Journal of
open research software

Smurf: System for Modelling with Uncertainty Reduction, and Forecasting

SOFTWARE
METAPAPER

ISABELLE MIROUZE
SOPHIE RICCI
Author affiliations can be found in the back matter of this article

ubiquity press

ABSTRACT

Smurf is an open source modular system developed in Python for running and cycling data assimilation (DA) systems. It is organised around three super classes for numerical model management, assimilation schemes and observation instruments. Any new item can be easily plugged in by defining a child class that will override as many methods as necessary. Non intrusive, Smurf can be used in any applicative domain for numerical models written in any language.

CORRESPONDING AUTHOR:
Isabelle Mirouze
CECI, CNRS UMR 5318
isabelle.mirouze31@gmail.com

KEYWORDS:
Data Assimilation; EnKF; History Matching; Python; Hydraulics; Toy Model

TO CITE THIS ARTICLE:
Mirouze, I and Ricci, S 2021 Smurf: System for Modelling with Uncertainty Reduction, and Forecasting. *Journal of Open Research Software*, 9: 2. DOI: *https://doi.org/10.5334/jors.312*

INTRODUCTION

Data assimilation (DA) aims to reduce the uncertainties in a modelled system by correcting the initial and boundary conditions, the current state, and the parameters of this system. Firstly developed in the sixties by meteorologists [12, 2, 4], it has widely spread through different domains within geosciences (ocean, hydrology, atmospheric chemistry, ...), and different schemes have been designed (variational, filtering, ...). DA is schematically presented in ***Figure 1*** and DA variables are represented along with their uncertainty. A focus is made on filtering algorithm hereafter for illustrative purposes only, but it should be noted that any other scheme can nevertheless be implemented in SMURF. The classical form of the Kalman Filter (KF) [7] is illustrated in ***Figure 1a*** and the ensemble Kalman filter (EnKF) [3] is illustrated in ***Figure 1b***. KF stands in two steps: the prediction and the analysis that are sequenced over assimilation cycles indexed by k in the following. The first step of DA consists of defining the *control vector* $\boldsymbol{x}$ that gathers the control variables, *i.e.* the quantities to correct. The control vector is propagated from time t_k to time t_{k+1} by the operator $M_{k,k+1}$, a numerical model generally. At DA cycle k, the values firstly assigned to these quantities are provided either by the user, a previous simulation, or an estimate, and form the *background vector* $\mathbf{x}^b_k$ (represented by blue dots in ***Figure 1*** along with its associated uncertainties whose covariance matrix is noted $\mathbf{B}_k$). When the quantities are of the same nature as the available observations $\mathbf{y}^o_k$ they can easily be compared to these observations (represented by green stars in ***Figure 1*** along with their associated uncertainties whose covariance matrix is noted $\mathbf{R}_k$), possibly by using interpolation. When the corrected quantities and the observations are of a different nature, an *observation operator* H_k (or tangent linear $\mathbf{H}_k$) must be used to define the model counterpart in the observation space. Whether the observation operator is simple (interpolation) or complex (a dedicated model or the model itself), it is used to calculate the *innovation vector* $\mathbf{d}_k$, *i.e.* the difference between the observations and the model counterparts. The analysis step then provides an *analysis vector* $\mathbf{x}^a_k$ (represented by red squares in ***Figure 1*** along with its associated uncertainties whose covariance matrix is noted $\mathbf{A}$), *i.e.* the control vector whose values are corrected by adding an *increment* to the *background vector* as displayed in Eq. 1. Calculating the *increment* is then done by defining a gain matrix $\mathbf{K}_k$ (filtering DA) to weight the innovations as displayed in Eq. 2. In the prediction step, the analysis is propagated over time and provides the background vector $\mathbf{x}^b_{k+1}$ for the next assimilation cycle k+1 (Eq. 3). The analysis can be computed at each observation time, or over a time window within which several observations are available:

$$\mathbf{x}^a_k = \mathbf{x}^b_k + \mathbf{K}_k \mathbf{d}_k, \tag{1}$$

$$\mathbf{K}_k = \mathbf{B}_k \mathbf{H}_k^T (\mathbf{H}_k \mathbf{B}_k \mathbf{H}_k^T + \mathbf{R}_k)^{-1}, \tag{2}$$

$$\mathbf{x}^b{}_{k+1} = M_{k1,k+1}(\mathbf{x}^b_k). \tag{3}$$

The background and observation accuracy is accounted for by involving the covariance matrices of their respective errors. The background error covariance matrix $\mathbf{B}_k$ can either be static by defining a unique matrix that is used all along the simulation, or it can be recomputed using the errors of the moment. For example, in the EnKF, an ensemble of analysis vectors (red squares in ***Figure 1b***) is computed from an ensemble of background vectors (blue dots in ***Figure 1b***). The background (resp. analysis) error covariance matrix is computed stochastically using a Monte Carlo technique from the ensemble of N background (resp. analysis) values represented in ***Figure 1b*** (Eq. 4):

$$\mathbf{B}_k = \frac{1}{N-1} \sum (\mathbf{x}^b{}_k - \overline{\mathbf{x}^b{}_k})(\mathbf{x}^b{}_k - \overline{\mathbf{x}^b{}_k})^T. \tag{4}$$

DA systems are often constructed around a specific numerical model and for a particular scheme. Although this structure allows the implementation of appropriate

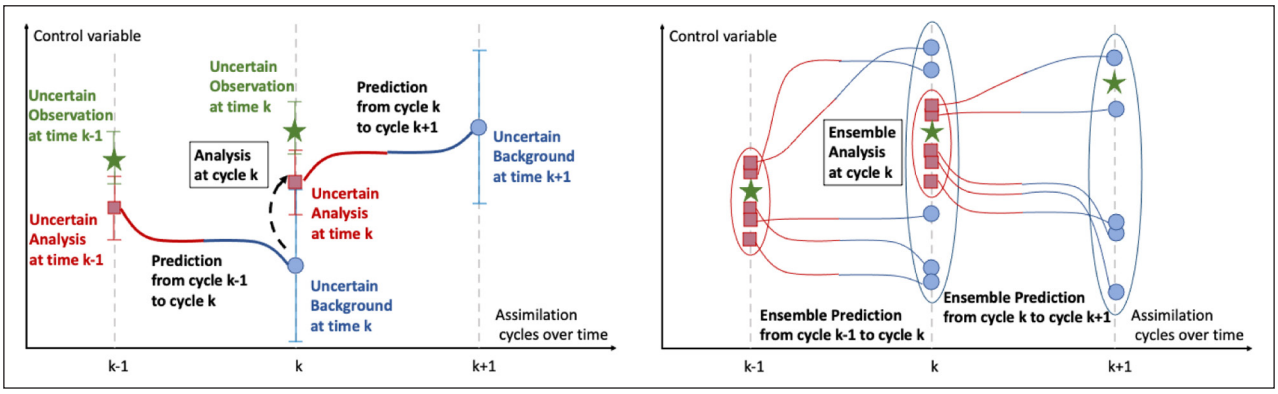

Figure 1 Data assimilation scheme. **(a)** At each cycle k of the KF, an analysis (red squares) is calculated from the comparison of the background (blue dots) with the observation (green stars). **(b)** At each cycle k of the EnKF, an ensemble of analyses (red squares) is calculated from the comparison of each background (blue dots) with the observation (green stars).

and ad hoc algorithms, it can make it difficult and time consuming to implement new schemes, to take into account new observation types, or to change the numerical solver for the physics involved, especially for huge systems such as Numerical Weather Prediction (NWP) or Ocean Prediction. To address this issue, technical infrastructures such as OOPS[1] (Object Oriented Prediction System) at ECMWF (European Center for Medium-range Weather Forecasting) and JEDI[2] (Joint Effort for Data assimilation Integration) at NCAR (National Center for Atmospheric Research) have been designed. These infrastructures aim to couple different components of a DA system.

A lot of different institutes have developed their own DA codes for their own purposes. Some systems have also been developed with a view to being shared, either for research or training purposes. For example DART[3] (Data Assimilation Research Testbed) [1] at NCAR, written in Fortran 90 and distributed via a subversion repository (SVN), has already been used with different geophysical models (atmospheric chemistry, ocean, climate, ...). It provides different sequential assimilation algorithms and particle filters, with Matlab post-processing scripts. For small to medium systems, DAPPER[4] (Data Assimilation with Python: a Package for Experimental Research) [9] at NERSC (Nansen Environmental and Remote Sensing Center), written in Python and available under github, provides a large choice of assimilation algorithms (variational, sequential and particle filters) to be used with some academic models (*e.g* Lorenz). It has been used in research studies on assimilation methodology and for training purposes. A list of alternative projects is available on the DAPPER page.

The Smurf project comes from the requirement to design a DA system in Python for different hydraulic models with current and future observations, with as little as possible intrusion in the solvers. It seemed to us that designing a new architecture under this requirement would be more appropriate than reshaping an existing DA system. Even though the first desired method was an EnKF, other assimilation schemes are planned. Smurf answers the need for a modular system where new models, new assimilation schemes and new observation types can be plugged in easily. Smurf is thus not restricted to hydraulics but can be used in any other domain for small to medium systems. Non intrusive, it does not constrain the language the model is written in, it only requires to be able to launch a simulation with a set of new values for the *control vector* and to retrieve output quantities of interest. Smurf also manages the cycling of the assimilation corrections during the simulation, and the archiving of data. For flexibility purposes, the experiments are set up easily using external configuration files.

APPLICATIONS

Smurf is currently used in a hydraulic research project to prepare for the SWOT (Surface Water and Ocean Topography) mission[5] that should be launched in 2021. The altimetry instrument on board SWOT is expected to provide water level observation maps at high-resolution for rivers over 100 meters wide. The project aims to study the merits of assimilating SWOT data to represent and forecast river discharge for ungauged catchments. So far, the one-dimensional solver Mascaret[6] [5] has been used with Smurf to show the benefit of using SWOT-like observations in terms of mean and root mean square error for the water elevation [8].

On-going work in collaboration with CSIRO-Data61 (Commonwealth Scientific and Industrial Research Organisation, Data61) aims to improve crowd evacuation modelling with the PiXIE[7] solver [13]. In this study, Smurf is used to correct parameters related to social interaction and forces that influence the behaviour of agents in a room they must evacuate. The correction is calculated by assimilating snapshots of the evacuation (cumulated number of evacuated people) as well as the total time of evacuation [10].

In the domain of geological reservoir modelling, OPM[8] (Open Porous Media) has been plugged in to Smurf. The aim is to study history matching, *i.e.* to try and recover a geological configuration from oil/gas/water rates measurements in a well.

DEMONSTRATION

For pedagogical purposes, the toy model Barbatruc[9] developed at CERFACS has been plugged in to Smurf. Barbatruc solves the two-dimensional Navier-Stokes equations to study the expansion of a passive tracer source, *e.g.* a pollutant. The tracer source is specified as a simple scalar at a particular location within a squared domain. The model parameters ρ (density) and ν (kinematic viscosity) are set up, and a forcing term is defined at the western domain boundary to mimic the effect of a pressure-driven channel (*u_west*). The Navier-Stokes equations are integrated over time and the solver outputs the horizontal components of the velocity, pressure and passive tracer fields.

Validation of a DA system, is conveniently achieved in the framework of twin experiments. First, a deterministic **reference** experiment is set up assuming "true" values for the model parameters (MP), the initial (IC) and the boundary conditions (BC). Observations are then generated by extracting values of observed fields at chosen locations and times. A random noise is added to these values to simulate an observation error. Assuming

now that the MP, the IC and the BC are unknown, a priori values are set up for a deterministic **control** experiment. The output fields of the control show errors compared with the reference outputs. Assimilating observations in a third experiment (**assimilation** experiment), aims to correct the MP, the IC and the BC in order to reduce these errors, and hence brings output fields closer to the reference fields.

For the demonstration, twin experiments are carried out for two different cases: 1) MP and BC correction; 2) IC correction. The setup for each experiment is summarised in ***Table 1***. The Navier-Stokes equations are integrated for 0.1 s, and the fields are recorded every 0.01 s. Observations are generated from the reference tracer field every 10 grid points at 0.05 s and 0.09 s, and an unbiased random normal error with a standard deviation of 10^{-5} is added. The assimilation experiments use an EnKF DA algorithm with 12 members split on 4 processors. The size of the ensemble is limited for the demonstration to avoid any computation resource requisite. Each member of the ensemble is generated using a *control vector* whose values are drawn from a uniform law in the range defined in ***Table 1***.

The experiments are assessed using classical DA diagnostics for both test cases. Firstly, the final tracer field (at t = 0.1 s) is plotted for the reference (***Figure 2***) experiment, using green cross symbols to show the position of the observations assimilated in both test cases. Histograms of the *background* and *analysis* values are plotted for the different variables of the *control vector*, showing the possible reduction of the ensemble spread (uncertainty reduction) and recentering of the ensemble towards the reference values (***Figures 4*** and ***6***). The root mean square (RMS) of the error between the control or assimilation final tracer field and the reference final tracer field is plotted (***Figures 3*** and ***5***).

In the first testing case, the MP ρ and ν, and the BC *u_west* are included in the *control vector* while the location and magnitude of the source (IC) are assumed to be known. As expected, assimilating the observations (***Figure 3b***) significantly reduces the RMS of the final tracer field error with respect to the reference experiment, compared to the control experiment (***Figure 3a***). As shown in ***Figure 4b*** and ***4c***, the ensemble spread for ν and *u_west* is smaller and centered closer to the reference values after DA analysis. No significant correction is seen for ρ (***Figure 4a***), suggesting that within the considered range, it has little significant influence on the tracer field.

In the second testing case, the MP ρ and ν, and the BC *u_west* are assumed to be known, while the location and magnitude of the source (IC) are included in the *control vector*. It should be noted that moving the source south west of the reference position reduces the RMS error north east of the true position (***Figure 5a***). As expected, assimilating the observations (***Figure 5b***) significantly reduces the RMS of the final tracer field error with respect to the reference experiment, compared to the control experiment (***Figure 5a***). We note, however, that there is still an important RMS error west of the true position. This is explained by the corrected position that is properly centered around the reference value 10 (***Figure 6a***) on the y-axis while en error remains for the correction on the x-axis (***Figure 6b***) with analysed value lower (around 8.5) than the reference value (10), *i.e.* west to the reference value. Regarding the source magnitude (***Figure 6c***), the correction allows for a significant reduction for all members, bringing the analysis closer to the reference value as expected.

Other DA diagnostics are used for both test cases although not shown in the paper. The uncertainty reduction is measured by comparing the norm of the

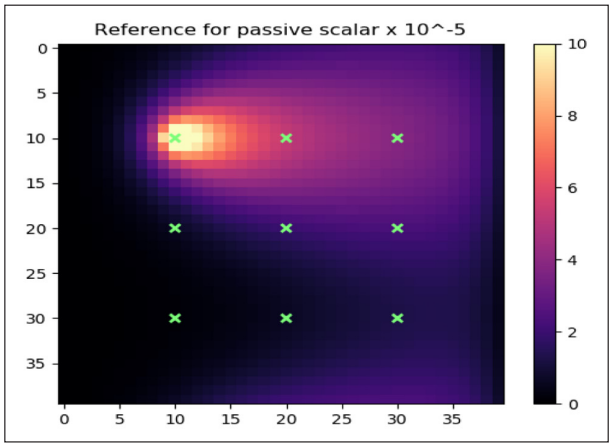

Figure 2 Final tracer field (t = 0.1 s) of the reference experiment. The green cross symbols show the locations of the observations.

	MP		BC	IC		
	ρ	ν	*u_west*	*y*	*x*	magnitude
Reference	1.0	1.0	20.0	10	10	1.0
Control 1	5.0	0.5	10.0	10	10	1.0
Assim 1	u[0.5, 8.5]	u[0.2, 1.2]	u[0.0, 20.0]	10	10	1.0
Control 2	1.0	1.0	20.0	12	8	2.0
Assim 2	1.0	1.0	20.0	u[10, 14]	u[6, 10]	u[1.0, 3.0]

Table 1 Set up of the twin experiments for the model parameters (MP), the boundary conditions (BC) and the initial condition (IC).

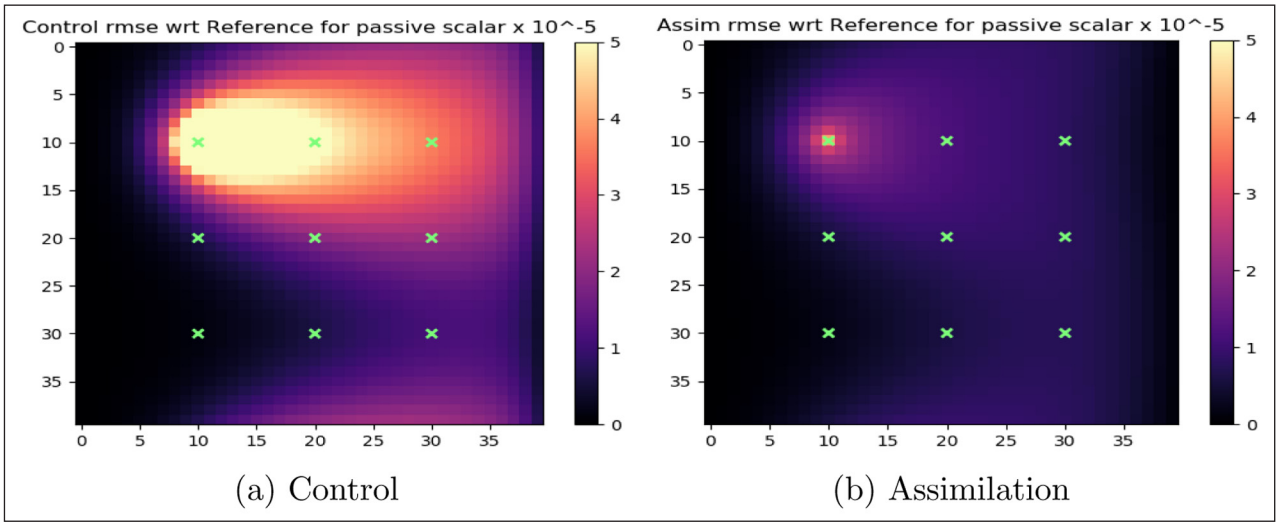

Figure 3 Test case 1: root mean square of the final tracer field error (t = 0.1 s) with respect to the reference experiment. The green cross symbols show the locations of the observations.

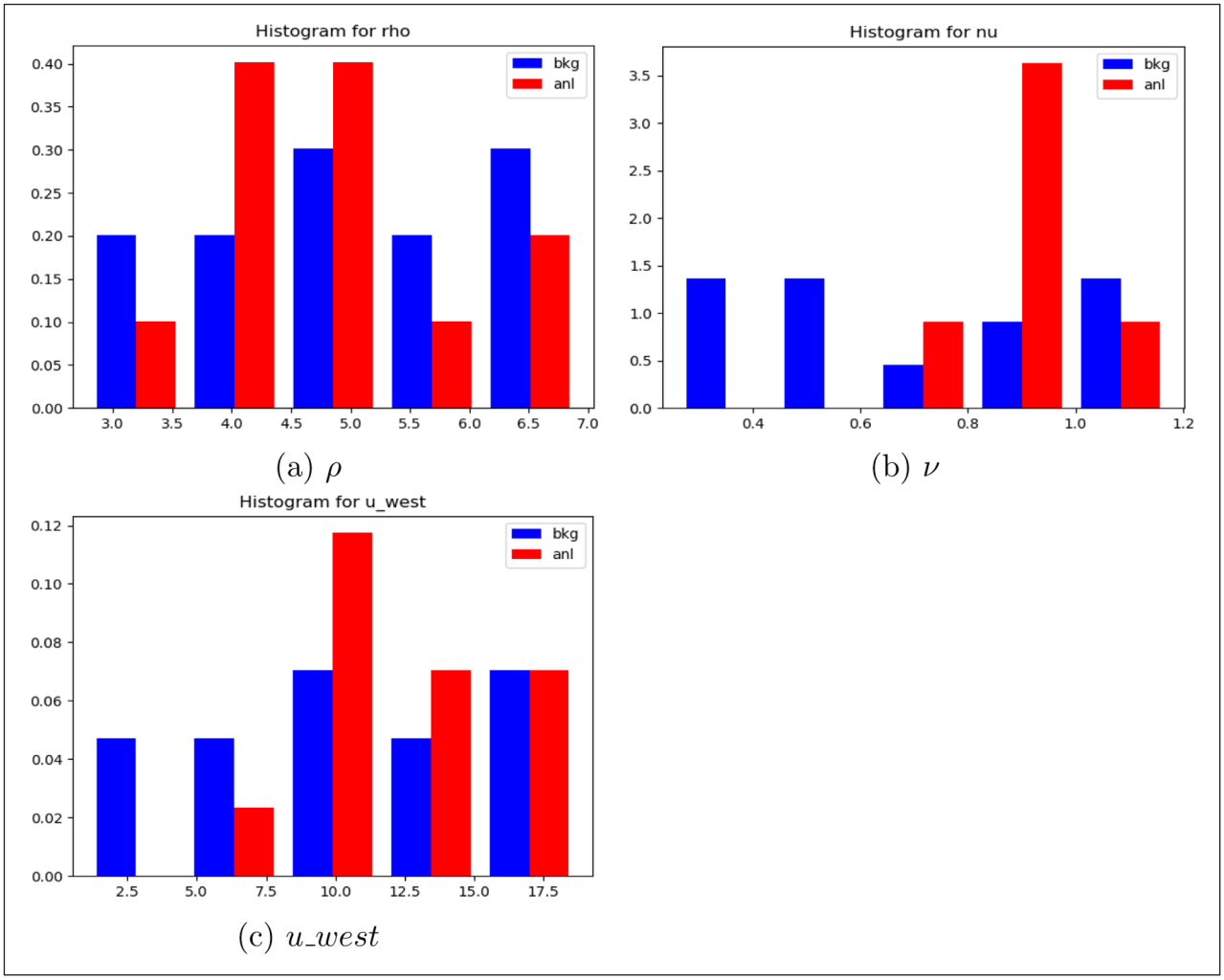

Figure 4 Test case 1: histogram of the parameter values before (blue) and after (red) DA analysis.

background and *analysis* error covariance matrices. The correlations between the *control vector* variables are plotted, showing spurious off-diagonal correlations due to the small size of the ensemble. The RMS of the error between observations and *background* (*innovations*) or *analysis* model counterpart are plotted at the observation locations, showing a reduction of the error for the latter. Another diagnostics shows the rank diagram of the observations, *i.e.* the distribution of the observations compared to the distribution of the ensemble members. This diagnostics allows us to check if the ensemble is correctly generated.

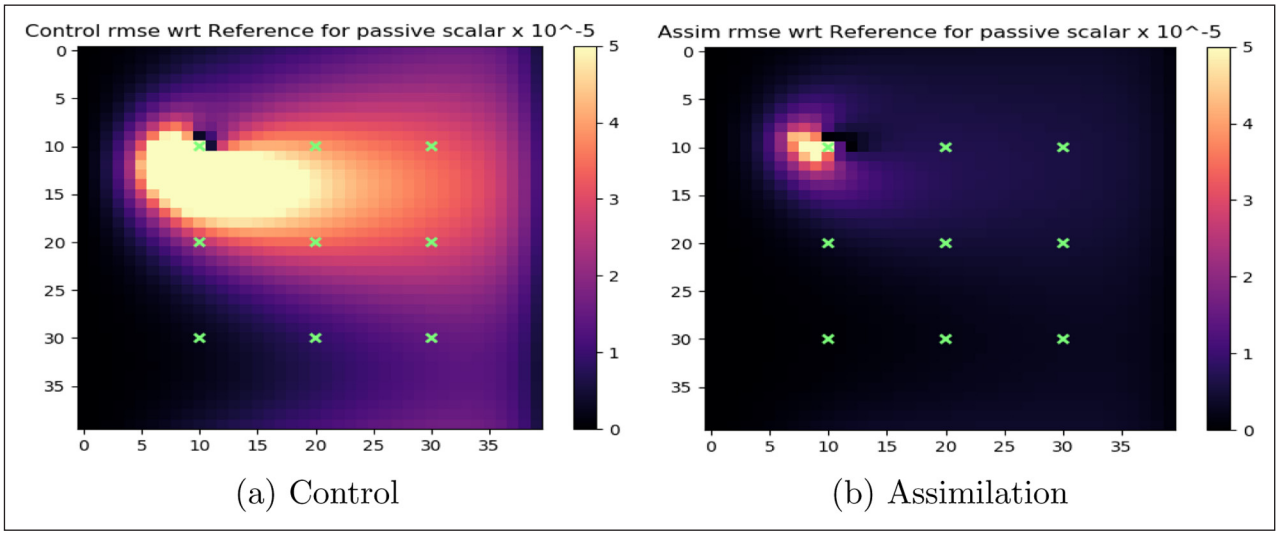

Figure 5 Test case 2: root mean square of the final tracer field error (t = 0.1 s) with respect to the reference experiment. The green cross symbols show the locations of the observations.

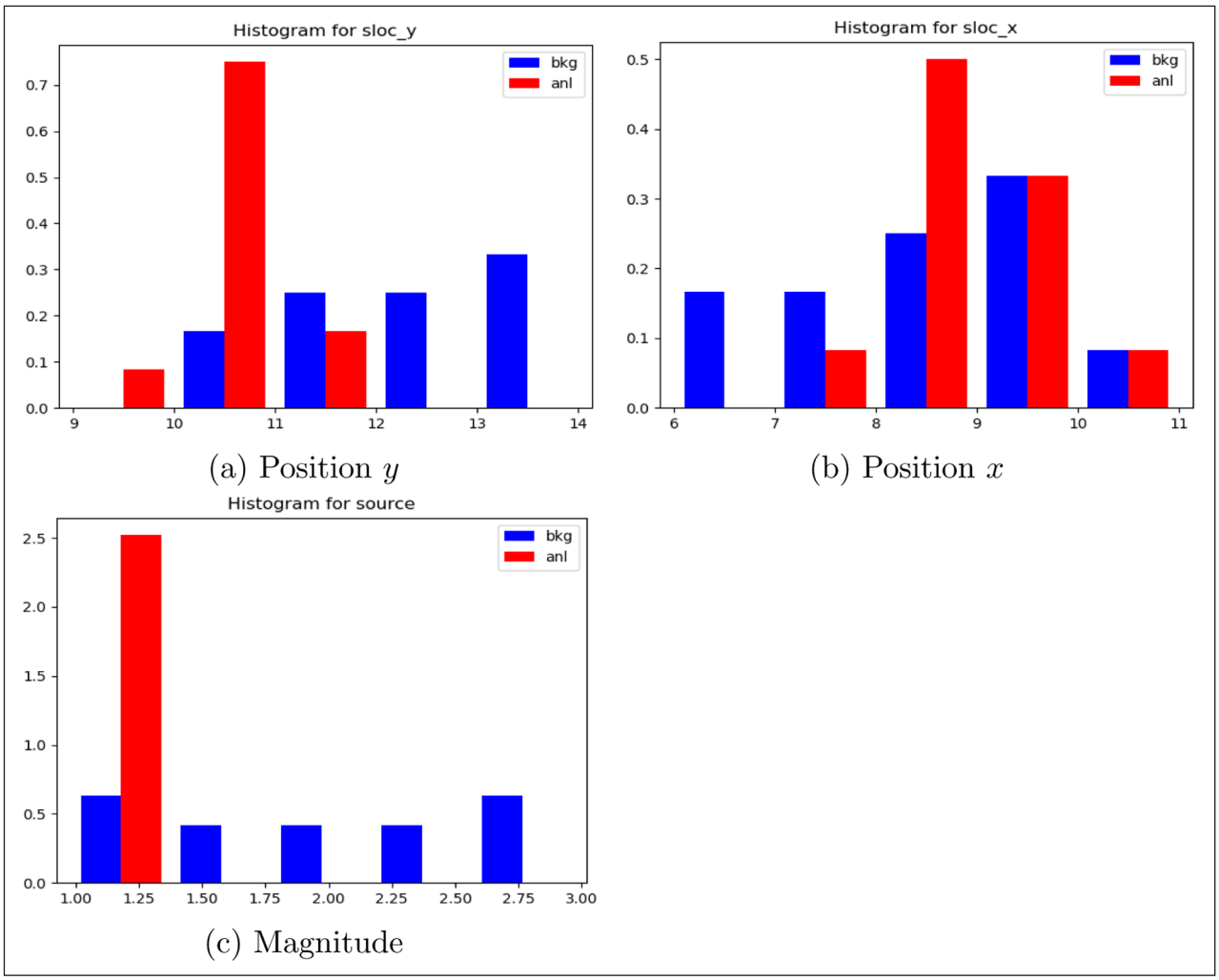

Figure 6 Test case 2: histogram of the source position and magnitude before (blue) and after (red) DA analysis.

IMPLEMENTATION AND ARCHITECTURE

The Smurf repository contains three main sub-directories: *Smurf* that contains the source code divided into several directories sheltering the python files depending on their nature (models, assim, observations, ...); *testing_case* that contains some examples (see section Quality Control); *templates* that contains skeletons to help plugging in other models or observation instruments to Smurf as well as configuration file templates.

Smurf is driven by the class *Experiment* that manages the cycling of the experiment, the directories, the archiving, and the calls required to perform the simulation, as shown in ***Figure 7***. The mother classes *Model*, *Assim* and *Instrument* define the attributes and methods required to plug in a model, an assimilation scheme and an observation instrument, respectively. A daughter class must be defined each time a new item is plugged in, inheriting methods from its mother class, but also overriding as many methods as required. A class *Perturbation* allows Monte Carlo methods such as the EnKF to generate the different members of the ensemble, using, for example uniform laws as in the section Demonstration. This class relies mainly on the Python package *ot-batman* [11] to sample the perturbations, although alternative methods could be added to read external perturbations. The *common* sub-directory contains basic functions and redefines classes such as *Vector* and *Matrix*. The background and observation error covariance matrices are daughter classes of *Matrix*, although sheltered by the sub-directory *covariances*. The class *ObsVector* in the sub-directory *observations* represents the observation vector and is a daughter class of *Vector*. Finally a class *PostProcessing* carries out the assessment of the experiments.

So far, four different numerical models have been plugged in Smurf: Barbatruc (toy model for fluid dynamics, Python), Mascaret (one-dimensional hydraulics, Fortran 90 with python application programming interfaces), PiXIE (crowd dynamics and evacuation, C++) and OPM (porous media processes, C++). The numerical models themselves are not distributed within Smurf, except for Barbatruc, and must be installed before they can be used with Smurf. Associated with the numerical models, different observation instruments are available: *Barbametre* (tracer values; Barbatruc), *Gauge* (limnimetric stations; Mascaret), *Chronos* (final time of evacuation; PiXIE) and *Clicker* (cumulated number of people evacuated at a specific time; PiXIE), *WellInstrument* (well measurements; OPM). Whilst the Smurf daughter classes *Barbatruc* and *Mascaret* are available publicly along with their instruments, the model and instrument classes for PiXIE and OPM are available on demand, only on dedicated git branches. A *Swot* instrument and its three variants *SwotPixelCloud*, *SwotRiverReach* and *SwotRiverNode* are currently under development and will be available once the definitive format of the SWOT data is confirmed. In terms of assimilation schemes, the stochastic EnKF is the only algorithm available at the moment. Smurf is in constant evolution and other items will be added to the existing structure.

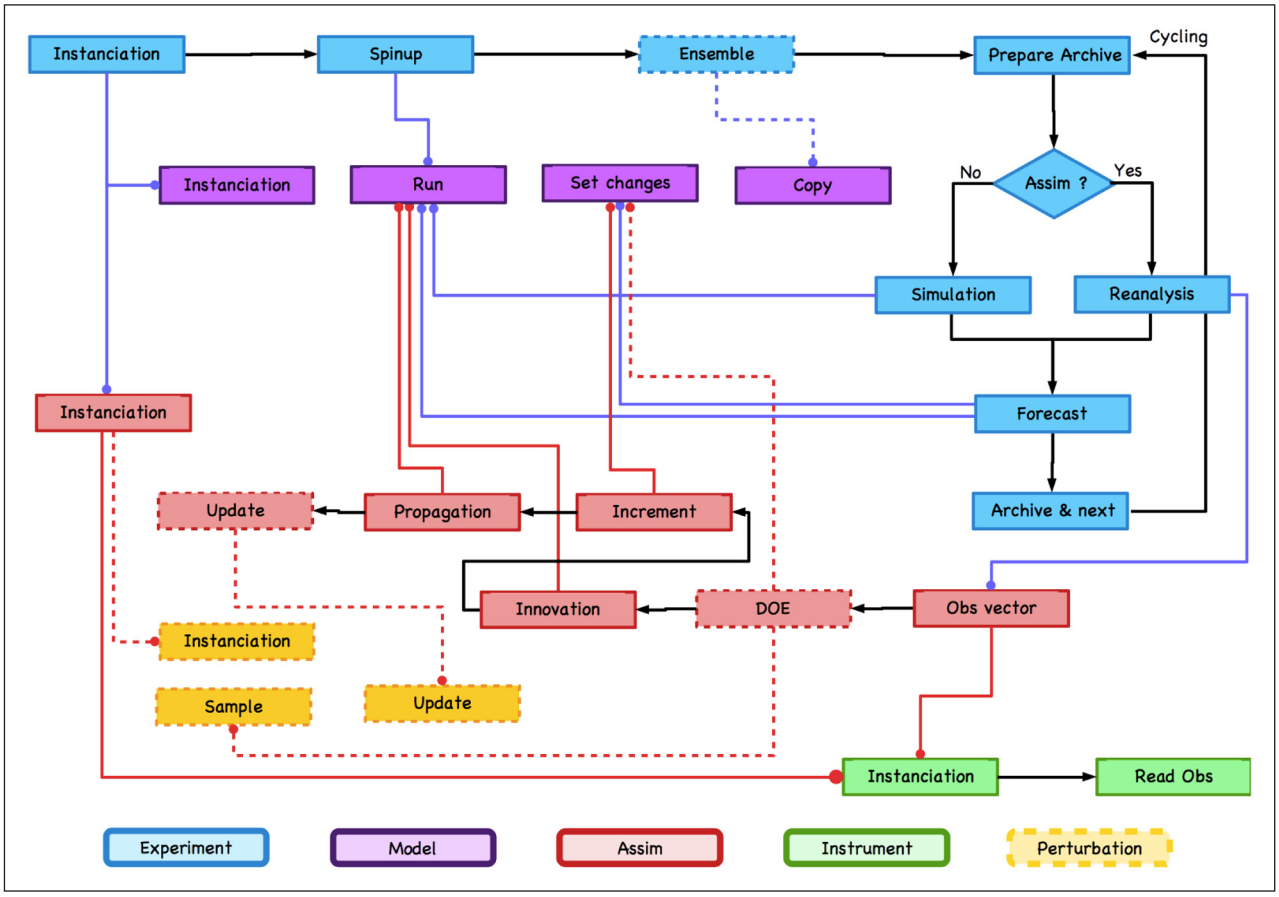

Figure 7 Smurf architecture. The different colors refer to the different classes: *Experiment* (blue), *Model* (purple), *Assim* (red), *Instrument* (green) and *Perturbation* (orange). Lines ending with an arrow indicate the next step in the chain, whereas lines ending with a dot indicate a call to another class method. Dashed lines are specific to ensemble systems such as the EnKF.

When running an ensemble method (EnKF), Smurf is able to launch several numerical simulations (Innovation, Propagation and Forecast steps of ***Figure 7***) simultaneously in parallel, using the *pathos* package. Since some classes might not be serialisable (Fortran code, NetCDF4 data, ...), the lists of model and instrument instantiations are kept out of the classes *Experiment* and *Assim* and used as global variables during the instantiation. A list of the object identifiers rather than the objects themselves is then defined as an attribute within the classes. The objects are recovered when needed using the package *ctypes*.

Smurf conforms to the PEP8[10] coding convention except for a few variable names that contain capital letters, because of their obvious meaning for data assimilators (B, R, Hxb, ...) [6]. Smurf is developed using the JetBrains Community Edition IDE (Integrated Development Environment) PyCharm.[11]

QUALITY CONTROL

Smurf classes are tested on their own during their development. Moreover, in the main sub-directory *testing_case*, configuration files are provided to run and assess twin experiments for each plugged model (Pixie and Opm are available on demand only). A README.md file helps understanding and running the tests.

For Barbatruc, the test consists of running the twin experiments described previously in the section Demonstration. For Mascaret, the test proposes to run the model on a 50-kilometre reach of the Garonne (South of France) from Tonneins to La Réole, via Marmande. Hourly water height observations at Le Mas d'Agenais and Marmande are assimilated to correct the Strickler coefficients and the upstream discharge. This test requires the numerical code Mascaret.

For each model, the testing case contains different files to configure the experiment:

- *configrun.yml*: configuration of the experiment
- *config_EnKF.yml*: configuration of the EnKF
- *parameter_Model.yml*: description of the Model parameters
- *test.py*: script for running Smurf
- *construct_obs.py*: script for constructing observations from a reference experiment
- *postproc.py*: script for assessing the experiment

A directory *Observations* contains a default observation file. Depending on the model, some extra files or directories are available. For Barbatruc, *config_case.yml* file is provided to configure the model. For Mascaret, a directory Garonne contains the configuration and restart files required by the model.

The testing cases are also used as non-regressive tests and are hence performed after each new implementation or modification in Smurf on linux personal computers (fedora 26), MacBook laptops and HPC clusters, depending on the physical model. Results are then compared to previous results obtained before the changes.

(2) AVAILABILITY

PROGRAMMING LANGUAGE

Smurf is written in Python 3 and has been tested with Anaconda Python 3.6 and 3.7.

ADDITIONAL SYSTEM REQUIREMENTS

The requirement in terms of memory depends mainly on the physical model used.

DEPENDENCIES

The following packages are required for Smurf

- numpy >= 1.15
- pathos >= 0.2
- matplotlib >= 3.0
- pyyaml >= 3.12
- ot-batman >= 1.9
- barbatruc == 0.0.2

For the Swot instruments, the package netCDF4 >= 1.5 is also required.

LIST OF CONTRIBUTORS

Authors:

- Isabelle Mirouze (CECI, CNRS UMR5318): author
- Sophie Ricci (CECI, CERFACS/CNRS UMR5318): co-author

Collaborators:

- Barbatruc:
  - Antoine Dauptain (CERFACS)
- Mascaret:
  - Andrea Piacentini (CERFACS)
  - Nicole Goutal (LNHE-EDF/LHSV)
  - Vanessya Laborie (CEREMA/LHSV)
  - Anne-Laure Tiberi (CEREMA)
- OPM:
  - Camille Besombes (CERFACS)
  - Corentin Lapeyre (CERFACS)
  - Rabeb Selmi (TOTAL)
- PiXIE:
  - Matt Bolger (CSIRO-Data61)
  - Vincent Lemiale (CSIRO-Data61)
- General purposes:
  - Antoine Dauptain (CERFACS)
  - Corentin Lapeyre (CERFACS)

SOFTWARE LOCATION

The master branch, the development branch and the Opm and Pixie branches are archived on the CERFACS nitrox repository and are available under specific registration only.

CODE REPOSITORY

Name: nitrox.cerfacs
Persistent identifier: *https://nitrox.cerfacs.fr/globc/Smurf.git*
Licence: Cecill B
Publisher: Isabelle Mirouze
Version published: 1.0.1
Date published: 15/11/2019

The master branch is mirrored on gitlab repository and is publicly available without any registration.

MIRROR

Name: gitlab
Persistent identifier: *https://gitlab.com/cerfacs/Smurf.git*
Licence: Cecill B
Date published: 15/11/2019

The master branch v1.0.1 has been released on the Python Package Index under the name Smurf-CERFACS and is available for installation using pip. It is also available under the Zenodo reference *http://doi.org/10.5281/zenodo.4432513*.

LANGUAGE

Smurf is integrally written in English.

(3) REUSE POTENTIAL

Using the models already plugged in but for a different configuration than the one defined in the *testing_case* sub-directory is straightforward. It is simply done by modifying the configuration files *configrun.yml*, *config_EnKF.yml*, and *parameter_Model.yml* (where Model is the name of the physical model), and possibly adapting the scripts *construct_obs.py* and *postproc.py*. The input files classically required by the physical model must also be provided. In the sub-directory *templates*, templates for the experiment configuration, the assimilation configuration and the model parameters are provided with an explanation of the items.

Assimilating observations from a new instrument can be done using the template *template_new_instrument.py*. This template is a skeleton describing the different attributes and methods in order to guide the definition of the new instrument. The template must be copied in the *Smurf/observations* directory under the name of the new instrument.

Similarly, it is possible to plug in a new model using the template *template_new_model.py*. This template is a skeleton describing the different attributes and methods in order to guide the definition of the new model. The template must be copied in the *Smurf/models* directory under the name of the new model. Associated with the new model, a parameter file must be defined using the template *template_parameter.yml*. This file is the file pointed to by the item *model: parameter* of the experiment configuration file.

No template currently exists for plugging in a new DA scheme. This can however be done following the mother class *assim* and overriding methods for calculating the *innovations*, the *increments*, ...

Thanks to its architecture detailed previously, Smurf can be reused in any applicative domain requiring DA. New projects are currently starting, in hydrology and two-dimensional hydraulics. Moreover, new collaborations are envisaged for flood extent and micro-meteorology, as well as with the OpenTURNS[12] initiative. Any other collaboration is welcome and can be initiated by contacting Sophie Ricci (ricci@cerfacs.fr). As previously mentioned, the master branch is available freely under the Cecill B license. It can therefore be used and developed outside any collaboration. However, due to limited human resources, the authors do not guarantee support for the use and development of Smurf, for instance through gitlab issues. In case help is needed, the users are welcome to contact the authors to investigate possible collaboration.

NOTES

1 *https://www.ecmwf.int/en/newsletter/153/news/progress-running-ifs-4d-var-under-oops*.
2 *https://jointcenterforsatellitedataassimilation-jedi-docs.readthedocs-hosted.com/en/latest/index.html*.
3 *http://www.image.ucar.edu/DAReS/DART/index.html*.
4 *https://github.com/nansencenter/DAPPER*.
5 *https://swot.cnes.fr/en/mission-1* and *https://swot.jpl.nasa.gov/mission.htm*.
6 *www.opentelemac.org*.
7 *https://research.csiro.au/pixie/*.
8 *https://opm-project.org/*.
9 *https://cerfacs.fr/coop/barbatruc*.
10 *https://www.python.org/dev/peps/pep-0008/*.
11 *https://www.jetbrains.com/pycharm/*.
12 *http://www.openturns.org*.

ACKNOWLEDGEMENTS

The authors acknowledge the CSG team at CERFACS for their suport with software and computational environment management. They are also grateful to Pamphile Roy who helped using the *ot-batman* package.

FUNDING STATEMENT

The development of Smurf has been funded by CERFACS and by the TOSCA program from CNES, with great contribution from CNES and EDF collaborators.

COMPETING INTERESTS

The authors have no competing interests to declare.

AUTHOR AFFILIATIONS

Isabelle Mirouze *orcid.org/0000-0002-5954-3056*
CECI, CNRS UMR 5318, FR

Sophie Ricci *orcid.org/0000-0002-4232-5626*
CECI, CERFACS/CNRS UMR 5318, FR

REFERENCES

1. **Anderson JL, Hoar T, Raeder K, Liu H, Collins N, Torn R, Arellano A.** The Data Assimilation Research Testbed: A community facility. *Bull. Amer. Meteor. Soc.* 2009; 90: 1283–1296. DOI: *https://doi.org/10.1175/2009BAMS2618.1*
2. **Cressman GP.** An operational objective analysis system. *Mon. Weather Rev.* 1959; 87: 367–374. DOI *https://doi.org/10.1175/1520-0493(1959)087<0367:AOOAS>2.0.CO;2*
3. **Evensen G.** Sequential data assimilation with a nonlinear quasigeostrophic model using Monte Carlo methods to forecast error statistics. *J. Geophys. Res.* 1994; 99: 10143–10162. DOI: *https://doi.org/10.1029/94JC00572*
4. **Gandin L.** *Objective Analysis of Meteorological Fields.* English translation by israel program for scientific translation, jerusalem, 1965 éd. Gridromet, Leningrad; 1963.
5. **Goutal N, Maurel F.** A finite volume solver for 1D shallow-water equations applied to an actual river. *Int. J. Numer. Methods Fluids.* 2001; 38: 1–19. DOI: *https://doi.org/10.1002/fld.201*
6. **Ide K, Courtier P, Ghil M, Lorenc A.** Unified notation for data assimilation: operational, sequential and variational. *J. Meteorol. Soc. Japan.* 1997; 75: 181–189. DOI: *https://doi.org/10.2151/jmsj1965.75.1B_181*
7. **Kalman RE.** A new approach to linear filtering and prediction problems. *Trans. ASME, J. Basic Eng.* 1960; 82: 35–45. DOI: *https://doi.org/10.1115/1.3662552*
8. **Mirouze I, Ricci S** and **Goutal N.** The impact of observation densification in an ensemble Kalman Filter. *Telemac User Conference,* Toulouse, France; 2019. DOI: *https://doi.org/10.5281/zenodo.3549572*
9. **Raanes PN,** et al. nansencenter/dapper: Version 0.8. 2018. DOI: *https://doi.org/10.5281/zenodo.2029296*
10. **Ricci S, Lemiale V, Mirouze I, Bolger M, Thio NA.** Sensitivity analysis for flood evacuation model. *9th International Conference on Sensitivity Analysis of Model Output,* Barcelona, Spain. 2019.
11. **Roy TP, Ricci S, Dupuis R, Campet R, Jouhaud JC, Cyril F.** BATMAN: Statistical analysis for expensive computer codes made easy. *JOOS*. 2018; 3(21): 493. *https://doi.org/10.21105/joss.00493*
12. **Sasaki Y.** An objective analysis based on the variational method. *J. Meteorol. Soc. Japan*. 1958; II-36, 77–88. DOI: *https://doi.org/10.2151/jmsj1923.36.3_77*
13. **Andrés-Thio N, Ras C, Bolger M, Lemial V.** A study of the role of forceful behaviour in evacuations via microscopic modelling of evacuation drills. *Safety Science*. 2021; 134: 105018. DOI: *https://doi.org/10.1016/j.ssci.2020.105018*

TO CITE THIS ARTICLE:
Mirouze, I and Ricci, S 2021 Smurf: System for Modelling with Uncertainty Reduction, and Forecasting. *Journal of Open Research Software,* 9: 2. DOI: *https://doi.org/10.5334/jors.312*

Submitted: 05 December 2019 Accepted: 18 November 2020 Published: 05 February 2021

COPYRIGHT:
© 2021 The Author(s). This is an open-access article distributed under the terms of the Creative Commons Attribution 4.0 International License (CC-BY 4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited. See *http://creativecommons.org/licenses/by/4.0/*.

Journal of Open Research Software is a peer-reviewed open access journal published by Ubiquity Press.