Groundwater

Methods Brief/

CFPy—A Python Package for Pre- and Postprocessing of the Conduit Flow Process of MODFLOW

by Thomas Reimann[1,2], Max Gustav Rudolph[1], Leonard Grabow[1], and Torsten Noffz[3]

Abstract

The conduit flow process (CFP) for MODFLOW's groundwater flow model is an advanced approach for investigating complex groundwater systems, such as karst, with coupled discrete-continuum models. CFP represents laminar and turbulent flow in a discrete pipe network coupled to a matrix continuum. However, the preprocessing demand is comparatively high to generate the conduit network and is usually performed with graphical user interfaces. To overcome this limitation and allow a scalable, reproducible, and comprehensive workflow, existing and new routines were aggregated to a Python package named CFPy, to allow script-based modeling that harmonizes well with the available and widely used FloPy package. CFPy allows information about the location and geometry of the conduit network to be considered by user-specific approaches or by sophisticated methods such as stochastic conduit network generators. The latter allows the automatic generation of many model variants with differing conduit networks for advanced investigations like multi-model approaches in combination with automatic parameter estimation. Additional postprocessing routines provide powerful control and valuable insights for CFP applications. In this methods note, a general technical description of the approach is complemented with two examples that guide users and demonstrate the main capabilities of CFPy.

Introduction

Karst systems are of great importance for water supply. Approximately 10% of the world's population relies on karst geology for water supply (Stevanović 2019).

[1]Institute for Groundwater Management, Technische Universität Dresden, 01062, Dresden, Germany; thomas.reimann@tu-dresden.de

[2]Corresponding author: Thomas Reimann, Institute for Groundwater Management, Technische Universität Dresden, 01062 Dresden, Germany; thomas.reimann@tu-dresden.de

[3]Department of Applied Geology, Georg-August-Universität Göttingen, 37077, Göttingen, Germany

Received December 2022, accepted May 2023.

doi: 10.1111/gwat.13331

Karst aquifers can be conceptualized as highly conductive karst conduits embedded in a less conductive and water-soluble host rock, represented by a matrix continuum (e.g., Liedl et al. 2003). Due to fractures and dissolution, resulting in enlarged cavities, karst systems are highly heterogeneous and anisotropic (Ford and Williams 2007). Process-based distributed numerical groundwater flow models are valuable tools for karst characterization, system identification, and hypothesis testing (e.g., Birk et al. 2006; Reimann et al. 2011; Geyer et al. 2013). Various karst numerical groundwater flow modeling approaches are available (Teutsch and Sauter 1998; Shoemaker et al. 2008). Thereof, discrete-continuum model approaches (DCM, sometimes denoted as hybrid models; e.g., Liedl et al. 2003) come with significant benefits because this method accounts for the specific hydraulic characteristics of karst conduits, such as laminar or turbulent flow in discrete structures that are hydraulically coupled to the matrix continuum, as well as for the distribution and structure of the conduit network in space (e.g., Kuniansky 2016). The Conduit Flow Process (CFP) Mode 1 for MODFLOW-2005 (Liedl et al. 2003; Shoemaker et al. 2008; Boyce et al. 2020) is an open-source discrete-continuum numerical simulation capable of modeling discrete features with nonlinear flow in a porous matrix environment, for example, karst systems, horizontal wells, and others. Besides Mode 1, CFP also allows accounting for nonlinear flow in the matrix continuum as Mode 2. The combination of Mode 1 (nonlinear flow in discrete pipes) and Mode 2 (nonlinear flow in the matrix continuum) is implemented as Mode 3.

A challenging issue when using CFP Mode 1 for karst simulation is the definition of the conduit network (see Shoemaker et al. 2008), which is built up of nodes that are connected by pipes. Based on user-defined information about the conduit nodes' spatial location, a relatively complex dataset that describes how the nodes and pipes are connected to the MODFLOW continuum cells needs to be generated before simulation, for example, by graphical user interfaces or manual construction. Generation of conduit networks can be further advanced with stochastic conduit network generation. The pseudo-generic approach of the Stochastic Karst Simulator (SKS; Borghi et al. 2016) generates hierarchical conduit networks based on a Fast Marching Algorithm (FMA, Sethian 2008). More recently, the open-source method pyKasso became available, generating stochastic karst conduit networks in an open-source Python library (pyKasso; Fandel et al. 2022). Accordingly, this network generator provides many possible conduit networks for further processing with numerical models (Borghi et al. 2016; Fandel et al. 2022). Those stochastically generated structures, however, could not yet be straightforwardly used in CFP due to the already mentioned relatively complex input data, limiting an efficient connection between the methods.

Script-based groundwater modeling is available for the MODFLOW family of numerical software with a Python package named FloPy (Bakker et al. 2016). FloPy covers the whole stack of the groundwater modeling workflow from preprocessing to running the model and postprocessing. Many applications and examples demonstrate the benefits of FloPy concerning scalable (i.e., running a large number of model variants) and comprehensive numerical groundwater modeling (FloPy 2022). Another significant benefit of script-based groundwater modeling with Python is the possibility to include many available Python libraries that can significantly improve the preprocessing and postprocessing of numerical groundwater flow models. However, FloPy does not support CFP Mode 1 and, therefore, limits the efficient use of discrete-continuum models for karst system investigation.

To overcome the current limitation, existing routines for karst conduit network input data (Reimann 2013) were transferred into a Python package named CFPy and complemented with other routines to allow efficient preprocessing and postprocessing of CFP Mode 1 groundwater flow models, also within the existing FloPy framework. Additionally, CFPy offers utilities to create stochastically generated conduit networks from pyKasso in numerical simulations with CFP. The manuscript describes the main functionality of CFPy for preprocessing and postprocessing and demonstrates the efficient use of CFPy with two examples. CFPy and Jupyter Notebooks of the examples are freely available and open-source.

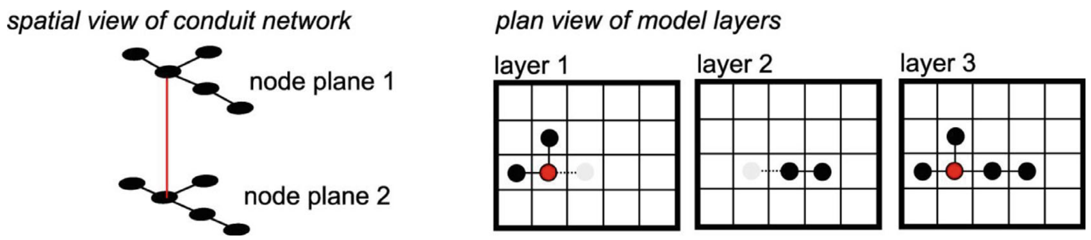
Pre- and Postprocessing of Data for the Conduit Flow Process

The core functionality of CFPy is based on an earlier routine to generate the conduit network input information, which is named CONGEN (CONduit network input GENerator; Reimann 2013). CONGEN is a tool to generate the CFP Mode 1 dataset that describes how nodes and pipes are connected to the MODFLOW continuum and how node-pipe connections are formed; see line 8 of the CFP input file (documented in Shoemaker et al. 2008, p. 27 and following). CONGEN functionality is based on a matrix with one element for each active MODFLOW cell. This CONGEN matrix considers for each node location the respective node height (elevation of the pipe center above a specific datum as a floating point number, see also Shoemaker et al. 2008, p. 28) to calculate the node network (node number, neighboring nodes, and tubes). Nodes are organized in planes where individual nodes with user-defined node heights are linked by horizontal tubes. CFPy considers a CONGEN matrix for each node plane. A proxy marks MODFLOW cells without conduit nodes (i.e., the number −999). CONGEN can generate complex conduit network input data, whereas the conduits are organized in planes, which are independent of MODFLOW layers, with pipes connecting the adjacent nodes. Vertical pipes can link individual conduit planes to form 3D conduit networks. The nodes defining the connection between different planes by vertical pipes are specifically marked by a leading letter "c" for both nodes. Figure 1 illustrates a simple conduit network with CONGEN input data and the resulting dataset to define the conduit network in CFP Mode 1.

New Python Routines and Coupling with Existing Routines (CONGEN)

The existing CONGEN routines, written in Fortran, were translated to Python code (see the CFPy.cfp.cfp submodule). While we do not show and discuss all the routines here, some general descriptions are given as follows.

To generate the dataset describing the conduit network (see Shoemaker et al. 2008), general information about the spatial discretization of the MODFLOW continuum are needed (i.e., number of rows, columns, and layers). In addition, information about the number of node planes, the MODFLOW layer elevations, and the CONGEN matrices for node locations and elevations are needed (see Figure 1). All this information is aggregated to a *.nbr-file for CFPy to process, which can be automatically performed using a preprocessing utility in CFPy. Subsequently, CFPy generates the required input for CFP Mode 1.



Example CONGEN input

```
#plane 1
-999  -999  -999  -999  -999
-999  20.1  -999  -999  -999
19.1 c20.0    19  18.9  -999
-999  -999  -999  -999  -999

#plane 2
-999  -999  -999  -999  -999
-999     5  -999  -999  -999
   5    c5     5     5  -999
 -999 -999  -999  -999  -999
```

NBR data within the output control file:

NO	MC	MR	ML	NB1	NB2	NB3	NB4	NB5	NB6	TB1	TB2	TB3	TB4	TB5	TB6
1	2	2	1	3	0	0	0	0	0	1	0	0	0	0	0
2	1	3	1	3	0	0	0	0	0	2	0	0	0	0	0
3	2	3	1	1	4	2	8	0	0	1	3	2	9	0	0
4	3	3	2	5	3	0	0	0	0	4	3	0	0	0	0
5	4	3	2	4	0	0	0	0	0	4	0	0	0	0	0
6	2	2	3	8	0	0	0	0	0	5	0	0	0	0	0
7	1	3	3	8	0	0	0	0	0	6	0	0	0	0	0
8	2	3	3	6	9	7	3	0	0	5	7	6	9	0	0
9	3	3	3	10	8	0	0	0	0	8	7	0	0	0	0
10	4	3	3	9	0	0	0	0	0	8	0	0	0	0	0

Figure 1. Simple example of a conduit network (top left), the input data structure for CFPy and CONGEN specifying node locations (bottom left), the location of the conduit network nodes in the parent MODFLOW model (top right), and the data structure used by CFP Mode 1 (bottom right).

Starting from that information, the algorithm iterates over (1) node planes, (2) columns, and (3) rows in the CONGEN matrices. For each cell at location [ROW, COL, PLANE], a corresponding MODFLOW layer is assigned based on the node elevation, and the node is given a node number. Then, for all surrounding cells, it is checked whether a neighboring node is present. If a neighboring node is present, its node number is added as a neighbor (see columns NB1 to NB6 in Figure 2) to the current node, and a pipe is added as a pipe number, connecting the active node and the neighboring node (see columns TB1 to TB6 in Figure 2). Vertical connections are generally not converted to pipes unless the user specifies a vertical connection with a leading "c" before the node elevation in both connected node planes (e.g., "c120.3", as shown in Figure 1). With this information, the matrix describing the conduit network structure is filled. Afterward, the user can parameterize the individual pipes with a corresponding diameter, wall roughness, wall permeability, tortuosity, and lower and upper critical Reynolds numbers. Further, defined head boundaries can be set for user-specified nodes. The remaining CFP Mode1 input files (*.crch for conduit recharge and *.coc for conduit output control, see the CFPy.cfp.crch and CFPy.cfp.coc submodules) are generated similarly as in FloPy.

Integration with FloPy and Other Methods (pyKasso)

Even though CFP Mode 1 readable conduit structures can be obtained from existing network structures with CFPy as described above, the coupling between stochastic conduit network generators, such as pyKasso, and CFPy was implemented as well. In a preprocessing routine (see the CFPy.utils.preprocessing submodule), such generated networks can be validated for structural consistency and can directly be transformed to a .cfp-file. This feature now enables the integration of stochastic network generation on the one hand and script-based automatable model development, on the other hand, harboring great potential for future studies. For example, CFPy allows distributed numerical simulation with large numbers of different feasible conduit networks for given site characteristics within a single framework, which was previously not supported by readily available (Python) tools.

Suppose an existing FloPy model is used, and CFP Mode1 functionality is desired: in that case, existing model information such as the number of layers, rows, columns, and layer elevations, can be extracted automatically and subsequently used in all calculations. This functionality enables the seamless integration of CFP Mode 1 to pre-existing MODFLOW models within the FloPy framework. It is noted that CFPy can only process rectangular or cuboid model domains. If the relevant model domain deviates from a rectangle or cuboid, it must be extended accordingly. After CFPy processing, the model domain can be altered by setting unwanted model cells inactive with FloPy to obtain a model domain that deviates from a rectangle or cuboid.

Description of Output Routines and Capabilities

Conduit network-specific results, such as flow rate in a pipe at a given time or head in a node, are always provided by the standard output listing

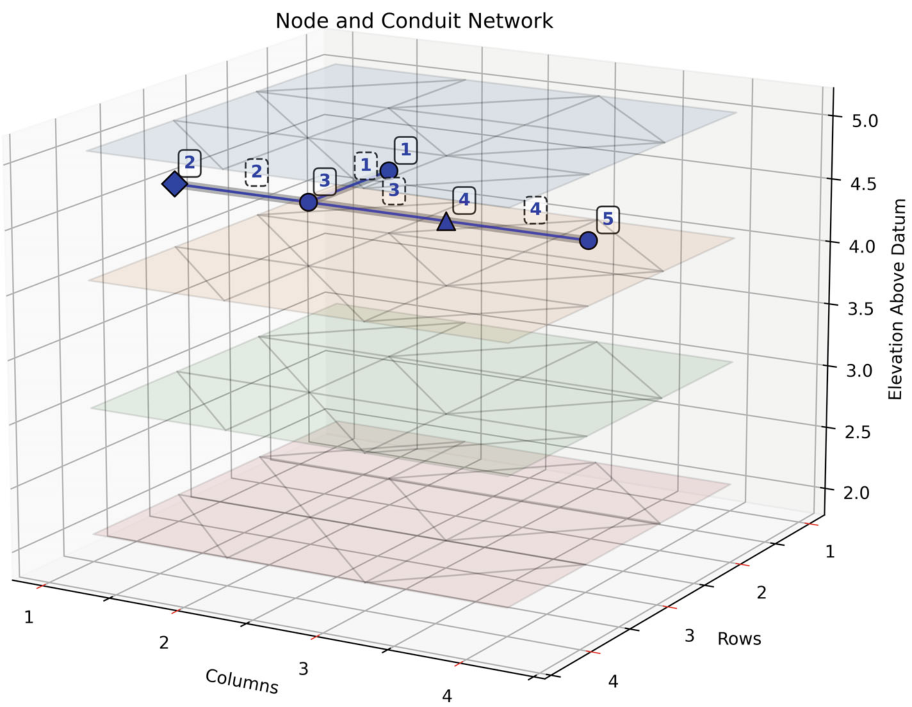


Figure 2. The conduit network of Example 1, plotted with CFPy-internal plotting utility; the colored areas represent layer boundaries that are interpolated via triangulation (gray grid lines), points represent nodes, boxes indicate node numbers, dashed boxes indicate tube numbers, elevation (z) is given in the units specified with the MODFLOW model (here, meters). Please note that CFPy uses internal zero-based indexing (Python standard) but translates input and output to the common MODFLOW indexing beginning with one. Further, CFPy marks defined water inlets with a triangle and water outlets with a diamond.

file of MODFLOW-2005 CFP. Besides that, more detailed information about the conduit hydraulics can be provided by formatted text files for each node or pipe, as specified in the CFPy.cfp.coc submodule. The CFPy.utils.postprocessing utilities can read this data for further processing, export, or custom plotting. Additionally, a projection of the three-dimensional node network can be plotted via the CFPy.utils.plots utilities, where the visualization can optionally include node and pipe numbers as well as matrix heads. Furthermore, it is possible to dynamically move the view (i.e., change elevation and azimuth angle) in the plot, enabling a more versatile way of generating visually attractive figures of three-dimensional conduit networks. The plotting utility is an additional novelty in CFP Mode 1 postprocessing and, for the first time, enables users to inspect the whole model domain dynamically and visually.

Example 1—Idealized Model from Shoemaker et al. (2008)

The first example application reflects the example from Shoemaker et al. (2008). A simple conduit network with five nodes and four tubes (similar to node plane 1 in Figure 1) is linked to a 3D matrix continuum with three layers, four rows, and four columns (see Figure 2). The model simulation is carried out for five transient stress periods with a uniform period of 100 days. The conduit network receives temporally variable and spatially uniform inflow from indirect (all cells in the uppermost MODFLOW layer), direct recharge (into node 4 with varying rates in periods 1–5:0.3, 0.1, 0.2, 0.3, and 0.2 m^3day^{-1}), and transfers water with the continuum matrix (in- or outflow according to the head difference between node and respective continuum cell). The conduit network contains a defined head boundary at node 2 that serves as a water outlet. Further details about the model can be found in Shoemaker et al. (2008). The respective Jupyter notebook of the example is open-source and available in the CFPy GitHub repository (https://github.com/iGW-TU-Dresden/CFPy).

The first step is the model design for the matrix continuum based on the FloPy functionality. General instructions are widely available, for example, in Bakker et al. (2016). The underlying MODFLOW/FloPy model contains information about the spatial extent of the model and its spatial and temporal discretization. CFPy considers this information to define the conduit network at the correct locations in the MODFLOW parent model. After creating the MODFLOW parent model, CFP-specific input files are generated with CFPy, similarly to in FloPy. The forward simulation is performed using all input files (i.e., general MODFLOW and specific CFP Mode 1 input files).

Results are presented here as time series (Figure 3). Characteristically, at node 2, the head within the matrix and the pipe is 20 m, showing that the boundary condition was respected. The heads within the conduit network are

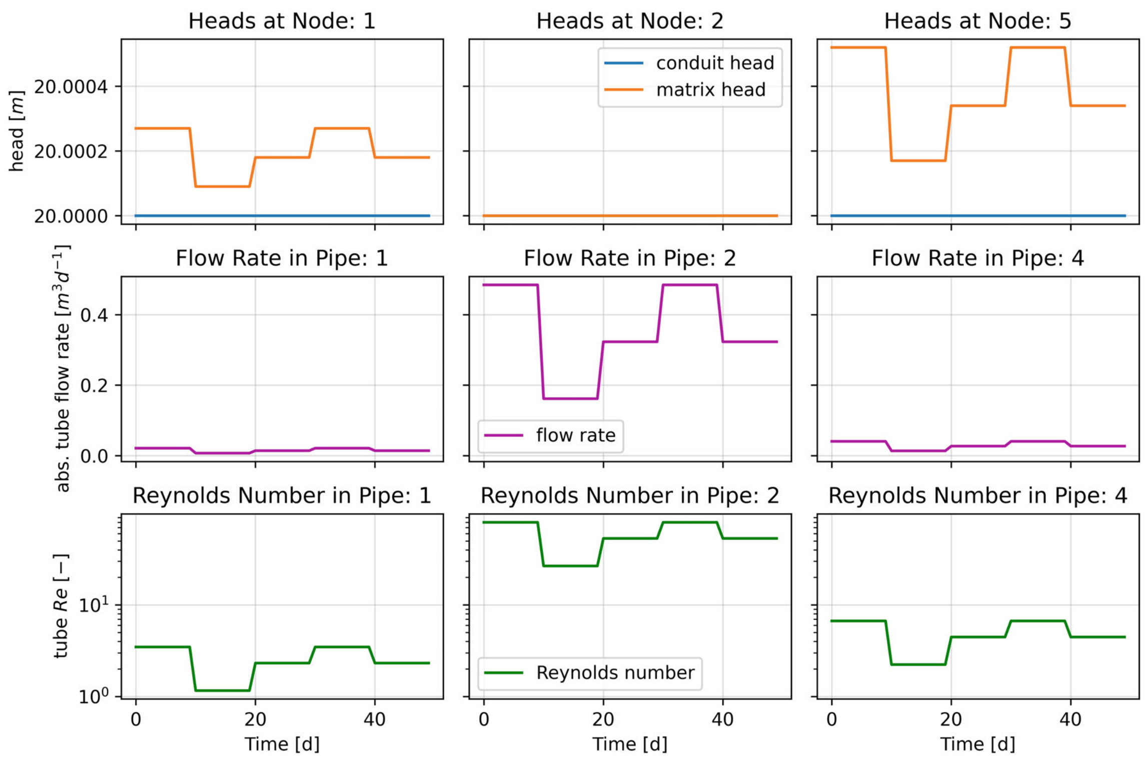


Figure 3. Results from Example 1; (top row) matrix and conduit head time series for specific nodes; (middle row) flow rate time series for specific pipes; (bottom row) time series of Reynolds numbers for specific pipes; the node and pipe numbers are equivalent to Figure 2; the spring is represented by node 2 and fed by pipe 2.

all highly influenced by the Dirichlet (specified head) boundary condition and vary only within a range of seven magnitudes smaller than the matrix heads. However, the flow rates in the conduits show characteristic dynamics, which are influenced further by direct recharge and water entering the conduits through conduit walls. It is evident that water enters the conduits through the walls as the matrix head is higher than the corresponding conduit head.

Example 2—Idealized Catchment with Conduit Network Variation

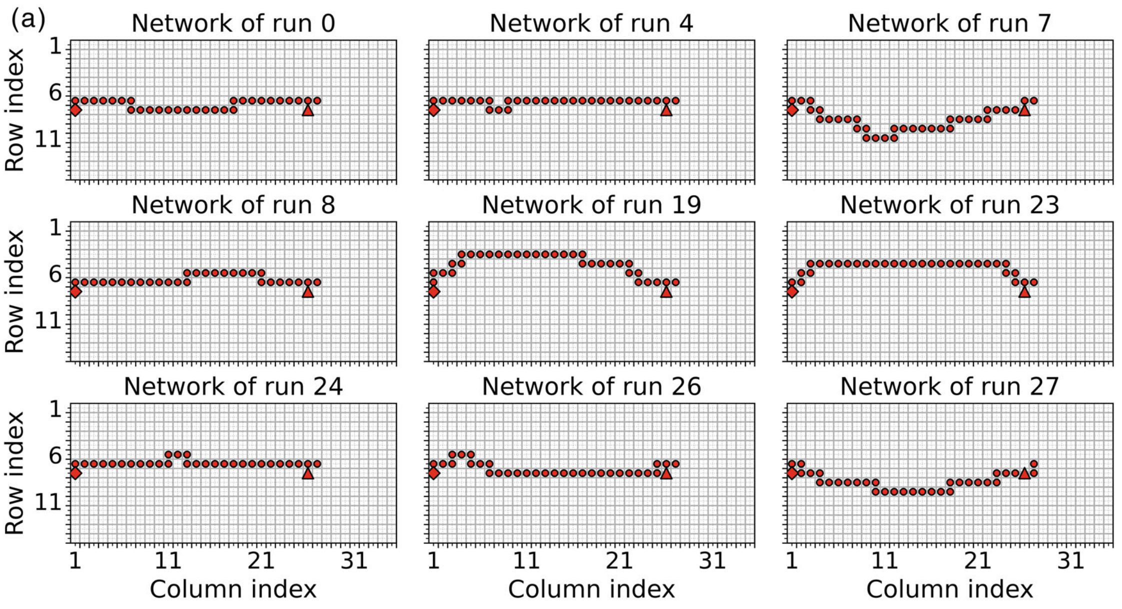
The second application example considers an idealized karst aquifer similar to a setup previously used by Birk et al. (2006). Accordingly, the model considers a rectangular catchment of 750 m × 1750 m, discretized into 15 rows and 35 columns with cell widths of 50 m each. The catchment receives uniform areal recharge of 1×10^{-8} ms^{-1} and is drained by a karst conduit, coupled to the matrix continuum by a head-dependent linear water transfer (Shoemaker et al. 2008). The karst conduit comprises several nodes connected by pipes, where nodes allow water transfer with matrix continuum cells. Contrary to Birk et al. (2006), a stochastic framework is adopted here (see below) to define the karst conduit position within the catchment. Only the location of the conduit inlet (i.e., a sinkhole) and outlet (i.e., a spring) is considered as available information about the underground karst structure.

Table 1
Model parameters for the example application 2

Parameter	Value	Unit
Pipe diameter	0.5	m
Pipe roughness	0.05	m
Pipe elevation	20	m (above datum)
Water transfer coefficient (pipe conductance)	0.001	m^{2}s^{-1}
Matrix hydraulic conductivity	1e-5	ms^{-1}
Specified head boundary (matrix and conduit)	50	m NHN

The spring outlet of the karst conduit is represented by a first-type boundary condition (specified head). Further information about model parameters is provided in Table 1. As an initial step in numerical model development, a MODFLOW continuum model is designed using FloPy. Subsequently, necessary information about the discrete karst conduits needs to be added (see Shoemaker et al. 2008, p. 27 ff), as in Example 1.

Depending on the study site, the spatial distribution of karst conduits can be assumed based on expert knowledge. Still, it can also be derived in an automatized way with stochastic network generators, for example, SKS (Borghi et al. 2016) and pyKasso (Fandel et al. 2021). These network generators typically connect inlets and



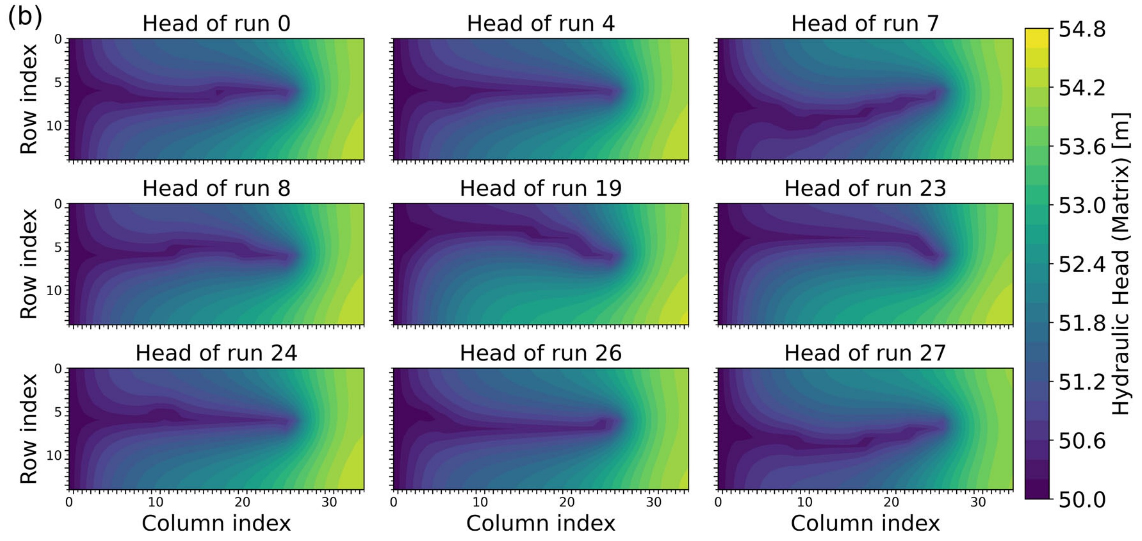


Figure 4. Computed conduit networks (above) and hydraulic heads in the matrix continuum for nine randomly selected model variants. CFPy marks defined water inlets with a triangle and water outlets with a diamond.

outlets of a karst system (i.e., sinkholes and springs) with stochastically generated karst structures influenced by fractures, which can also be generated stochastically. PyKasso is an open-source stochastic network generator for Python and can, therefore, be easily integrated into the CFPy workflow to generate different karst conduit structures stochastically.

This methodology of generating a number of different conduit networks and simulating them with CFP Mode 1 is followed here by combining the capabilities of FloPy, pyKasso, and CFPy. A single outlet is defined for pyKasso at the right domain boundary in the eighth row (see Figure 4). A single inlet is furthermore located in the eighth row and 26th column of the model domain. Fractures are generated randomly with densities of 0.00005 and 0.0001 for each fracture family, with minimum orientation angles of 340° and 70° and maximum orientation angles of 20° and 110°, respectively. Based on this information, PyKasso can generate an arbitrary number of karst conduit networks for further consideration in CFP Mode1.

The demonstration example considers a user-defined number of 100 model variants by a computational loop. Accordingly, the information on the stochastically generated spatial distribution of the individual karst conduit networks is complemented with the required hydraulic parameters like pipe diameter, conduit-matrix conductance, roughness height, and so on. These parameters are equal for all of the model variants. Subsequently, the numerical model is solved, and results are saved for further processing.

Figure 4 provides an impression of the various stochastically generated karst conduit networks and the corresponding results for hydraulic heads. Figure 5 presents the standard deviation of matrix heads and the mean matrix head as an example of CFPy postprocessing

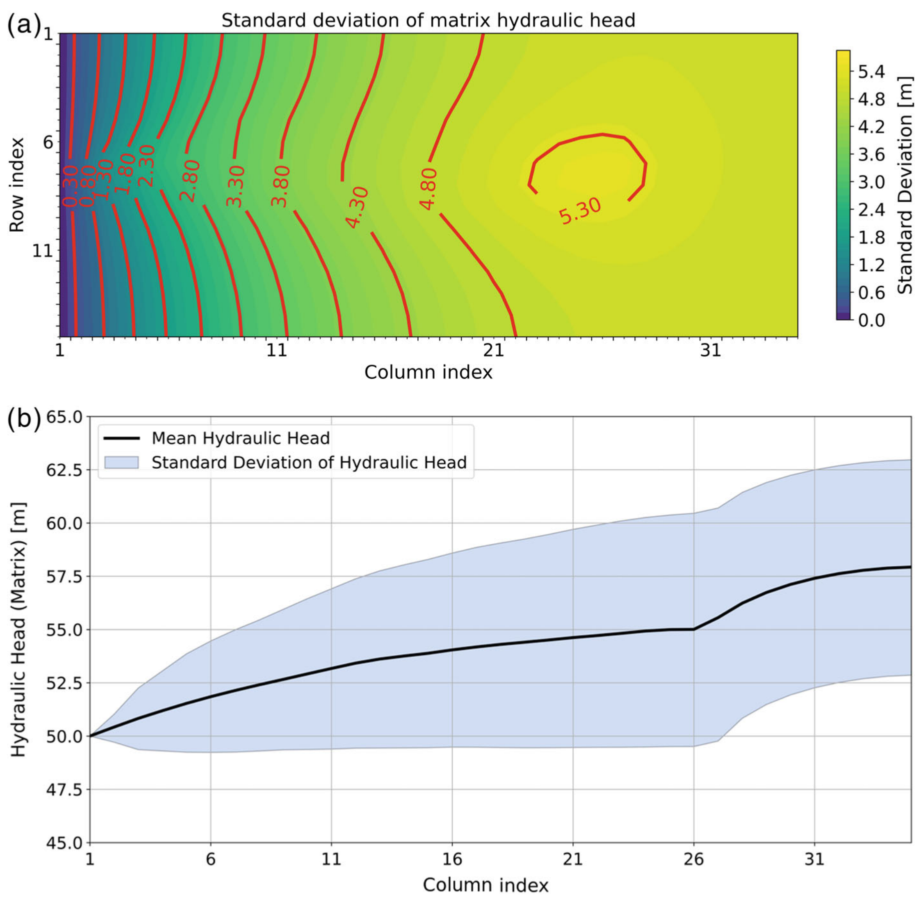


Figure 5. (a) Standard deviation of matrix hydraulic heads from all model variants and (b) mean matrix hydraulic head along columns.

capabilities (more results like the variation of spring discharge is provided in the example Jupyter notebook on the CFPy GitHub repository https://github.com/iGW-TU-Dresden/CFPy). In conclusion, Example 2 demonstrates the ability of CFPy to couple with other existing Python-based routines for advanced karst system modeling and demonstrates the significantly widened use of CFP Mode1 for distributed numerical modeling.

Dependencies

The CFPy module relies on several open-source Python libraries that are readily available and can easily be downloaded from the respective GitHub repositories or installed using common distribution software (e.g., Anaconda Distribution or pip). It has to be noted that copies of these libraries are not included in the CFPy package itself and have to be installed and imported by the user. Therefore, we refer the reader to the GitHub repository of CFPy (https://github.com/iGW-TU-Dresden/CFPy) for the corresponding documentation.

Conclusion and Outlook

CFPy is an open-source script-based approach to generate, run, and process distributed numerical groundwater models for the MODFLOW-2005 Conduit Flow Process (CFP, Shoemaker et al. 2008). CFPy may be used, for example, to simulate groundwater flow in karst systems in a reproducible and script-based framework. CFPy allows for an automated and scalable workflow for the preprocessing and postprocessing of MODFLOW-2005 CFP Mode 1 models. In addition, the script-based approach allows using a wide range of existing Python routines like FloPy (Bakker et al. 2016) for preprocessing and postprocessing of the MODFLOW continuum, or pyKasso (Fandel et al. 2021), which can stochastically generate spatially distributed karst conduit networks. Further, general Python functionality, for example, for data handling, structured model processing (e.g., computational loops), visualization, and more can be easily included. With this, CFPy opens the distributed numerical model CFP for many innovative applications, for example, multi-model investigations in combination with automatic calibration (e.g., Borghi et al. 2016).

Current limitations of CFPy comprise missing functionality for adaptive spatial grid optimization for the MODFLOW continuum. Further, current networks are built up from nodes, which are centrally placed in cells, and are further connected by vertical or horizontal pipes but without the option for diagonal connections. Because the stochastic network generation is adopted from pyKasso, CFPy is currently limited to work for one- and two-dimensional model domains. However, the flexibility of the open-source and Python-based approach potentially allows for overcoming these limitations with manageable effort. CFPy is open-source and available under https://github.com/iGW-TU-Dresden/CFPy.

Acknowledgments

This work was supported by the Deutsche Forschungsgemeinschaft, DFG for the iKarst project (Grant Numbers: LI727/31-1, RE4001/2-1, SA501/28-1, and KO5359/3-1) and by the German Federal Ministry of Education and Research (BMBF) for the project MEWAC-FEMAR. The authors highly appreciate comments and suggestions on that manuscript from Chloe Fandel and W. Barclay Shoemaker. The authors would also like to thank the reviewers and editors for their thoughtful and constructive comments. Open Access funding enabled and organized by Projekt DEAL.

Supporting Information

Additional supporting information may be found online in the Supporting Information section at the end of the article. The supporting information lists dependencies respectively packages that were used for CFPy and the example notebooks. Further information is also available on GitHub (https://github.com/iGW-TU-Dresden/CFPy) and within the example notebooks. There, users will find a watermark that precisely states which package and Python versions were used.

Disclaimer

Supporting Information is generally not peer-reviewed.

Supporting Information

Additional supporting information may be found online in the Supporting Information section at the end of the article. Supporting Information is generally *not* peer reviewed.

Data S1. Supporting Information

References

Bakker, M., V. Post, C.D. Langevin, J.D. Hughes, J.T. White, J.J. Starn, and M.N. Fienen. 2016. Scripting MODFLOW model development using Python and FloPy. *Groundwater* 54, no. 5: 733–739.

Birk, S., R. Liedl, and M. Sauter. 2006. Karst spring responses examined by process-based modeling. *Groundwater* 44, no. 6: 832–836.

Borghi, A., P. Renard, and F. Cornaton. 2016. Can one identify karst conduit networks geometry and properties from hydraulic and tracer test data? *Advances in Water Resources* 90: 99–115.

Boyce, S.E., R.T. Hanson, I. Ferguson, W. Schmid, W. Henson, T. Reimann, S.M. Mehl, and M.M. Earll. 2020. One-water hydrologic flow model: A MODFLOW based conjunctive-use simulation software: U.S. Geological Survey. *Techniques and Methods* 6–A60: 435. https://doi.org/10.3133/tm6A60

Fandel, C., T. Ferré, Z. Chen, P. Renard, and N. Goldscheider. 2021. A model ensemble generator to explore structural uncertainty in karst systems with unmapped conduits. *Hydrogeology Journal* 29, no. 1: 229–248.

Fandel, C., F. Miville, T. Ferré, N. Goldscheider, and P. Renard. 2022. The stochastic simulation of karst conduit network structure using anisotropic fast marching, and its application to a geologically complex alpine karst system. *Hydrogeology Journal* 30, no. 3: 927–946.

FloPy. 2022. FloPy—A Python package to create, run, and post-process MODFLOW-based models. https://github.com/modflowpy/flopy (accessed November 4, 2022).

Ford, D., and P.D. Williams. 2007. *Karst Hydrogeology and Geomorphology*. Hoboken, NJ: John Wiley & Sons.

Geyer, T., S. Birk, T. Reimann, N. Dörfliger, and M. Sauter. 2013. Differentiated characterization of karst aquifers: Some contributions. *Carbonates and Evaporites* 28, no. 1: 41–46.

Kuniansky, E.L. 2016. Simulating groundwater flow in karst aquifers with distributed parameter models—Comparison of porous-equivalent media and hybrid flow approaches. U.S. Geological Survey Scientific Investigations Report 2016-5116, https://doi.org/10.3133/sir20165116.

Liedl, R., M. Sauter, D. Hückinghaus, T. Clemens, and G. Teutsch. 2003. Simulation of the development of karst aquifers using a coupled continuum pipe flow model. *Water Resources Research* 39, no. 3: 1057.

Reimann, T. 2013. CONGEN: CONduit network input GENerator. Internal documentation, Institute for Groundwater Management, TU Dresden. https://tu-dresden.de/bu/umwelt/hydro/igw/forschung/downloads/congen (accessed November 4, 2022)

Reimann, T., T. Geyer, W.B. Shoemaker, R. Liedl, and M. Sauter. 2011. Effects of dynamically variable saturation and matrix-conduit coupling of flow in karst aquifers. *Water Resources Research* 47, no. 11: W11503.

Sethian, J.A. 2008. *Level Set Methods and Fast Marching Methods Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science (re-Edition)*. Cambridge, UK: Cambridge University Press.

Shoemaker, W.B., E.L. Kuniansky, S. Birk, S. Bauer, and E.D. Swain. 2008. Documentation of a conduit flow process (CFP) for MODFLOW-2005: U.S. Geological Survey. *Techniques and Methods* 6-A24: 50. https://doi.org/10.3133/tm6A24

Stevanović, Z. 2019. Karst waters in potable water supply: A global scale overview. *Environmental Earth Science* 78: 662. https://doi.org/10.1007/s12665-019-8670-9

Teutsch, G., and M. Sauter. 1998. Distributed parameter modelling approaches in karst-hydrological investigations. *Bulletin Hydrogeologie* 16: 99–109.