Supplement of Hydrol. Earth Syst. Sci., 29, 4251–4279, 2025
https://doi.org/10.5194/hess-29-4251-2025-supplement

Hydrology and Earth System Sciences
Open Access
EGU

Supplement of

Enhancing inverse modeling in groundwater systems through machine learning: a comprehensive comparative study

Junjun Chen et al.

Correspondence to: Zhenxue Dai (dzx@jlu.edu.cn) and Shangxian Yin (yinshx03@126.com)

Contents of this file

Implementation procedures and theoretical foundations for the four metaheuristic algorithms (**MHA S1 to MHA S4**).

Table S1 to Table S3

Figures S1 to S44

Introduction

MHA S1 Particle swarm optimization algorithm (PSO)

MHA S2 Genetic algorithm (GA)

MHA S3 Simulated Annealing (SA)

MHA S4 Differential evolution (DE)

Table S1 is the optimal hyperparameters for MSVR by four metaheuristic algorithms

Table S2 and **S3** are $\mathrm{RMSE}_{(\mathrm{All})}$ and R^2_{All} values of FC-DNN with different number of hidden layers

Table S4 is the RMSE values of estimated log-permeability fields for the four metaheuristic algorithms and the TNNA algorithm under Scenario 1-4.

Fig.S1. to **Fig.S2** are detail architectures of LeNet and ResNet.

Fig.S3. to **Fig.S6.** are pair-wise comparisons for four surrogate modeling methods.

Fig.S7. to **Fig.S10.** are spatial distributions of log-permeability field estimation results and absolute errors by four metaheuristic algorithms for Scenarios 1~4

Fig.S11. to **Fig.S14.** are spatial distributions of log-permeability field estimation results and absolute errors by the TNNA method for Scenarios 1~4

Fig.S15. to **Fig.S44.** are spatial distributions of calibrated numerical simulation results and absolute errors for solute concentrations using the TNNA algorithm and four metaheuristic algorithms (from day 2 to day 60 recorded every two days).

MHA S1 to MHA S4 Metaheuristic algorithms

Here, we briefly introduce the main steps of the four metaheuristic algorithms used in this study. In the following methods, the symbol $\boldsymbol{m}$ denotes the model parameters to be optimized, and $\boldsymbol{F}_{obj}(\cdot)$ represents the objective function for inversion.

MHA S1 Particle swarm optimization algorithm (PSO)

PSO is a population-based intelligent optimization algorithm inspired by the foraging behavior of birds (Eberhart and Kennedy, 1995). It is realized through the following steps:

Step 1: Initialize a population with N_{Me} particles of a N_m-dimensional space $\boldsymbol{m}=(\boldsymbol{m}_1, \boldsymbol{m}_2,\ldots,\boldsymbol{m}_{N_{Me}})$. For an arbitrary particle $\boldsymbol{m}_i$, denote its position, velocity and best position at the k-th iteration as $\boldsymbol{m}_i^k=(m_{i1}^k,\ldots,m_{iN_m}^k)$, $V_i^k=(v_{i1}^k,\ldots,v_{iN_m}^k)$, and $P_i^k=(p_{i1}^k,\ldots,p_{im}^k)$, respectively.

Step 2: Calculate the best solution for each particle $\boldsymbol{m}_{i,pbest}^k$ according to equation (S1):

$$\boldsymbol{m}_{i,pbest}^k=\begin{cases}\boldsymbol{m}_{i,pbest}^{k-1}, & \boldsymbol{F}_{obj}(\boldsymbol{m}_i^k)\geq\boldsymbol{F}_{obj}(\boldsymbol{m}_{i,pbest}^{k-1})\\ \boldsymbol{m}_i^k, & \boldsymbol{F}_{obj}(\boldsymbol{m}_i^k)<\boldsymbol{F}_{obj}(\boldsymbol{m}_{i,pbest}^{k-1})\end{cases} \tag{S1}$$

where $\boldsymbol{F}_{obj}(\cdot)$ is the objective function, also known as the fitness function.

Step 3: Calculate the best position of the population $\boldsymbol{m}_{i,gbest}^k$ according to equation (S2).

$$\boldsymbol{m}_{gbest}^k=\text{argmin}\{\boldsymbol{F}_{obj}(\boldsymbol{m}_1^k),\ldots,\boldsymbol{F}_{obj}(\boldsymbol{m}_{N_{Me}}^k)\} \tag{S2}$$

Step 4: Updated the velocity and position for each particle (i) according to equations (S3) and (S4):

$$V_i^{k+1}=w_i V_i^k+r_1c_1(\boldsymbol{m}_{i,pbest}^k-\boldsymbol{m}_i^k)+r_2c_2(\boldsymbol{m}_{gbest}^k-\boldsymbol{m}_i^k) \tag{S3}$$

$$\boldsymbol{m}_i^{k+1}=\boldsymbol{m}_i^k+V_i^{k+1} \tag{S4}$$

where c_1 and c_2 are learning parameters, generally taken as two equal non-negative constants and are set to 0.5 and 0.1 here; r_1 and r_2 are two random values within the range of [0, 1]; w_i is the inertia weight and set to 0.8 for this study.

MHA S2 Genetic algorithm (GA)

GA is initially introduced by Holland John (1975). It draws inspiration from natural evolution and genetics, where individuals within a population are selected or eliminated based on their adaptability to the environment. The GA is realized through the following steps:

Step 1: Generate an initial population $\boldsymbol{m}=(\boldsymbol{m}_1, \boldsymbol{m}_2,\ldots,\boldsymbol{m}_{N_{Me}})$ randomly.

Step 2: Perform binary encoding on all individuals in the population X to obtain their respective binary symbol strings. These binary symbol strings are called chromosomes, and each value (“0” or “1”) on a symbol string is called a gene.

Step 3: Crossover: Perform crossover operations on randomly paired combinations of individuals in X. The essence of crossover is to exchange some values in the symbol strings of a pair of individuals.

Step 4: Mutation: Perform mutation operations on some random individuals in $\boldsymbol{m}$ by changing some values of their symbol strings.

Step 5: Selection: Perform selection operations based on the fitness values of each individual ($\boldsymbol{m}_i$) to generate the next generation population. This step is realized through the roulette wheel selection method, where individuals with higher fitness values are more likely to be selected.

Step 6: Determine whether the current results satisfy the iteration termination condition. If not, return to step (2); otherwise, output the optimal individual in the current population as the final result.

MHA S3 Simulated Annealing (SA)

The SA method is a Monte Carlo-based stochastic optimization algorithm proposed by Metropolis et al. (1953) and initially applied to combinational optimization problems by Kirkpatrick et al. (1983). The realization steps for SA method are as follows:

Step 1: Set the starting temperature as T_0 and draw an initial optimal solution as $\boldsymbol{m}_i$.

Step 2: Generate a new solution $\boldsymbol{m}_j$ X_j from the neighborhood of the current solution $\boldsymbol{m}_i$.

Step 3: Calculate the objective function values $\boldsymbol{F}_{obj}(\boldsymbol{m}_i)$ and $\boldsymbol{F}_{obj}(\boldsymbol{m}_j)$. If $\boldsymbol{F}_{obj}(\boldsymbol{m}_i) \geq \boldsymbol{F}_{obj}(\boldsymbol{m}_j)$, then $\boldsymbol{m}_j$ becomes the current solution $\boldsymbol{m}_i$; otherwise, $\boldsymbol{m}_j$ becomes the current solution $\boldsymbol{m}_i$ with a probability calculated as:

$$P(\boldsymbol{m}_i \rightarrow \boldsymbol{m}_j) = \exp\left(\frac{\boldsymbol{F}_{obj}(\boldsymbol{m}_i) - \boldsymbol{F}_{obj}(\boldsymbol{m}_j)}{a^t T_0}\right) \tag{S5}$$

where t is the current time and a is the temperature decay constant.

Step 4: Under the current temperature conditions, repeat steps (2) and (3) until reaching the predetermined number of internal iterations. Then, update the temperature and time as follows: set $t=t+1$ and $T_t=a_t T_0$, then proceed to the next step.

Step 5: Return to step (2) and continue the iteration according to the new temperature (T_t) and time (t) until the termination conditions are met. The iterations in this step can be considered outer iterations, distinguished from step (4).

MHA S4 Differential evolution (DE)

DE is another evolutionary algorithm proposed by Storn and Price (1997). Similar to GA, DE also employs mutation, crossover and selection operators, but they update uncertain model parameters in different ways (Tran et al., 2022). The detailed steps for realizing DE are as follows:

Step 1: Generate the initial population $\boldsymbol{m}=(\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_{N_{Me}})$ randomly.

Step 2: Perform encoding for each individual in X. The encoding method used in DE is floating-point real encoding, rather than binary encoding used in GA.

Step 3: Mutation**:** After completing individual encoding, DE performs mutation operations to generate new individuals according to equation (S6):

$$\boldsymbol{m}_{perturbed}^{(g+1)} = \boldsymbol{m}_{rand_1}^{(g)} + F_{DE} \times (\boldsymbol{m}_{rand_2}^{(g)} - \boldsymbol{m}_{rand_3}^{(g)}) \tag{S6}$$

where $\boldsymbol{m}_{rand_1}$, $\boldsymbol{m}_{rand_2}$ and $\boldsymbol{m}_{rand_3}$ are randomly selected individuals among the candidate solutions of the current population and must be different from each other. F_{DE} is a scaling parameter within the range of [0,1], controlling differential variations. g represents the sequence number of iterations.

Step 4: Crossover: Perform crossover operations to generate the trial vector by combining the mutant and target vectors. The formula for this step is as follows:

$$\boldsymbol{m}_j^{(g+1,\mathrm{trail})} = \begin{cases} \boldsymbol{m}_{perturbed,\,j}^{(g+1)} & \text{if } P_j \leq CR \\ \boldsymbol{m}_j^{(g)} & \text{if } P_j > CR \end{cases} \tag{S7}$$

where P_j is a random number in the range of [0,1], CR is the crossover rate. If some variables of the trial vector have the same values, keep one of them and reset the others with random integer numbers in the range [1, D].

Step 5: Selection: Evaluate the objective function $\boldsymbol{F}_{obj}(\cdot)$ for both the new trail vector $\boldsymbol{m}_j^{(g+1,\mathrm{trail})}$ and the original vector $\boldsymbol{m}_j^{(g)}$. The candidate solution replaces its parent only if it yields a better (lower) objective function value.

Step 6: Return to step 3 until the convergence criteria are met.

Table S1 Optimal hyperparameters for MSVR by four metaheuristic algorithms

Training data number	Optimal algorithms	C	ε	σ	MSE
200	GA	18.640	6.0117E-03	0.398	6.2652E-02
	DE	**27.526**	**4.8503E-03**	**0.391**	**6.2498E-02**
	PSO	**27.526**	**4.8503E-03**	**0.391**	**6.2498E-02**
	SA	35.533	8.3451E-06	0.334	6.2499E-02
500	GA	54.278	4.9071E-03	0.509	4.9246E-02
	DE	39.979	3.0950E-03	0.867	4.9729E-02
	PSO	48.596	3.5939E-03	0.706	4.9215E-02
	SA	**32.241**	**5.5964E-03**	**0.615**	**4.8987E-02**
1000	**GA**	**23.296**	**4.2424E-03**	**0.724**	**4.3391E-02**
	DE	40.680	3.9406E-03	0.585	4.3556E-02
	PSO	25.317	6.1069E-03	0.820	4.3510E-02
	SA	71.104	4.0023E-05	0.561	4.3777E-02
2000	GA	61.888	1.1828E-03	0.918	3.5188E-02
	DE	53.579	1.6516E-03	0.964	3.5137E-02
	PSO	**50.431**	**9.4148E-04**	**0.921**	**3.5120E-02**
	SA	50.307	9.0781E-03	1.033	3.5265E-02

Note: The rows in bold represent the optimal hyperparameter configurations corresponding to the smallest MSE values.

Table S2 $RMSE_{(All)}$ values of FC-DNN with different number of hidden layers

Training data number	Hidden layer number						
	1	**2**	**3**	**4**	**5**	**6**	**7**
200	0.07588	**0.05882**	0.06870	0.17916	0.16125	0.13690	0.13340
500	0.07050	0.04308	0.03788	**0.03786**	0.05824	0.09567	0.10229
1000	0.05118	0.03571	**0.02703**	0.02732	0.02866	0.04213	0.07825
2000	0.03936	0.02944	**0.02090**	0.02168	0.02580	0.03064	0.06887

Note: The bold values represent the smallest MSE values among the considered seven hidden layer numbers.

Table S3 R^2_{All} values of FC-DNN with different number of hidden layers

Training data number	Hidden layer number						
	1	**2**	**3**	**4**	**5**	**6**	**7**
200	0.94140	**0.96479**	0.95197	0.67332	0.73539	0.80926	0.81890
500	0.94942	0.98111	0.98540	**0.98541**	0.96548	0.90685	0.89351
1000	0.97334	0.98703	**0.99256**	0.99240	0.99164	0.98194	0.93768
2000	0.98424	0.99118	**0.99555**	0.99522	0.99323	0.99045	0.95173

Note: The bold values represent the largest R^2_{All} values among the considered seven hidden layer numbers.

Table S4. RMSE values of estimated log-permeability fields for the four metaheuristic algorithms and the TNNA algorithm under Scenario 1-4.

Scenarios	**Metaheuristic algorithms**					**TNNA**	
		GA	**DE**	**PSO**	**SA**		
Scenario 1	N_{PC}=100	0.7844	0.5984	0.9423	0.7720	epoch=200	0.4895
	N_{PC}=500	0.8246	0.7639	0.6379	0.8980	epoch=1000	0.4748
	N_{PC}=1000	0.6659	0.6391	0.7127	0.8012		
Scenario 2	N_{PC}=100	0.9554	0.5223	0.8785	0.6987	epoch=200	0.4317
	N_{PC}=500	0.6164	0.4925	1.0293	1.1549	epoch=1000	0.4271
	N_{PC}=1000	0.5389	0.5322	0.9686	0.6288		
Scenario 3	N_{PC}=100	0.5386	0.3892	0.5486	0.5647	epoch=200	0.3161
	N_{PC}=500	0.4339	0.4271	0.5762	0.5714	epoch=1000	0.2970
	N_{PC}=1000	0.4060	0.5042	0.6295	0.5558		
Scenario 4	N_{PC}=100	0.4436	0.3841	0.5723	0.6459	epoch=200	0.2749
	N_{PC}=500	0.4265	0.3971	0.3770	0.5654	epoch=1000	0.2328
	N_{PC}=1000	0.3653	0.3459	0.5367	0.5033		

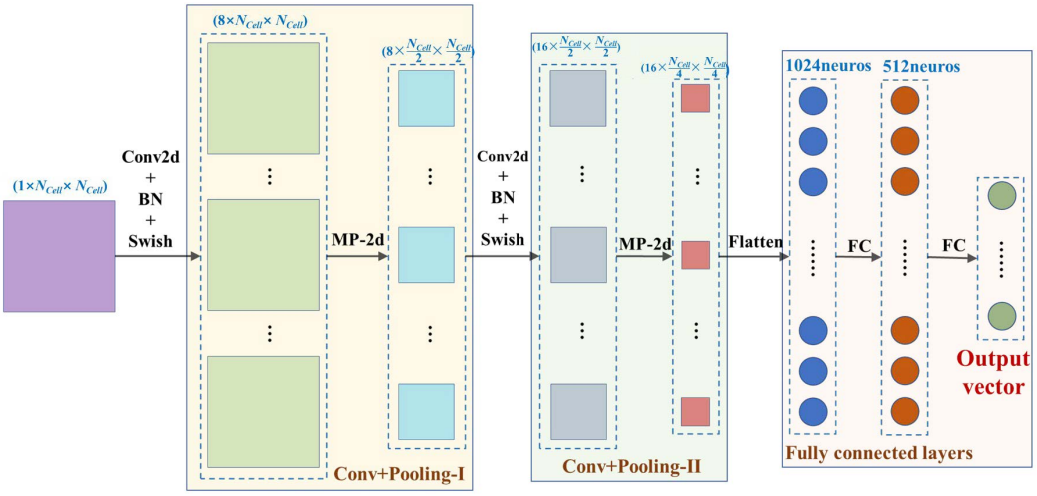

Fig.S1. Detailed architecture of a LeNet based CNN. The input matrix data ($1\times N_{Cell}\times N_{Cell}$) are obtained according to Figure 2(c) and subjected to feature extraction through a sequence of two convolutional and pooling layers, subsequently connected to the output layer using a flatten layer and two fully connected layers.

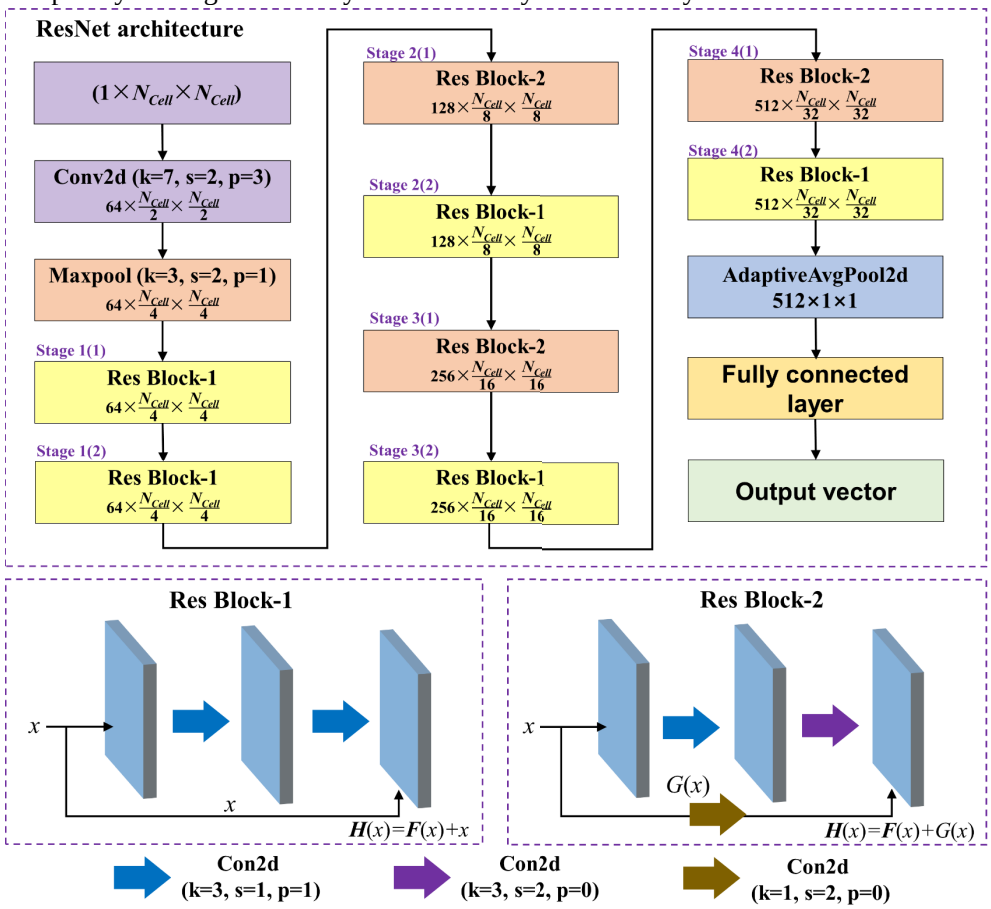

Fig.S2. Detailed architecture of a ResNet based CNN. The input matrix data ($1\times N_{Cell}\times N_{Cell}$) are obtained according to Figure 2(c). "Res Block-1" and "Res Block-2" are two different types of residual blocks used in this ResNet. Eight residual blocks in four stages are designed in this ResNet. "Stage *i* (*j*)" represents the *j*th residual block used in stage *i*. Note: In Stage 4, when dimensions cannot be evenly divided, the results are rounded up to the nearest integer (e.g., 5/2 is rounded up to 3).

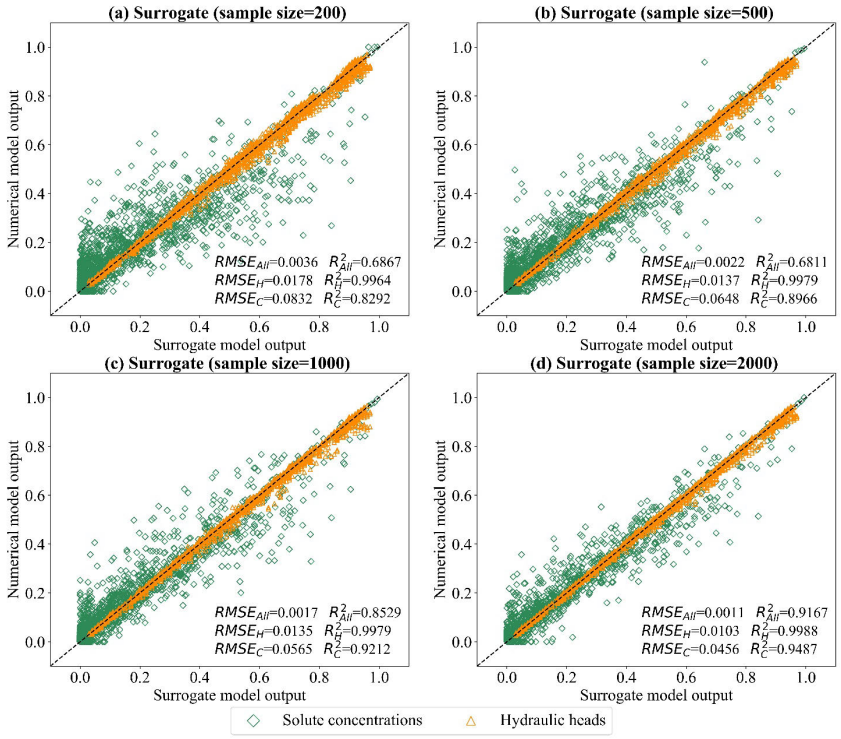

Fig.S3. Performance of MSVR based surrogate models for the solute concentration and hydraulic head prediction. (a~d) are pair-wise comparisons based on surrogate models trained by 200, 500, 1000, and 2000 training samples, respectively.

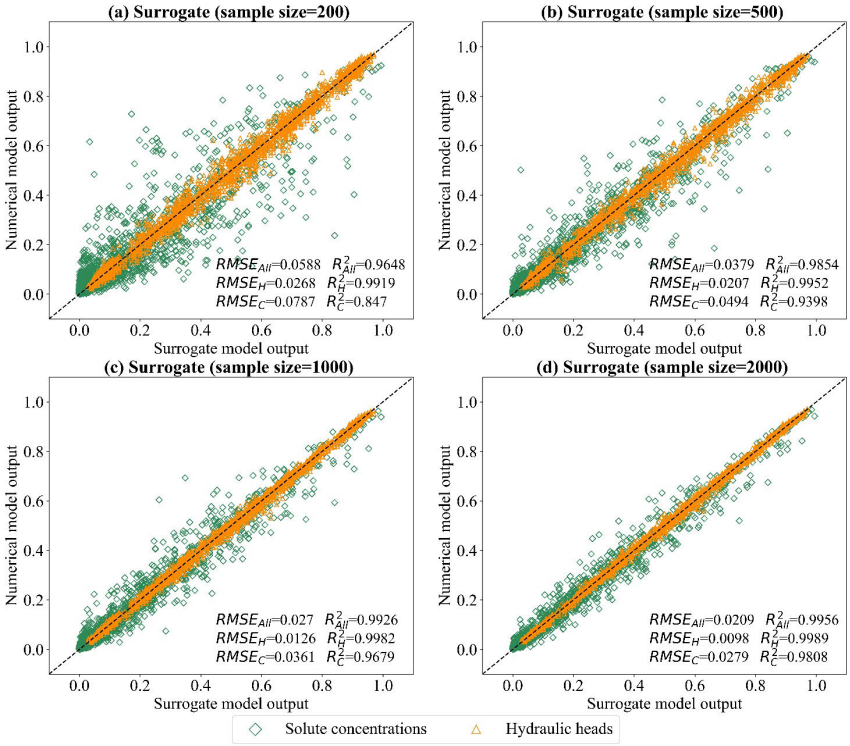

Fig.S4. Performance of FC-DNN based surrogate models for the solute concentration and hydraulic head prediction. (a~d) are pair-wise comparisons based on surrogate models trained by 200, 500, 1000, and 2000 training samples, respectively.

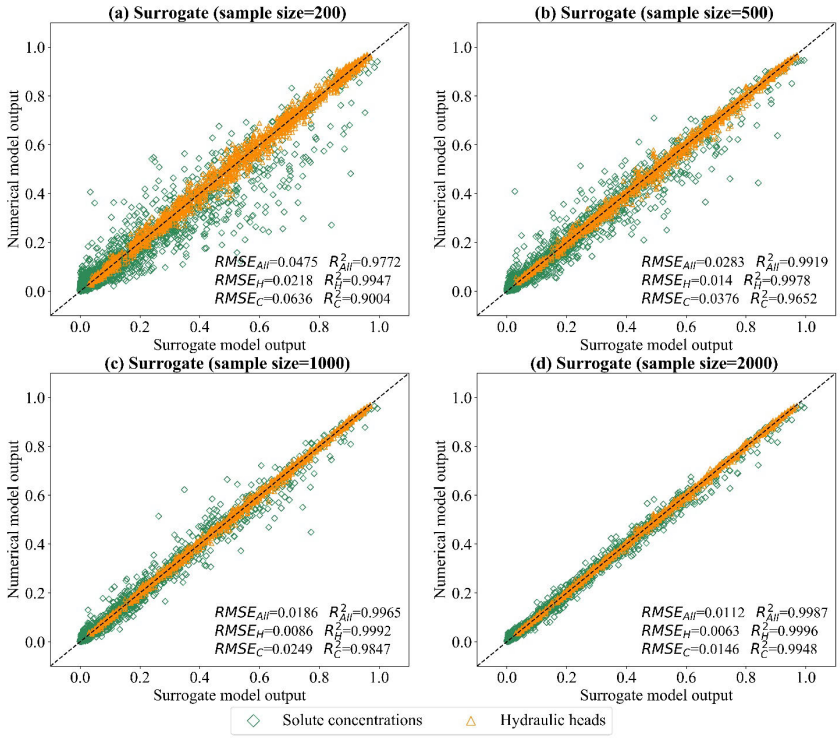

Fig.S5. Performance of LeNet CNN based surrogate models for the solute concentration and hydraulic head prediction. (a~d) are pair-wise comparisons based on surrogate models trained by 200, 500, 1000, and 2000 training samples, respectively.

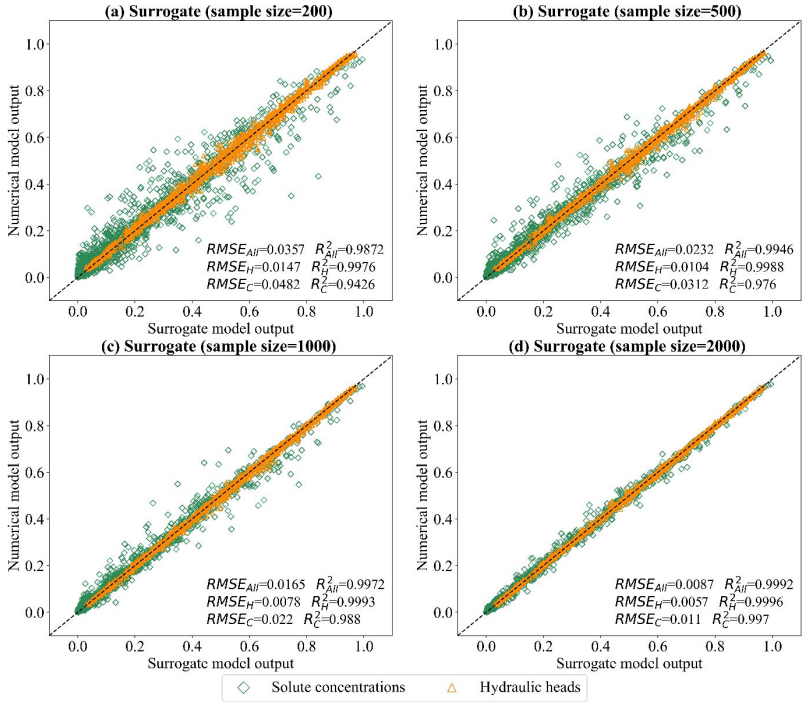

Fig.S6. Performance of ResNet CNN based surrogate models for the solute concentration and hydraulic head prediction. (a~d) are pair-wise comparisons based on surrogate models trained by 200, 500, 1000, and 2000 training samples, respectively.

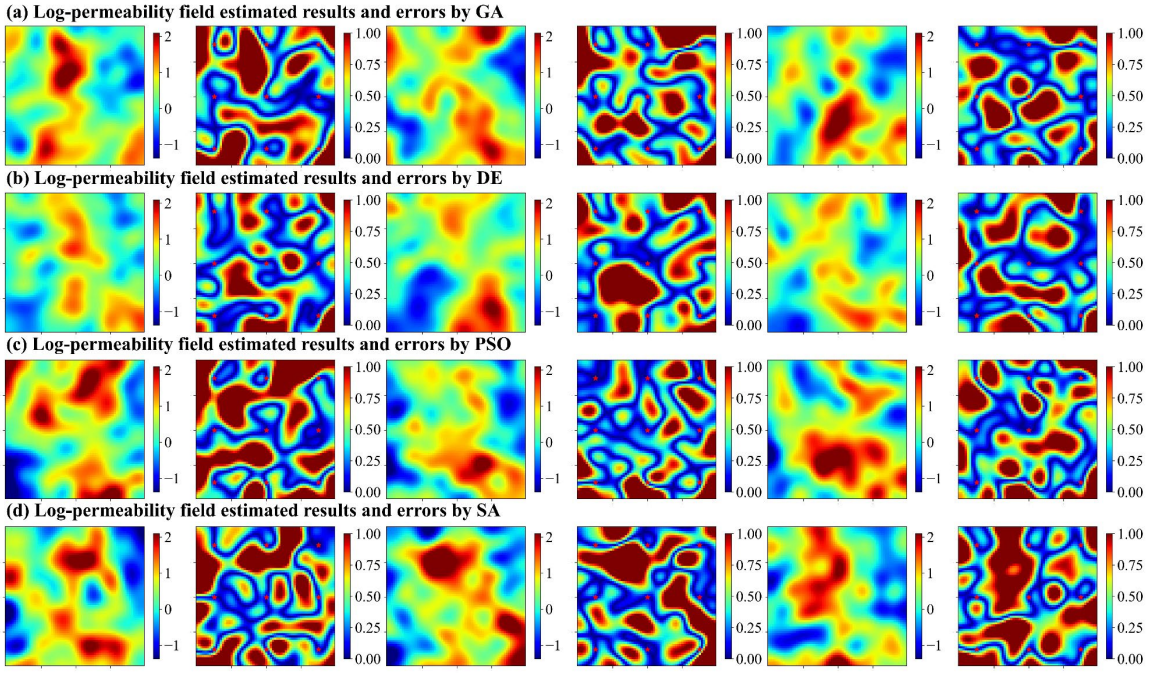

Fig.S7 Spatial distributions of log-permeability field estimation results (row 1, 3, and 5 for N_{PC}=100, 500, and 1000, respectively) and absolute errors (row 2, 4, and 6 for N_{PC}=100, 500, and 1000, respectively) for Scenario 1, achieved by four metaheuristic algorithms.

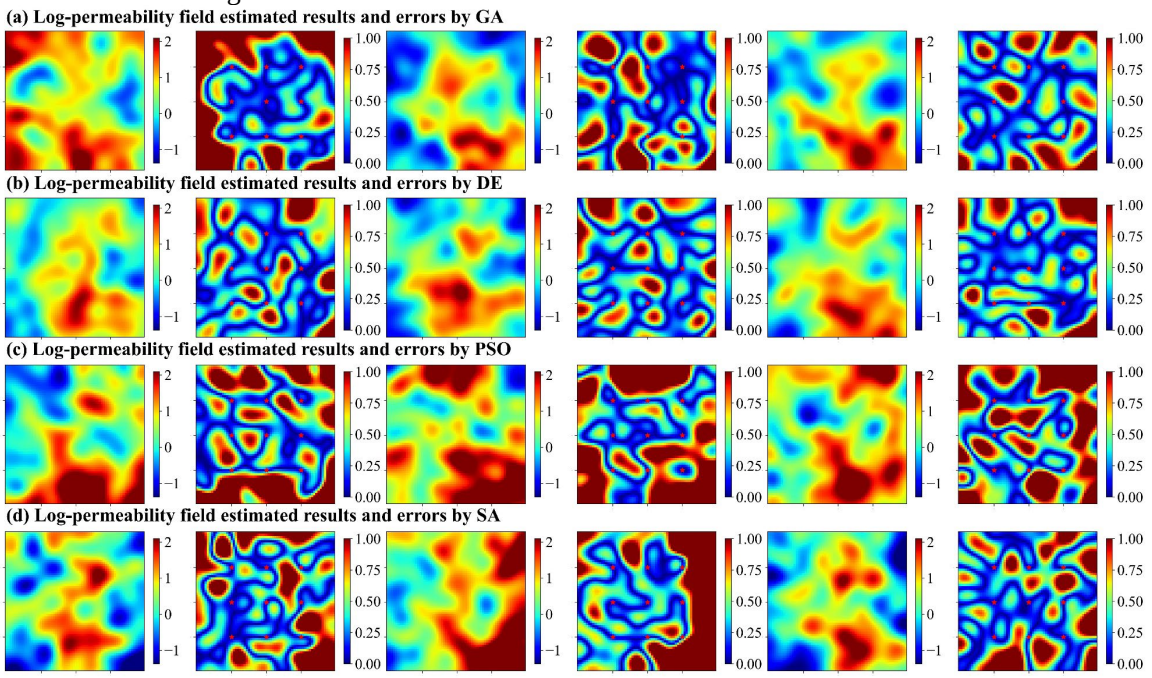

Fig.S8 Spatial distributions of log-permeability field estimation results (row 1, 3, and 5 for N_{PC}=100, 500, and 1000, respectively) and absolute errors (row 2, 4, and 6 for N_{PC}=100, 500, and 1000, respectively) for Scenario 2, achieved by four metaheuristic algorithms.

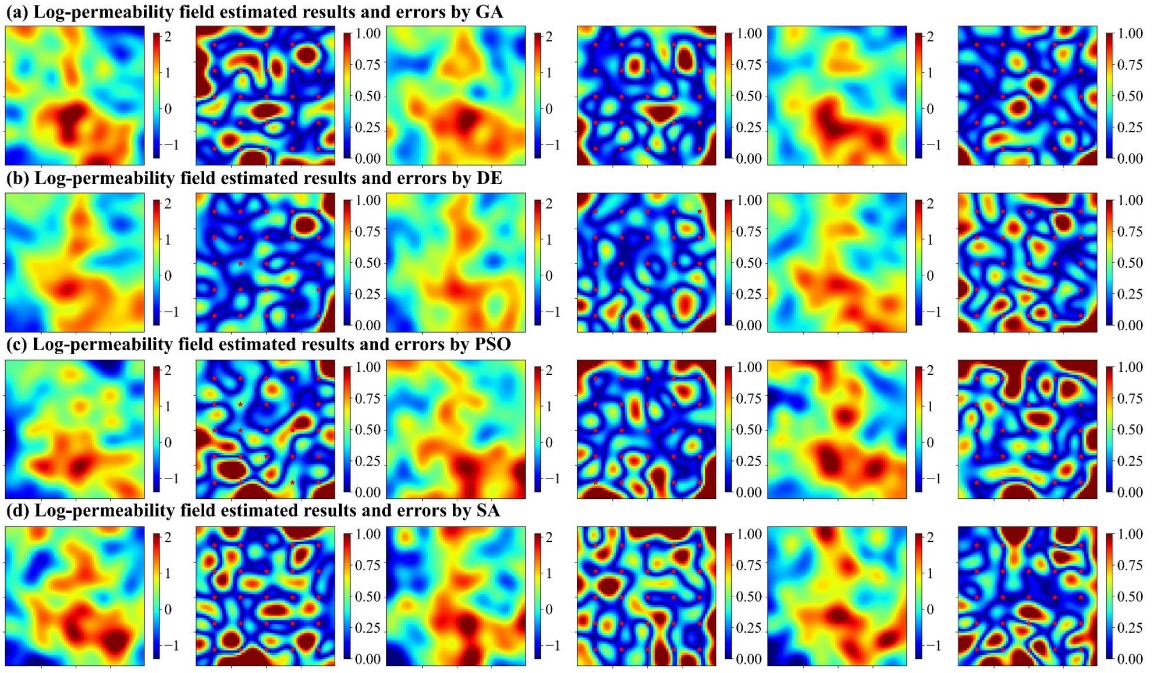

Fig.S9 Spatial distributions of log-permeability field estimation results (row 1, 3, and 5 for N_{PC}=100, 500, and 1000, respectively) and absolute errors (row 2, 4, and 6 for N_{PC}=100, 500, and 1000, respectively) for Scenario 3, achieved by four metaheuristic algorithms.

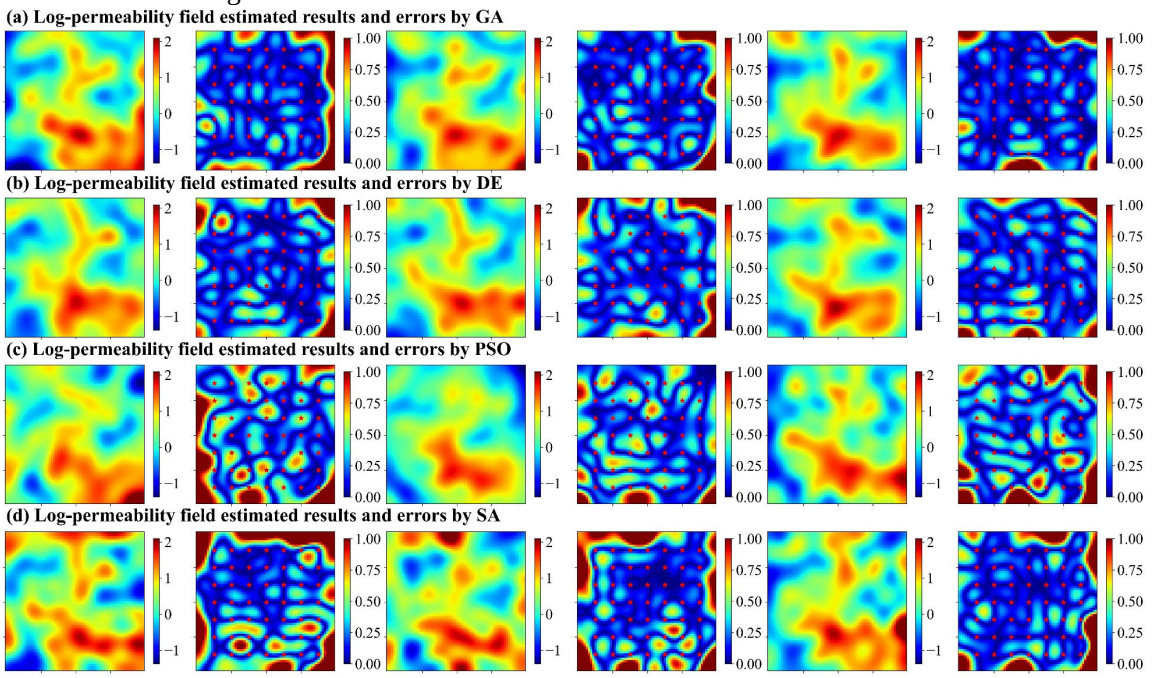

Fig.S10 Spatial distributions of log-permeability field estimation results (row 1, 3, and 5 for N_{PC}=100, 500, and 1000, respectively) and absolute errors (row 2, 4, and 6 for N_{PC}=100, 500, and 1000, respectively) for Scenario 4, achieved by four metaheuristic algorithms.

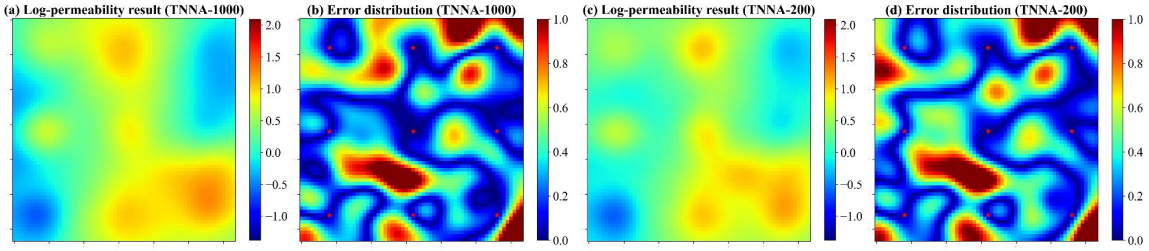

Fig.S11. Spatial distributions log-permeability field estimation results and absolute errors for Scenario 1, achieved by the TNNA inversion algorithm.

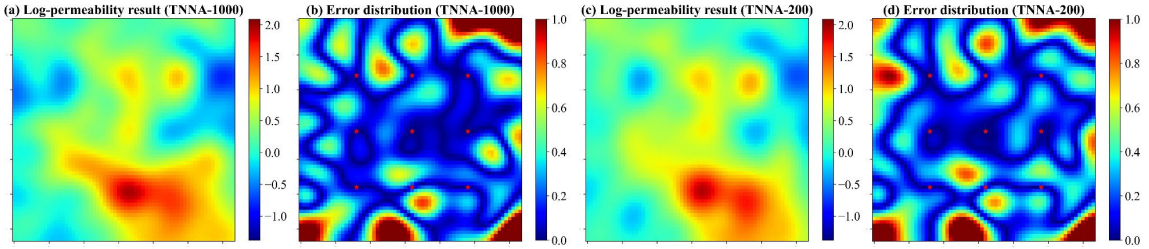

Fig.S12. Spatial distributions log-permeability field estimation results and absolute errors for Scenario 2, achieved by the TNNA inversion algorithm.

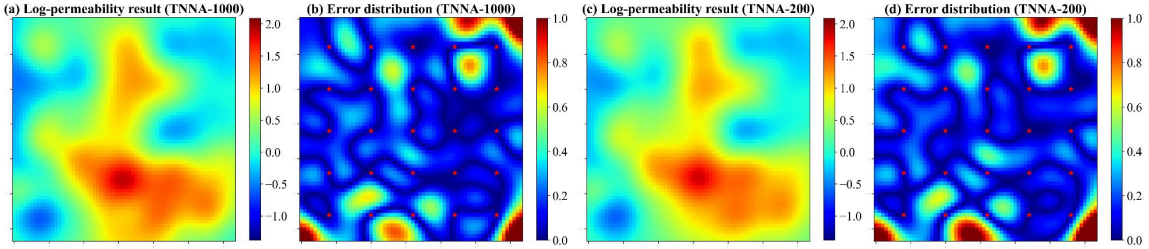

Fig.S13. Spatial distributions log-permeability field estimation results and absolute errors for Scenario 3, achieved by the TNNA inversion algorithm.

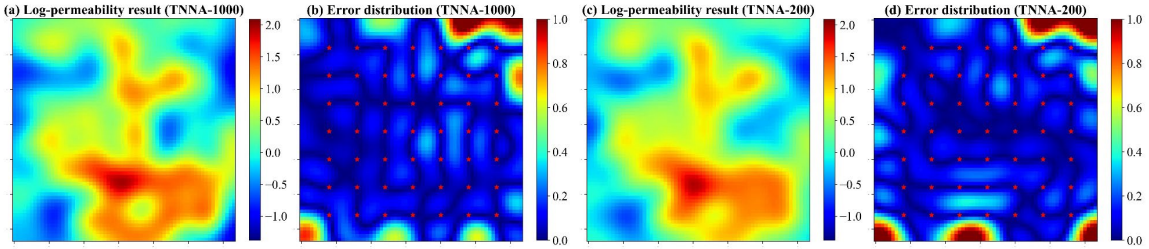

Fig.S14. Spatial distributions log-permeability field estimation results and absolute errors for Scenario 4, achieved by the TNNA inversion algorithm.

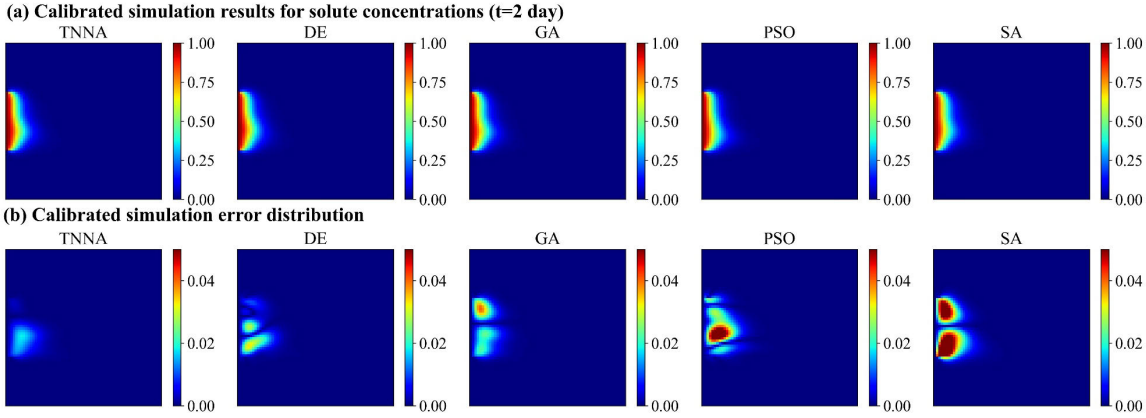

Fig.S15. Spatial distributions of calibrated numerical simulation results and absolute errors for solute concentrations (*t*=2 day) using the TNNA algorithm and four metaheuristic algorithms.

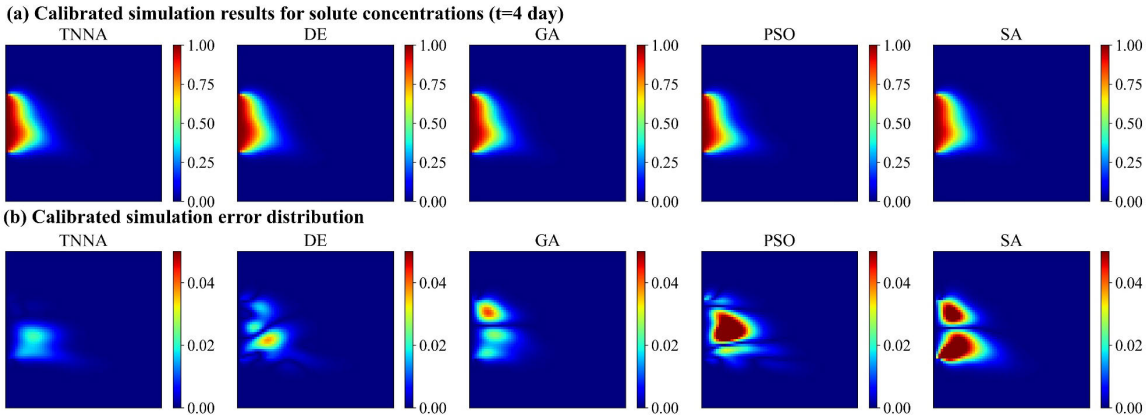

Fig.S16. Spatial distributions of calibrated numerical simulation results and absolute errors for solute concentrations (*t*=4 day) using the TNNA algorithm and four metaheuristic algorithms.

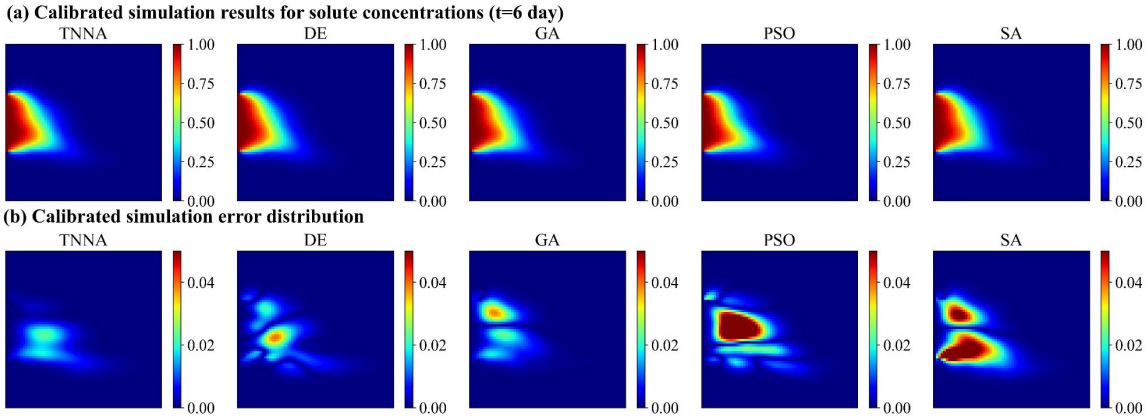

Fig.S17. Spatial distributions of calibrated numerical simulation results and absolute errors for solute concentrations (*t*=6 day) using the TNNA algorithm and four metaheuristic algorithms.

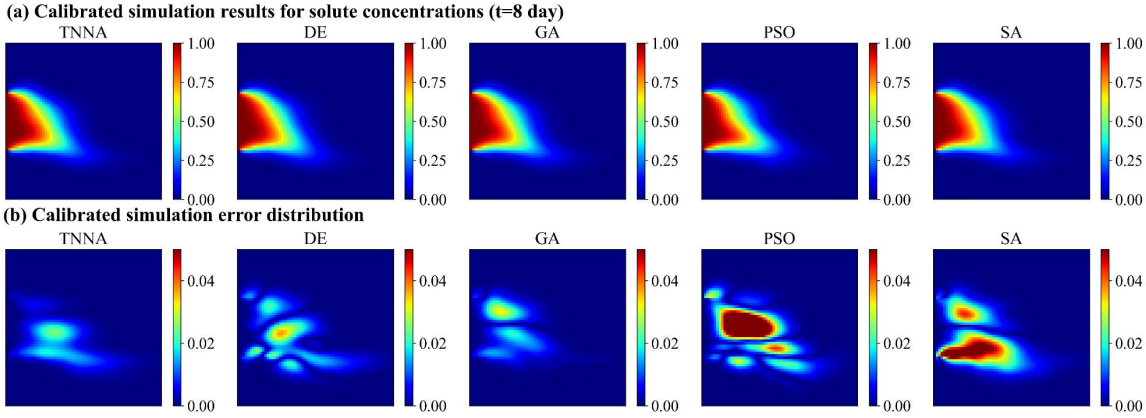

Fig.S18. Spatial distributions of calibrated numerical simulation results and absolute errors for solute concentrations (*t*=8 day) using the TNNA algorithm and four metaheuristic algorithms.

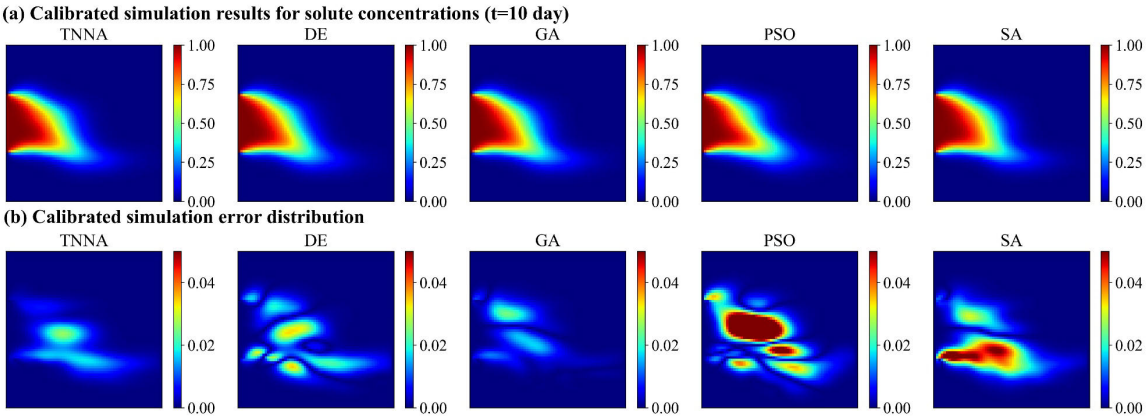

Fig.S19. Spatial distributions of calibrated numerical simulation results and absolute errors for solute concentrations (*t*=10 day) using the TNNA algorithm and four metaheuristic algorithms.

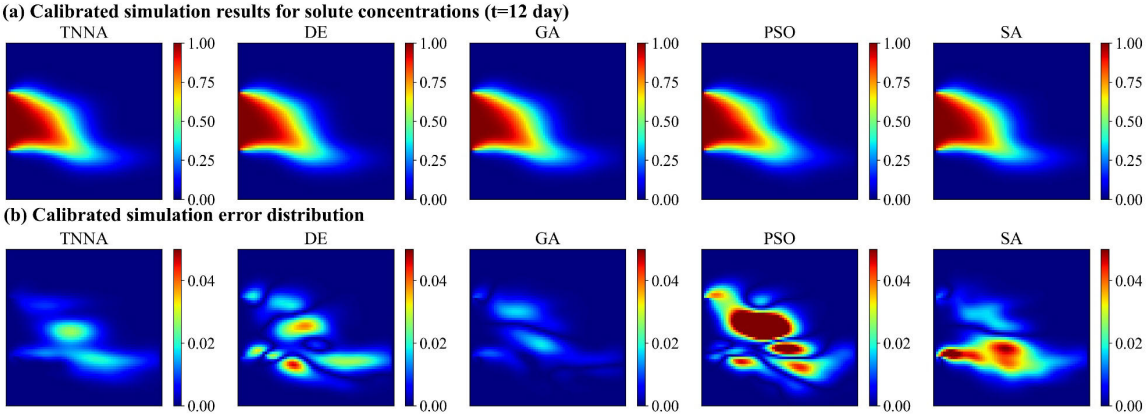

Fig.S20. Spatial distributions of calibrated numerical simulation results and absolute errors for solute concentrations (*t*=12 day) using the TNNA algorithm and four metaheuristic algorithms.

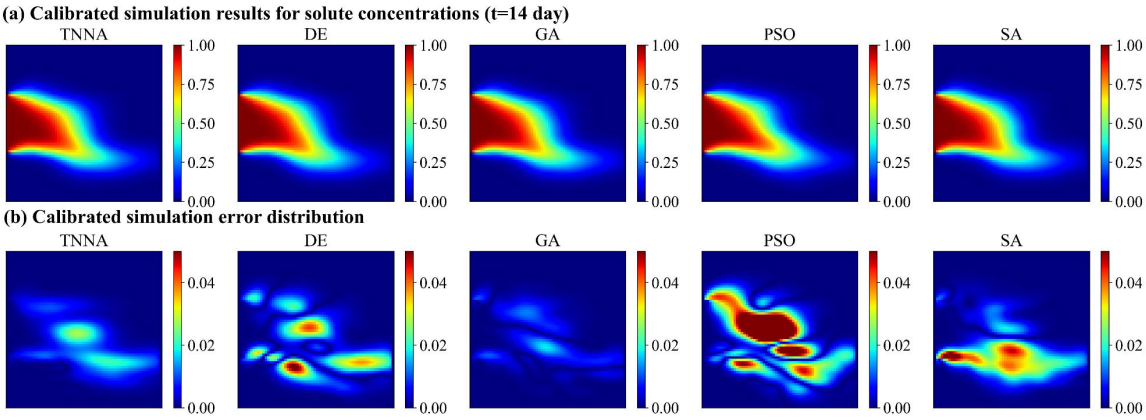

Fig.S21. Spatial distributions of calibrated numerical simulation results and absolute errors for solute concentrations (*t*=14 day) using the TNNA algorithm and four metaheuristic algorithms.

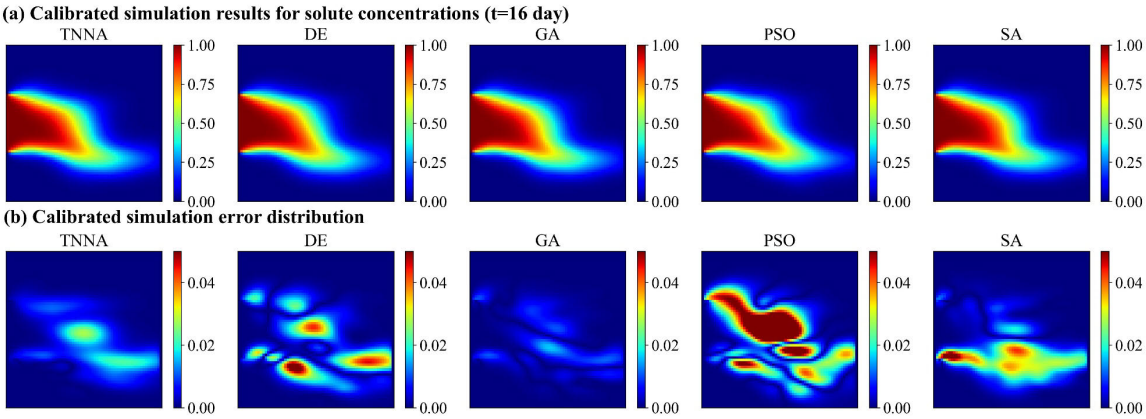

Fig.S22 Spatial distributions of calibrated numerical simulation results and absolute errors for solute concentrations (*t*=16 day) using the TNNA algorithm and four metaheuristic algorithms.

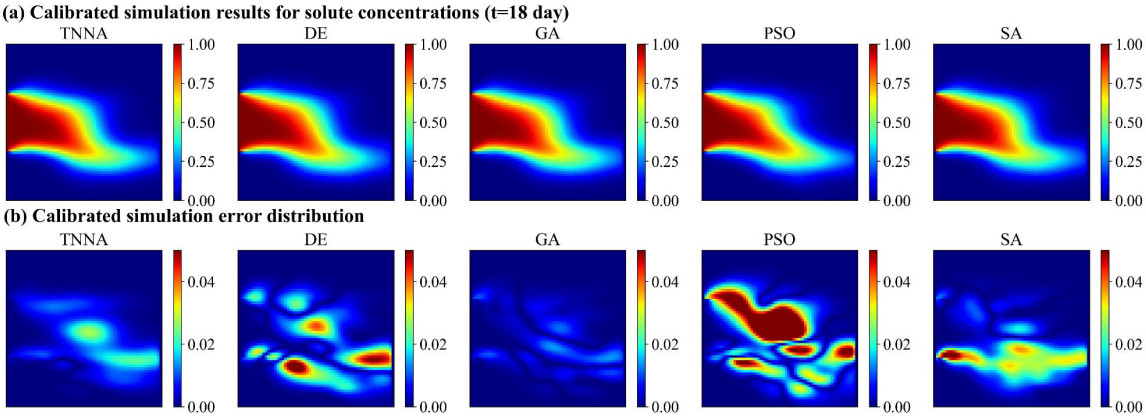

Fig.S23 Spatial distributions of calibrated numerical simulation results and absolute errors for solute concentrations (*t*=18 day) using the TNNA algorithm and four metaheuristic algorithms.

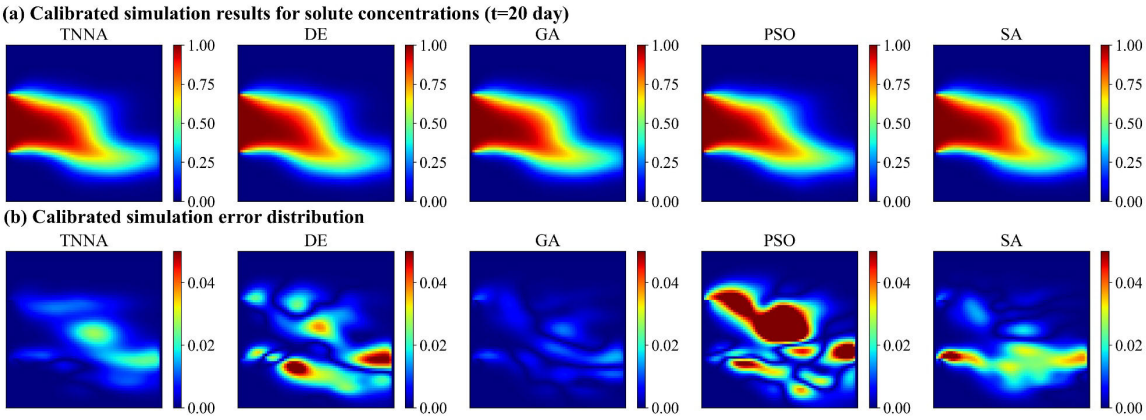

Fig.S24 Spatial distributions of calibrated numerical simulation results and absolute errors for solute concentrations (*t*=20 day) using the TNNA algorithm and four metaheuristic algorithms.

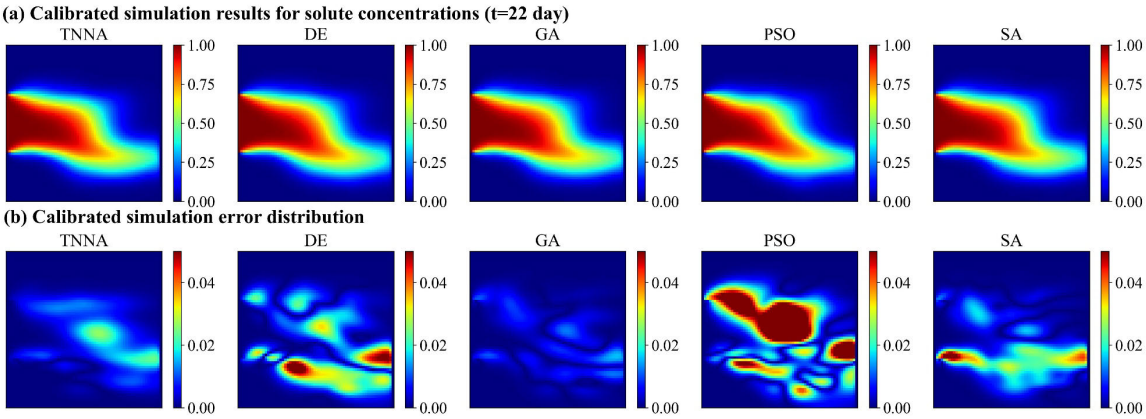

Fig.S25. Spatial distributions of calibrated numerical simulation results and absolute errors for solute concentrations (*t*=22 day) using the TNNA algorithm and four metaheuristic algorithms.

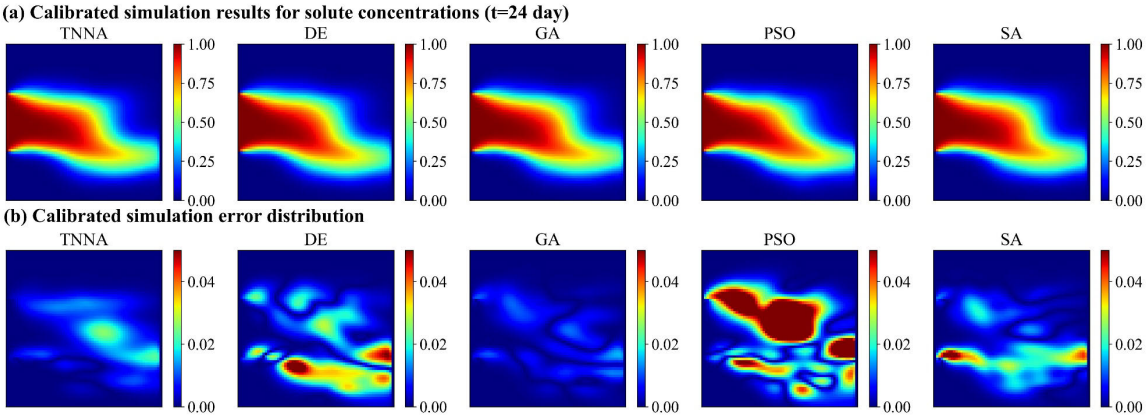

Fig.S26. Spatial distributions of calibrated numerical simulation results and absolute errors for solute concentrations (*t*=24 day) using the TNNA algorithm and four metaheuristic algorithms.

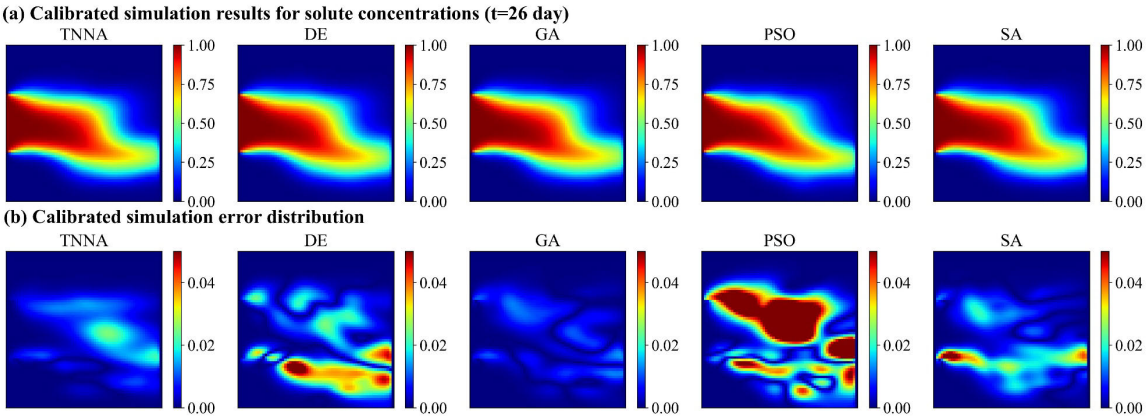

Fig.S27. Spatial distributions of calibrated numerical simulation results and absolute errors for solute concentrations (t=26 day) using the TNNA algorithm and four metaheuristic algorithms.

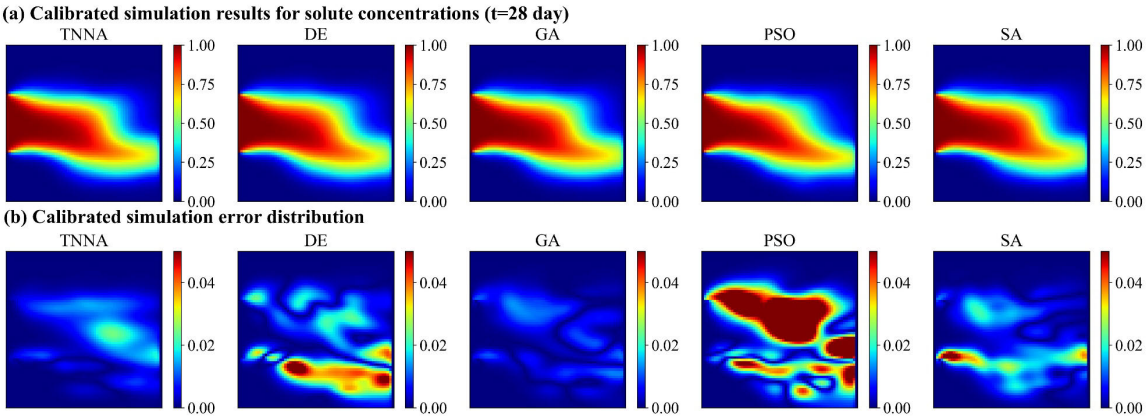

Fig.S28. Spatial distributions of calibrated numerical simulation results and absolute errors for solute concentrations (t=28 day) using the TNNA algorithm and four metaheuristic algorithms.

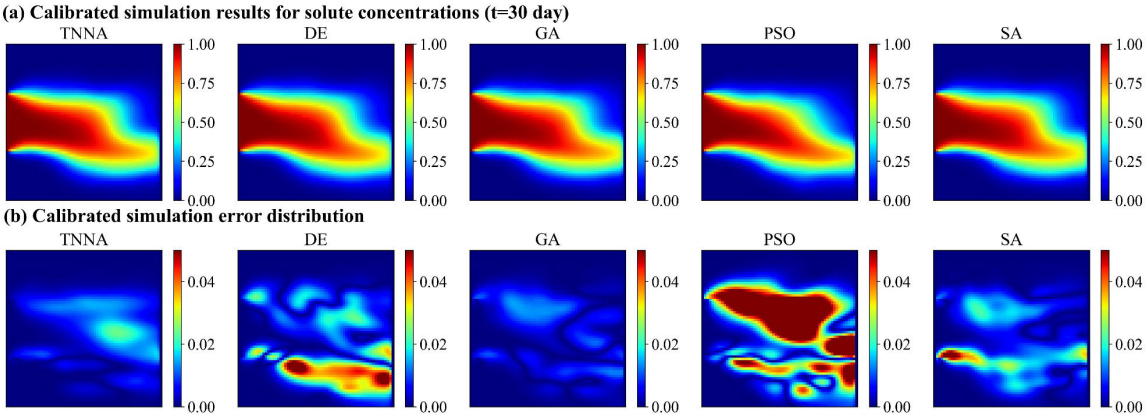

Fig.S29. Spatial distributions of calibrated numerical simulation results and absolute errors for solute concentrations (t=30 day) using the TNNA algorithm and four metaheuristic algorithms.

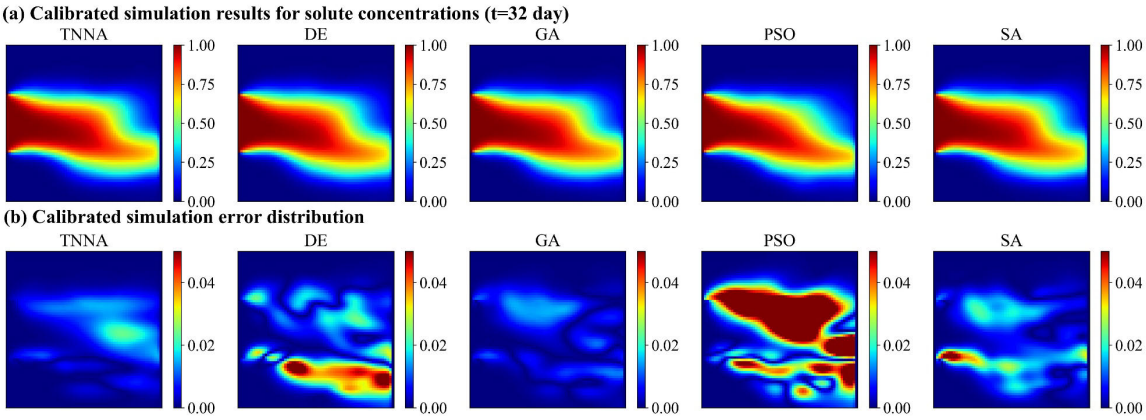

Fig.S30. Spatial distributions of calibrated numerical simulation results and absolute errors for solute concentrations (*t*=32 day) using the TNNA algorithm and four metaheuristic algorithms.

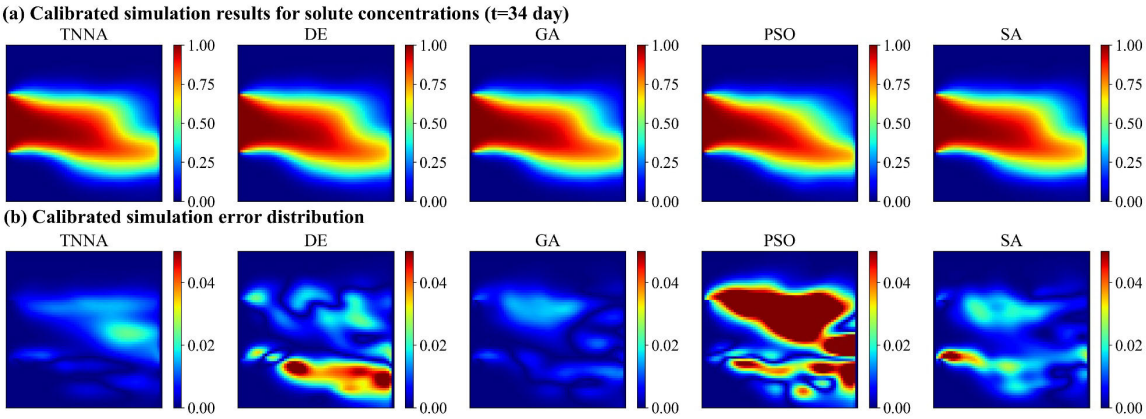

Fig.S31. Spatial distributions of calibrated numerical simulation results and absolute errors for solute concentrations (*t*=34 day) using the TNNA algorithm and four metaheuristic algorithms.

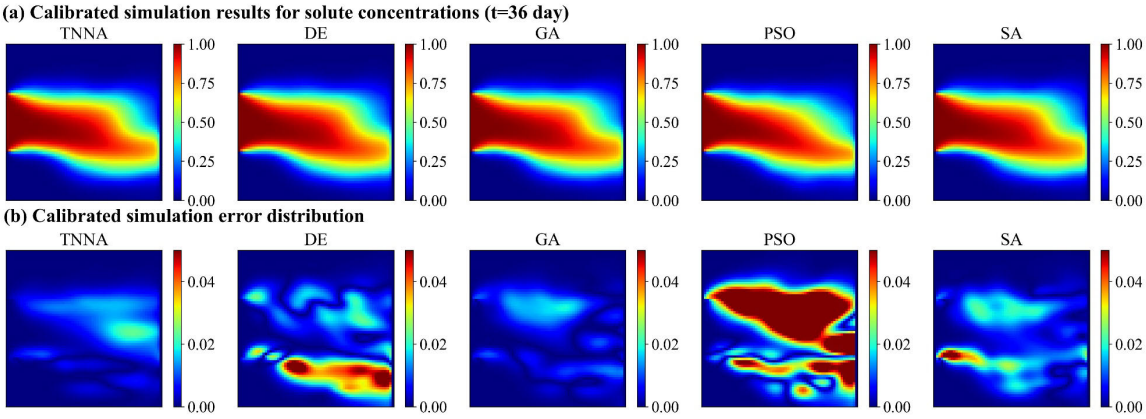

Fig.S32. Spatial distributions of calibrated numerical simulation results and absolute errors for solute concentrations (*t*=36 day) using the TNNA algorithm and four metaheuristic algorithms.

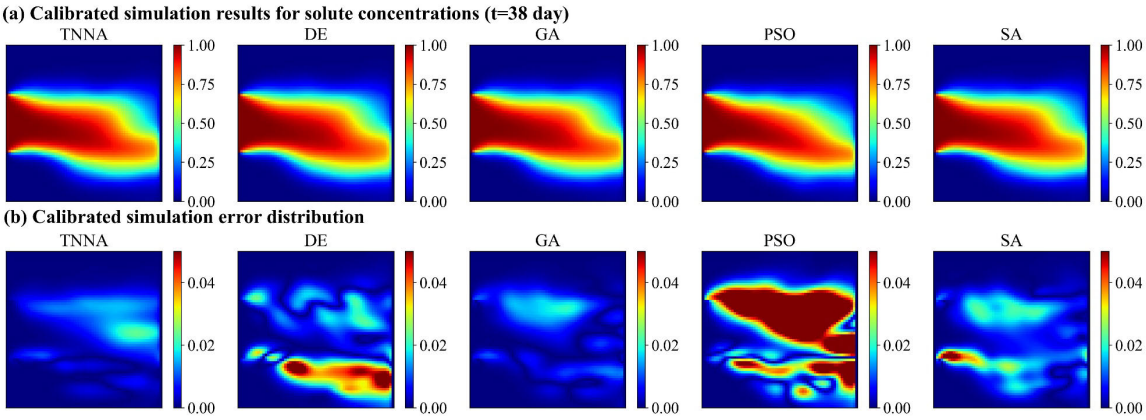

Fig.S33. Spatial distributions of calibrated numerical simulation results and absolute errors for solute concentrations (*t*=38 day) using the TNNA algorithm and four metaheuristic algorithms.

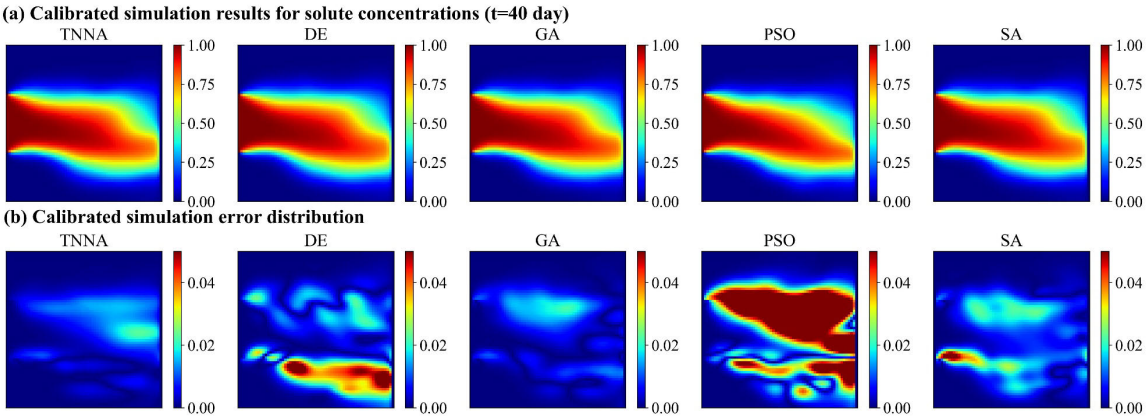

Fig.S34. Spatial distributions of calibrated numerical simulation results and absolute errors for solute concentrations (*t*=40 day) using the TNNA algorithm and four metaheuristic algorithms.

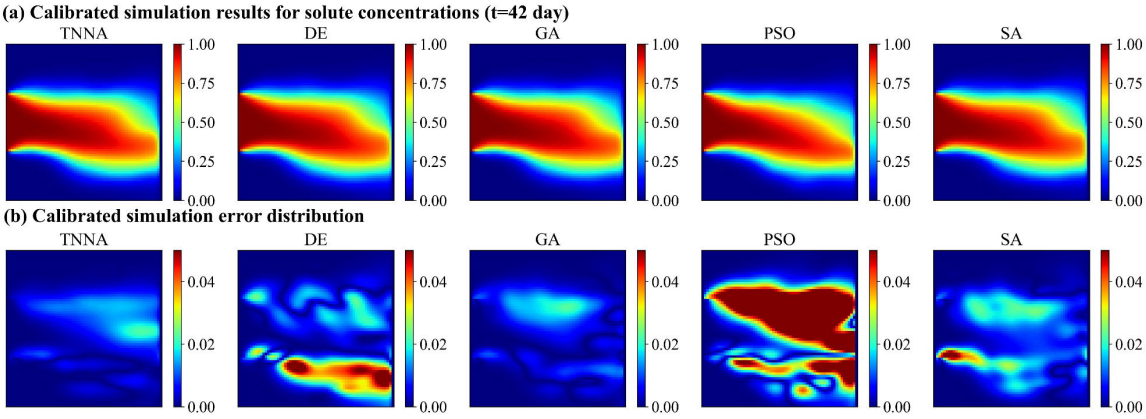

Fig.S35. Spatial distributions of calibrated numerical simulation results and absolute errors for solute concentrations (*t*=42 day) using the TNNA algorithm and four metaheuristic algorithms.

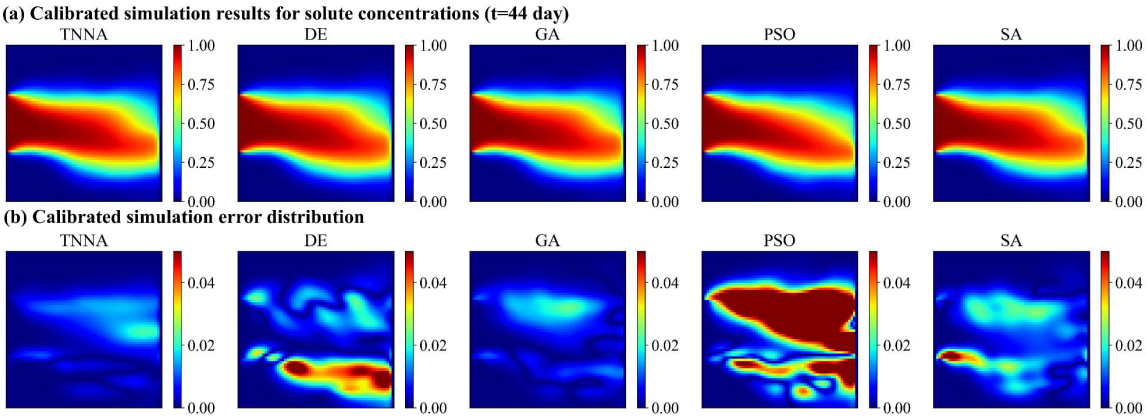

Fig.S36. Spatial distributions of calibrated numerical simulation results and absolute errors for solute concentrations (t=44 day) using the TNNA algorithm and four metaheuristic algorithms.

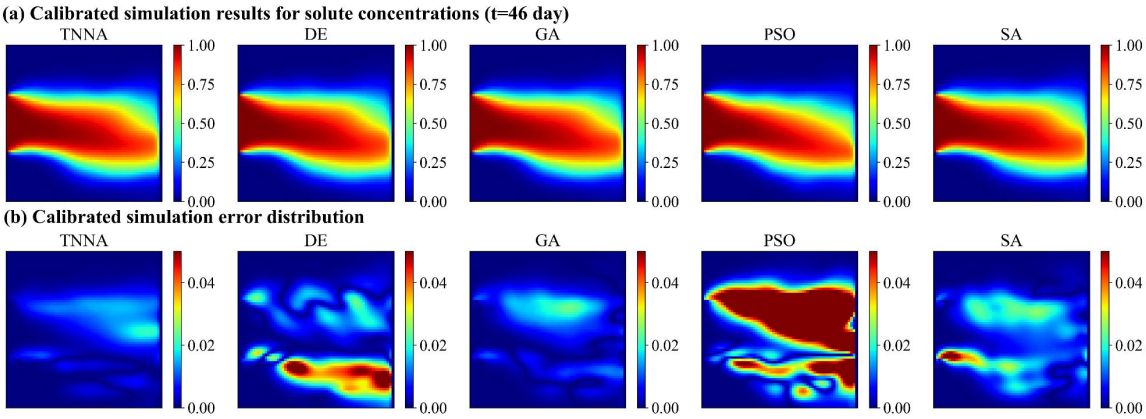

Fig.S37. Spatial distributions of calibrated numerical simulation results and absolute errors for solute concentrations (t=46 day) using the TNNA algorithm and four metaheuristic algorithms.

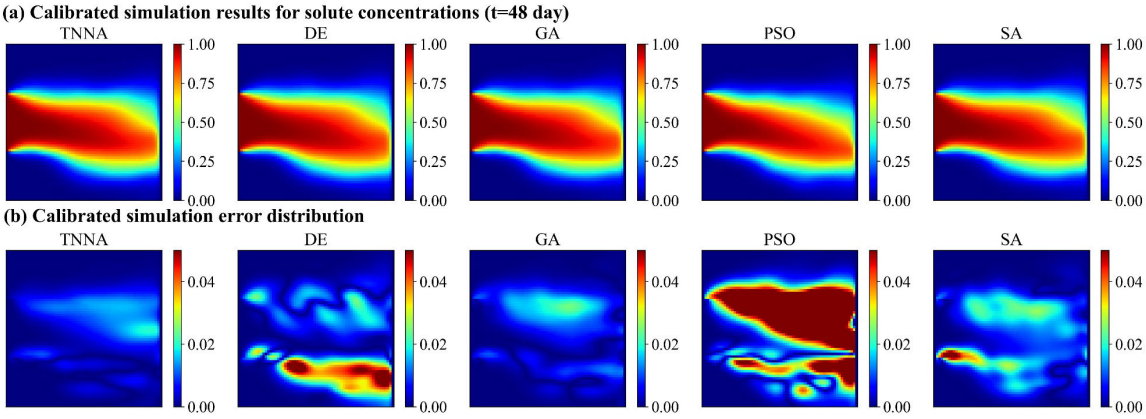

Fig.S38. Spatial distributions of calibrated numerical simulation results and absolute errors for solute concentrations (t=48 day) using the TNNA algorithm and four metaheuristic algorithms.

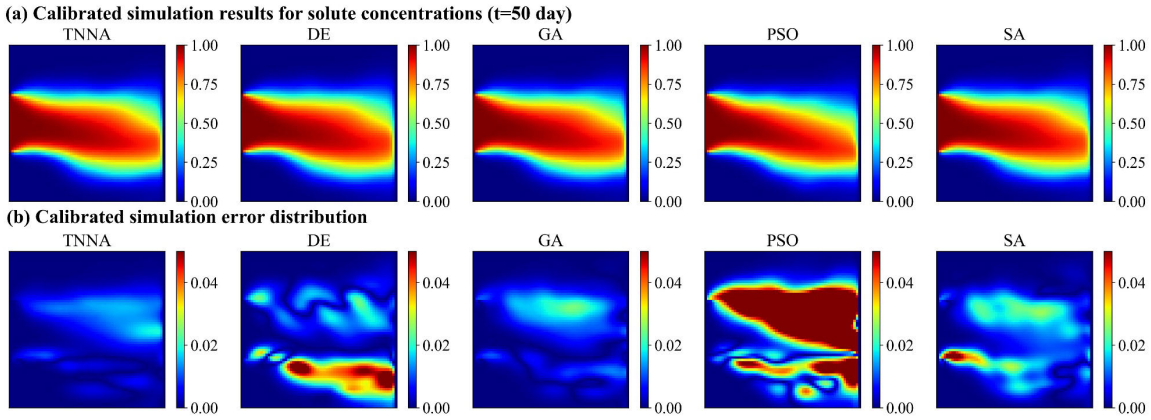

Fig.S39. Spatial distributions of calibrated numerical simulation results and absolute errors for solute concentrations (*t*=50 day) using the TNNA algorithm and four metaheuristic algorithms.

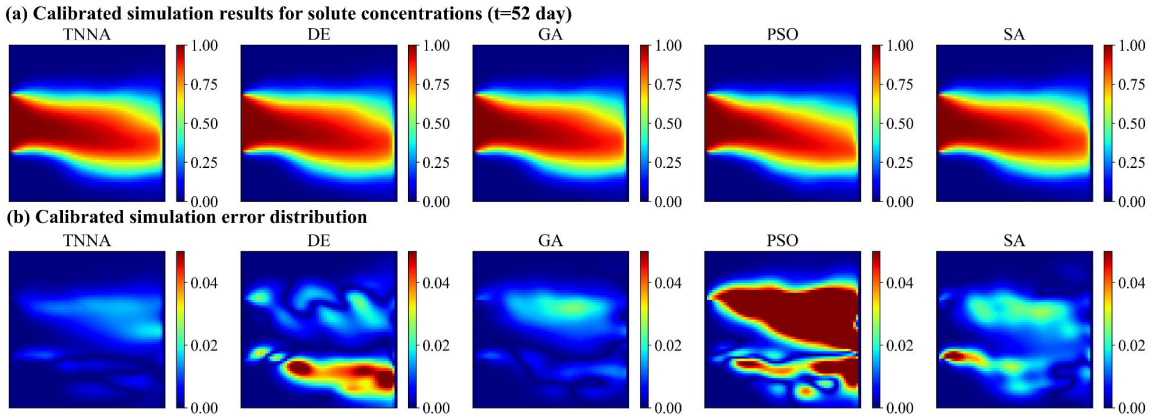

Fig.S40. Spatial distributions of calibrated numerical simulation results and absolute errors for solute concentrations (*t*=52 day) using the TNNA algorithm and four metaheuristic algorithms.

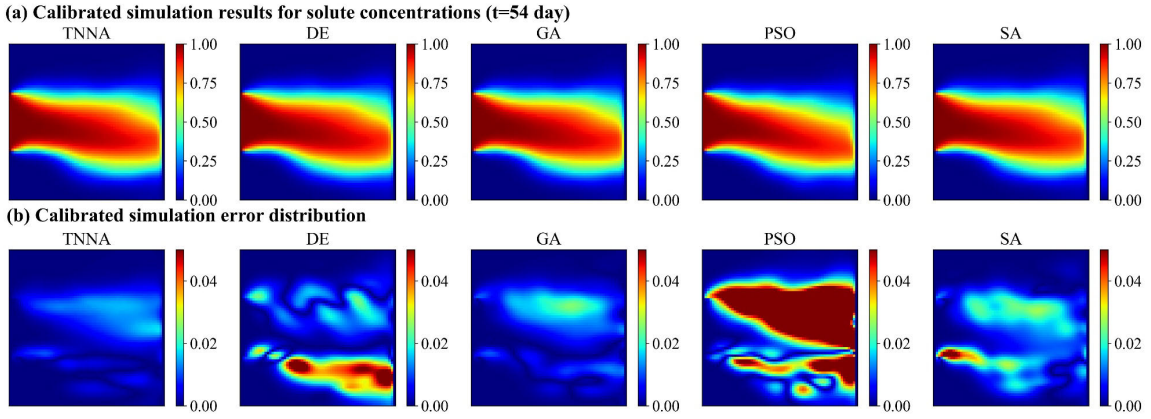

Fig.S41. Spatial distributions of calibrated numerical simulation results and absolute errors for solute concentrations (*t*=54 day) using the TNNA algorithm and four metaheuristic algorithms.

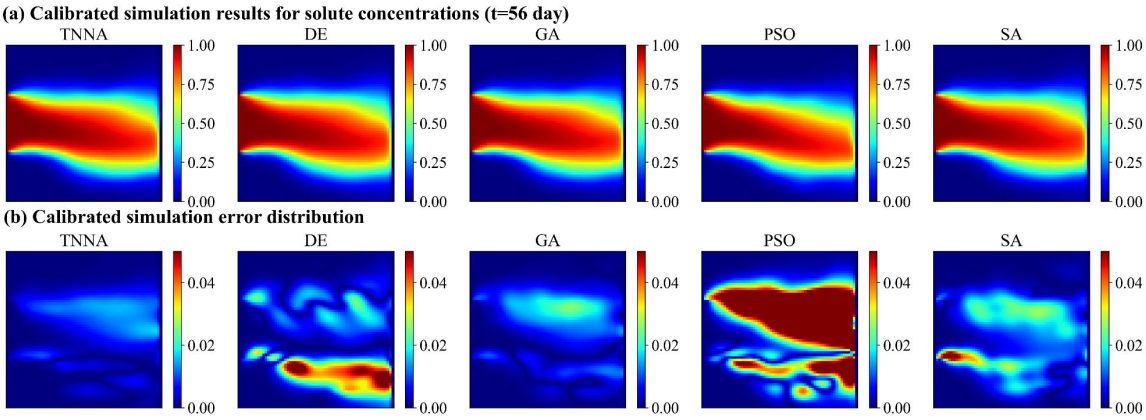

Fig.S42. Spatial distributions of calibrated numerical simulation results and absolute errors for solute concentrations (*t*=56 day) using the TNNA algorithm and four metaheuristic algorithms.

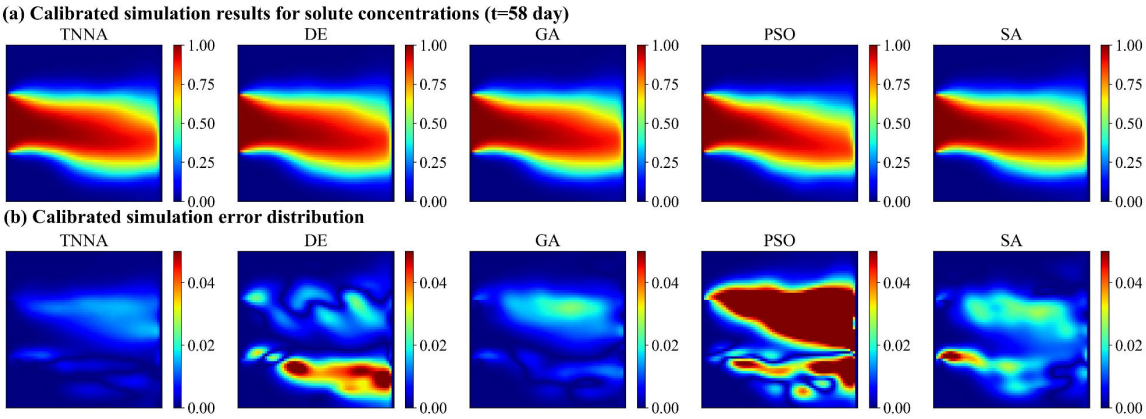

Fig.S43. Spatial distributions of calibrated numerical simulation results and absolute errors for solute concentrations (*t*=58 day) using the TNNA algorithm and four metaheuristic algorithms.

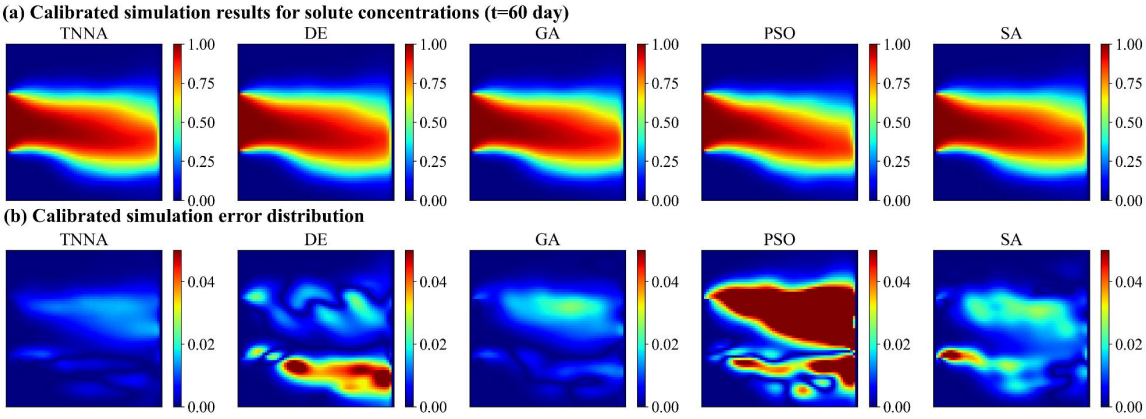

Fig.S44. Spatial distributions of calibrated numerical simulation results and absolute errors for solute concentrations (*t*=60 day) using the TNNA algorithm and four metaheuristic algorithms.