https://doi.org/10.5194/egusphere-2026-5708
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

KrigeKit v0.3.2: a parallel Python–Fortran engine for variogram analysis, kriging, and geostatistical simulation

Gengxin Ou; Vivek Bedekar; Matt Tonkin; Gilbert Barth; Marinko Karanovic

S.S. Papadopulos & Associates, Inc.
1801 Rockville Pike, Suite 220, Rockville, MD 20852

Corresponding author: Gengxin Ou — mou@sspa.com

Abstract. Kriging and geostatistical simulation underpin environmental modeling, yet practitioners must choose between aging file-driven Fortran codes such as GSLIB and convenient but slower high-level packages. We present KrigeKit, an open-source (MIT) library pairing an OpenMP-parallel Fortran engine with a dependency-light Python layer. A single engine provides ordinary, simple, and co-kriging, kriging with external drift, sequential Gaussian and indicator simulation, and space-time kriging. Its hallmark is the breadth of advanced capabilities it exposes together — multiplicative covariance structures, gradient constraints, spatially varying anisotropy, and positive-semidefinite co-kriging fitting — seldom available in one package. Performance comes from parallel block estimation, tabular variogram evaluation, and a per-thread Cholesky cache that cuts the per-block cost from $O(n^3)$ to $O(n^2)$ on repeated neighborhoods. We verify KrigeKit against analytic results and established packages, characterize its parallel scaling, and demonstrate it on three environmental case studies. KrigeKit ships as pre-built wheels on PyPI for Linux, macOS, and Windows.

Keywords: Geostatistics; Kriging; Sequential Gaussian simulation; Space-time interpolation; Groundwater mapping; Aquifer characterization

1. Introduction

Spatial interpolation and stochastic simulation are routine operations in environmental modeling: mapping contaminant concentrations, interpolating groundwater levels, characterizing subsurface heterogeneity, and generating conditional realizations for uncertainty propagation. Kriging — the family of best linear unbiased predictors built on a fitted variogram — remains the standard tool for these tasks (Cressie, 1993; Goovaerts, 1997; Chilès and Delfiner, 2012).

The software landscape, however, presents a recurring trade-off between speed and convenience. The long-established Fortran code GSLIB (Deutsch and Journel, 1998) remains the canonical, feature-rich reference, but it is command-line and file-driven, and its serial design is slow on the large grids common today. The R package gstat (Pebesma, 2004; Gräler et al.,

https://doi.org/10.5194/egusphere-2026-5708
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

2016) is widely used but tied to the R ecosystem. Modern Python packages — PyKrige (Murphy et al., 2025), GSTools (Müller et al., 2022), and scikit-gstat — integrate well with the scientific-Python stack, but their most demanding steps (neighborhood search, repeated dense linear solves, simulation paths) run in interpreted or JIT-compiled code and become a bottleneck on the large three-dimensional and space-time grids that environmental work increasingly requires. Sect. 7.3 compares the feature coverage of these packages directly.

We introduce KrigeKit to occupy the middle ground: the usability of a Python API with the throughput of a compiled, parallel backend. Its principal contribution is a unified computational architecture that exposes a broad range of advanced geostatistical methods — among them multiplicative (product) covariance structures (Sect. 2.8) — through a single high-performance solver and a consistent Python interface, an approach rarely taken in open-source geostatistics. Concretely, its contributions are:

- A single Fortran engine that unifies spatial and space-time kriging behind one solve framework, exposing ordinary and simple kriging, co-kriging, kriging with external drift (KED), sequential Gaussian and indicator simulation (SGSIM, SIS), and spatially varying anisotropy (SVA), with block support throughout.
- Multiplicative (product) covariance structures that compose valid covariances — for example an exponential decay multiplied by a cyclic hole-effect term — to represent decay combined with periodicity, with independent ranges per factor.
- A performance design combining OpenMP parallelism over the estimation grid, precomputed tabular variogram evaluation, and a per-thread cache of Cholesky factorizations that reuses identical neighborhood solves, with optional storage and replay of kriging weights for fast re-estimation when only data values change.
- A thin ctypes Python interface with a small dependency footprint and no run-time compilation, distributed as pre-built wheels for Linux, macOS, and Windows; the same C-binding API lets the engine be driven from other languages (Sect. 3.4).
- Verification against analytic and published results, parallel-scaling benchmarks, and three environmental case studies.

The remainder of the paper presents the geostatistical methods (Sect. 2), the software architecture and performance design (Sect. 3), verification and validation (Sect. 4), performance (Sect. 5), the environmental case studies (Sect. 6), a discussion of usability, reliability, and limitations (Sect. 7), and conclusions (Sect. 8).

2. Geostatistical methods

What sets KrigeKit apart is the breadth of methods it exposes through one solver, so this section is organized to make that breadth visible while keeping the standard material brief. The established estimators — ordinary and simple kriging with block support (Sect. 2.1), co-kriging (Sect. 2.2), and kriging with external drift (Sect. 2.3) — are stated only in the compact form needed to fix notation and to show that a single linear system serves them all; their derivations are standard (Goovaerts,

https://doi.org/10.5194/egusphere-2026-5708
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

1997; Chilès and Delfiner, 2012). The emphasis falls instead on the capabilities that are distinctive to KrigeKit and seldom found together in one package: gradient constraints (Sect. 2.3), spatially varying anisotropy (Sect. 2.7), and multiplicative covariance structures (Sect. 2.8), supported by the positive-semidefinite co-kriging fitting introduced in Sect. 2.2.

2.1 The unified kriging system

Every kriging method in KrigeKit is a special case of one linear system; the methods differ only in how that system is populated, not in how it is solved. For P cross-correlated variables, the estimate of variable i at a target $\mathbf{s}_0$ (a point or a block) is the linear combination

$$Z_i^*(\mathbf{s}_0) = \sum_{j=1}^{P} \sum_{\alpha=1}^{n_j} \lambda_\alpha^{(j)}\, Z_j(\mathbf{s}_\alpha) \tag{1}$$

of the n_j data available on each variable j (Goovaerts, 1997; Kitanidis, 1997; Wackernagel, 2003). The weights are obtained, together with a set of Lagrange multipliers μ, from a single symmetric block system,

$$\sum_{j=1}^{P} \sum_{\beta=1}^{n_j} \lambda_\beta^{(j)}\, C_{kj}(\mathbf{s}_\alpha - \mathbf{s}_\beta) + \sum_{l=0}^{L} \mu_l^{(k)}\, f_l(\mathbf{s}_\alpha) = \bar{C}_{ki}(\mathbf{s}_\alpha, V_0), \qquad \sum_{\beta=1}^{n_k} \lambda_\beta^{(k)}\, f_l(\mathbf{s}_\beta) = \delta_{ik}\, f_l(\mathbf{s}_0) \tag{2}$$

stated here for every datum (k, α) alongside the unbiasedness conditions on the right. The system has the partitioned form of a covariance block $\mathbf{C}$ — the auto- and cross-covariances C_{kj} among all data of all variables — bordered by a constraint block $\mathbf{F}$ that carries the unbiasedness and external-drift functions f_l; its right-hand side pairs the data-to-target covariances $\bar{C}_{ki}$ with the drift values $f_l(\mathbf{s}_0)$ at the target. KrigeKit assembles exactly this matrix and factorizes it once per neighborhood (Sect. 3). The associated kriging variance is

$$\sigma_i^2(\mathbf{s}_0) = \bar{C}_{ii}(V_0, V_0) - \sum_{j=1}^{P} \sum_{\alpha=1}^{n_j} \lambda_\alpha^{(j)}\, \bar{C}_{ij}(\mathbf{s}_\alpha, V_0) - \sum_{l=0}^{L} \mu_l^{(i)}\, f_l(\mathbf{s}_0) \tag{3}$$

Every target is treated as a block V_0, so the right-hand-side covariance is the point-to-block average

$$\bar{C}_{ij}(\mathbf{s}_\alpha, V) = \frac{1}{N} \sum_{m=1}^{N} C_{ij}\,(\mathbf{s}_\alpha - \mathbf{s}_m), \quad \mathbf{s}_m \in V \tag{4}$$

obtained by discretizing V_0 into N points $\mathbf{s}_m$; point kriging is simply the single-point case ($N = 1$) and uses the same code path. What changes between methods is only how $\mathbf{C}$ and $\mathbf{F}$ are filled:

- Simple kriging assumes a known mean, so the constraint block $\mathbf{F}$ is empty and the system reduces to $\mathbf{C}\,\boldsymbol{\lambda} = \mathbf{c}_0$.
- Ordinary kriging adds a single unit-sum constraint per variable to $\mathbf{F}$, making the predictor unbiased without assuming the mean.
- Co-kriging takes $P > 1$ correlated variables; the cross-covariances filling $\mathbf{C}$ come from the Linear Model of Coregionalization (Sect. 2.2), and the unbiasedness rows of $\mathbf{F}$ follow either the standard per-variable convention or the single-constraint Isaaks–Srivastava form, both selectable at solve time.

https://doi.org/10.5194/egusphere-2026-5708
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

- Kriging with external drift augments **F** with known trend functions to model a non-stationary mean; gradient observations augment **C** (Sect. 2.3).

2.2 Co-kriging and the linear model of coregionalization

Co-kriging — the $P > 1$ case of Sect. 2.1 — requires a valid joint model of the auto- and cross-covariances that fill the covariance block **C**. To keep **C** positive semi-definite for any configuration of data, KrigeKit models the variables as a Linear Model of Coregionalization (LMC; Wackernagel, 2003): every auto- and cross-covariance shares a common set of S basic structures ρ_u with matrix-valued sills,

$$C_{ij}(h) = \sum_{u=1}^{S} b_{ij}^{u}\ \rho_u(h) \tag{5}$$

where each coregionalization matrix $B_u = [b_{ij}^u]$ is symmetric and positive semi-definite, which ensures a valid multivariate model. KrigeKit can build this model either by fitting all auto- and cross-variograms jointly under a positive-semidefinite constraint or, for a sparsely sampled primary with a dense secondary covariate, through a Markov model (Sect. 2.9).

2.3 External drift and gradient constraints

Kriging with external drift (KED), a form of universal kriging, models a non-stationary mean of each variable as a linear function of L known drift variables,

$$Z_i(\mathbf{s}) = m_i(\mathbf{s}) + R_i(\mathbf{s}), \qquad m_i(\mathbf{s}) = \sum_{l=0}^{L} a_l^{(i)}\ f_l(\mathbf{s}) \tag{6}$$

where $f_0 \equiv 1$ and $f_1, \ldots, f_L$ are the drift (external-trend) functions evaluated at every data and prediction location, and R_i is a zero-mean residual. The drift coefficients are not fitted explicitly; instead the functions f_l become the columns of the constraint block **F**, and the unbiasedness conditions of Eq. (2) force the weights to reproduce the local drift. The residual covariance is inferred from the variogram of the residuals after removing the drift, so each estimate is conditioned on the drift value supplied at every prediction node.

The same engine also admits gradient (derivative) observations (Delhomme, 1979), but these enter through the covariance block **C** rather than through **F**. A directional gradient is represented internally as a finite-difference pair of nearby locations: its covariance with every other datum is formed as the corresponding difference of point covariances — a difference quotient approximating the derivative — and an optional measurement variance is added to its diagonal entry, ranging from an exact constraint to a soft one. Because the gradient is just another row and column of **C**, no extra term is needed in the system. A common environmental use is a no-flow (zero normal-gradient) boundary, imposed as a gradient pseudo-datum with value zero, so that interpolated heads honor an impermeable boundary without padding it with a dense ring of artificial data.

https://doi.org/10.5194/egusphere-2026-5708
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

2.4 Sequential Gaussian simulation

Sequential Gaussian simulation (SGSIM) generates conditional realizations under the multiGaussian model. Visiting every grid node once along a random path, simple kriging against the original data and all previously simulated nodes yields a conditional mean $Z^*_{SK}(\mathbf{s}_0)$ and standard deviation $\sigma_{SK}(\mathbf{s}_0)$, from which the node value is drawn,

$$Z^{(r)}(\mathbf{s}_0) = Z^*_{SK}(\mathbf{s}_0) + \sigma_{SK}(\mathbf{s}_0)\, w, \qquad w \sim N(0,1) \tag{7}$$

with w an independent standard-normal deviate. Each simulated value is added to the conditioning set before the next node is visited, so the realizations honor the data and reproduce the covariance model; different random paths (controlled by the seed) give independent realizations. A single realization must be built sequentially along its path; all realizations requested are produced together within one solve call, but that call is single-threaded — the engine disables OpenMP whenever simulation is requested, so unlike kriging, simulation does not currently make use of multiple cores.

For strongly non-Gaussian variables — concentrations, conductivities, and other quantities spanning orders of magnitude — KrigeKit provides two marginal transforms built from a weighted empirical CDF of the data: a normal-score (Gaussian anamorphosis) transform and a uniform quantile transform. The variogram is fitted and the solve performed in score space, and results are mapped back to data units with configurable tail extrapolation. Simulated realizations are back-transformed directly through the quantile table; for kriging, the score-space estimate and variance define a conditional distribution integrated back to data-unit moments numerically, by Gauss–Hermite quadrature applied through the inverse transformation. Like all numerical work, the transforms are implemented in the Fortran engine behind the C API, so every client language obtains the same bit-reproducible result (Sect. 3.1).

2.5 Space-time covariance models

Space–time kriging treats time as an additional coordinate of a random field $Z(\mathbf{s}, t)$. For a pair of points the spatial lag is $h = \| \mathbf{s}_i - \mathbf{s}_j \|$ and the temporal lag is $u = |t_i - t_j|$, with C_s and C_t the purely spatial and temporal covariances. KrigeKit provides two non-separable space–time families (Kyriakidis and Journel, 1999; De Iaco et al., 2001). The product–sum model superposes the product of a spatial and a temporal covariance on their sum,

$$C_{ST}(h,u) = k_1\, C_s(h)\, C_t(u) + k_2\, C_s(h) + k_3\, C_t(u) \tag{8}$$

with $k_1 > 0$ and $k_2, k_3 \geq 0$ guaranteeing a valid covariance. The sum–metric model instead adds a joint structure on a metric distance that rescales time onto a spatial-equivalent scale,

$$C_{ST}(h,u) = C_s(h) + C_t(u) + C_{st}\left(\sqrt{h^2 + (\kappa u)^2}\right) \tag{9}$$

where κ is the space–time anisotropy ratio. Neighbor search uses the same rescaling — time is mapped to κu — so candidates are ordered by a single space–time distance,

$$d_{ST}(h,u) = \sqrt{h^2 + (\kappa u)^2} \tag{10}$$

evaluated in the $(d+1)$-dimensional search tree (Sect. 3.2).

https://doi.org/10.5194/egusphere-2026-5708
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

2.6 Multiple-indicator kriging and simulation

A categorical variable $Z(\mathbf{s})$ taking one of K classes is recoded as K binary indicators (Journel, 1983; Goovaerts, 1997),

$$I_k(\mathbf{s}) = 1 \text{ if } Z(\mathbf{s}) = c_k, \text{ else } 0, \quad k = 1, \dots, K \tag{11}$$

each with expectation equal to the class proportion p_k and indicator cross-covariances $C_{kl}(h)$. Indicator co-kriging estimates the conditional probability that $\mathbf{s}_0$ belongs to class k as a linear combination of the surrounding indicators,

$$p_k^*(\mathbf{s}_0) = \sum_{\alpha=1}^{n} \sum_{l=1}^{K} \lambda_{\alpha l}^{(k)} \, I_l(\mathbf{s}_\alpha) \tag{12}$$

where the inner sum couples all K indicators and collapses to a single term for single-variable indicator kriging. For ordinary indicator kriging the class-k weights solve

$$\sum_{\beta=1}^{n} \lambda_\beta^{(k)} \, C_{kk}(\mathbf{s}_\alpha - \mathbf{s}_\beta) + \mu_k = C_{kk}(\mathbf{s}_\alpha - \mathbf{s}_0), \quad \sum_{\beta=1}^{n} \lambda_\beta^{(k)} = 1 \tag{13}$$

with μ_k the unit-sum Lagrange multiplier. The K indicators are estimated jointly as one multivariate system. Because kriging is unconstrained, the raw probabilities can fall outside $[0,1]$ and need not sum to one, so KrigeKit projects them onto the probability simplex,

$$\tilde{p}_k(\mathbf{s}_0) = \frac{\max\{p_k^*(\mathbf{s}_0),\, 0\}}{\sum_{l=1}^{K} \max\{p_l^*(\mathbf{s}_0),\, 0\}} \tag{14}$$

yielding non-negative probabilities that sum to one. In sequential indicator simulation the node is assigned a class by inverting the resulting distribution with a uniform draw $U \sim U(0,1)$ — class m is selected when

$$\sum_{k<m} \tilde{p}_k(\mathbf{s}_0) \;\le\; U \;<\; \sum_{k \le m} \tilde{p}_k(\mathbf{s}_0) \tag{15}$$

— and the simulated indicator joins the conditioning set before the next node is visited.

The three cross-variogram strategies differ only in how the sills are assigned: the proportional strategy uses the indicator variance with LMC-consistent cross-sills,

$$s_{kk} = p_k(1 - p_k), \qquad s_{kl} = \sqrt{s_{kk}\, s_{ll}} \quad (k \neq l) \tag{16}$$

the shared strategy applies one sill to all K^2 pairs, and the independent strategy sets $s_{kl} = 0$ for $k \neq l$. A continuous secondary variable may also be supplied as a collocated co-kriging covariate: it enters through its covariance with the indicators — conditioning the simulated probabilities — while being excluded from the category draw, letting soft data such as geophysical surveys constrain the simulation.

https://doi.org/10.5194/egusphere-2026-5708
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

2.7 Spatially varying anisotropy

Anisotropy enters the covariance through an affine standardization of the lag vector. Collecting the ranges along the principal axes in the diagonal matrix $\mathbf{D} = \mathrm{diag}(a_{major}, a_{minor1}, a_{minor2})$ and letting $\mathbf{R}(\theta)$ be the rotation that aligns those axes with the coordinate frame, the standardized (isotropic-equivalent) lag between two points $\mathbf{s}_i$ and $\mathbf{s}_j$ is

$$\tilde{h}(\mathbf{x}) = \left\| \mathbf{D}_{\mathbf{x}}^{-1}\, \mathbf{R}(\theta_{\mathbf{x}})\, (\mathbf{s}_i - \mathbf{s}_j) \right\| \tag{17}$$

and the covariance for that location is then $C_0\, \rho$ evaluated at this standardized lag, with ρ the unit-sill correlation function and C_0 the sill. The rotation follows the GSLIB convention (Deutsch and Journel, 1998), composed from the azimuth a, dip ψ and plunge φ,

$$\mathbf{R}(\theta) = \mathbf{R}_z(\varphi)\, \mathbf{R}_x(\psi)\, \mathbf{R}_z(a) \tag{18}$$

In spatially-varying-anisotropy mode the orientation $\theta_{\mathbf{x}}$ and the ranges $\mathbf{D}_{\mathbf{x}}$ both depend on the estimation location $\mathbf{x}$, so each block carries its own transform $\mathbf{M}_{\mathbf{x}} = \mathbf{D}_{\mathbf{x}}^{-1}\, \mathbf{R}(\theta_{\mathbf{x}})$ and the variogram ellipse rotates and stretches across the grid. Two lightweight per-node modifiers are also available: a range-scaling factor $r(\mathbf{x})$ that divides the standardized lag, and a local nugget $\eta(\mathbf{x})$ added to the diagonal of the kriging system at the estimation point, which raises the kriging variance to reflect local data density or measurement error.

This is a locally stationary (moving-window) construction: each estimate uses a single stationary anisotropic covariance evaluated at the target location, in the applied-geostatistics tradition of locally varying anisotropy (Deutsch and Journel, 1998; Schilling et al., 2022) in which the orientation field is prescribed from a conceptual model. It suits a known directional structure — such as channels mapped from imagery — and integrates directly with the (co)kriging engine, but it does not constitute a single globally positive-definite nonstationary covariance, because the two points' local anisotropies are not combined symmetrically. Where a coherent, provably valid nonstationary covariance is required — typically with the anisotropy field estimated rather than prescribed — the kernel-convolution construction of Paciorek and Schervish (2006), implemented for kriging by Risser and Calder (2017), is the more rigorous alternative. KrigeKit favors the prescribed-field, locally stationary route for its speed, interpretability, and fit to conceptual channel-following workflows.

2.8 Multiplicative (product) covariance structures

Composite covariance models in geostatistics are almost always built by additive nesting — summing several elementary structures. KrigeKit additionally allows structures to be multiplied. A model is expressed as a sum of product groups,

$$C(h) = \sum_g \prod_{m \in g} C_m\,(h),$$

where each group g is the product of one or more component covariances C_m; ordinary nested models are recovered when every group contains a single component. The validity argument is matrix-level and rests on the Schur product theorem (Horn and Johnson, 2013): for any fixed set of locations, the Hadamard (entrywise) product of positive-semidefinite matrices is itself positive-semidefinite. Hence, whenever every factor's covariance matrix over the sampled and target locations is positive-semidefinite, the assembled product covariance matrix is too, and therefore defines a valid covariance on those

https://doi.org/10.5194/egusphere-2026-5708
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

locations. A unique solve additionally requires a nonsingular kriging system. The positive-semidefiniteness guarantee is independent of spatial dimension.

Whether the corresponding continuum model is itself a valid covariance function depends on each factor being positive-definite in the working dimension. Three cases should be distinguished. (i) The exponential and Gaussian factors are positive-definite in arbitrary Euclidean dimension, while the standard spherical factor is positive-definite only in the one-, two-, and three-dimensional settings targeted by KrigeKit; products composed solely of these factors are therefore valid on the supported spatial domains. (ii) Oscillatory factors — the hole-effect (cosine), damped cosine, and periodic kernels — are positive-definite as functions of a one-dimensional lag; the intended and tested use is therefore to multiply such a cyclic factor onto a monotone envelope along a single axis, most commonly the temporal axis, as in the seasonal-on-trend space-time model of Sect. 6.1 (cf. De Iaco et al., 2001). (iii) For an oscillatory factor acting over a genuinely multi-dimensional lag, positive-definiteness in the continuum is not guaranteed in general — it depends on the damping-to-frequency ratio and the dimension — and KrigeKit does not certify it. An indefinite system is not automatically rejected, so such a model can still be solved but may yield negative kriging variances; confirming validity (for example empirically, by checking that the assembled matrices are positive-semidefinite and the kriging variances stay non-negative) rests with the user when oscillatory structures are combined outside their one-dimensional setting. In all cases a valid covariance need not be non-negative — the negative lobes of a cyclic product represent anticorrelation at periodic lags.

Few mainstream kriging packages expose product covariances as a first-class feature (Sect. 7.3, Table 5), yet they are valuable whenever a process combines decay with periodicity. A typical example is a long-range exponential decay multiplied by a cyclic (hole-effect) term, representing seasonal or otherwise periodic correlation superimposed on a slowly decaying trend — a pattern common in environmental time series and space-time data. Members of a product group may carry independent ranges (for instance a multi-year decay length and a one-year period) while sharing a common anisotropy orientation.

KrigeKit also offers fixed-form shapes that build an oscillation into a single elementary structure — a damped cosine (dco) and a Gaussian-process periodic / exponential-sine-squared kernel (cyc) — but these tie the decay and the oscillation to one shared range parameter. The product mechanism is the more general construction: it lets the decay length and the period be specified and fitted independently, and it can multiply any of the supported elementary shapes. The feature is exposed through a single product flag on set_vgm:

```
# exponential decay (5-unit range) x cyclic 1-unit period
k.set_vgm(1, 1, vtype="exp", sill=1.0, a_major=5.0)
k.set_vgm(1, 1, vtype="hol", sill=1.0, a_major=0.5, product=True)
```

Listing 1. Composing a multiplicative covariance: an exponential decay envelope multiplied by a cyclic hole-effect term.

https://doi.org/10.5194/egusphere-2026-5708
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

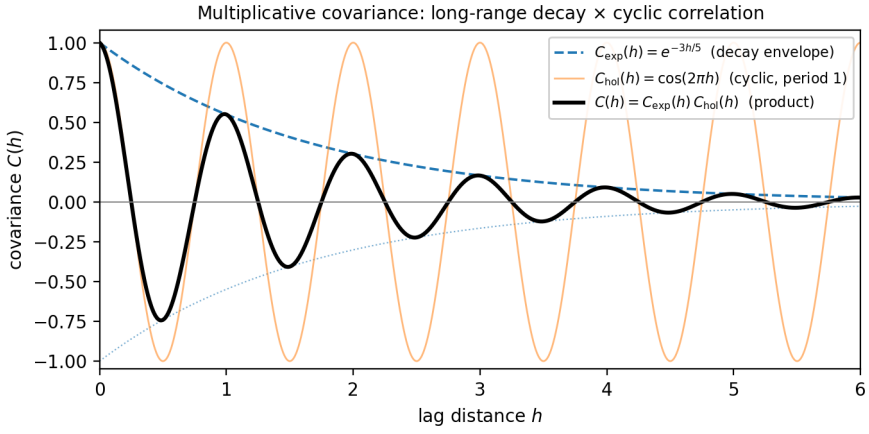


Fig. 1. *A multiplicative covariance $C(h) = C_{exp}(h) \cdot C_{hol}(h)$: a slowly decaying exponential envelope (range 5) modulating a cyclic hole-effect (period 1), shown with its two factors. This represents long-range decay combined with periodic correlation, as produced by Listing 1.*

2.9 Variogram analysis and model fitting

The kriging systems above require a covariance model, and KrigeKit includes an integrated variogram-analysis module for diagnosing and fitting it within the same workflow. From a sample dataset it forms the experimental variogram cloud in one, two, or three dimensions (and in space–time); from that cloud it derives directional experimental variograms along arbitrary axes, variogram maps (planar, polar, and a three-dimensional fence), and an automatic estimate of the anisotropy geometry — azimuth, dip, and axis ratios — by a weighted principal-component analysis of the lag vectors. The principal directions of continuity are thus read from the data rather than specified a priori, which is most valuable in three dimensions where they are hard to guess.

Theoretical models, including nested (multi-structure) models, are fit to the experimental variograms by weighted least squares. For multivariable (co-kriging) problems the auto- and cross-variograms are fit jointly as a Linear Model of Coregionalization: each coregionalization matrix B_u (Eq. 5) is parameterized through its Cholesky factor, $B_u = L_u L_u^{\mathsf{T}}$, during the fit, so every fitted sill matrix is positive semi-definite by construction and the multivariate model is guaranteed valid; an optional monotonicity constraint additionally keeps each fitted cross-variogram monotone. Fitted models apply directly to the solvers, keeping analysis and prediction in one place.

For a sparsely sampled primary co-kriged with a dense secondary covariate, where a jointly fitted LMC can drive the cross-covariance toward zero, KrigeKit also offers a simpler Markov-model (MM1) cross that scales the secondary's structure by the collocated correlation, $b_{ij}^u = \rho_{ij}\sqrt{b_{ii}^u\, b_{jj}^u}$ (Almeida and Journel, 1994) — valid for $|\rho| \leq 1$ without a joint fit.

https://doi.org/10.5194/egusphere-2026-5708
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

3. Software description

3.1 Overview and design

KrigeKit is organized as a compiled Fortran compute engine fronted by a Python layer with two parts: a thin ctypes binding that exposes the engine, and a self-contained pure-Python variogram-analysis module that prepares and validates the covariance models the engine consumes. The binding uses only ctypes to call the shared library and NumPy arrays for data exchange, so there is no run-time compilation or JIT step and the dependency footprint is small. The Fortran engine performs all heavy work: neighborhood search, covariance assembly, linear solves, simulation paths, and OpenMP scheduling.

The split follows the two tasks' different computational profiles. Variogram analysis is exploratory and iterative: the analyst bins the data into lag classes, inspects directional variograms and variogram maps, and refits candidate models by visual inspection and by least squares. This work is interactive and visual rather than compute-bound — it operates on compact lag summaries, not the full grid — and it relies on the plotting and optimization facilities of the scientific-Python ecosystem. Python is its natural home; reproducing it in Fortran would add little speed while pulling heavyweight dependencies into a core whose value lies in staying small and embeddable. Keeping the analysis in Python therefore lets the engine remain a lean, LAPACK-only shared library, while only the fitted models — a small number of structures, sills, and ranges — cross into the engine for the heavy estimation.

The engine exposes a C-compatible application programming interface through Fortran's iso_c_binding. The bundled Python wrapper is only one client of this C API: the same shared library can be called from any language with a C foreign-function interface (C/C++, R, Julia, MATLAB, and others) or embedded directly into larger modeling tools, so the engine is readily reusable beyond Python. The Python package therefore does not contain a Python extension module; it uses ctypes to load the platform-specific shared library.

https://doi.org/10.5194/egusphere-2026-5708
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

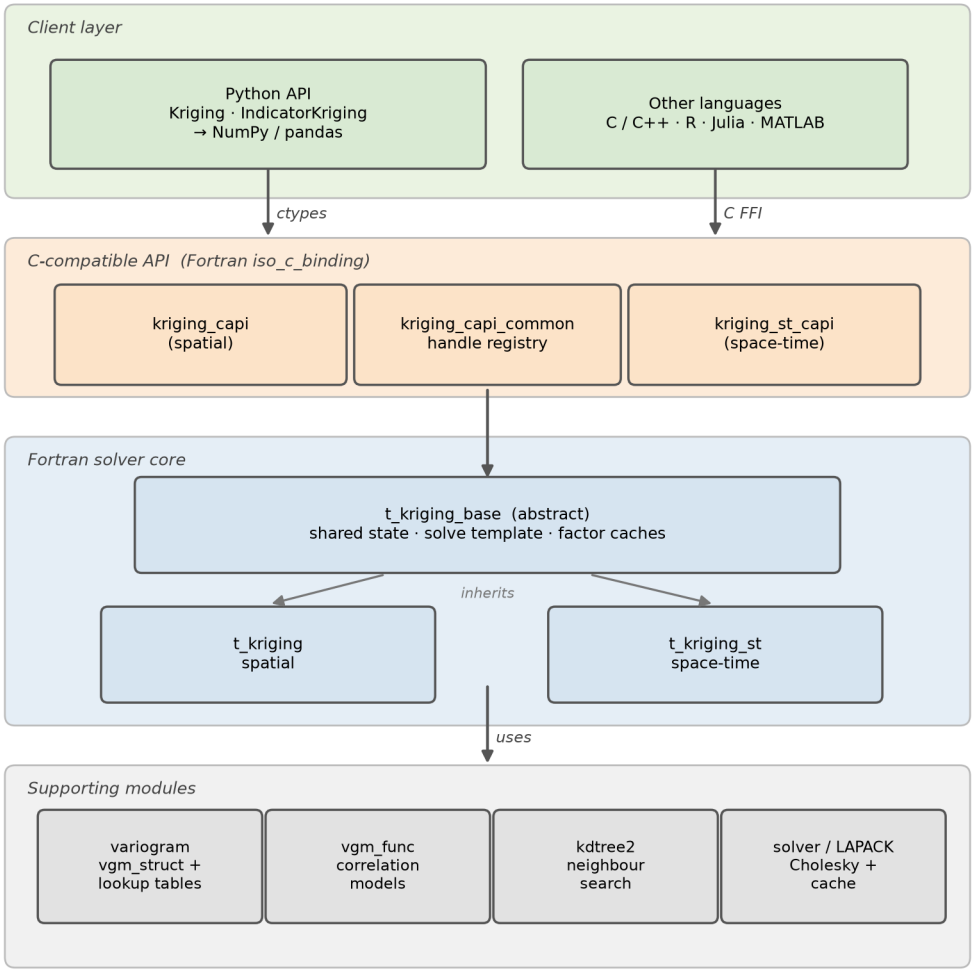


Fig. 2. *Layered architecture of KrigeKit: a thin Python/ctypes API and other languages (via the C foreign-function interface) call a C-compatible API built on Fortran iso_c_binding; a shared handle registry dispatches to the Fortran solver core, which builds on the variogram, neighbor-search, and linear-algebra modules. The solver class hierarchy is described in Sect. 3.2.*

3.2 Solver class hierarchy

The engine centers on an abstract base type, t_kriging_base, which owns state shared by all solvers (options, dimensions, observation/grid/block data, gradients, weight storage, and the factor caches) and the non-overridable solve template. Two concrete types specialize it: t_kriging for spatial kriging and t_kriging_st for space-time kriging. Type-specific behavior (initialization, neighbor search, LHS/RHS assembly, variance, finalization) is provided through deferred procedures, so the

https://doi.org/10.5194/egusphere-2026-5708
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

block loop, caching, drift handling, weight storage, and simulation bookkeeping reside in a single implementation while each concrete type defines only its covariance and search model. A shared C-API registry stores base-type pointers, and thin spatial and space-time C-API wrappers downcast to the expected concrete type.

Table 1. *KrigeKit capabilities and the originating solver type.*

Capability	Notes
Ordinary & simple kriging	Point and block support
Co-kriging	Linear Model of Coregionalization
Universal kriging / KED	External drift variables
Sequential Gaussian Simulation	Reproducible paths, multi-realization
Space-time kriging	Sum-metric and product-sum models
Multiple Indicator Kriging / SIS	Three cross-variogram strategies; secondary co-variate (collocated co-kriging)
Spatially Varying Anisotropy	Per-block variogram
Multiplicative (product) structures	Decay × cyclic correlation; Schur-product valid; independent ranges per factor
Cross-validation	Leave-one-out
Kriging weight reuse	Store and replay weights
OpenMP parallelism	Thread count controllable per solve() call

3.3 Covariance model library and evaluation

All variogram structures are built from a small library of elementary correlation functions $\rho(r)$ of the reduced, anisotropy-standardized distance $r = \tilde{h}/a$ (Table 2). Each structure scales ρ by a partial sill and may add a nugget; both the additive nesting of Sect. 2.1 and the multiplicative product groups of Sect. 2.8 are assembled from these shapes. The hole-effect and damped-cosine models can take negative values, which is what allows them to represent cyclic correlation.

Table 2. *Elementary correlation functions $\rho(r)$ available in KrigeKit (reduced distance $r = \tilde{h}/a$; $\rho = 0$ for $r \geq 1$ where a model has finite range).*

Code	Model	Correlation function $\rho(r)$
nug	Nugget	1 at $r = 0$, else 0
sph	Spherical	$1 - 1.5\,r + 0.5\,r^3 \;\; (r < 1)$
exp	Exponential	$\exp(-3r)$
gau	Gaussian	$\exp(-3.0625\,r^2)$
pow	Power	$1 - r^{1.5} \;\; (r < 1)$
bsq	Bi-square	$(1 - r^2)^2 \;\; (r < 1)$

https://doi.org/10.5194/egusphere-2026-5708
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

cir	Circular	$1-(2/\pi)[\,r\sqrt{1-r^2}+\arcsin r\,]\ \ (r<1)$
lin	Linear	$1-r\ \ (r<1)$
hol	Hole-effect	$\cos(\pi r)$
dco	Damped cosine	$\exp(-3r)\cos(\pi r)$
cyc	Periodic (exp-sine^2)	$\exp(-2\sin^2(\pi r))$

For speed, the covariance shape is stored as an integer tag and evaluated by a single pure-elemental dispatch routine, so the compiler inlines it, builds a jump table, and can vectorize (SIMD) the inner loop — there is no run-time polymorphism or heap allocation in the hot path. During system assembly KrigeKit precomputes a two-level piecewise-uniform lookup table — one per structure, or a single composite table when all structures share the same anisotropy — which keeps the maximum interpolation error below about 5×10^{-7} while removing transcendental-function calls from the innermost loop.

3.4 Solve template

Both concrete types execute the same solve template (Fig. 3): a pre-solve preparation hook, a parallel loop over grid blocks, and a post-loop save of any persistent cache. Within the loop each block assembles its linear system (neighbor search, then LHS/RHS assembly), solves it through the shared factorization/cache path, and computes the conditional variance. The steps that depend on covariance geometry — neighbor search, LHS/RHS assembly, and variance — are deferred procedures (hooks) supplied by the spatial or space-time type, while the block loop, caching, drift handling, weight storage, and simulation bookkeeping are shared. This keeps the parallelization and caching logic identical for spatial and space-time problems.

https://doi.org/10.5194/egusphere-2026-5708
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

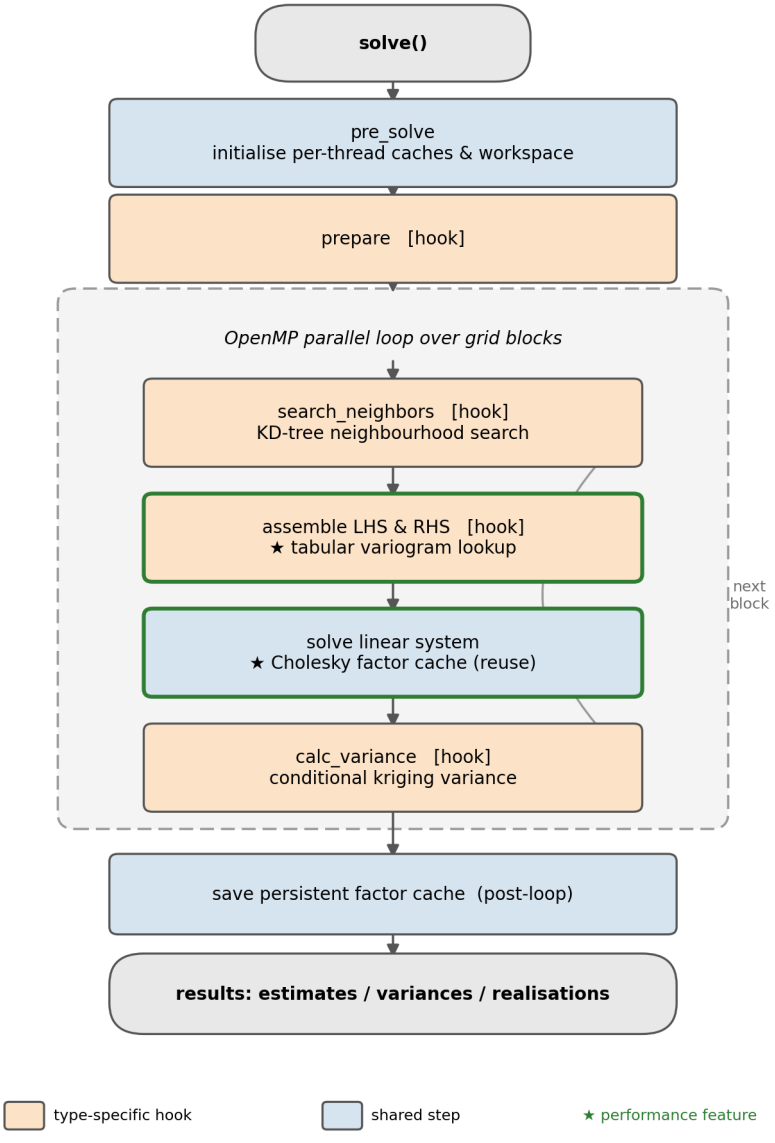


Fig. 3. *The unified solve() workflow shared by the spatial and space-time solvers. Type-specific steps are deferred procedures (hooks); the block loop and caching are shared. ★ marks the two performance features — tabular variogram evaluation (Sect. 3.3) and the Cholesky factor cache (Sect. 3.5).*

3.5 Performance design

Three mechanisms drive performance. First, the block loop is parallelized with OpenMP; the number of threads is selectable per solve() call. Second, the neighborhood covariance matrix is determined by the neighbor indices together with the

https://doi.org/10.5194/egusphere-2026-5708
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

covariance parameters, observation-error variances, and any local modifiers. Once the Cholesky factorization of that neighborhood covariance matrix has been computed ($O(n^3)$), it can be reused for any later block sharing the same neighbor set and unchanged matrix-defining parameters, reducing the per-block cost to a triangular solve ($O(n^2)$). Each thread keeps an independent single-entry cache and a bounded multi-slot hash cache (default 64 slots); an optional persistent cache on the kriging object retains assembled systems for inspection. Because contiguous blocks tend to share neighbors, per-thread hit rates are high in practice. After each solve(), hit/miss counts are reported in solver_stats, enabling the user to tune the thread count and cache size (Sect. 5).

Third, covariance assembly avoids repeated transcendental-function evaluation. As described in Sect. 3.3, each variogram structure is precomputed as a piecewise-uniform lookup table, so building the kriging matrices reduces to an anisotropic-distance transform followed by a table lookup with linear interpolation rather than evaluating exponentials, cosines, or square roots for every matrix entry. When all structures share the same anisotropy, KrigeKit collapses them into a single composite table, so the per-pair covariance cost becomes independent of the number of nested structures — one distance transform and one lookup — at a controlled accuracy (maximum error below $\sim 5 \times 10^{-7}$). This tabular evaluation is especially effective for space-time and multi-structure models, where the analytic covariance would otherwise dominate assembly time.

A complementary facility stores the kriging weights for each block on a first solve and replays them when only the observed values change (for example across simulation realizations or time snapshots with a fixed monitoring network), avoiding re-factorization entirely.

3.6 Python interface and workflow

A complete ordinary-kriging workflow illustrates the API:

```
import numpy as np
from krigekit import Kriging

k = Kriging()
k.set_obs(ivar=1, coord=obs_coord, value=obs_value)
k.set_grid(coord=grid_coord)
k.set_vgm(ivar=1, jvar=1, vtype="sph", sill=1.0, a_major=1.0)
k.set_search()
k.solve(nthread=4, ncache=64)
df = k.get_result_df()
```

Listing 2. Minimal ordinary-kriging workflow.

3.7 Distribution, testing, and documentation

KrigeKit is released under the MIT license, with two bundled permissive third-party components retained under their own terms — the LAPACK linear-algebra routines (BSD-3-Clause) and the kdtree2 nearest-neighbor search (Academic Free License v1.1), documented in the repository's THIRD_PARTY_LICENSES file. Pre-built binary wheels are published on

https://doi.org/10.5194/egusphere-2026-5708
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

PyPI for Linux (x86-64, manylinux_2_28), macOS (arm64 and x86-64), and Windows (x86-64), for Python 3.10–3.14, so most users install without a Fortran toolchain. Each platform wheel is tagged py3-none because the ctypes interface is independent of the Python extension ABI; the wheel remains platform-specific because it bundles the compiled shared library. Documentation is hosted on Read the Docs and comprises a quickstart, a topic-organized user guide — ordinary and simple kriging, universal kriging / external drift, indicator kriging, co-kriging, sequential Gaussian and indicator simulation, space-time kriging, spatially varying anisotropy, variogram fitting, data transforms, weight reuse, and performance tuning — an API reference generated from the docstrings, and a developer guide covering the layered architecture and the C API. A particular strength is the Sphinx-Gallery of runnable worked examples, organized into kriging, simulation, space-time, and variogram-analysis sections and spanning the full feature set: two-variable co-kriging on the Walker Lake data, kriging with external drift and with spatially varying anisotropy on groundwater levels, sequential Gaussian and indicator / co-indicator simulation of lithofacies, and one-, three-, and space-time-dimensional variogram fitting, among others. Because Sphinx-Gallery executes every example when the documentation is built, the gallery doubles as a continuously exercised regression layer that keeps the published code and figures reproducible, complementing the pytest suite of Sect. 4. Continuous integration builds and tests the wheels across the supported platforms.

4. Verification and validation

KrigeKit is verified by a suite of more than 600 automated tests (606 at the time of writing) together with subprocess tests of the command-line tool. The full suite runs in continuous integration (GitHub Actions) on Linux and Windows for Python 3.11 and 3.12 on every new git commit; in addition, the release workflow builds binary wheels for Linux, Windows, and macOS (both Intel and Apple Silicon) and import-tests them across Python 3.10–3.14 before publishing to PyPI. The tests fall into five groups: theoretical-property checks, leave-one-out cross-validation, comparison against reference results and other packages, simulation validation, and reproducibility/determinism.

4.1 Theoretical-property verification

A first group of tests checks exact mathematical properties of the kriging equations, independent of any reference implementation:

- Exact interpolation — at a node coincident with an observation and no measurement error, the estimate equals the observed value and the kriging variance is zero; with a prescribed observation-error variance the kriging variance equals that error.
- Unbiasedness — ordinary kriging of a constant field returns the constant exactly, confirming that the weights sum to one (Eq. (2)).
- Non-negative variance — the kriging variance is non-negative at every node for point, block, co-kriging, indicator, and space-time solves.

https://doi.org/10.5194/egusphere-2026-5708
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

- Block regularization — for the tested stationary models and block configurations, a block variance never exceeds the point variance at the block centroid, a small block reproduces the centroid point estimate, and Gaussian-quadrature discretization matches explicit sub-node discretization (Eq. (4)).
- Co-kriging consistency — the supplied LMC is checked for validity, the co-kriging variance is bounded by the primary sill, and adding an abundant secondary variable reduces the variance relative to ordinary kriging (Eqs. (3) and (5)).
- Neighborhood filtering — restricting the search radius gives the same estimate as physically removing the excluded data, and an empty neighborhood raises a controlled error rather than a silent result.

4.2 Cross-validation

KrigeKit provides leave-one-out cross-validation (LOO-CV) as a built-in diagnostic: each observation is removed in turn and re-estimated from the remainder, yielding the mean error (bias), the RMSE, the mean standardized squared error (MSSE, expected ≈ 1), the fraction of standardized residuals within $\pm 3\sigma$, and the correlation against the held-out values. It is worth being precise about what this checks. These statistics respond chiefly to how well the chosen variogram model fits the data, so LOO-CV assesses the adequacy of the model on a given dataset rather than the correctness of the solver — a misspecified model yields a poorly calibrated MSSE even from a perfectly correct implementation, and vice versa. We therefore report quantitative LOO-CV results where model adequacy is the question, in the case studies (Sections 6.1 and 6.3), and rely on the property and reference checks of Sections 4.1 and 4.3 to verify the implementation itself. As a coarse regression guard the continuous-integration suite nonetheless runs a LOO-CV smoke test on a fixed dataset, asserting near-zero bias, an MSSE below 4, and more than 85 % of residuals within $\pm 3\sigma$.

4.3 Comparison with reference results and other packages

Beyond internal properties, KrigeKit estimates are regression-tested against pre-computed reference fields (a correlation above 0.99 is required on a 62-observation percent-coarse dataset), and the command-line tool is checked against the Python library on the same inputs. To position the engine against established software, we additionally compare estimates and variances against independent implementations on identical data and variograms; Table 3 reports the agreement.

Table 3. *Agreement with independent implementations on identical data, variogram, and 30-neighbor sets (FourCounties benchmark; root-mean-square difference of estimate and variance). KrigeKit uses parity settings (no output bounds, no positive-weight correction).*

Comparison	Estimate RMSE	Variance RMSE
KrigeKit vs PyKrige — OK	1.0e-7	9.6e-10
KrigeKit vs gstat — OK	1.0e-7	9.6e-10
KrigeKit vs GSLIB KT3D — OK	3.8e-6	3.7e-7

https://doi.org/10.5194/egusphere-2026-5708
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

KrigeKit vs gstat — CK (standard)	1.0e-7	1.6e-9
KrigeKit vs GSLIB COKB3D — CK	3.7e-6	4.8e-2 *

Estimates agree to about 10^{-7} *— effectively machine precision — once the range conventions, anisotropy, and 30-neighbor sets are matched; the larger GSLIB differences (~* 10^{-6}*) are consistent with its legacy single-precision arithmetic and formatted-text output. * GSLIB COKB3D reports its co-kriging variance under a different convention, so that entry is not directly comparable, although the estimates still match.*

4.4 Simulation validation

Sequential Gaussian simulation is validated against the kriging solution from the same engine, and separately for exact reproducibility. On the percent-coarse test dataset (62 conditioning points, a 4800-node grid, a spherical model of sill 0.12 and 5 km range), an ensemble of 500 conditional realizations reproduces the kriging solution: the ensemble mean matches the ordinary-kriging estimate (correlation 0.989, RMSE 0.033 against a data standard deviation of 0.34), and the ensemble variance matches the kriging variance (correlation 0.986, regression slope 0.97), the slight shortfall being consistent with the simple-kriging variance that conditional simulation targets and with finite-ensemble sampling (Fig. 4a, b). The experimental variogram confirms that the engine imposes the specified covariance: averaged over unconditional (prior) realizations it reproduces the input model, reaching the 0.12 sill near the 5 km range (Fig. 4c). The conditional realizations fall slightly below the model in the structured range — their spatial variance is reduced from 0.126 to 0.102 — which is the expected conditioning effect, the variance removed by honoring the 62 data, and not a deficiency of the algorithm. The ensemble mean converges toward the kriging estimate as the ensemble grows (Fig. 4d).

Reproducibility was measured, not assumed. A fixed seed (or an explicit random path and uniform-sample set) yields bit-for-bit identical realizations across repeated runs, whereas a different seed yields a distinct field. The result is moreover invariant to the OpenMP thread count: on this problem the 500-realization ensemble, and the deterministic kriging estimate and variance, are bit-for-bit identical at 1 and at 24 threads (maximum absolute difference exactly zero), though for two different reasons. Kriging parallelizes over independent grid blocks with no cross-thread reduction, so the estimate and variance carry no floating-point summation-order or scheduling dependence. Sequential Gaussian simulation, by contrast, is inherently ordered — each node conditions on previously simulated nodes — so the engine runs the simulation path single-threaded, disabling OpenMP whenever realizations are requested (Sections 2.4, 7.4); invariance to the thread count is then immediate, because the simulated ensemble cannot depend on a thread count that the simulation path does not use. In both cases the result draws from a deterministic random stream fixed by the master seed (or an explicit stored path and uniform-sample set) rather than by the order in which threads happen to run. Bit-identity therefore holds for a fixed seed on a given build and platform (the same compiler and floating-point settings); cross-platform reproduction is guaranteed in the regression suite for a stored explicit path and sample set, which reproduces a realization bit-for-bit.

https://doi.org/10.5194/egusphere-2026-5708
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

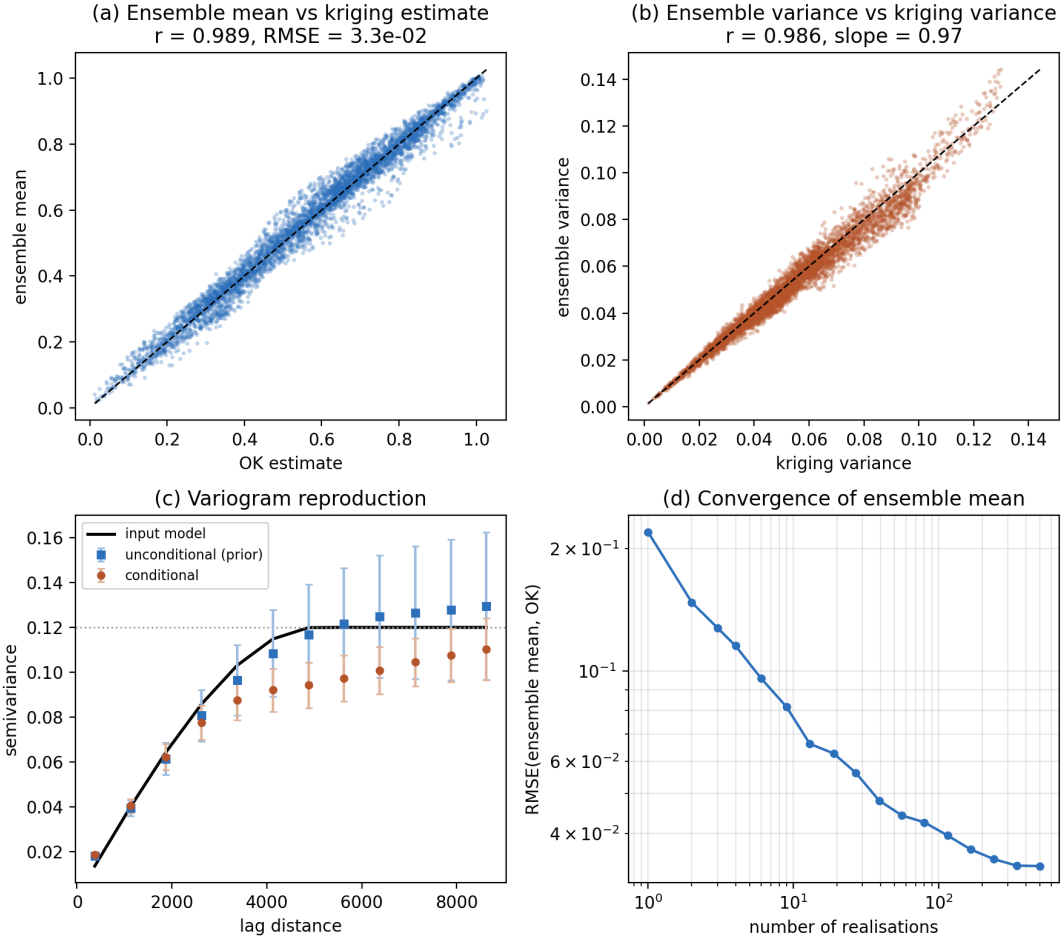


Fig. 4. *Sequential Gaussian simulation validation on the percent-coarse dataset (500 conditional realizations). (a) Ensemble mean versus the ordinary-kriging estimate; (b) ensemble variance versus the kriging variance; (c) experimental variograms of unconditional (prior) and conditional realizations (mean ± standard deviation over realizations) against the input model — the prior reproduces the model, while conditioning to data reduces the structured variance; (d) convergence of the ensemble mean to the kriging estimate as the number of realizations grows. Dashed lines in (a) and (b) are the 1:1 reference.*

4.5 Reproducibility and numerical determinism

Parallelism does not change results. Because the engine parallelizes over independent grid blocks with no cross-thread reduction, a parallel solve returns bit-for-bit the same estimates and variances as the single-threaded solve (Sect. 4.4), and repeated calls are bitwise reproducible; the thread count and the per-thread factor cache change the schedule, not the answer. The weight-store round-trip (store, persist to a factor file, reload) reproduces estimates and variances bit-for-bit, and reusing

https://doi.org/10.5194/egusphere-2026-5708
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

a Kriging object with new data or a new grid reproduces the result of a freshly constructed object. These checks run on every supported platform in continuous integration.

5. Performance

KrigeKit's throughput rests on the design choices of Sect. 3 — OpenMP parallelism over the estimation grid, precomputed tabular variogram evaluation, and a per-thread cache of Cholesky factorizations. We characterize its parallel scaling on a large three-dimensional co-kriging problem: a coarse-fraction field co-kriged with airborne-EM resistivity over the FourCounties Central Valley dataset (506,645 primary and 1,205,193 secondary observations; exponential model with practical ranges 9854 / 3809 / 60 m), estimated over the complete 3,411,800-cell grid on an Intel Core i7-13700K workstation (24 logical cores). A controlled comparison against established packages on the same problem is reported in Appendix A, kept separate so as not to interrupt the description of the software; its headline result is summarized below.

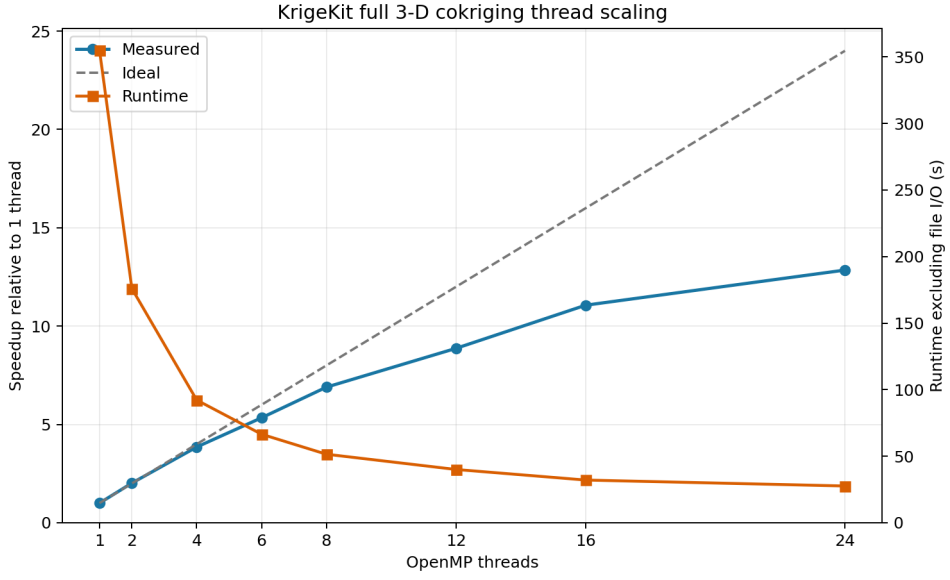


Fig. 5. *Parallel thread scaling for the three-dimensional co-kriging benchmark, measured over the complete estimation grid (3,411,800 cells, both variables solved jointly): measured speed-up relative to a single thread against ideal linear scaling (left axis), and wall-clock runtime excluding file I/O (right axis), versus OpenMP thread count.*

Figure 5 shows the thread scaling over the complete 3,411,800-cell grid with both variables co-kriged: speed-up tracks the ideal line closely at low thread counts and then tapers as the dense solves become memory-bandwidth bound, reaching roughly 13× at 24 threads while wall-clock runtime falls by the same factor.

Against established packages on the same estimation problem — with the full protocol, configurations, and per-package results given in Appendix A — single-threaded KrigeKit is 13.5× faster than the R package gstat for ordinary kriging and 9.3× for co-kriging; at 24 threads those margins widen to 60× and 67×. Under this configuration — the tested executables,

https://doi.org/10.5194/egusphere-2026-5708
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

moving-neighborhood settings, and Windows process and text-I/O conventions detailed in Appendix A — recompiled GSLIB is roughly another order of magnitude slower than gstat, and two Python packages (PyKrige and GSTools) do not complete the full-source problem at all. Every package that finishes agrees with KrigeKit to the precision reported in Table 3.

6. Environmental case studies

6.1 Space-time mapping of groundwater levels in the Middle Republican River basin

The principal case study reconstructs groundwater-level dynamics across the Middle Republican Natural Resources District in the Republican River basin of southern Nebraska, which overlies the High Plains (Ogallala) aquifer. The data are 23,075 manual water-level measurements at 988 monitoring wells over 1978–2022, recorded mostly at half-year or annual intervals (University of Nebraska–Lincoln Conservation and Survey Division). Because hydraulic head carries a strong topographic trend, we model the depth to water (land-surface elevation minus measured level) and convert back to head for reporting. Reconstructing the full space-time field from these sparse, irregularly timed records is the environmental problem: a purely spatial map at any single date discards the information carried by neighboring dates, yet capturing the seasonal cycle needs a space-time covariance that is at once valid and quasi-periodic — a combination most kriging tools do not provide (gstat offers product-sum space-time models but not a multiplicative seasonal term, and GSLIB has no space-time kriging at all).

Water levels combine two temporal signals: a slowly decaying multi-year trend — driven by sustained groundwater abstraction and multi-year drought — and a weaker but persistent annual cycle, with drawdown during the summer irrigation season and partial recovery over winter. This seasonal-on-trend structure is what the multiplicative covariance of Sect. 2.8 represents: we model the temporal marginal as a long-range Gaussian decay plus a second Gaussian decay multiplied by an annual hole-effect term, $C_T(u) = A\,G(u) + B\,G(u)\cos(2\pi u)$, a valid quasi-periodic covariance whose seasonal amplitude decays with the same long-term envelope. The spatial and temporal marginals are combined additively through the space-time model of Sect. 2.5 with zero interaction sill, $C(h,u) = C_S(h) + C_T(u)$.

The spatial marginal, fitted to the long-term mean depth to water at each well, is a near-pure spherical structure (sill $\approx 3.5 \times 10^3$ ft², practical range $\approx 1.0 \times 10^5$ ft, negligible nugget). The temporal marginal, fitted from within-well pairs, has a background Gaussian sill of 38.5 ft² and a seasonal sill of 2.6 ft² over a shared Gaussian range of about 35 yr, a one-year period (annual hole-effect, 0.5-yr practical range), and a 3.4 ft² nugget; the annual component is therefore real but small relative to the multi-year variation. The multi-year decay range exceeds the 44-year record and is only weakly identified — the data constrain the near-origin behavior rather than the long-term envelope.

KrigeKit produces groundwater-level estimates and kriging variances at unmonitored locations and times, enabling continuous hydrograph reconstruction and gap-filling. Reconstructed half-year hydrographs at well-sampled wells track the observed records closely (RMSE 0.7–1.1 ft) with uncertainty bands that widen over unmonitored periods, and sparse wells holding a single measurement are reconstructed from neighboring records to within 1.7–4.6 ft in leave-one-well-out tests.

https://doi.org/10.5194/egusphere-2026-5708
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

The clearest demonstration is a contemporaneous snapshot: withholding a reproducible 20% of the 325 wells measured on the same date (mid-2008), a contemporaneous-only spatial map predicts the withheld depths with an RMSE of 26.6 ft, whereas the space-time model — which additionally draws on each well's measurements from other dates — achieves 1.7 ft. Isolating the multiplicative annual term at equal total temporal sill, the product model improves on a smooth (non-cyclic) temporal covariance from 1.9 ft to 1.7 ft: a modest but consistent gain that reflects the weak annual signal while confirming the value of representing it with a valid quasi-periodic structure.

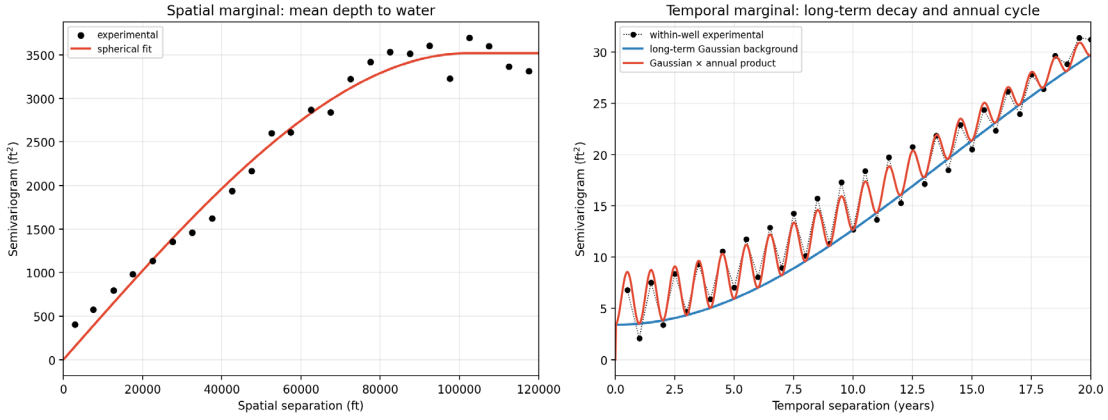


Fig. 6. *Fitted space-time marginals: the spatial spherical variogram of mean depth to water (left) and the temporal variogram (right), showing the long-term Gaussian background (blue) and the full Gaussian × annual-hole-effect product (red), whose alternating half-year response decays with the long-term envelope.*

https://doi.org/10.5194/egusphere-2026-5708
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

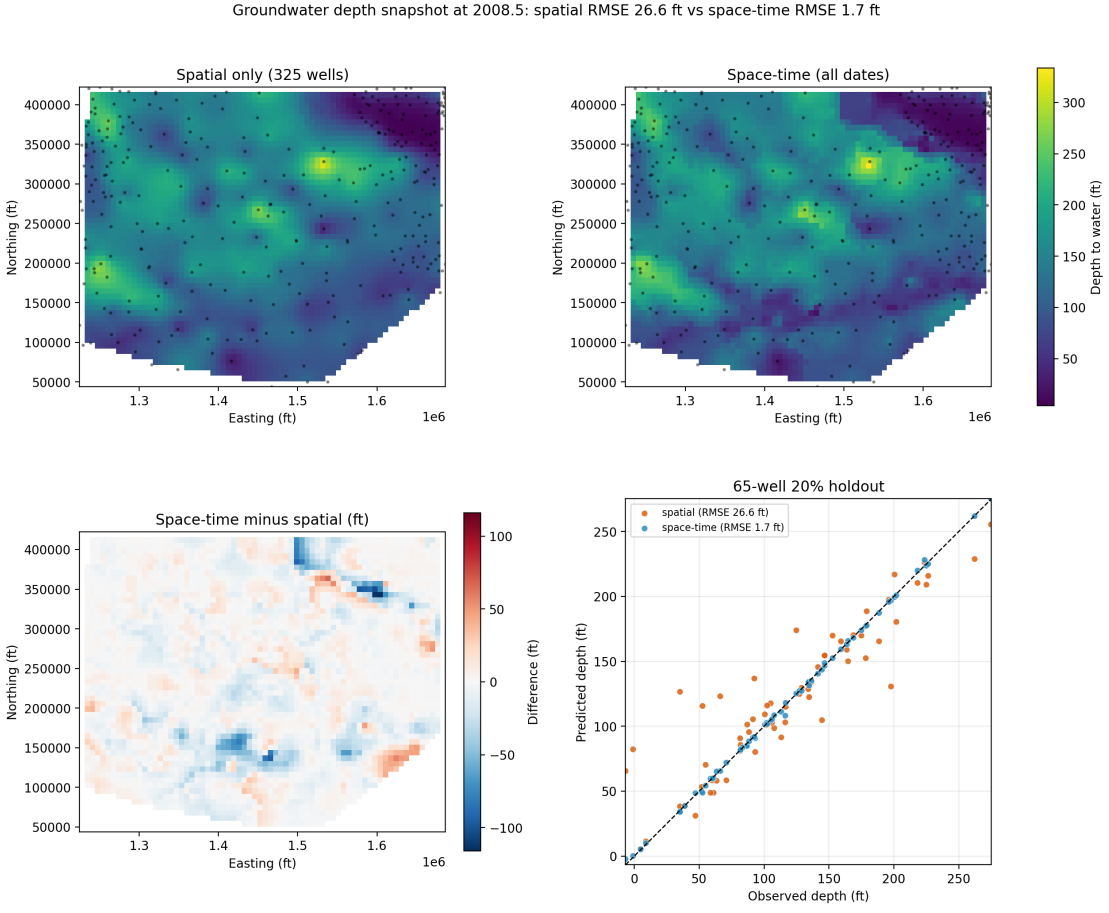


Fig. 7. *Space-time reconstruction at the mid-2008 snapshot: contemporaneous-only spatial map, all-dates space-time map, and their difference, with the 20% holdout cross-plot. Drawing on other-date records reduces the holdout RMSE from 26.6 ft (spatial only) to 1.7 ft (space-time).*

Two distinct gains should be separated. The decisive improvement in point prediction comes from the space-time formulation itself: borrowing strength across dates cuts the held-out snapshot RMSE more than fifteen-fold, from 26.6 ft for a contemporaneous-only spatial map to 1.7 ft for the space-time map (Fig. 7). The multiplicative covariance plays a different, more specific role — seasonal reconstruction. The annual component carries a sill of only 2.6 ft² against a 38.5 ft² background, so it is a real but secondary part of the temporal variance; representing it as a valid quasi-periodic structure lets the reconstruction follow the summer-drawdown / winter-recovery cycle rather than absorbing it into the smooth trend, which a non-cyclic or purely separable model would do. The case study therefore demonstrates the space-time engine as the source of predictive accuracy and the multiplicative covariance as the means of rendering the seasonal signal a modeled, valid part of that prediction. The complete workflow is available as a runnable gallery example (st_ok2d_gwlevel), with the marginal and joint variogram fitting shown separately (st_variogram_fitting_gwlevel).

https://doi.org/10.5194/egusphere-2026-5708
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

6.2 Multiple-indicator simulation of aquifer texture from well logs and airborne electromagnetic data in a California watershed

Characterizing the three-dimensional distribution of coarse- and fine-grained deposits is central to sustainable groundwater management, for example under California's Sustainable Groundwater Management Act (SGMA). This case study applies multiple-indicator simulation (MIS/SIS) to map aquifer texture in the Sacramento River alluvial aquifer of Butte County, California (west of Chico) — the same aquifer system used for the spatially varying anisotropy study of Sect. 6.3. Three ordered hydrofacies (fine, sand/mixed and coarse gravel) are derived from the drillers' lithology descriptions by coarse-fraction scoring. The methodological challenge is to let the dense but indirect AEM constrain a categorical texture model that is honored at the sparse logs — that is, to co-simulate categorical indicators with a continuous geophysical covariate under a provably valid coregionalization. Few general kriging packages support this combination: indicator simulation conditioned on a continuous secondary through a positive-semidefinite LMC (or a Markov cross), with separate per-variable search neighborhoods for the sparse hard data and the dense covariate.

Two complementary data sources are used: lithology descriptions from drillers' well-completion logs, which provide direct but sparse and irregularly distributed texture observations, and airborne electromagnetic (AEM) resistivity surveys, which provide dense, laterally continuous coverage that discriminates coarse (resistive) from fine (conductive) sediments. The hard data are 433 drillers' logs composited to a regular 5 m vertical support; the secondary is a 30-layer airborne-EM resistivity inversion at 6 327 soundings (California Department of Water Resources statewide AEM Surveys, https://data.cnra.ca.gov/dataset/aem), retained as log-resistivity for layers above the conservative depth of investigation. Because the soundings are spaced about 34 m along flight lines — far finer than the simulation grid — they are first averaged into 200 m along-line supports, never blending separate flight lines.

The two data sets are coupled through a four-variable linear model of coregionalization (LMC): the three texture indicators plus standardized log-resistivity as a fourth, densely sampled co-kriging variable. KrigeKit's VariogramSystem.fit_lmc parameterizes each nested coregionalization matrix as $L L^T$, so every fitted nugget and sill matrix is positive semidefinite by construction while still admitting the signed cross-sills the data require — negative between fine texture and resistivity, positive between coarse texture and resistivity; a monotonicity constraint keeps each fitted cross-variogram monotone (Fig. 8). This validity guarantee is exactly what an unconstrained least-squares cross-variogram fit cannot provide.

The drillers' indicators are nearly pure-nugget at the well spacing, whereas the AEM carries lateral continuity to several kilometers. The LMC therefore uses two nested structures sharing one anisotropy: a short (about 2 km) texture structure and a long (about 7 km) structure that carries the AEM continuity. The indicator–AEM cross is fit on the collocated subset — each well paired with its nearest sounding within 200 m — while the AEM auto-variogram is taken from the full dense field so that the long structure is populated; the fitted model is then applied to the full heterotopic simulation, in which logs and AEM occupy different locations.

https://doi.org/10.5194/egusphere-2026-5708
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.



```
    # K texture indicators + 1 secondary (standardized log-resistivity)
    sysv = VariogramSystem(nvar=K + 1)
    for i in range(1, K + 2):
        sysv.set_obs(i, coord, values[:, i - 1])      # collocated subset
    for i in range(1, K + 2):                          # two nested structures
        for j in range(i, K + 2):
            sysv.set_vgm(i, j, "sph", **short)             # ~2 km texture
            sysv.set_vgm(i, j, "sph", append=True, **long)  # ~7 km AEM
    res = sysv.fit_lmc(fit_ranges=False)               # L L^T per structure -> PSD
    fitted = res.target

    ik = IndicatorKriging(ncat=K, nvar=K + 1, ndim=3, nsim=NSIM, std_ck=True)
    ind = encode_indicator_matrix(log_litho, labels)
    for k in range(1, K + 1):
        ik.set_obs(ivar=k, coord=log_xyz, value=ind[:, k - 1])
    ik.set_obs(ivar=K + 1, coord=aem_xyz, value=aem_logrho)   # dense secondary
    fitted.apply(ik, observations=False)
    ik.set_grid(coord=grid_xyz); ik.set_sim()
    for k in range(1, K + 2):
        ik.set_search(ivar=k, nmax=32)
    ik.solve()
    sims, _ = ik.get_results()      # (ngrid, K, NSIM) one-hot realizations
```

Listing 3. Four-variable indicator–resistivity cokriging: a valid (PSD) LMC fitted with fit_lmc and applied to a heterotopic indicator simulation conditioned on the dense AEM secondary.

At the collocated wells the log-resistivity correlates with the coarse indicator at $\rho \approx 0.28$ and with the fine indicator at $\rho \approx -0.21$ (mixed ≈ 0); the fitted cross-variograms (Fig. 8) resolve this co-structure to about 1–3 km. Including the AEM changes the most-probable texture in roughly a quarter of grid cells relative to the logs-only baseline (Fig. 9, left versus center). The effect is real but modest, and necessarily so: a valid LMC bounds each indicator–AEM cross-sill by the geometric mean of the two structured sills, so with near-structureless indicators the co-located coupling is carried largely by a cross-nugget (the small near-origin offset in Fig. 8); with $\rho \approx 0.28$ the covariate can refine, but not dominate, the texture field.

To test whether the AEM materially improves prediction rather than merely changing it, we ran a five-fold leave-wells-out cross-validation: in each fold the LMC is refit on the training wells and facies probabilities are estimated at the held-out composites with and without the AEM secondary (10,827 held-out composites; coarse fraction 0.31). Adding the AEM improves every metric, consistently but modestly. For the coarse indicator the held-out area under the ROC curve rises from 0.656 to 0.667, the Brier score falls from 0.204 to 0.201, and the log score from 0.626 to 0.612; three-class accuracy rises from 47.4 % to 48.6 % and the three-class log score falls from 1.341 to 1.317. That all five metrics move in the favorable direction, across independently held-out wells, confirms that the covariate carries real — if small — predictive information about texture; the magnitude is consistent with the $\rho \approx 0.28$ point correlation and the positive-semidefinite cross-sill bound, and is a property of the data, not a limitation of the cokriging.

https://doi.org/10.5194/egusphere-2026-5708
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

To illustrate the stronger spatial influence expected when the secondary variable has a substantially larger validated cross-correlation with the indicators, Fig. 9 (right) repeats the simulation as a sensitivity case with a deliberately strong, long-range cross ($\rho = 0.7$) on a smoothed covariate — not a fit to these data: the cokriging then imposes coherent, resistivity-consistent texture bodies between the wells. The Markov-model cross (set_markov_cross) gives essentially the same result here, confirming that the covariate's value is set by its correlation with texture rather than by the choice of cross construction.

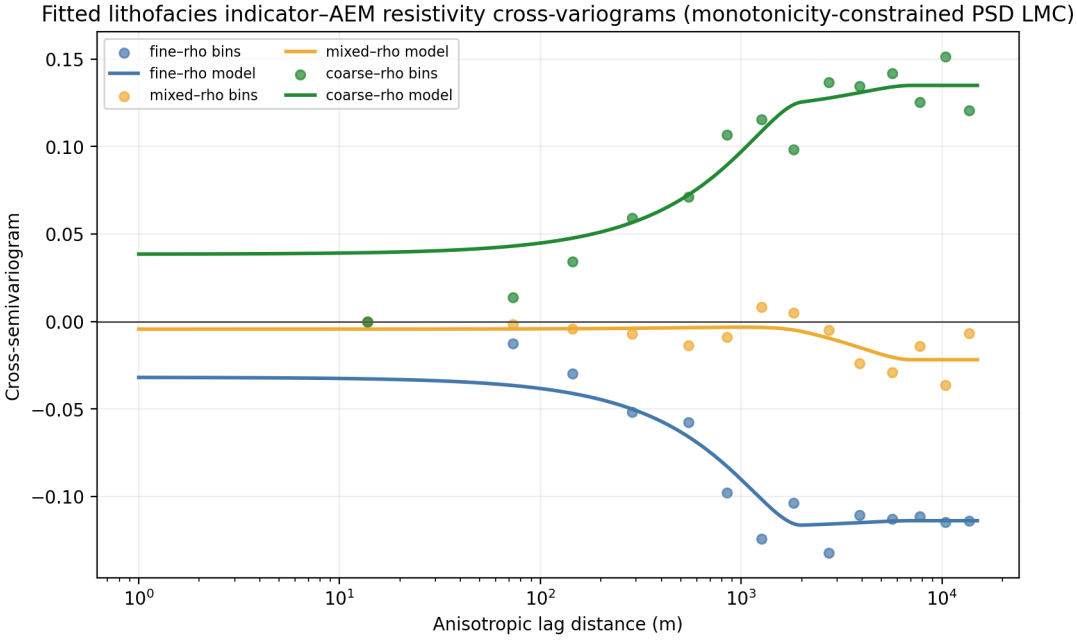
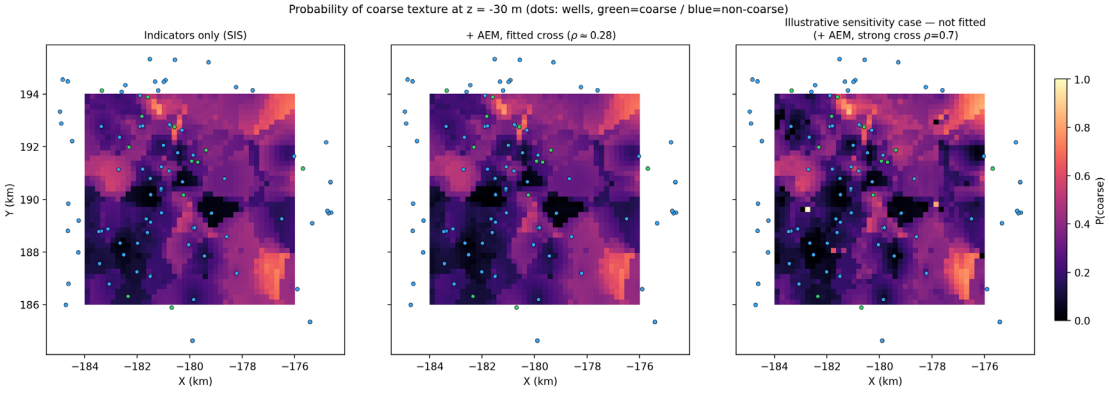


Fig. 8. *Fitted lithofacies indicator–AEM resistivity cross-variograms for the Butte aquifer. Points: experimental cross-semivariances on the collocated subset; curves: the monotonicity-constrained, positive-semidefinite LMC fit (the near-origin offset of each cross is a co-located cross-nugget). Coarse texture is positively coregionalized with resistivity, fine negatively, mixed near zero.*



https://doi.org/10.5194/egusphere-2026-5708
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

Fig. 9. *Probability of coarse texture at $z = -30$ m. Left: indicator simulation from well logs only. Center: cokriging with AEM resistivity at its fitted correlation ($\rho \approx 0.28$), a modest refinement. Right: illustrative sensitivity case — not fitted to observations — a deliberately strong, long-range cross ($\rho = 0.7$) on a smoothed covariate, showing the coherent texture structure the cokriging imposes when the secondary is strongly informative. Dots: wells (green = coarse, blue = non-coarse).*

The full Butte AEM survey is too large to bundle with the package, so the co-indicator-simulation technique used here — indicator simulation conditioned on a continuous secondary through co-kriging — is provided as a self-contained, runnable gallery example (s_coik_lithofacies) on a smaller lithofacies dataset.

6.3 Spatially varying anisotropy in the Sacramento River alluvial aquifer

The first two case studies use a single, globally constant anisotropy. In many environmental settings, however, the direction of maximum spatial continuity rotates across the domain — for example aquifer and sediment properties in a meandering fluvial system, where continuity follows the sinuous channel and valley axis rather than a fixed compass bearing. Imposing one global anisotropy then smears correlation across channel boundaries and mis-orients the interpolation. Most kriging packages expose only a single, domain-wide anisotropy, so this rotating geometry cannot be honored without manually partitioning the domain into quasi-stationary zones; a per-block (locally varying) anisotropy that integrates directly with the kriging neighborhood is uncommon outside specialist workflows. This case study is a targeted proof of concept for KrigeKit's spatially varying anisotropy on a real but modest dataset (62 wells), not a claim of general superiority.

This case study maps the coarse fraction (the proportion of coarse-grained sediment) of the alluvial aquifer along the Sacramento River in Butte County, California, west of Chico, from 62 well measurements onto a 4 800-cell grid. Following the general principle of geologically informed, spatially varying anisotropy used by Schilling et al. (2022), we prescribe the direction of maximum spatial continuity from mapped river geometry, that is, the direction of maximum continuity is taken to follow the river. Each well is assigned a local anisotropy azimuth together with major and minor correlation ranges (Fig. 10), set manually with reference to the mapped channels: the azimuth tracks the bearing of the nearest channel, and the along-channel range and major-to-minor ratio are largest where the well sits within the active channel belt. Assigning these attributes per well, rather than from a single distance-to-river decay rule, lets the conceptual model accommodate local variations in channel form and connectivity.

These per-well attributes are then carried to every estimation cell by kriging the azimuth, major range, and anisotropy ratio across the grid in two successive iterations. Iteration 0 uses an isotropic helper variogram; iteration 1 re-kriges the same attributes using the spatially varying field from iteration 0 as its helper, so the interpolated orientation and ranges themselves honor the channel-parallel structure. Azimuth is interpolated as an axial quantity (period 180°) to avoid wrap-around at the 0°/180° boundary, giving a smoothly rotating orientation field. The resulting along-channel range varies from about 3 km away from the river to nearly 8 km within the channel belt, and the major-to-minor ratio from near 1:1 (isotropic) to about 4:1 (strongly channelized).

https://doi.org/10.5194/egusphere-2026-5708
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

KrigeKit's spatially varying anisotropy (SVA) mode then assigns each estimation cell its own rotated variogram. With varying_vgm=True, set_vgm_block supplies the cell's local azimuth together with its major and minor ranges (here a spherical model); per-cell range scaling and a local nugget are also available.

```
k = Kriging(varying_vgm=True, bounds=[0.0, 1.0])
k.set_obs(ivar=1, coord=well_xy, value=coarse_fraction)
k.set_grid(coord=grid_xy)
# per-cell azimuth and ranges; both greatest near the river (see text)
for ib in range(len(grid_xy)):
    k.set_vgm_block(ib + 1, 1, 1, vtype="sph", sill=0.12,
                    a_major=grid_a_major[ib], a_minor1=grid_a_minor[ib],
                    azimuth=grid_azimuth[ib])
k.set_search(ivar=1, nmax=30)
k.solve()
```

Listing 4. Spatially varying anisotropy: a per-cell variogram whose azimuth follows the interpolated channel orientation.

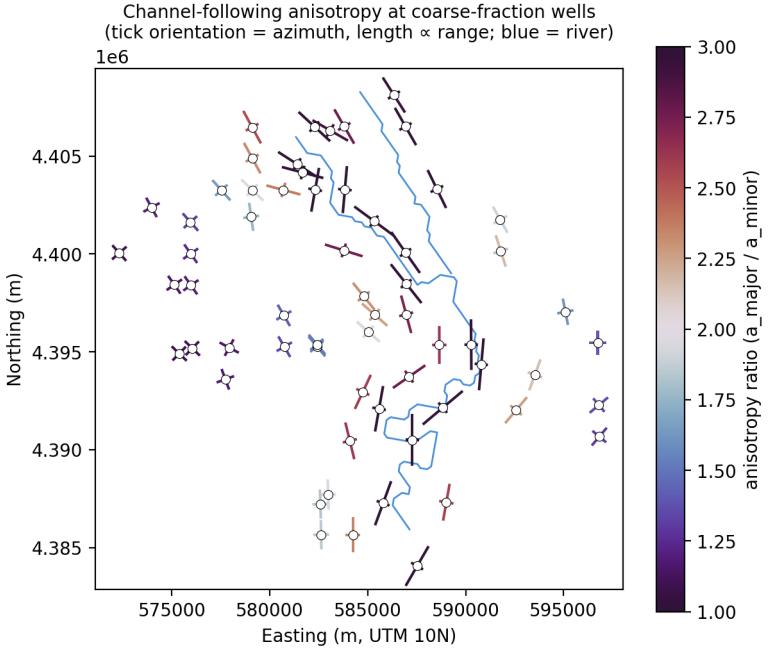


Fig. 10. *Channel-following anisotropy at the 62 coarse-fraction wells: tick orientation is the local major-axis azimuth, tick length is proportional to the correlation range, and color is the anisotropy ratio. Both range and ratio are largest for wells within the channel belt of the mapped Sacramento River (blue). The per-well azimuth and ranges are then interpolated to every grid cell by a two-iteration kriging bootstrap for the SVA solve.*

https://doi.org/10.5194/egusphere-2026-5708
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

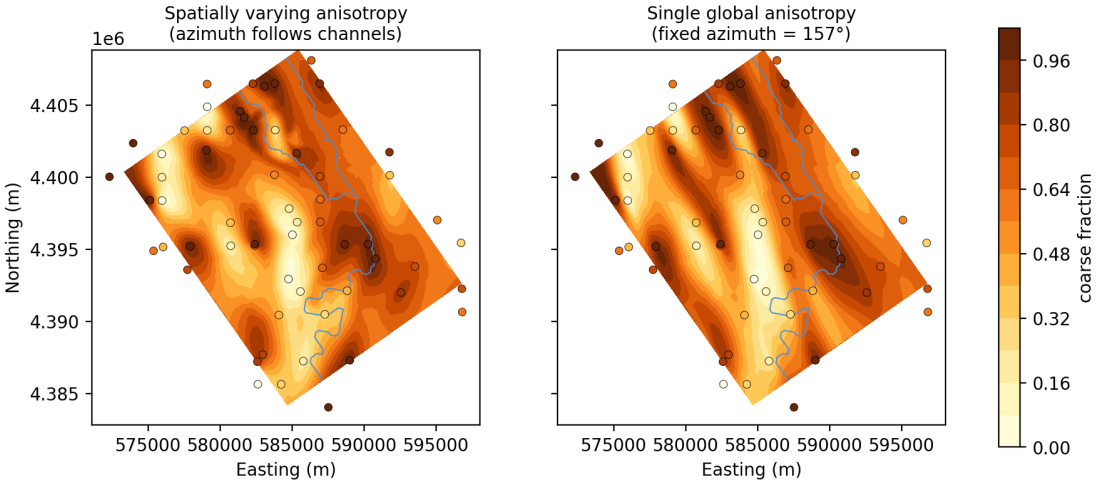


Fig. 11. *Coarse-fraction interpolation with spatially varying anisotropy (left) versus a single global anisotropy (right). SVA reproduces curvilinear, channel-following continuity, whereas the global model imposes a uniform directional banding across channel boundaries.*

Leave-one-out cross-validation (Table 4) gives the SVA model the lowest error, improving on both the isotropic model and a single global anisotropy. The mean standardized squared error (MSSE), which equals one for a well-calibrated variogram, also moves closest to one under SVA, indicating the per-cell variograms describe the prediction uncertainty more faithfully than a single stationary model.

Table 4. *Leave-one-out cross-validation of coarse-fraction interpolation, Sacramento River alluvial aquifer (62 wells). MSSE is the mean standardized squared error (ideal value 1).*

Anisotropy model	RMSE	MSSE
Isotropic	0.354	2.62
Single global (azimuth 157°)	0.337	1.87
Spatially varying (channel-following)	0.316	1.49

SVA improves on the global anisotropy in both accuracy and calibration (Table 4). The RMSE gain is modest because most of the 4 800 cells lie away from the river, where all three models agree; the improvement is concentrated in the channel belt, where texture governs flow and transport. The MSSE change is the more telling: a value near 1 means the kriging variance faithfully measures the actual error, whereas the isotropic and global models are badly over-confident — reporting uncertainty standard deviations roughly 1.4–1.6 times too small (the square root of the mean squared standardized error, a standard-deviation rather than a variance scaling) — so SVA not only predicts better but reports better-calibrated uncertainty, although the MSSE of 1.49 still indicates underestimation of prediction uncertainty. This matters when the estimates feed risk or transport assessments. The cause is geometric: a single fixed azimuth mis-orients correlation wherever the channel swings away from the domain-mean bearing, whereas SVA follows the channel locally. The maps make this clear (Fig. 11): SVA reproduces curvilinear, channel-following continuity that is strongest in the channel belt and fades to

https://doi.org/10.5194/egusphere-2026-5708
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

near-isotropic away from the river, whereas the global model imposes uniform directional banding across channel boundaries. The per-well azimuth and range assignment is a conceptual choice that could be refined with directional variograms. The full comparison is available as a runnable gallery example (s_ok2d_sva).

7. Discussion

7.1 Usability and adoption

The Python API mirrors common geostatistical workflows in a few method calls, returns results as NumPy arrays or pandas DataFrames, and installs from PyPI without a compiler. Beyond Python, the C-binding API lets the engine be called from other languages and embedded in other modeling frameworks, broadening its potential for integration into existing environmental-modeling pipelines.

Beyond standalone use, the engine has recently been adopted as the geostatistical back-end of two applied groundwater-modeling tools developed at the authors' organization. Texture2Par (Scantlebury et al., 2025), which interpolates borehole texture onto model grids and derives MODFLOW and IWFM aquifer parameters, and Groundwater Desktop (S.S. Papadopulos & Associates, 2026), a pre- and post-processing environment for groundwater models, both call the KrigeKit engine through its C API in releases subsequent to their original publication. Because the two build on the same interface — one from compiled code, the other from a scripted workflow — they exercise the engine across the exact boundary the architecture was designed around, and their ongoing use drives the package's continued maintenance. These are in-house applications rather than independent third-party adoptions; the shared affiliation is recorded in the Declaration of competing interest.

7.2 Reliability and maintenance

Correctness is guarded by the suite of more than 600 automated tests described in Sect. 4, run on every push and pull request through GitHub Actions. The single-engine architecture also limits duplicated code paths between the spatial and space-time solvers; just as important, improvements to numerical stability, caching, parallelization, and validation made once in the shared solver benefit every estimation method automatically, which aids maintainability, reproducibility, and future extension; the same shared interface is now exercised in production by the two downstream tools described in Sect. 7.1. Source code, release history, and the issue tracker for bug reports and feature requests are hosted publicly at https://github.com/ougx/krigekit.

7.3 Comparison with existing software

Table 5. *Feature availability through documented high-level workflows in the versions evaluated (PyKrige 1.7.3, GSTools 1.7.0, gstat 2.1.6, gstlearn 1.10.1, GSLIB). Entries reflect the tested high-level APIs and documentation, not an exhaustive statement of everything each code base can be made to do.*

https://doi.org/10.5194/egusphere-2026-5708
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

Feature	KrigeKit	PyKrige	GSTools	gstat (R)	GSLIB	gstlearn
Language / backend	Py + Fortran	Py + Cython	Py + Cython	R + C	Fortran	Py/R + C++
OpenMP parallel solve	Yes	No	Yes	No	No	Yes
Factorization cache	Yes	No	No	No	No	No
Co-kriging (LMC)	Yes	Not native	Not native	Yes	Yes	Yes
KED / external drift	Yes	Yes	Yes	Yes	Yes	Yes
Gradient constraints	Yes	Not native	Not native	Not native	Not native	Not native
Local moving neighborhood (nmax)	Yes	Yes	Not native	Yes	Yes	Yes
Per-variable search limits	Yes	Not native	Not native	Yes	Yes	Not native
SGSIM	Yes	Not native	Limited[e]	Yes	Yes	Limited[g]
Space-time	Yes	Not native	Limited[c]	Yes	Not native	Yes
Indicator kriging / SIS	Yes	Not native	Not native	Yes	Yes	Yes
Spatially varying anisotropy	Yes	Not native	Not native	Not native	Limited[d]	Yes
Multiplicative (product) structures	Yes	Not native	Not native	Limited[f]	Workaround	Not native
Normal-score / quantile transform	Yes	Not native	Limited[b]	Workaround[a]	Yes	Yes
Weight reuse	Yes	No	No	No	No	No

Entries describe each package's high-level public API at the versions tested in the Appendix A benchmark (PyKrige 1.7.3, GSTools 1.7.0, gstat 2.1.6, gstlearn 1.10.1, and the recompiled GSLIB). To avoid unfair comparison with packages that provide related functionality under different names, cells distinguish:

Yes — native, first-class support; Limited — partial or constrained support; Not native — not a built-in feature, though related functionality may exist under a different name or be assembled by composition; Workaround — achievable only through manual or external steps; No — absent with no close equivalent. The entries reflect our testing and documentation review and may understate capabilities reachable through lower-level interfaces or third-party extensions.

[a] gstat applies the normal-score transform with separate R functions, outside its kriging call. [b] GSTools provides Normalizer transforms (e.g. Box–Cox, Yeo–Johnson) but not a rank-based normal-score anamorphosis tied to the kriging workflow. [c] GSTools supplies spatio-temporal covariance models and random fields; space-time kriging is less developed than gstat's. [d] Standard GSLIB kt3d uses a single anisotropy; locally varying azimuth is available in specific programs and published extensions. [e] GSTools conditions fields by the randomization (spectral) method (CondSRF), not sequential Gaussian simulation. [f] gstat's product term appears only in its space-time

https://doi.org/10.5194/egusphere-2026-5708
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

product-sum model, not as a general spatial product covariance. [g] gstlearn conditions by the turning-bands method, not sequential Gaussian simulation.

The comparison shows KrigeKit matching the established packages on the common estimators while retaining several capabilities none of them provide: the factorization cache, gradient constraints, and stored-weight reuse are unique to KrigeKit in this set, and it is the only package with general multiplicative (product) covariances as a first-class feature. Among the five comparison packages, gstlearn is the most feature-complete — a compiled C++ library that also offers OpenMP-parallel solves, co-kriging, external drift, space-time and indicator kriging, non-stationary anisotropy, and Gaussian anamorphosis — but it conditions by turning bands rather than sequential simulation, caps the total co-kriging sample count rather than enforcing per-variable search limits, and, in our benchmark, its linear-scan neighbor search did not complete the full-source problem (Appendix A). gstat is comparably broad but tied to the R ecosystem and single-threaded; PyKrige and GSTools are lighter Python libraries centered on point kriging and random-field simulation respectively, without native co-kriging; and GSLIB remains feature-rich but file-driven and serial.

7.4 Limitations and future work

Sequential simulation is sequential by construction — each node conditions on the previously simulated nodes — so a realization cannot be parallelized over the grid. A multi-realization run (nsim > 1) produces all of its realizations within a single solve, but that solve executes in one thread: the engine disables OpenMP whenever simulation is requested, so simulation does not benefit from the multicore scaling that accelerates kriging. Because the realizations are mutually independent, parallelizing them across threads is a natural enhancement for future work. The spatially-varying-anisotropy mode is a locally stationary (moving-window) approximation rather than a globally valid nonstationary covariance. For the prescribed-orientation, channel-following continuity typical of standard environmental applications this is practically sufficient and highly interpretable: the anisotropy field is read directly from a mapped conceptual model, and each estimate uses a single, well-defined local ellipse, which is usually preferable to the smoother but opaquer output of a fully nonstationary kernel. Where the anisotropy must instead be inferred from the data and a provably valid global nonstationary covariance is required, benchmarking against the kernel-convolution model of Paciorek and Schervish (2006) on the same data is a useful direction for future work.

KrigeKit's parallelism is shared-memory only: OpenMP distributes the estimation grid across the cores of a single machine, but the engine has no distributed-memory (MPI) layer and so does not span the nodes of a cluster. For the grid sizes typical of environmental studies a single multi-core workstation is generally sufficient (Sect. 5), and a problem too large for one node's memory can still be partitioned into independently solved tiles. Because estimation blocks are mutually independent, the same decomposition that drives the OpenMP scaling would extend naturally to a distributed implementation, which we leave to future work.

https://doi.org/10.5194/egusphere-2026-5708
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

8. Conclusions

KrigeKit delivers a single, OpenMP-parallel engine for spatial and space-time geostatistics behind an accessible Python interface, combining the throughput of compiled Fortran with the convenience of the scientific-Python ecosystem. Its caching and weight-reuse mechanisms target the dominant cost of repeated dense solves, and its breadth — from co-kriging and KED to SGSIM, indicator simulation, space-time models, and spatially varying anisotropy — covers a wide range of environmental-modeling needs in one package. The Middle Republican River groundwater, California aquifer-texture, and spatially-varying-anisotropy case studies illustrate its applicability to real environmental problems — including multiplicative covariance, which captures seasonal correlation superimposed on a slowly decaying trend, and per-block anisotropy that follows curvilinear geological structure. Pre-built wheels, an executable example gallery spanning the full feature set, and a topic-organized user guide lower the barrier to adoption and keep the published workflows reproducible, supporting KrigeKit's use both as an end-user tool and, through its C API, as an embeddable engine within larger environmental-modeling pipelines.

Appendix A: Cross-package performance comparison

This appendix gives the full cross-package speed comparison summarized in Sect. 5: the benchmark configurations, the per-package wall-clock results, and the complete protocol needed to reproduce them.

The comparison benchmark interpolates a coarse-fraction field (primary), optionally co-kriged with airborne-EM resistivity (secondary), over the FourCounties Central Valley dataset using an exponential model (a = 9854 / 3809 / 60 m, azimuth 0) with a 30-neighbor search and a 20 km secondary search radius. Two configurations are reported (Table A1): a full-source case in which every tool receives the complete source data, and a clipped case representing a practical, externally pre-filtered workflow. Each tool's variogram, anisotropy, and neighborhood are matched after reconciling range conventions, and reported timings exclude file input/output. Hardware: Intel Core i7-13700K, 64 GB RAM, 24 logical cores, Windows; Python 3.12.12, R 4.3.3, GNU Fortran 16.1.0. The absolute timings are hardware- and implementation-specific; the observed relative performance applies to this benchmark configuration and should not be assumed to generalize to every dataset or package configuration.

Table A1. *Comparison-benchmark configurations (FourCounties 3-D aquifer texture; exponential model, nmax = 30).*

Case	Primary obs	Secondary obs	Targets	nmax
Full-source OK / CK	506,645	1,205,193	100,000	30
Clipped OK / CK	2,026	3,796	10,000	30

Table A2. *Wall-clock comparison on the 100,000-target full-source benchmark (complete 506,645 primary / 1,205,193 secondary observations supplied to every tool; runtime excludes file I/O, except the GSLIB executables, which expose only combined wall time). KrigeKit is shown at 1 and 24 threads; the others are single-threaded. GSLIB was recompiled with*

https://doi.org/10.5194/egusphere-2026-5708
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

gfortran -O3 and run with a standard 3×-range search radius. ✗ did not complete — memory exhaustion or no moving-neighborhood search; n/a = no co-kriging.

Package	Backend	Threads	OK (s)	CK (s)
KrigeKit	Fortran + OpenMP	1	1.34	8.51
KrigeKit	Fortran + OpenMP	24	0.30	1.18
gstat (R)	C	1	18.1	79.1
GSLIB	Fortran -O3	1	245	978
PyKrige	NumPy/Cython	1	✗	n/a
GSTools	Cython	—	✗	n/a

At one thread KrigeKit was 13.5× faster than gstat for OK and 9.3× for CK; with 24 threads, 60× and 67×. Recompiled with gfortran -O3 and run with a standard 3×-range search radius, the legacy GSLIB executables complete the full source in 4.1 min (OK) and 16.3 min (CK) — still an order of magnitude slower than gstat and far slower than parallel KrigeKit, but they complete. PyKrige exhausts memory building global source-pair structures before applying a local neighborhood, and GSTools has no native moving-neighborhood search, so neither finishes — a neighborhood-count argument does not by itself imply scalable full-source execution. Estimates agree to the precision in Table 3. A fuller comparison, including the newer gstlearn and TFInterpy packages, is provided in the supplementary benchmark.

A1 Benchmark protocol and reproducibility

For full reproducibility, we state the benchmark protocol in detail. All runs used one workstation — Intel Core i7-13700K, 64 GB RAM, 24 logical cores, Windows — on 19 June 2026. KrigeKit 0.3.2 was built with GNU Fortran 16.1.0 at -O3 and links a bundled reference LAPACK/BLAS compiled with the same toolchain (no machine-tuned BLAS); its Python interface was benchmarked with Python 3.12.12. The comparison tools were R gstat 2.1.6 (R 4.3.3, with R's default reference BLAS/LAPACK) and GSLIB kt3d / cokb3d recompiled from source with GNU Fortran 16.1.0 at -O3; the supplementary comparison additionally used PyKrige 1.7.3, GSTools 1.7.0, gstlearn 1.10.1, and TFInterpy 1.1.3 on the same Python 3.12.12 interpreter. The absolute seconds are hardware- and implementation-specific; the observed relative performance applies to this benchmark configuration and should not be assumed to generalize to every dataset or package configuration.

Every tool solved the identical estimation problem. All received the same primary — and, for co-kriging, secondary — observations, the same exponential covariance model (practical ranges a = 9854 / 3809 / 60 m, azimuth and dip 0°; OK sill 0.20 with zero nugget; CK structured sills 0.20 / 0.15 / 0.12 with a 0.03 secondary nugget), the same 30-nearest moving neighborhood (n_{max} = 30 per variable, 20 km secondary search radius), and the same target grid. Two convention reconciliations ensured the same model was solved everywhere: GSTools and gstat parameterize the exponential by its e-folding scale, so their range input was set to $a/3$; and gstat's legal-LMC test rejects an exactly singular nugget matrix, so the primary nugget was set to 10^{-10} — a numerical jitter, not a different model. Tools without an intrinsic anisotropic search

https://doi.org/10.5194/egusphere-2026-5708
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

(PyKrige, GSTools, gstat) received coordinates pre-transformed to the isotropic metric, so their neighbor geometry matches the anisotropic search used natively by KrigeKit and GSLIB. For strict numerical parity KrigeKit's optional output bounds and negative-weight correction were disabled and standard (not the optional Isaaks–Srivastava) co-kriging was used.

The reported runtime measures the same operations for every package: data filtering and transformation, object construction, model and search setup, prediction, and result retrieval. File reading and result writing are excluded for all tools. The one exception is the GSLIB executables, which expose only a single combined wall time that necessarily includes their text-file input and formatted output; those values are footnoted as executable wall time and, if anything, overstate GSLIB's compute time relative to the I/O-excluded figures for the other tools.

The GSLIB configuration deserves particular detail, because its default parameterization is misleading. The widely distributed GSLIB binaries date from 2000 and were produced by a late-1990s compiler; run as distributed they completed only a few thousand targets in five minutes, projecting to roughly 1.5 h (OK) and 6.7 h (CK). Two configuration artifacts — not the GSLIB algorithm — caused this. First, the binaries were 24 years old. Second, the distributed parameter files set a search radius of 10^8, about 10,000× the variogram range, which defeats kt3d's super-block index and forces an $O(n)$ scan of every observation per target (the same failure mode seen in two other packages). We therefore recompiled kt3d and cokb3d from the original GSLIB source with GNU Fortran 16.1.0 at -O3 — the only source edit being to comment out a Compaq-Fortran-only 'use msflib' line ('getarg' being a gfortran intrinsic) — and set a standard search radius of three times the range (30000 / 11595 / 182.7 m). Together these gave roughly a 20× speed-up, after which kt3d / cokb3d complete the full source in 4.1 min (OK) and 16.3 min (CK). Bounding the radius changes only a small number of cells versus the unbounded search (estimate RMSE 2.7×10^{-3}, a dense vertical-tie effect, identical at 30 km and 50 km radii), so the bounded search is not what makes GSLIB competitive — it merely removes a pathological parameterization. Even so, GSLIB runs one executable per case with process start-up and text I/O and remains serial, keeping it far behind gstat and parallel KrigeKit. The recompilation commands and parameter files are in the supplementary benchmark.

Where a tool does not complete, the cause is identified rather than left blank. PyKrige exposes a neighbor-count argument, but its OrdinaryKriging3D constructor first computes pairwise distances and leave-one-out diagnostics over all supplied observations; at 506,645 primary observations those vectors alone need about 1.9 TiB, so it fails on memory before any local neighborhood is applied — a resource guard, not a timeout. GSTools has no native moving-neighborhood (n_{max}) interface and forms a single global conditioning system, infeasible at this observation count. In the supplementary set, gstlearn 1.10.1 is a compiled C++ library with native 3-D moving neighborhoods, but its search scans $O(n)$ per target and its ball-search option did not accelerate it, so full-source OK did not finish within the 600 s censoring window; its heterotopic co-kriging neighborhood also uses one total sample cap rather than separate per-variable limits, so that row is a capability note, not a parity result. PyKrige, GSTools, and TFInterpy provide no general two-variable geostatistical co-kriging. These four causes — memory exhaustion, censoring timeout, missing moving-neighborhood search, and absent co-kriging API — are exactly what the ✗ and n/a entries in Table A2 denote.

https://doi.org/10.5194/egusphere-2026-5708
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.

Code and data availability

KrigeKit is free and open-source software released under the MIT license. The exact version described in this paper (v0.3.2) is archived on Zenodo (Ou et al., 2026; DOI: https://doi.org/10.5281/zenodo.22691837). Ongoing development is hosted at https://github.com/ougx/krigekit, release wheels are distributed on PyPI (install with 'pip install krigekit'), and documentation is available at https://krigekit.readthedocs.io. The package comprises an OpenMP-parallel Fortran compute engine with a Python (ctypes) interface and depends on numpy, pandas, scipy, matplotlib, and scikit-learn.

The illustrative datasets bundled with the source repository (under test_data/) are included in the Zenodo archive. The Middle Republican groundwater-level case study (Sect. 6.1) uses records from the University of Nebraska–Lincoln Conservation and Survey Division statewide groundwater-level database (University of Nebraska–Lincoln Conservation and Survey Division, n.d.) (https://csd.unl.edu/water/groundwater/NebGW_Levels). The California aquifer-texture case studies (Sections 6.2 and 6.3) use the California Department of Water Resources statewide Airborne Electromagnetic (AEM) Surveys (California Department of Water Resources, n.d.), distributed openly through the California Natural Resources Agency open-data portal (https://data.cnra.ca.gov/dataset/aem), together with public drillers' well-completion lithology logs compiled by the same program.

Author contributions

Gengxin Ou: Conceptualization, Methodology, Software, Writing – original draft, Writing – review & editing. Vivek Bedekar: Conceptualization, Validation, Writing – review & editing. Matt Tonkin: Conceptualization, Writing – review & editing. Gilbert Barth: Conceptualization, Writing – review & editing. Marinko Karanovic: Software, Validation, Writing – review & editing.

Competing interests

The authors are affiliated with S.S. Papadopulos & Associates, Inc., which develops KrigeKit and the downstream tools (Texture2Par, Groundwater Desktop) referenced in Sect. 7.1. Beyond this affiliation, the authors declare that they have no competing interests.

Acknowledgements

The authors thank the University of Nebraska–Lincoln Conservation and Survey Division and the California Department of Water Resources for providing the open datasets used in the case studies.

During preparation of this manuscript, the authors used large language models to suggest language revisions, check consistency and compliance with the journal template, and assist with the assembly and formatting of the document.

https://doi.org/10.5194/egusphere-2026-5708
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.



Financial support

This research received no external grant funding; KrigeKit was developed with internal technical-development time provided by S.S. Papadopulos & Associates, Inc.

References

Almeida, A. S. and Journel, A. G.: Joint simulation of multiple variables with a Markov-type coregionalization model, Math. Geol., 26, 565–588, https://doi.org/10.1007/BF02089242, 1994.

California Department of Water Resources: Statewide Airborne Electromagnetic (AEM) Surveys, California Natural Resources Agency Open Data Portal [data set], https://data.cnra.ca.gov/dataset/aem (last access: 18 September 2026), 2026.

Chilès, J.-P. and Delfiner, P.: Geostatistics: Modeling Spatial Uncertainty, 2nd edn., Wiley, Hoboken, New Jersey, USA, https://doi.org/10.1002/9781118136188, 2012.

Cressie, N. A. C.: Statistics for Spatial Data, revised edn., Wiley, New York, USA, https://doi.org/10.1002/9781119115151, 1993.

De Iaco, S., Myers, D. E., and Posa, D.: Space–time analysis using a general product–sum model, Stat. Probabil. Lett., 52, 21–28, https://doi.org/10.1016/S0167-7152(00)00200-5, 2001.

Delhomme, J. P.: Spatial variability and uncertainty in groundwater flow parameters: a geostatistical approach, Water Resour. Res., 15, 269–280, https://doi.org/10.1029/WR015i002p00269, 1979.

Deutsch, C. V. and Journel, A. G.: GSLIB: Geostatistical Software Library and User's Guide, 2nd edn., Oxford University Press, New York, USA, http://claytonvdeutsch.com/wp-content/uploads/2019/03/GSLIB-Book-Second-Edition.pdf, 1998.

Goovaerts, P.: Geostatistics for Natural Resources Evaluation, Oxford University Press, New York, USA, https://doi.org/10.1093/oso/9780195115383.001.0001, 1997.

Gräler, B., Pebesma, E., and Heuvelink, G.: Spatio-temporal interpolation using gstat, R J., 8, 204–218, https://doi.org/10.32614/RJ-2016-014, 2016.

Horn, R. A. and Johnson, C. R.: Matrix Analysis, 2nd edn., Cambridge University Press, Cambridge, UK, https://doi.org/10.1017/CBO9781139020411, 2013.

Journel, A. G.: Nonparametric estimation of spatial distributions, Math. Geol., 15, 445–468, https://doi.org/10.1007/BF01031292, 1983.

https://doi.org/10.5194/egusphere-2026-5708
Preprint. Discussion started: 3 November 2026
© Author(s) 2026. CC BY 4.0 License.



Kitanidis, P. K.: Introduction to Geostatistics: Applications in Hydrogeology, Cambridge University Press, Cambridge, UK, https://doi.org/10.1017/CBO9780511626166, 1997.

Kyriakidis, P. C. and Journel, A. G.: Geostatistical space–time models: a review, Math. Geol., 31, 651–684, https://doi.org/10.1023/A:1007528426688, 1999.

Müller, S., Schüler, L., Zech, A., and Heße, F.: GSTools v1.3: a toolbox for geostatistical modelling in Python, Geosci. Model Dev., 15, 3161–3182, https://doi.org/10.5194/gmd-15-3161-2022, 2022.

Murphy, B., Yurchak, R., and Müller, S.: GeoStat-Framework/PyKrige: v1.7.3, Zenodo [Computer software], https://doi.org/10.5281/zenodo.17372225, 2025.

Ou, G., Bedekar, V., Tonkin, M., Barth, G., and Karanovic, M.: KrigeKit v0.3.2, Zenodo [code], https://doi.org/10.5281/zenodo.22691837, 2026.

Paciorek, C. J. and Schervish, M. J.: Spatial modelling using a new class of nonstationary covariance functions, Environmetrics, 17, 483–506, https://doi.org/10.1002/env.785, 2006.

Pebesma, E. J.: Multivariable geostatistics in S: the gstat package, Comput. Geosci., 30, 683–691, https://doi.org/10.1016/j.cageo.2004.03.012, 2004.

Risser, M. D. and Calder, C. A.: Local likelihood estimation for covariance functions with spatially-varying parameters: the convoSPAT package for R, J. Stat. Softw., 81, 1–32, https://doi.org/10.18637/jss.v081.i14, 2017.

S.S. Papadopulos & Associates: Groundwater Desktop, https://www.groundwaterdesktop.com/ (last access: 18 September 2026), 2026.

Scantlebury, L., Bedekar, V., Tonkin, M. J., Karanovic, M., and Harter, T.: Texture2Par: a texture-driven tool for estimating subsurface hydraulic properties, Environ. Modell. Softw., 187, 106372, https://doi.org/10.1016/j.envsoft.2025.106372, 2025.

Schilling, O. S., Partington, D. J., Doherty, J., Kipfer, R., Hunkeler, D., and Brunner, P.: Buried paleo-channel detection with a groundwater model, tracer-based observations, and spatially varying, preferred anisotropy pilot point calibration, Geophys. Res. Lett., 49, e2022GL098944, https://doi.org/10.1029/2022GL098944, 2022.

University of Nebraska–Lincoln Conservation and Survey Division: Nebraska Groundwater-Level Database [data set], https://csd.unl.edu/water/groundwater/NebGW_Levels (last access: 18 September 2026), 2026.

Wackernagel, H.: Multivariate Geostatistics: An Introduction with Applications, 3rd edn., Springer, Berlin, Germany, https://doi.org/10.1007/978-3-662-05294-5, 2003.