens)
    return (eps,R)
```

- `forward_transf` is used to set up the parameter space transformation. It uses the Transformation.py module and the only argument is the ensemble of parameters. It returns the ensemble of parameters in the transformed

space.

For instance, to log transform the parameters, the function is written

```
def forward_transf(xx):
    from Tools import Transformation as T
    xx=T.Log_forward(xx)
    return xx
```

- `backward_transf` is used to back-transform the parameters into their physical space. Like `forward_transf`, it makes use of the Transformation.py module. The function argument is the updated ensemble of parameters in the transformed space, returning it back-transformed.

  For instance, to back-transform from the log space, the function is written

```
def forward_transf(xx):
    from Tools import Transformation as T
    xx=T.Log_backward(xx)
    return xx
```

- `localization` is used to set up the covariance localization; it makes use of the Localization.py module. The template function provided in the repository only needs to be modified to provide specific values for the space and time correlation distances in Eq. (13) in case localization is to be applied.

InputSettings.py module includes two functions that allow defining the performance metrics:

- `Metrics_obs` is used to select the metrics to be used to evaluate the performance by comparing actual observations and predicted values. It makes use of the Metrics.py module. The function arguments are the ensemble of parameter estimates, `Xprev`, the ensemble of state variable predictions at observation location `pred`, and the observations `obs`. It returns a dictionary containing the selected metrics.

  For instance, if the user wants to compute the RMSE between observations and model prediction and the AES, the function is

```
def Metrics_obs(Xprev,pred,obs):
    from Tools import Metrics as m
    par=Xprev
    RMSE_obs=[m.RMSE(obs.flatten(), pred.mean(1))]
    AES=[m.AES(par)]
```

```
    metrics_dict ={}
    for variable in ['RMSE_obs','AES']:
        metrics_dict[variable]=eval(variable)
    return metrics_dict
```

- `Metrics_obs_par` is used when a reference solution is available. The RMSE of the parameters can also be calculated. The function is like the previous one with an additional argument `True_par` containing the reference parameter values.

Template files for all of these modules are provided in the software repository, together with the specific files for the three example applications described next.

4. Example applications

genES-MDA has been tested in three case studies. All the examples are available on GitHub as explained in the Code Availability section. The first example presents the reverse flow routing problem in a river reach; then, two groundwater case studies are discussed: the identification of a conductivity field using hydraulic head information, and the estimation of the release history of a contaminant spill in an aquifer based on concentration data.

4.1. Case1: reverse flow routing

The reverse flow routing is the process to estimate the inflow hydrograph at ungauged locations in a river reach on the basis of available information at downstream gauged stations (D'Oria and Tanda, 2012; D'Oria et al., 2014). The numerical model describing the forward routing is the Hydrologic Engineering Center River Analysis System (HEC-RAS v. 5.0.7) of the US Army Corps of Engineers (Brunner, 2010). genES-MDA is tested using one of the HEC-RAS example projects, freely downloadable from the site `http://www.hec.usace.army.mil/software/hec-ras/download.aspx`. After downloading HEC-RAS, the test project is contained in the subfolder "Example Data\1D Unsteady Flow Hydraulics" and labeled "Multiple Reaches with Hydraulic Structures". The river system, sketched in Fig. 2, consists of two rivers (Butte Creek and Fall River), which merge at the Sutter Junction; Fall River is divided into two reaches named Upper Reach and Lower Reach. Two bridges, one on the Butte Creek River and one on the Lower Reach, and a culvert on the Upper Reach complete the system.

genES-MDA is applied to estimate the inflow hydrograph to the Butte Creek River (River station 0.4 in Fig. 2). The upstream boundary condition at River station 10.4 in Fig. 2 is a known flow hydrograph and the downstream boundary condition at station 9.3 in Fig. 2 is the normal depth, computed using Manning's equation based on a known bed slope. The observations are water levels collected on the Lower Reach downstream of the bridge section (River station 9.6 in Fig. 2). The reference observations were obtained through a forward routing of the actual inflow hydrograph to the

Butte Creek River provided by the HEC-RAS example. The total simulation time is three days, the upstream discharge hydrograph and the observed stage hydrograph were both discretized in intervals of 1 hour, resulting in a total number of parameters to identify (N_p) of 73, and the same number of observations (m).

For this case study, genES-MDA requires four Python libraries; numpy, os, win32com and pydsstools. The latter is freely downloadable from `https://github.com/gyanz/pydsstools`. Note that the examples provided have been prepared to run under a Windows operating system computer; the package can be used with a MacOS computer or a Linux one, in which cases, the libraries to interface with the operating systems will be different. The pydsstools python library is used to manage the HEC-DSS data base files.

Mod.py is adapted for this specific problem as follows. The `write_input` routine copies the latest estimate of the inflow hydrograph to the Butte Creek River in the input file of HEC-RAS with extension .u##. The original file in the example ("3ReachUnsteady.u01") is copied to a template file ("3ReachUnsteady.tpl"), which is read and used as the baseline to build the new input file. The function `run` contains the command to run the current HEC-RAS model making use of the HECRAS Controller (Goodell, 2014). The function `read_output` reads the stage hydrograph at the station where observations are collected. The results of the HEC-RAS are stored in a HEC-DSS database file.

InputSettings.py is modified to set up an optimal algorithm configuration. The ES-MDA was performed using an ensemble of 35 realizations; the initial ensemble is constituted of Gaussian functions generated using the function `PdfNormal` of the Ensemble_Generator.py module. For each ensemble, the coefficients of the Gaussian function (Eq. (17)) are selected randomly from uniform distributions over fixed ranges: A $\in$ U[2,7] m^3/s, B $\in$ U[$1.5\cdot10^{-6}$,$1.5\cdot10^{-5}$] m^3, μ $\in$ U[$5.2\cdot10^{-4}$,$1.4\cdot10^{-5}$] s, and σ $\in$ U[$1.0\cdot10^{-4}$,$3.9\cdot10^{-4}$] s. The observation errors were generated using the `NormalError` function of the Errors_Generator.py module; these are normally distributed with zero mean and variance equal to $1\cdot10^{-4}$ m^2. The total number of iterations is set to 6 and a decreasing α series is obtained with α_{geo}=3 (Eq. (3)). The module `Localization` is used to apply temporal covariance localization with a correlation time length of 10 h (b_t in Eq. (13)). Covariance inflation is applied with an inflation factor equal to 1.01 (r in Eq. (14)). Fig. 3 shows the actual and estimated inflow hydrographs (as the median of the predicted hydrographs) with its uncertainty interval. Fig. 4 shows the observed and predicted stage hydrographs. Both the discharge and stage hydrographs are well reproduced.

The ES-MDA performance is evaluated using the root mean square errors between observed and predicted water levels (RMSE$_{obs}$ = $3 \cdot 10^{-3}$ m) and between actual and estimated discharge values (RMSE$_{par}$=0.10 m^3/s), the Nash-Sutcliffe efficiency coefficient for the estimated parameters (NSE$_{par}$=99.99 %) and the parameter average ensemble spread (AES=0.55 m^3/s). All the metrics confirm the capability of genES-MDA to solve the inverse problem.

4.2. Estimation of a hydraulic conductivity field

The second application is the estimation of a zoned hydraulic conductivity field from hydraulic head data obtained through a pumping test. The reference permeability field (Fig. 5) is taken from the literature (Ayvaz, 2010); it is composed of 5 homogeneous zones, the conductivity values of which can be read in Table 1. The groundwater flow was modeled with MODFLOW 2005 (Harbaugh, 2005) under steady-state flow conditions. The two-dimensional model has a grid spacing in the x and y directions of 100 m and the aquifer saturated thickness is 30 m. The aquifer has constant hydraulic heads on the upper-left (100 m) and lower-right (80 m) boundaries are no-flow condition at the other boundaries. The flow rate extracted by the well is 3000 m^3/day. Observations (hydraulic heads) were collected at the eleven monitoring points (m=11) shown in Fig. 5; their values were obtained by a forward run of MODFLOW using the actual hydraulic conductivity field. The parameters to be estimated are the homogeneous conductivity values of each zone (N_p=5).

For this case study, genES-MDA requires the numpy, math and os libraries and the FloPy package. FloPy is a Python interface to MODFLOW, developed by Bakker et al. (2016), that allows to build model input files, run the model and read outputs through Python scripts. The functions of module Mod.py, `write_input`, `run` and `read_output`, are adapted making use of the FloPy package to update the permeability values by modifying the Layer Property Flow (LPF) file, run MODFLOW and extract the predicted hydraulic heads at observation locations.

InputSettings.py was built for the following ES-MDA configuration. The ensemble of realizations is 30, which are initially generated as uniform random values in the range [5, 150] m/day using the function `Random` of the Ensemble_Generator.py module. The observation errors were generated using the `NormalError` function of the Errors_Generator.py module; they are normally distributed with zero mean and variance equal to $1{\cdot}10^{-4}$ m^2. The total number of iterations is equal to 5 and a series of decreasing α is obtained with α_{geo}=3 (Eq. (3)). Covariance localization was not applied, and covariance inflation was applied with an inflation factor equal to 1.01 (r in Eq. (14)).

Table 1 reports the actual and estimated parameters, with their 90% confidence interval computed from the ensembles. The estimated field reproduces well the conductivities of the reference aquifer as well as the observed hydraulic heads (Table 2). The RMSE between observations and predictions, at the last iteration, is equal to 0.01 m. The conclusion is that the code is able to reconstruct the aquifer with little uncertainty.

4.3. Recovering the release history of two simultaneous point-source groundwater contamination events

genES-MDA has been applied to the Ayvaz (2010) case study. It deals with the estimation of the release history of two simultaneous point-source contaminant events from observed concentration data. The groundwater system is the same as the previous example with two pollution sources (of known location) and seven concentration monitoring

Table 1
Case 2 - Actual and estimated (ensemble median) hydraulic conductivities with its 90% confidence interval (5th-95th percentiles).

Parameter	Actual (m/day)	Estimated (m/day)
K_1	34.56	34.52 (33.91, 35.51)
K_2	17.28	17.24 (16.74, 17.75)
K_3	8.64	8.74 (8.39, 9.07)
K_4	25.92	26.08 (24.79, 27.78)
K_5	60.48	61.2 (58.01, 64.25)

Table 2
Case 2 - Observed and predicted (ensemble median) hydraulic heads with its 90% confidence interval (5th-95th percentiles).

Observation	Actual (m)	Estimated (m)
O1	98.07	98.06 (98.05, 98.07)
O2	97.28	97.25 (97.24, 97.29)
O3	93.27	93.26 (93.24, 93.27)
O4	90.77	90.75 (90.72, 90.78)
O5	92.72	92.71 (92.68, 92.77)
O6	84.79	84.78 (84.75, 84.81)
O7	82.15	82.14 (82.12, 82.16)
O8	84.98	84.97 (84.92, 85.01)
O9	82.04	82.03 (82.01, 82.04)
O10	80.80	80.79 (80.76, 80.81)
O11	80.39	80.38 (80.37, 80.39)

wells (Fig. 6).

MODFLOW 2005 (Harbaugh, 2005) and MT3DMS (Zheng and Wang, 1999) are combined to model the groundwater flow and mass transport, respectively. Groundwater flow is under steady-state condition. The hydraulic parameters and geometry characteristics of the model are the same as described above; effective porosity is equal to 0.3, longitudinal and transversal dispersivities are 40 m and 4 m, respectively. The initial condition is zero concentration everywhere. The total simulation time is 5 years, divided into 10 equal stress periods of 6 months. A conservative pollutant is released simultaneously from two sources during the first four stress periods with different fluxes. The parameters to be estimated are the four values of the released fluxes for each source (N_p=8). The observations are pollutant concentrations collected at the seven monitoring wells (Fig. 6) at the end of each year (m=28), which were obtained by a forward run using the actual release history.

genES-MDA uses the FloPy package to couple the Python code with MODFLOW and MT3DMS. The Source-Sink Mixing (SSM) file is modified by the `write_input` function of the module Mod.py, in order to update the contaminant release history at each iteration. MT3DMS is run from the `run` function. The predicted concentrations at the locations where observations are available is performed in the `read_output` function.

InputSettings.py is written for the following ES-MDA configuration. The number of realizations is 50, which are initially generated as random values in the range [1000,9000] kg/day using the function `Random` of the Ensem-

Table 3
Case 3 - Actual and estimated (ensemble median) source fluxes with their 90% confidence interval (5th-95th percentiles).

Source	Stress period	Actual (kg/day)	Estimated (kg/day)
S1	SP1	2074	2090 (1948, 2242)
	SP2	4838	4878 (4527, 5168)
	SP3	3715	3738 (3293, 3986)
	SP4	3024	3009 (2886,3214)
S2	SP1	3024	3026 (2994, 3050)
	SP2	7776	7764 (7560, 7967)
	SP3	5616	5612 (5215, 5957)
	SP4	4061	4027 (3798, 4332)

ble_Generator.py module. The observation errors were generated using the `PercNormalError` function of the Errors_Generator.py module; they are normally distributed with zero mean and variances computed according to the percentage error p=2%. The total number of iterations was equal to 5 and a decreasing α obtained with α_{geo}=2 (Eq. (3)) was adopted. The spatiotemporal covariance localization was applied considering a correlation time length of 5 years (b_t, Eq. (13)) and a correlation space length of 2500 m (b_s, Eq. (13)). The covariance inflation is applied with a factor equal to 1.01 (r, Eq. (14)).

In Table 3, the actual and estimated parameters with their uncertainty intervals are compared. The estimated mass fluxes reproduce well the reference solution, the RMSE between observed and estimated concentrations is $2{\cdot}10^{-3}$ g/l. The results are in agreement with other literature works (see e.g., Ayvaz, 2010; Jamshidi et al., 2020). The conclusion is that the code can also address problems of contaminant source identification.

5. Code running

To verify the results presented using genES-MDA, the software package should be downloaded and, from the subfolder corresponding to each one of the three cases, the python command `ES_MDA` should be run using the default settings.

6. Conclusions

This paper presents the Python software package genES-MDA, a model-independent and open-source code to solve generic inverse problems by means of the Ensemble Smoother with Multiple Data Assimilation. genES-MDA provides several functionalities that allow implementing alternative configurations of the algorithm suited for diverse problems. genES-MDA was tested using three synthetic examples in the context of surface and subsurface hydrology: the solution of the reverse flow routing problem in a river reach, the estimation of the hydraulic conductivity field, and the identification of the release history of a contaminant spill in an aquifer. In addition, genES-MDA has been already applied for the solution of more complex real case studies, see e.g., Todaro et al. (2021) and D'Oria et al. (2021). The

good results of all tests demonstrate the capabilities of genES-MDA and its flexibility to solve different types of inverse problems with good performance.

Furthermore, genES-MDA has a flexible workflow that can be easily adapted for other variants of the Ensemble Kalman filter. Future works will focus on the improvement of the software package by integrating different EnKF methods, including alternative covariance inflation approaches, and implementing a parallelized version to take advantage that, being an ensemble-based method, several realizations can be evaluated at once as done in Xu et al. (2013).

7. Limitations and future developments

The software package presented in this paper is the first version of genES-MDA and has some limitations that will be addressed in future versions of the code. For instance, several approaches to performing covariance localization and covariance inflation exist in the literature, not all of which are available in the code; they will be incorporated in future versions.

The Transformation module could include new functions to perform the parameter space transformation. In particular, the normal-score transformation (Zhou et al., 2011) will be added to handle non-Gaussian distributed parameters. Although Ensemble Kalman filters have been demonstrated to be able to achieve good solutions even working with non-Gaussian parameters, they are optimal for multi-Gaussian distributed parameters. The normal-score transformation has been proven to give very good results for highly non-Gaussian parameters.

One of the main disadvantage of ES-MDA is the need to define a priori the number of iterations N and the inflation coefficients α_i; their optimal values are determined by trial and error. Automatic procedures will be integrated into genES-MDA to adaptively select N and α_i following the works by other researchers (see e.g. Le et al., 2016; Emerick, 2019).

Another possible improvement of the software package could be the introduction of a more general input control file that interacts (or replaces) the InputSettings module. For example, an external file that groups the inputs following the JSON file format, which can be easily managed within Python, could be used.

8. Acknowledgments

The fourth author would like to acknowledge grant PID2019-109131RB-I00 funded by MCIN/AEI/10.13039/501100011033 and the University of Parma for hosting him as a visiting professor, during which time the paper was finalized.

The authors are thankful to the anonymous Reviewers for their valuable comments.

Code availability section

Name of code: genES-MDA

Developer: Valeria Todaro

Contact address: Department of Engineering and Architecture, University of Parma, Parco Area delle Scienze 181/A, 43124 Parma, Italy

E-mail: valeria.todaro@unipr.it

Year first available: 2022

Hardware required: none

Software required: Python

Program language: Python

Program size: 21.4 MB

Source codes: https://github.com/ValeriaTodaro/genES-MDA

References

Anderson, J., Hoar, T., Raeder, K., Liu, H., Collins, N., Torn, R., Avellano, A., 2009. The data assimilation research testbed: A community facility. Bulletin of the American Meteorological Society 90, 1283–1296. doi:10.1175/2009bams2618.1.

Anderson, J.L., 2001. An ensemble adjustment kalman filter for data assimilation. Monthly Weather Review 129, 2884–2903. doi:10.1175/1520-0493(2001)129<2884:aeakff>2.0.co;2.

Anderson, J.L., 2007. Exploring the need for localization in ensemble data assimilation using a hierarchical ensemble filter. Physica D: Nonlinear Phenomena 230, 99–111. doi:10.1016/j.physd.2006.02.011.

Anderson, J.L., Anderson, S.L., 1999. A monte carlo implementation of the nonlinear filtering problem to produce ensemble assimilations and forecasts. Monthly Weather Review 127, 2741–2758. doi:10.1175/1520-0493(1999)127<2741:amciot>2.0.co;2.

Ayvaz, M.T., 2010. A linked simulation-optimization model for solving the unknown groundwater pollution source identification problems. Journal of Contaminant Hydrology 117, 46–59. doi:10.1016/j.jconhyd.2010.06.004.

Bakker, M., Post, V., Langevin, C.D., Hughes, J.D., White, J.T., Starn, J.J., Fienen, M.N., 2016. Scripting MODFLOW model development using python and FloPy. Groundwater 54, 733–739. doi:10.1111/gwat.12413.

Brunner, G.W., 2010. HEC-RAS, River Analysis System Hydraulic Reference Manual Version 4.1. US Army Corps of Engineers, Institute for Water resource. Hydrologic Engineering Center. Davis, California.

Butera, I., Gómez-Hernández, J.J., Nicotra, S., 2021. Contaminant-source detection in a water distribution system using the ensemble kalman filter. Journal of Water Resources Planning and Management 147. doi:10.1061/(asce)wr.1943-5452.0001383.

Capilla, J.E., Gömez-Hernández, J.J., Sahuquillo, A., 1998. Stochastic simulation of transmissivity fields conditional to both transmissivity and piezometric head data—3. application to the culebra formation at the waste isolation pilot plan (wipp), new mexico, usa. Journal of Hydrology 207, 254–269.

Capilla, J.E., Rodrigo, J., Gómez-Hernández, J.J., 1999. Simulation of non-gaussian transmissivity fields honoring piezometric data and integrating soft and secondary information. Mathematical Geology 31, 907–927.

Chen, Y., Oliver, D.S., 2009. Cross-covariances and localization for EnKF in multiphase flow data assimilation. Computational Geosciences 14, 579–601. doi:10.1007/s10596-009-9174-6.

D'Oria, M., Mignosa, P., Tanda, M.G., 2014. Bayesian estimation of inflow hydrographs in ungauged sites of multiple reach systems. Advances in Water Resources 63, 143–151. doi:10.1016/j.advwatres.2013.11.007.

D'Oria, M., Mignosa, P., Tanda, M.G., Todaro, V., 2021. Estimation of levee breach discharge hydrographs: comparison of inverse approaches. Hydrological Sciences Journal , 1–11doi:10.1080/02626667.2021.1996580.

D'Oria, M., Tanda, M.G., 2012. Reverse flow routing in open channels: A bayesian geostatistical approach. Journal of Hydrology 460-461, 130–135. doi:10.1016/j.jhydrol.2012.06.055.

Emerick, A.A., 2019. Analysis of geometric selection of the data-error covariance inflation for ES-MDA. Journal of Petroleum Science and Engineering 182, 106168. doi:10.1016/j.petrol.2019.06.032.

Emerick, A.A., Reynolds, A.C., 2012. History matching time-lapse seismic data using the ensemble kalman filter with multiple data assimilations. Computational Geosciences 16, 639–659. doi:10.1007/s10596-012-9275-5.

Emerick, A.A., Reynolds, A.C., 2013. Ensemble smoother with multiple data assimilation. Computers & Geosciences 55, 3–15. doi:10.1016/j.cageo.2012.03.011.

Evensen, G., 1994. Sequential data assimilation with a nonlinear quasi-geostrophic model using monte carlo methods to forecast error statistics. Journal of Geophysical Research 99, 10143. doi:10.1029/94jc00572.

Evensen, G., 2018. Analysis of iterative ensemble smoothers for solving inverse problems. Computational Geosciences 22, 885–908. doi:10.1007/s10596-018-9731-y.

Feyen, L., Gómez-Hernández, J., Ribeiro Jr, P., Beven, K.J., De Smedt, F., 2003. A bayesian approach to stochastic capture zone delineation incorporating tracer arrival times, conductivity measurements, and hydraulic head observations. Water resources research 39.

Franssen, H., Gómez-Hernández, J., 2002. 3d inverse modelling of groundwater flow at a fractured site using a stochastic continuum model with multiple statistical populations. Stochastic Environmental Research and Risk Assessment 16, 155–174.

Gaspari, G., Cohn, S.E., 1999. Construction of correlation functions in two and three dimensions. Quarterly Journal of the Royal Meteorological Society 125, 723–757. doi:10.1002/qj.49712555417.

Godoy, V.A., Napa-García, G.F., Gómez-Hernández, J.J., 2021. Ensemble smoother with multiple data assimilation as a tool for curve fitting and parameter uncertainty characterization: Example applications to fit nonlinear sorption isotherms. Mathematical Geosciences doi:10.1007/s11004-021-09981-7.

Gómez-Hernández, J., Franssen, H.J.H., Sahuquillo, A., 2003. Stochastic conditional inverse modeling of subsurface mass transport: a brief review and the self-calibrating method. Stochastic Environmental Research and Risk Assessment 17, 319–328.

Gómez-Hernández, J.J., Xu, T., 2021. Contaminant source identification in aquifers: A critical view. Mathematical Geosciences doi:10.1007/s11004-021-09976-4.

Goodell, C., 2014. Breaking the HEC-RAS code : a user's guide to automating HEC-RAS. h2ls, Portland, Or.

Hamill, T.M., Whitaker, J.S., Snyder, C., 2001. Distance-dependent filtering of background error covariance estimates in an ensemble kalman filter. Monthly Weather Review 129, 2776–2790. doi:10.1175/1520-0493(2001)129<2776:ddfobe>2.0.co;2.

Harbaugh, A.W., 2005. MODFLOW-2005: the u.s. geological survey modular ground-water model–the ground-water flow process. doi:10.3133/tm6a16.

Houtekamer, P.L., Mitchell, H.L., 1998. Data assimilation using an ensemble kalman filter technique. Monthly Weather Review 126, 796–811. doi:10.1175/1520-0493(1998)126<0796:dauaek>2.0.co;2.

Hunt, B.R., Kostelich, E.J., Szunyogh, I., 2007. Efficient data assimilation for spatiotemporal chaos: A local ensemble transform kalman filter. Physica D: Nonlinear Phenomena 230, 112–126. doi:10.1016/j.physd.2006.11.008.

Jamshidi, A., Samani, J.M.V., Samani, H.M.V., Zanini, A., Tanda, M.G., Mazaheri, M., 2020. Solving inverse problems of unknown contaminant source in groundwater-river integrated systems using a surrogate transport model based optimization. Water 12, 2415. doi:10.3390/w12092415.

Kalman, R.E., 1960. A new approach to linear filtering and prediction problems. Journal of Basic Engineering 82, 35–45. doi:10.1115/1.3662552.

Le, D.H., Emerick, A.A., Reynolds, A.C., 2016. An adaptive ensemble smoother with multiple data assimilation for assisted history matching. SPE Journal 21, 2195–2207. doi:10.2118/173214-pa.

Li, H., Kalnay, E., Miyoshi, T., 2009. Simultaneous estimation of covariance inflation and observation errors within an ensemble kalman filter. Quarterly Journal of the Royal Meteorological Society 135, 523–533. doi:10.1002/qj.371.

Li, L., Zhou, H., Hendricks Franssen, H.J., Gómez-Hernández, J.J., 2012. Modeling transient groundwater flow by coupling ensemble kalman filtering and upscaling. Water Resources Research 48.

Liang, X., Zheng, X., Zhang, S., Wu, G., Dai, Y., Li, Y., 2011. Maximum likelihood estimation of inflation factors on error covariance matrices for ensemble kalman filter assimilation. Quarterly Journal of the Royal Meteorological Society 138, 263–273. doi:10.1002/qj.912.

Nerger, L., Hiller, W., 2013. Software for ensemble-based data assimilation systems—implementation strategies and scalability. Computers & Geosciences 55, 110–118. doi:10.1016/j.cageo.2012.03.026.

Ridler, M.E., van Velzen, N., Hummel, S., Sandholt, I., Falk, A.K., Heemink, A., Madsen, H., 2014. Data assimilation framework: Linking an open data assimilation library (OpenDA) to a widely adopted model interface (OpenMI). Environmental Modelling & Software 57, 76–89. doi:10.1016/j.envsoft.2014.02.008.

Silva, T.M., Bela, R.V., Pesco, S., Barreto, A., 2021. ES-MDA applied to estimate skin zone properties from injectivity tests data in multilayer reservoirs. Computers & Geosciences 146, 104635. doi:10.1016/j.cageo.2020.104635.

Sun, N.Z., 1994. Inverse Problems in Groundwater Modeling. Kluwer Academic.

Tarantola, A., 2005. Inverse Problem Theory and Methods for Model Parameter Estimation. CAMBRIDGE.

Tippett, M.K., Anderson, J.L., Bishop, C.H., Hamill, T.M., Whitaker, J.S., 2003. Ensemble square root filters. Monthly Weather Review 131, 1485–1490. doi:10.1175/1520-0493(2003)131<1485:esrf>2.0.co;2.

Todaro, V., D'Oria, M., Tanda, M.G., Gómez-Hernández, J.J., 2019. Ensemble smoother with multiple data assimilation for reverse flow routing. Computers & Geosciences 131, 32–40. doi:10.1016/j.cageo.2019.06.002.

Todaro, V., D'Oria, M., Tanda, M.G., Gómez-Hernández, J.J., 2021. Ensemble smoother with multiple data assimilation to simultaneously estimate the source location and the release history of a contaminant spill in an aquifer. Journal of Hydrology 598, 126215. doi:10.1016/j.jhydrol.2021.126215.

Wen, X.H., Capilla, J.E., Deutsch, C., Gómez-Hernández, J., Cullick, A., 1999. A program to create permeability fields that honor single-phase flow rate and pressure data. Computers & Geosciences 25, 217–230.

White, J.T., Hunt, R.J., Fienen, M.N., Doherty, J.E., 2020. Approaches to highly parameterized inversion: PEST++ version 5, a software suite for parameter estimation, uncertainty analysis, management optimization and sensitivity analysis. doi:10.3133/tm7c26.

Xu, T., Gómez-Hernández, J.J., 2016. Joint identification of contaminant source location, initial release time, and initial solute concentration in an aquifer via ensemble kalman filtering. Water Resources Research 52, 6587–6595. doi:10.1002/2016wr019111.

Xu, T., Gómez-Hernández, J.J., Chen, Z., Lu, C., 2020. A comparison between ES-MDA and restart EnKF for the purpose of the simultaneous identification of a contaminant source and hydraulic conductivity. Journal of Hydrology , 125681doi:10.1016/j.jhydrol.2020.125681.

Xu, T., Gómez-Hernández, J.J., Li, L., Zhou, H., 2013. Parallelized ensemble kalman filter for hydraulic conductivity characterization. Computers

& Geosciences 52, 42–49. doi:10.1016/j.cageo.2012.10.007.

Zheng, C., Wang, P.P., 1999. MT3DMS : a modular three-dimensional multispecies transport model for simulation of advection, dispersion, and chemical reactions of contaminants in groundwater systems; documentation and user's guide.

Zheng, X., 2009. An adaptive estimation of forecast error covariance parameters for kalman filtering data assimilation. Advances in Atmospheric Sciences 26, 154–160. doi:10.1007/s00376-009-0154-5.

Zhou, H., Gómez-Hernández, J.J., Franssen, H.J.H., Li, L., 2011. An approach to handling non-gaussianity of parameters and state variables in ensemble kalman filtering. Advances in Water Resources 34, 844–864. doi:10.1016/j.advwatres.2011.04.014.

Zhou, H., Gómez-Hernández, J.J., Li, L., 2012. A pattern-search-based inverse method. Water Resources Research 48.

Zhou, H., Gómez-Hernández, J.J., Li, L., 2014. Inverse methods in hydrogeology: Evolution and recent trends. Advances in Water Resources 63, 22–37. doi:10.1016/j.advwatres.2013.10.014.

List of Figures

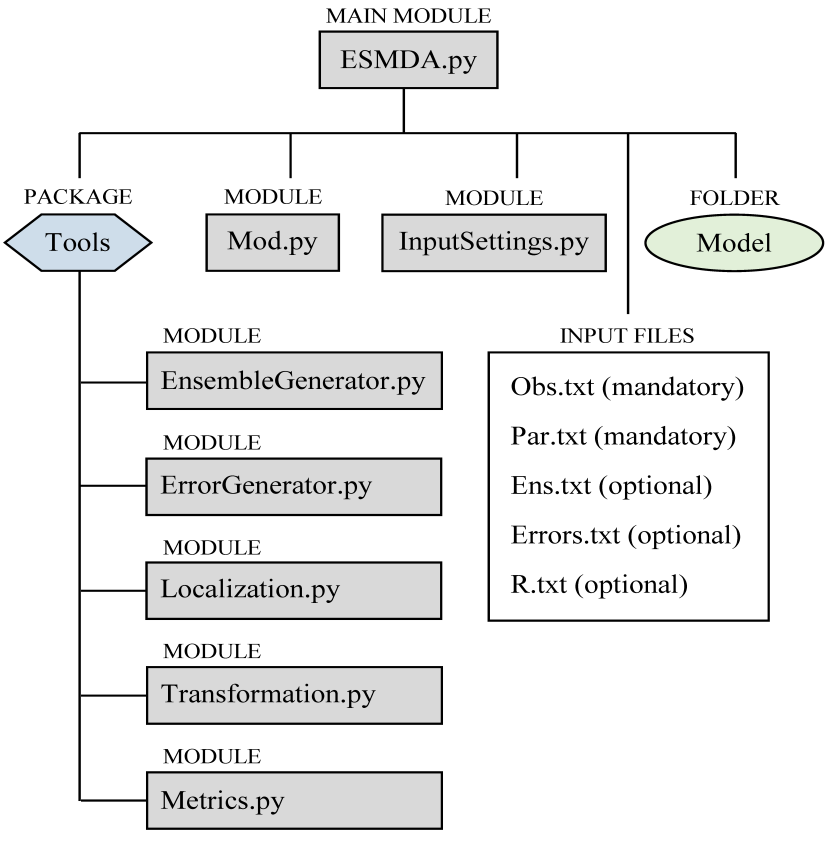

Figure 1: Software package structure.

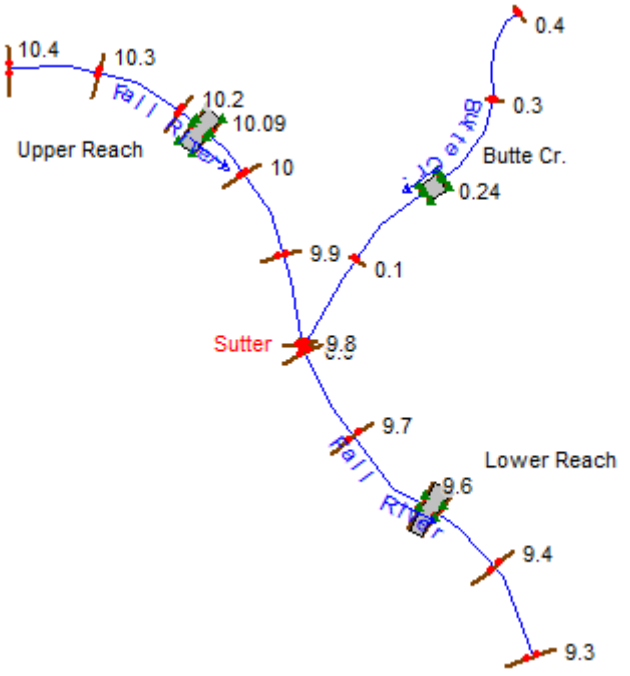

Figure 2: Case 1 - HEC-RAS geometry.

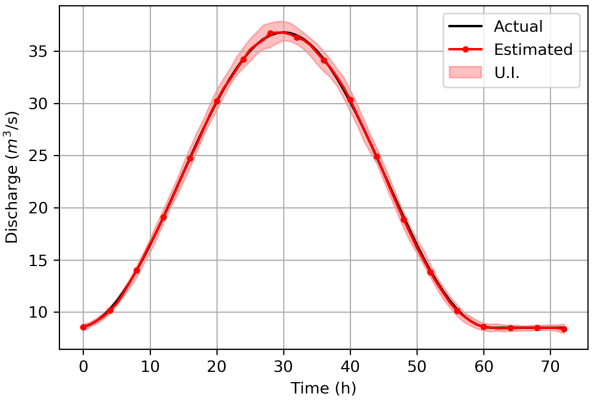

Figure 3: Case 1 - Actual and estimated inflow hydrographs. The ensemble of estimated hydrographs is summarized by its median value and the 90% confidence interval (5th-95th percentiles).

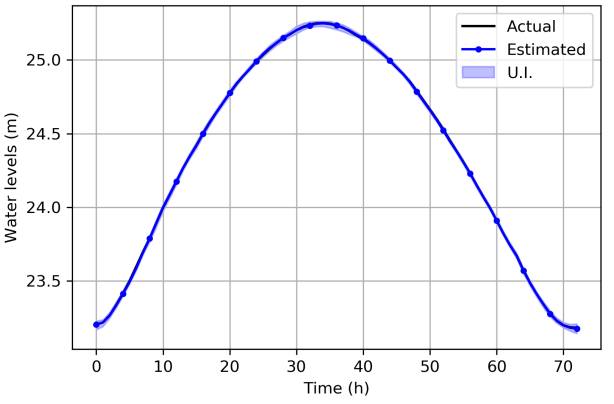

Figure 4: Case 1 - Observed and predicted water levels. The ensemble of predicted water levels is summarized by its median value and the 90% confidence interval (5th-95th percentiles).

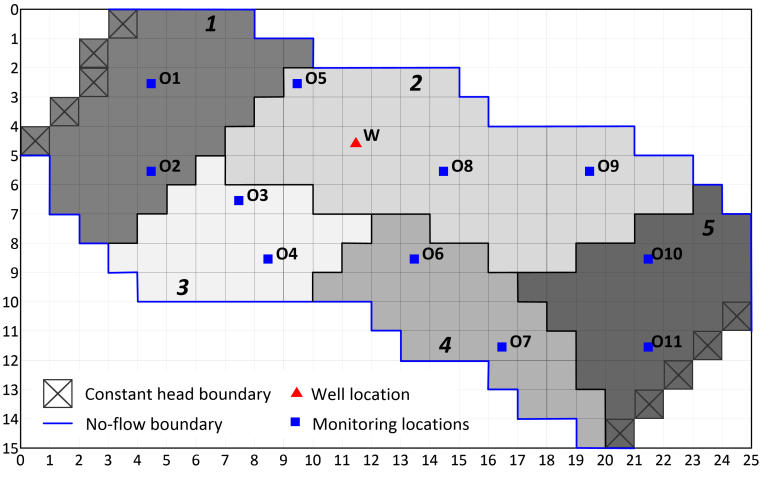

Figure 5: Case 2 - Reference permeability field; the squares mark the monitoring locations and the triangles, the well locations.

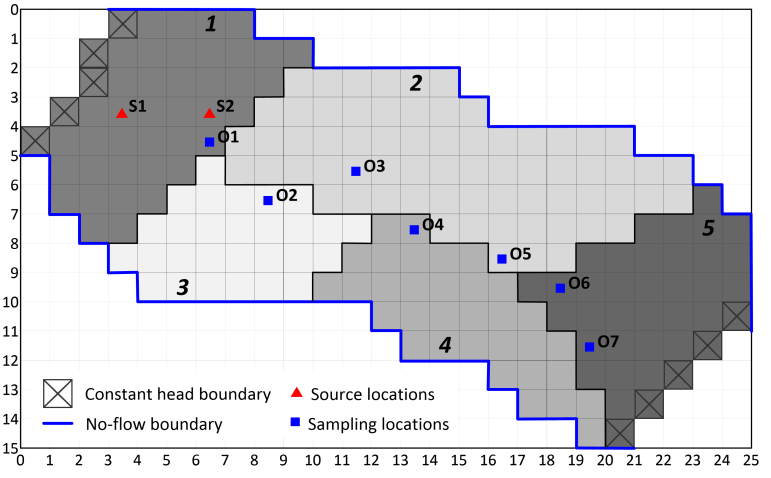

Figure 6: Case 3 - Aquifer model; the squares denote the monitoring locations and the triangles indicate the pollution sources.