

# INVERSE METHOD TO DETERMINE HYDRAULIC CONDUCTIVITY FROM A VELOCITY FIELD USING GRAPH THEORY

Michael E. Mont-Eton<sup>1</sup>, Steffen Borgwardt<sup>2</sup>, David C. Mays<sup>1</sup>

<sup>1</sup>University of Colorado Denver, Department of Civil Engineering; <sup>2</sup>University of Colorado Denver, Department of Mathematical and Statistical Sciences

## Table of Contents

|                                                |   |
|------------------------------------------------|---|
| S1 Graph Theory .....                          | 1 |
| S2 Well-Posedness .....                        | 2 |
| S3 Details on FlowPaths .....                  | 2 |
| S4 Cyclomatic Complexity .....                 | 5 |
| S5 The Forward and Inverse Problem Cycle ..... | 7 |
| S6 4x4 Example Data Tables .....               | 9 |
| References .....                               | 9 |

## 1. GRAPH THEORY

In the language of graph theory, nodes are called *vertices*, links are called *edges*, and together they constitute a *graph* with symbol  $G$ . Formally,  $V$  is the vertex set of  $G$ ,  $E$  is the edge set of  $G$ , and  $G$  is defined as the tuple  $G = (V, E)$ . A directed graph (*digraph*)  $D = (V, E)$  consists of the same set  $V$  of vertices together with the same set  $E$  of edges, but links them through ordered pairs  $(u, v)$  of vertices called *directed edges*. A digraph  $H$  is a *subdigraph* of a digraph  $D$  if  $V(H) \subseteq V(D)$  and  $E(H) \subseteq E(D)$  and every edge in  $E(H)$  has both end-vertices in  $V(H)$ . Vertices  $u, v \in V$  in a digraph that are joined by an edge  $e \in E$  are *adjacent* and  $e$  is called *incident* to the predecessor vertex  $u$  and the successor vertex  $v$ .

In the context of this study, one of the most important concepts is a *path*. A path  $p$  in a digraph  $D$  is a sequence of two or more vertices connected by directed edges, with no repeated edges. A *simple path* has no repeated vertices (22). A *cycle* is a path that starts and ends at the same vertex. A *directed acyclic graph* (DAG), is the type of graph that we use in this study since it generally matches the conditions of steady groundwater flow.

Graphs and digraphs may also have an attribute assigned to the edges, which we call *weights*. The weight  $w_i$  for edge  $e_i$  is defined to be the intercellular specific discharge  $q_i$ , which is equal to the total flow,  $Q_i$ , divided by the area of the cell face  $A_i$ . The complete description of the digraph  $D$  with weights assigned to each edge is  $D = (V, E, w)$ , where  $w$  is the set of weights for the edge set of  $D$ , or  $E(D)$ . For any vertex except the root or sink in a weakly connected digraph the law of *continuity* (there is no change in mass of any element of the system, whether it is a vertex or a grid cell) applies (Eq. S1):



$$\sum_{i=1}^{i=d} w_i(u) = 0 \quad (S1)$$

where  $d$  is the *degree* of the vertex  $u$ , *i.e.* the number of edges that are incident to  $u$ . This condition is established in porous media when the groundwater flow reaches a steady state. FlowPaths considers only the specific discharges entering the vertices, implying that all edge weights must be non-negative.

## 2. WELL-POSEDNESS

Here we elaborate on Tikhonov's third criterion for well-posedness (that the solution is stable) using the following argument. Zijl et al. (27) distinguish between two cases. In case 1, an inverse model is being used to calibrate a forward model with a priori information, such as certain known values of  $K(x, y)$ . In case 2, the inverse model is used to directly identify  $K(x, y)$  using some other observational data, such as the measured specific discharges which we use in FlowPaths. In case 2 (but not case 1), a stability analysis which uses a combined error result is appropriate. According to Higham (13, p. 7), this is a *mixed forward-backward error* result, where the modeled input  $\hat{x} + \Delta x$  that is close to the actual input  $\hat{x}$  produces a modeled output  $\hat{y} + \Delta y$  that is close to the actual output  $\hat{y}$ . In our approach the computed value is the inverse solution  $K(x, y)$ , and the input is the array of specific discharges  $q(x, y)$ . We use a recursive error checker (9) that computes the cumulative sum of the mixed forward-backward error for every recursive step and then examines the regression equation to see if the solution parameter is stable within the significance level  $\alpha = 0.05$  as implemented in the MATLAB function "cusumtest" (17). If the regression equation of the cumulative forward-backward error exceeds either the upper or lower error bounds, that indicates a structural change in the system, rejecting the null hypothesis of instability in the model, shown as the data points between the dotted 5% critical lines shown, for example, in **Figure 8** in the [main text](#) (3, 14). Thus, the mixed forward-backward error, implemented with the MATLAB function "cusumtest", confirms that small changes in the inputs produce small changes in the outputs for both the ADI forward model and the FlowPaths inverse model.

## 3. DETAILS ON FLOWPATHS

FlowPaths determines the hydraulic conductivity in a square flow matrix, starting with tables of specific discharge data for each cell face, through four steps. Step 1 is to create a digraph  $D = (V, E, w)$ , where the weight  $w$  of each edge  $e \in E$  is assigned the specific discharge  $q$  of the corresponding cell face. Step 2 is to decompose the digraph  $D$  into a set of subgraphs, labeled  $H_{s,t}$ : one for each pair of source-target vertices  $(s, t)$ . For each  $(s, t)$  pair, FlowPaths finds a maximal linearly independent set of vectors (26) as vertex paths,  $P_{\perp}$ , based on the reachable vertices and edges in  $H_{s,t}$ . Using the head-drop equation for each path, the  $q^*$  coefficient for each vertex is calculated using **Equation 12** from the [main text](#). Step 3 is to assemble the output of the subroutine for each  $(s, t)$  pair into an over-determined array **A** of rows of paths  $p_i$  and columns of vertices  $v_j$ , where each element  $A_{i,j}$  contains a  $q^*$  coefficient or a zero (the latter of which indicates that vertex  $v_j$  does not belong to path  $p_i$ ). **A** is reduced to a square matrix **M** of  $q^*$  coefficients and zeros containing linearly independent rows and columns. And finally, Step 4 is to solve the system of equations ( $\mathbf{M}\mathbf{x}=\mathbf{b}$ ), where  $\mathbf{x}$  is the unknown column vector of resistances,  $\mathbf{R}$ , and  $\mathbf{b}$  is the total head drop of the system divided by the length of each cell (edge). The solution  $K(x, y)$  is the cell-by-cell reciprocal of  $\mathbf{R}$ . The following subsections elaborate on each of these four steps.

### Step 1: Transformation from the Cell Format to a Digraph

FlowPaths is based on head drop equations corresponding to independent paths identified using graph theory as shown in **Figure 3** of the [main text](#).

## Step 2: Identify the Linearly Independent Paths in the Sub-digraphs $H_{st}$

Many paths exist between the sources and sinks. As an example, four *simple* directed paths are shown in **Figure S1**, each starting at a high-head source vertex. Paths 1-4 are identified by the vertices that they traverse. Each path generates a head-drop equation in the form of **Equation 13**. In order to accentuate the relationship between the  $R$  and  $q^*$  coefficients as triplets, we identify the central vertex as belonging to  $R$  and the subscripts in each  $q^*$  coefficient belonging to the predecessor and successor vertices, as (**Eq. S2**):

$$\text{Path 1: } \frac{h_1 - h_{22}}{l_{edge}} = R_5(q_{1,9}^*) + R_9(q_{5,13}^*) + R_{13}(q_{9,14}^*) + R_{14}(q_{13,18}^*) + R_{18}(q_{14,22}^*) \quad (\text{S2})$$

$$\text{Path 2: } \frac{h_3 - h_{22}}{l_{edge}} = R_7(q_{3,6}^*) + R_6(q_{7,10}^*) + R_{10}(q_{6,14}^*) + R_{14}(q_{10,18}^*) + R_{18}(q_{14,22}^*)$$

$$\text{Path 3: } \frac{h_3 - h_{23}}{l_{edge}} = R_7(q_{3,11}^*) + R_{11}(q_{7,15}^*) + R_{15}(q_{11,19}^*) + R_{19}(q_{15,23}^*)$$

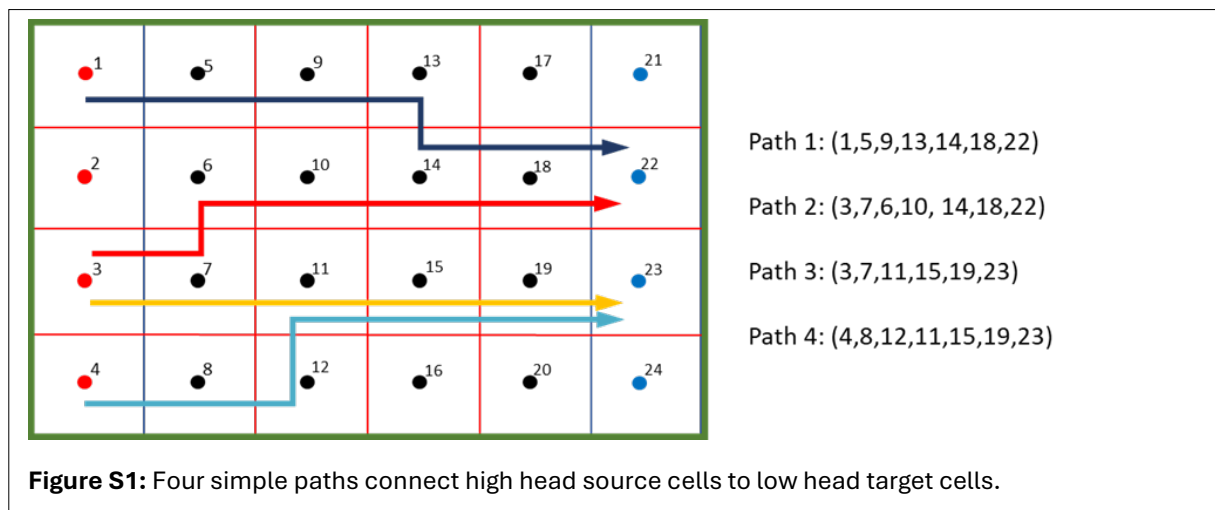
$$\text{Path 4: } \frac{h_4 - h_{23}}{l_{edge}} = R_8(q_{4,12}^*) + R_{12}(q_{8,11}^*) + R_{11}(q_{12,15}^*) + R_{15}(q_{11,19}^*) + R_{19}(q_{15,23}^*)$$

The path-finding subroutine in FlowPaths considers each pair of source-target ( $s, t$ ) vertices from  $D$ , using the digraph and subdigraphs generated using graph functions in Matlab (15). The subroutine comprises three blocks as follows:

**Block 1:** Identify the sub-digraph  $H$  based on the source and target vertices: The first step is to identify all of the vertices in  $D_\Omega$  that are reachable from  $s$  and all of the vertices that lead to  $t$ . Let  $X(s)$  be the set of successor vertices in  $D_\Omega$  of  $s$  and  $Y(t)$  be the set of vertices which are predecessors in  $D_\Omega$  of  $t$ . Let the vertex set  $V(H)$  be the intersection of  $X(s)$  and  $Y(t)$ :  $V(H) = X(s) \cap Y(t)$ . If  $V(H)$  has no members, return a vertex path matrix of all zeros. If  $V(H)$  is not empty, let the edge set  $E(H)$  be the edges in  $D_\Omega$  that are incident to any pair of vertices in  $H_{s,t}$ .

**Block 2:** Identify the Minimum Directed Spanning Tree (MDST) in  $H$ , rooted at  $s$ : As detailed in Block 3 below, independent paths are identified using two concepts from graph theory, first, minimum directed spanning trees, and second, chords. A *tree*  $T$  is a connected acyclic graph, and its edges are called *branches* (7, p. 27). The other edges are called *chords*, and are elements of the co-tree  $\bar{T}$  (4, p. 80). Here we use the term tree loosely to indicate a subdigraph that is rooted at a vertex on the boundary of the flow matrix. Every chord is identified as an edge of  $H$  which is not in the MDST (the co-tree). The name of the chords and their weights (specific discharges) are stored in a table, such as **Table S1**.

**Block 3:** Find the independent set of edge paths, going both ways from the chord: The chords identified in Block 2 provide the starting point to find the independent paths. The subroutine starts with the first



chord in the table, which for the example of **Table S1** is  $e$ . A path from the source to the in-vertex of  $e$  within the MDST is formed. Then a shortest path is found from the out-vertex of  $e$  to  $t$ . The two sets of vertices and incident edges along with the chord are concatenated into a full path. This process is repeated for all chords. The second segment of any path  $p_a$  starting at chord  $c_a$  may also traverse some other chord (or chords)  $c_b$ , but a path  $p_b$  generated by  $c_b$  cannot include  $c_a$ , because each chord is the first one in its generated path. This is the reason why the constructed set of paths is linearly independent. The Matlab function "shortestpath" (16) implements the shortest path algorithm (22) through the MDST for each chord, where that chord is the first one the path. The subroutine continues through the table of chords, finding all of the unique paths generated by the chords. The number of paths at this point equals the number of chords, and thus is equal to the cyclomatic number  $v(G)$ . The final path to be added to the path set  $P$  is the *trunk* path, which has no chords, by implementing a depth-first search through the MDST itself. For the example of **Table 1** and **Figure 7** (both in the [main text](#)), this trunk path is [3, 7, 8, 12, 11, 10, 9, 13, 17, 18, 22]. Now we have that  $P_{LI} = 8$  for this subdigraph.

**Table S1:** Chords belonging to the co-tree  $\bar{T}$  from the sub-digraph  $H_{3,22}$ .

| Chord Name | In-vertex | Out-vertex | Weight (cm/day) |
|------------|-----------|------------|-----------------|
| e          | 5         | 9          | 1.38956         |
| f          | 6         | 10         | 1.52726         |
| g          | 7         | 11         | 1.44822         |
| m          | 10        | 14         | 1.57701         |
| n          | 11        | 15         | 1.37437         |
| p          | 13        | 14         | 0.12137         |
| t          | 14        | 15         | 1.70985         |

### Step 3: Pass the paths from the sub-digraph back to the main program

The set of paths from each subdigraph  $H$  are assembled into a vertex path matrix (VPM), which is similar to an edge path matrix as defined by Foulds (7, p. 90). The VPM consists of a row for each path between  $s$  and  $t$  and a column for each vertex in  $H$ . If vertex  $k$  is in path  $i$ , then  $p_{ik} = 1$ , otherwise  $p_{ik} = 0$ . Recall that the number of LI paths in  $H$  is equal to the number of chords in the co-tree of the rooted tree plus 1 (**Eq. S3**) where  $|\sim|$  is the number of elements in set  $\sim$ :

$$|P_{LI}| = |\bar{T}| + 1 = C \quad (S3)$$

If  $T \subseteq H$  is a spanning tree, and  $e$  is a chord in  $H$ , a fundamental cycle  $C_e$  is the simple cycle made by the union of  $e$  and the path in  $T$  that connects the endpoints of  $e$ . The set of all the fundamental cycles in  $H$  constructed consecutively in this fashion form the fundamental (linearly independent) cycle basis of  $H$ ,  $C(H)$ , with  $e - v + 1$  cycles (5, p. 27). The set of paths developed using the MDST and its chords (the chord paths plus the trunk path) are also linearly independent, forming a path basis of  $H, P(H)$ . The resulting VPM is thus a fundamental vertex path matrix (FVPM). All other paths between  $s$  and  $t$  are linear combinations of these independent paths. For  $H_{3,22}$ , the FVPM is (**Eq. S4**) where the source and target vertices are not shown in the matrix:

$$P_{f,v}(3,22) = \begin{matrix} & \begin{matrix} 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{matrix} & \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{matrix} \quad (S4)$$

Now that the FVPM has been identified for each sub-digraph  $H$ , the  $q^*$  coefficients are determined for each vertex on a path using **Equation 12** and the serial head-drop (**Eq. 13**) for each path. The coefficients of the edge polynomials are mapped to the coefficients of the vertex polynomials. For any path  $P_i \in P_{f,v}(s, t)$  (**Eq. S5**):

$$P_i(R_k, q_{k-1,k+1}^*) \Big|_{k=2}^{k=n-1} = R_2 q_{1,3}^* + R_3 q_{2,4}^* + \cdots + R_{n-2} q_{n-3,n-1}^* + R_{n-1} q_{n-2,n}^* \quad (\text{S5})$$

Each vertex polynomial now becomes a member of the system of equations,  $\mathbf{Gm}_{est} = \mathbf{d}$  that describes the resistance problem, that is, the reciprocal of the hydraulic conductivity problem. Before the system can be solved for  $\mathbf{R}$ , using matrix methods (*i.e.*,  $\mathbf{Ax} = \mathbf{b}$ ), where  $\mathbf{x} = \mathbf{R}$  and  $\mathbf{b} = \frac{\Delta H}{l}$ , the  $q_{k-1,k+1}^*$  coefficients of each vertex polynomial  $P_i(q^*, R)$  must be mapped to the correct  $j^{th}$  element in row  $i$  of the coefficient matrix  $A_{ij}$ . The mapping is accomplished by using a set of logical descriptors built into the attributes of every vertex of the digraph, because each vertex may have up to four values of  $q^*$ , depending on the combinations of flow into and out of the vertex. Once all of the paths in the other sub-digraphs have been aggregated, the union of the FVPMs produces an over-determined array ( $\mathbf{A}$ ) of  $q^*$  coefficients, because some of the paths may be linear combinations of other paths.

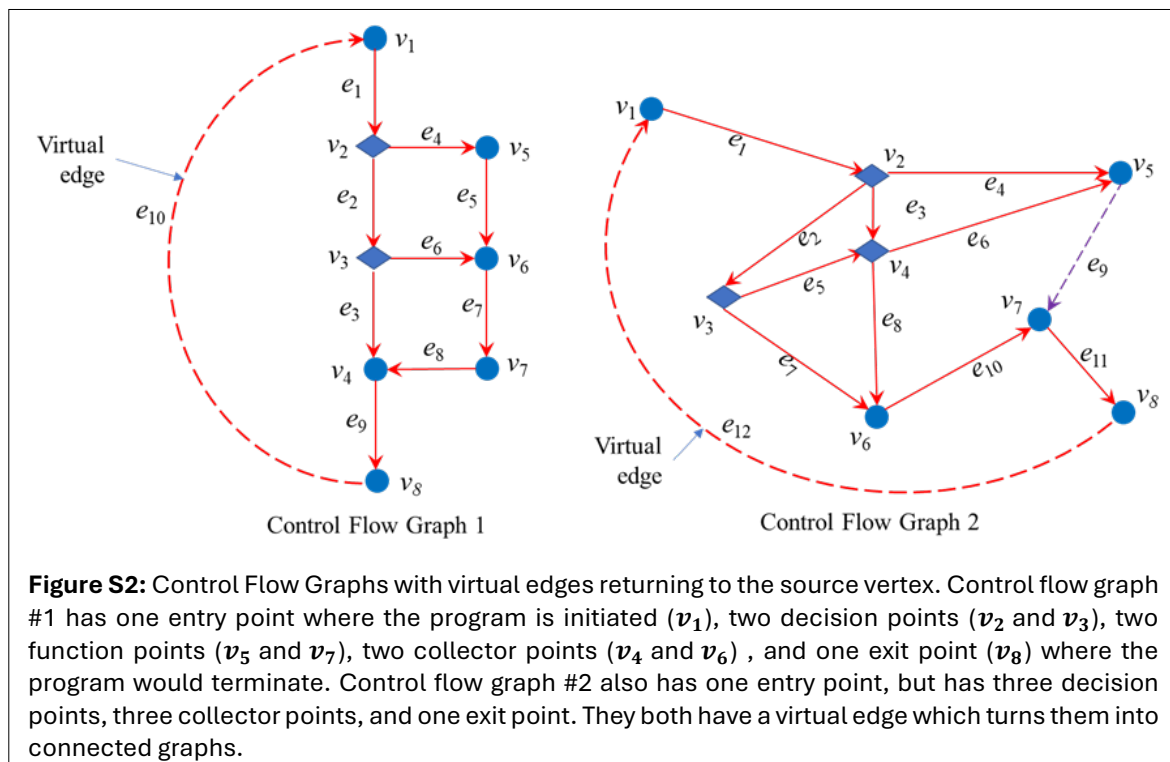
#### Step 4: Solve the System for the Hydraulic Conductivity

When all of the  $q_k^*$  coefficients have been sorted into the proper elements for every path, the redundant rows are removed from  $\mathbf{A}$ . The solution step for computing  $\mathbf{R}$  is accomplished using a function in Matlab that uses two measures of a least-norm solution, called "lsqminnorm" (18). The reciprocal of  $\mathbf{R}$  is computed, producing a column vector of grid cell hydraulic conductivities,  $\mathbf{K}$ , then  $\mathbf{K}$  is converted into an array of dimension  $(r \times c - 2)$ , where  $r$  is the number of rows in  $\mathbf{A}$  and  $c$  is the number of columns in  $\mathbf{A}$ . The first and last columns of  $\mathbf{K}$  are reflected across the  $x$ -direction boundaries, producing a final array of size  $r \times c$ , which matches the size of the closure of the model domain,  $\bar{\Omega}$ .

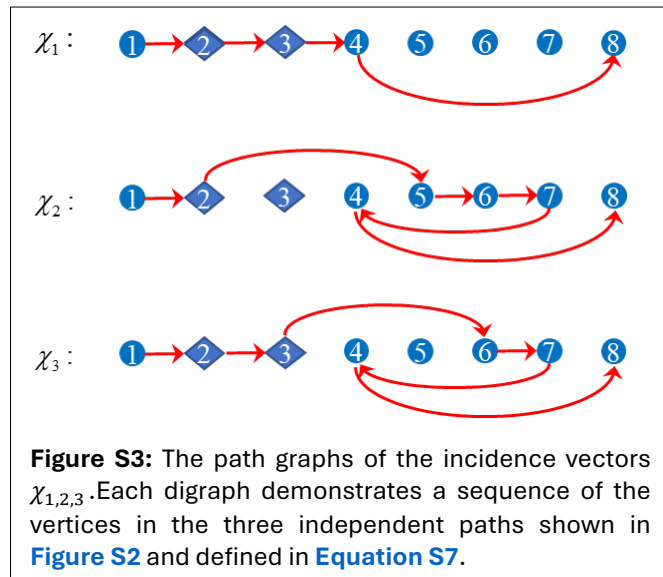
## 4. CYCLOMATIC COMPLEXITY

The maximum number of LI Paths in each subdigraph is equal to the cyclomatic complexity,  $C$ , (19), which is defined for weakly connected, single-source single-sink digraphs as (**Eq. S6**):

$$C = v(G) + 1 = m - n + 2 \quad (\text{S6})$$



where  $v(G)$  is the cyclomatic number, defined by Berge (1, p. 16) as the number of elementary cycles in a graph or digraph needed to form a cycle basis using the associated vectors for each cycle. The size of a path basis is the size of the cycle basis plus one; see, e.g., Gleiss et al. (10, p. 3). Since a basis of a vector space  $\mathbb{V}$  must span  $\mathbb{V}$  and be formed from linearly independent vectors (6, p. 182). The spanning property implies that a linearly independent set  $P$  of  $(s, t)$  paths  $\{p_1, p_2, \dots, p_n\}$  is a path basis in the single-source single-sink digraph  $H$  if every  $(s, t)$  path not belonging to  $P$  can be formed by a linear combination of other  $(s, t)$  paths in  $P$  (12).

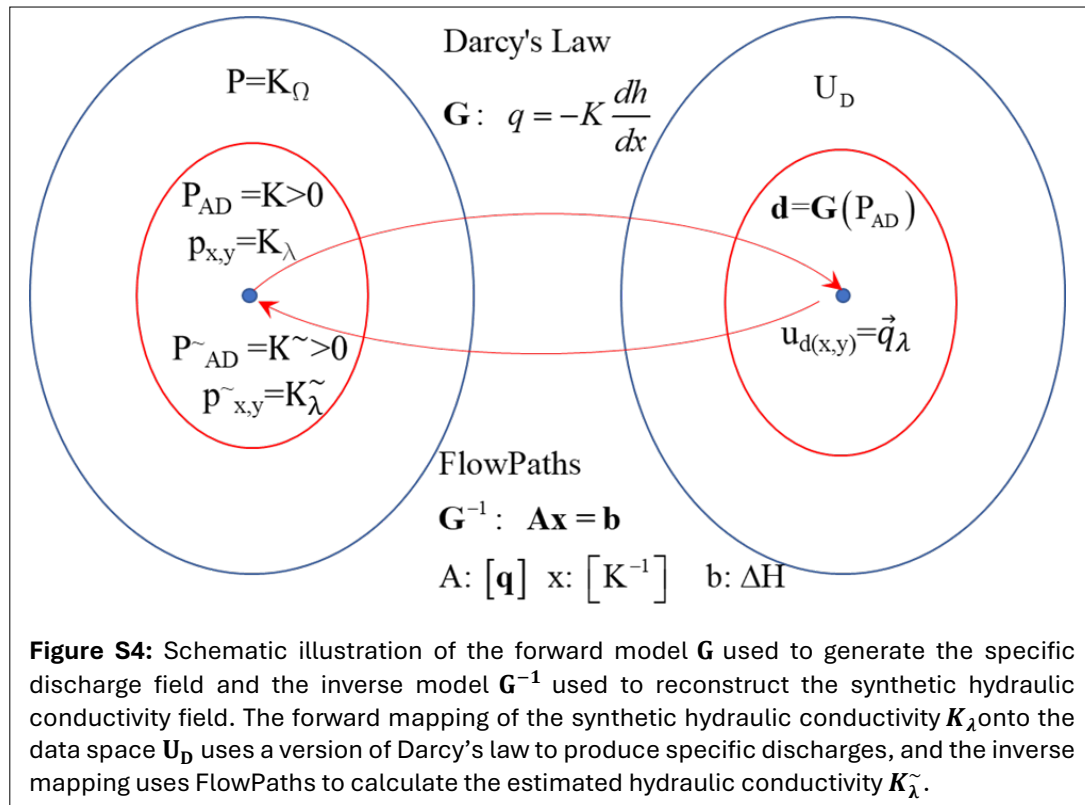


A corollary of Berge's definition of  $v(G)$  is that in a strongly connected graph  $G$ , the cyclomatic number is equal to the maximum number of independent cycles (1, p. 17). The result was a diagram called a Control Flow Graph (25) that added a virtual edge to the structured program. One of the corollaries of the definition given earlier is that every vertex in the Control Flow Graph has to be reachable from the entry vertex and each vertex can reach the exit vertex. Two examples are shown in Figure S2.

One can represent the information on a path basis either by listing the incident vertices or the incident edges. Each set of specific discharges (Fig. 3 in the main text) represents a multi-source multi-sink (MSMS) network as described by Deo (4, p. 478) and by Borradaile et al. (2). Paths are visualized as a set of alternating distinct vertices and edges starting with a vertex on the high head side (source vertices:  $s \in S$ ) and connecting to a vertex on the low head side (target vertices:  $t \in T$ ). For example, there are 69 simple, directed paths in the digraph of Figure 3b in the main text, determined using a "find all paths" search between all  $(s, t)$  pairs. This brute force approach to identify paths is certainly valid, but costly in terms of computer resources and time (11). To minimize this cost, the FlowPaths method takes advantage of a minimalistic approach to finding the linearly independent paths, which agrees with an approach taken by Vatinlen et al. (24) to solve a single  $(s, t)$  flow for forward problems. They found that, for a unique optimal solution, the incidence vectors of the paths must be linearly independent. Here, optimal means that the total flow value of the set of paths is optimized with respect to the feasible flows. The incidence vector  $\chi_P$  of a path  $P$  of the digraph  $D$  having a single source and single sink,  $D(s, t)$ , is a sequence of integers, such that if a vertex  $v$  is incident to an edge  $e$  on a path  $p$ , then it is assigned a one, zero otherwise. For the control flow graph of Figure S2, three such paths exist, consisting of the vertex sets  $\{1, 2, 3, 4, 8\}$ ,  $\{1, 2, 5, 6, 7, 4, 8\}$ , and  $\{1, 2, 3, 6, 7, 4, 8\}$  so the incidence vectors are (Eq. S7):

$$\begin{aligned} \chi_1 &= [1 \ 1 \ 1 \ 1 \ 0 \ 0 \ 0 \ 1]^T \\ \chi_2 &= [1 \ 1 \ 0 \ 1 \ 1 \ 1 \ 1 \ 1]^T \\ \chi_3 &= [1 \ 1 \ 1 \ 1 \ 0 \ 1 \ 1 \ 1]^T \end{aligned} \quad (S7)$$

The path graphs of the incidence vectors  $\chi_{1,2,3}$  may be shown in a canonical, or planar form as in Figure S3.



## 5. THE FORWARD AND INVERSE PROBLEM CYCLE

According to Tarantola (23, p. 10), solving an error-free forward problem means to perfectly predict the values of a set of observations  $\mathbf{d} = [d_1, d_2, d_3 \dots d_N]$  (an element of the data space  $D$ , where  $N$  is the number of observations) that came from some set of model parameters  $\mathbf{m} = [m_1, m_2, m_3 \dots m_M]$  (an element of the model space  $M$ , where  $M$  is the number of model parameters), using a forward operator,  $\mathbf{g}(\cdot)$ , which may or may not be linear, as (Eq. S8) which represents the set of equations  $d_i = g_i(m_j)$ .

$$\mathbf{d} = \mathbf{g}(\mathbf{m}) \quad (\text{S8})$$

If the operator is linear, as in groundwater modeling of a confined aquifer, then  $\mathbf{g}$  is a linear operator mapping from  $M$  (a vector) into  $D$  (also a vector). According to Menke (21), the discrete forward problem may be stated as shown in Equation S9 where the matrix  $\mathbf{G}$  is the representation of the mapping function.

$$d_i = \sum_{j=1}^M G_{ij} m_j \quad (\text{S9})$$

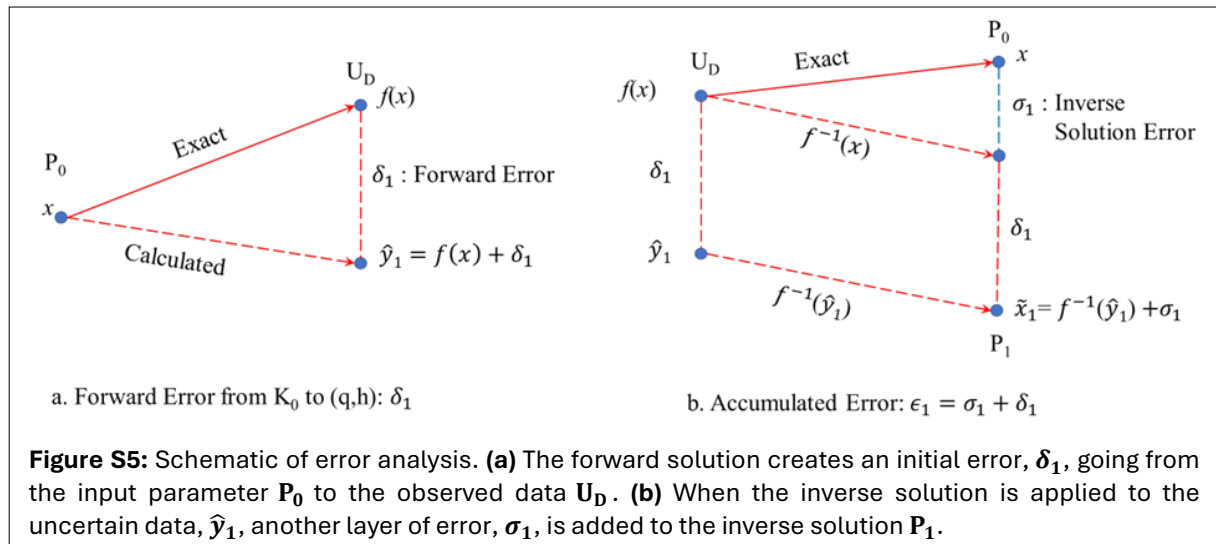
The forward mapping produces a matrix of discrete head values, and a matrix of intercellular flows. In FlowPaths, we have that the number of observations of specific discharges ( $\vec{q}_i$ ) is equal to the number of model parameters ( $K_j$ ), so  $i = j$  and  $N = M$ . The discrete inverse problem solution can then be stated as shown in Equation S10 or as in Equation S11 where  $\mathbf{m}_{\text{est}}$  is the estimated version of the true model.

$$\mathbf{m}_{\text{est}} = \mathbf{G}^{-1} \mathbf{d} \quad (\text{S10})$$

$$\mathbf{G} \mathbf{m}_{\text{est}} = \mathbf{d} \quad (\text{S11})$$

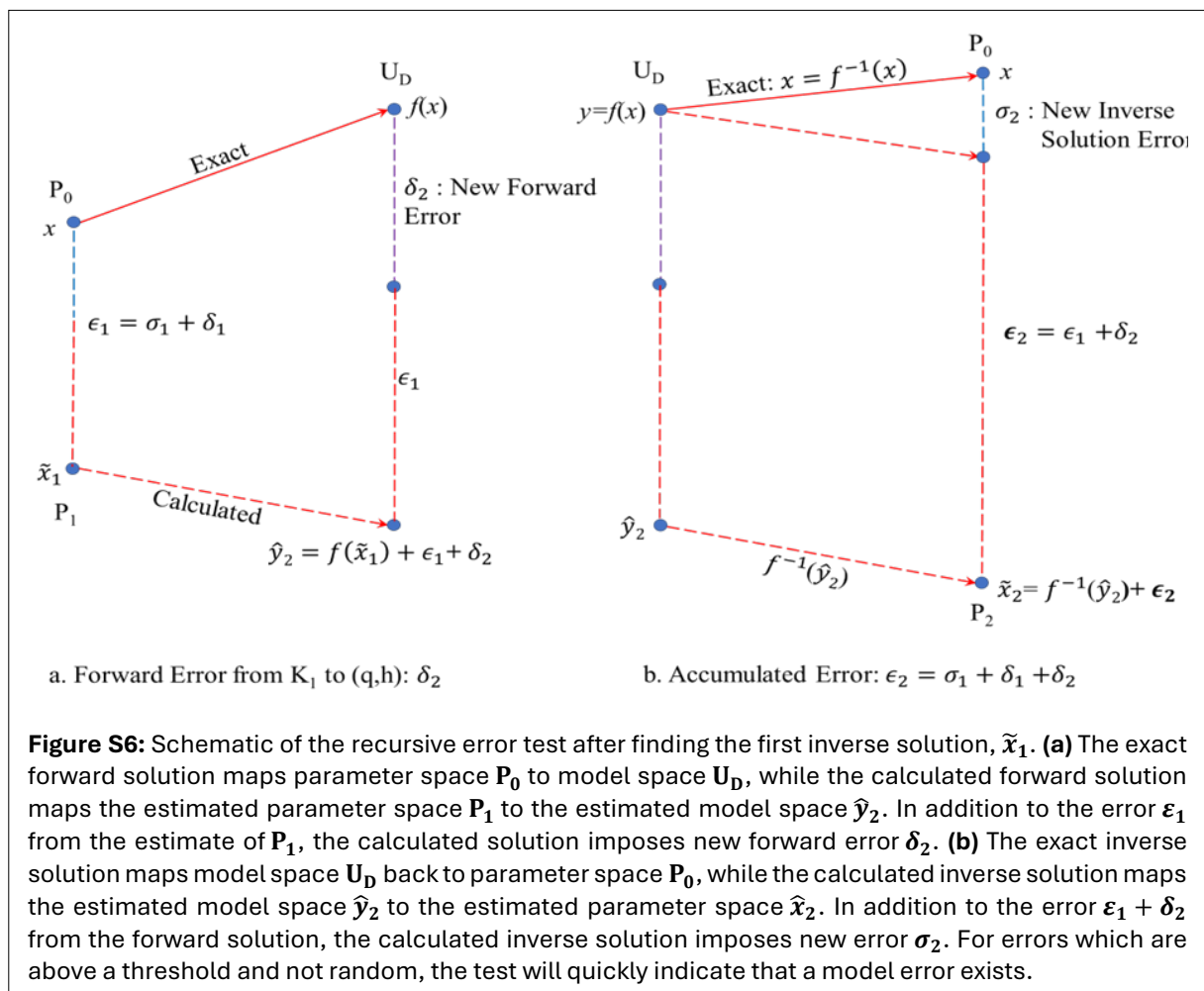
There are several possible models,  $(m_i)$ , that solve the forward problem for head at discrete locations, so that  $m_i \in M$ . Our forward solution of the groundwater flow equation uses the alternating direction implicit (ADI) method to calculate head in the domain ( $h_\Omega$ ) and to find the corresponding specific discharge field ( $\vec{q}_\lambda$ ). Conversely, our inverse method FlowPaths uses the specific discharge field  $\vec{q}_\lambda$  to compute the estimated parameter  $K_\lambda^\sim$ . Figure S4 shows the forward map, using Darcy's law, from the





parameter space  $P$  to the data space  $U_D$  and the complimentary inverse map, using FlowPaths, from the data space  $U_D$  to the estimated parameter space  $P^*$ . In particular, the admissible parameter space,  $P_{AD} = K_\Omega > 0$  is constrained to be strictly positive to produce the data  $u_{d(x,y)} = \tilde{q}_\lambda$  (20).

The mapping  $G$  will also contain error (Figure S5 and Figure S6). The admissible data,  $(P_{AD})$ , that is generated by the forward mapping is the head and specific discharge in cells of finite size,  $h(i,j)$ , and  $q(i,j)$ . The admissible set,  $P_{AD}$ , of  $K$  usually found in porous media is between  $10^{-11}$  cm/sec and  $10^2$  cm/sec (8). The admissible set of head observations,  $h_{ad}$ , is strictly bounded by the static head measurements on the two boundaries. The range of  $q_{ad}$  is as follows in Equation S12, where  $Q_{total}$  is the





total volumetric flow of water through the porous media,  $A_{total}$  is the total area of the porous media perpendicular to the macroscopic flow direction, and  $N_c$  is the number of cells perpendicular to the macroscopic flow direction. Thus, the maximum admissible specific discharge corresponds to the special case where all flow passes through a single cell (Eq. S12).

$$0 < q_{ad} < \frac{Q_{total}}{(1/N_c)A_{total}} \quad (S12)$$

## 6. 4X4 EXAMPLE DATA TABLES

For the 4×4 running example, the specific discharge entering cells from the left,  $q_L$ , is shown on Table S2; the specific discharge entering cells from the top,  $q_T$ , is shown in Table S3.

**Table S2:** For the 4×4 example, the specific discharge entering cells from the left,  $q_L$ .

|   | 1 | 2           | 3           | 4           | 5           | 6           |
|---|---|-------------|-------------|-------------|-------------|-------------|
| 1 | 0 | 1.233406316 | 1.389561673 | 1.440069412 | 1.318699565 | 1.154178457 |
| 2 | 0 | 1.607555994 | 1.527264142 | 1.577009080 | 1.709846106 | 1.852747944 |
| 3 | 0 | 1.623388820 | 1.448221000 | 1.374366672 | 1.471149851 | 1.636089196 |
| 4 | 0 | 1.196349797 | 1.295654112 | 1.269255763 | 1.161005406 | 1.017685330 |

**Table S3:** For the 4×4 example, the specific discharge entering cells from the top,  $q_T$  (negative denotes specific discharge leaving from the top).

|   | 1 | 2            | 3            | 4            | 5            | 6 |
|---|---|--------------|--------------|--------------|--------------|---|
| 1 | 0 | 0            | 0            | 0            | 0            | 0 |
| 2 | 0 | -0.156155357 | -0.050507739 | 0.121369847  | 0.164521108  | 0 |
| 3 | 0 | -0.075863505 | -0.100252677 | -0.011467179 | 0.021619270  | 0 |
| 4 | 0 | 0.099304315  | -0.026398349 | -0.108250357 | -0.143320076 | 0 |

## REFERENCES

- Berge, C. (1973). *Graphs and hypergraphs*. North-Holland Publishing Company.
- Borradaile, G., Klein, P. N., Nussbaum, Y., & Wulf-Nilsen, C. (2021). Multiple-source multiple-sink maximum flow in directed planar graphs in near-linear time. 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science, Palm Springs, CA, USA.
- Brown, R. L., Durbin, J., & Evans, J. M. (1975). Techniques for testing the constancy of regression relationships over time. *Journal of the Royal Statistical Society: Series B (Methodological)*, 37(2), 149-163. <https://doi.org/10.1111/j.2517-6161.1975.tb01532.x>
- Deo, N. (2016). *Graph theory with applications to engineering & computer science*. Dover Publications, Inc.
- Diestel, R. (2005). *Graph theory* (3 ed., Vol. 173). Springer.
- Farlow, J., Hall, J. E., McDill, J. M., & West, B. H. (2002). *Differential equations and linear algebra*. Prentice-Hall.
- Foulds, L. R. (Ed.). (1994). *Graph theory applications* (2 ed.). Springer-Verlag, Inc.
- Freeze, R., & Cherry, J. (1979). *Groundwater*. Prentice-Hall, Inc.
- Galpin, J. S., & Hawkins, D. M. (1984). The use of recursive residuals in checking model fit in linear-regression. *American Statistician*, 38(2), 94-105. <https://doi.org/10.2307/2683242>
- Gleiss, P. M., Leydold, J., & Stadler, P. F. (2005). Minimum path bases and relevant paths. *Networks*, 46(3), 119-123. <https://doi.org/10.1002/net.20080>
- Gosnell, D. K., & Broecheler, M. (2020). *The practitioner's guide To graph data: Applying graph thinking and graph technologies to solve complex problems* (N. Barber, Ed. 1 ed.). O'Reilly Media, Inc.
- Green, E. L. (1983). Graphs with relations, coverings and group-graded algebras. *Transactions of the American Mathematical Society*, 279(1), 297-310. <https://doi.org/10.2307/1999386>
- Higham, N. J. (2002). *Accuracy and stability of numerical algorithms* (2 ed.). Society for Industrial and Applied Mathematics. <https://doi.org/10.1137/1.9780898718027>
- Lin, D. Y., Wei, L. J., & Ying, Z. (2002). Model-checking techniques based on cumulative residuals. *Biometrics*, 58(1), 1-12. <https://doi.org/10.1111/j.0006-341X.2002.00001.x>

15. MathWorks. (2020a). *Row-major and column-major array layouts - MATLAB & Simulink*. <https://www.mathworks.com/help/coder/ug/what-are-column-major-and-row-major-representation-1.html>
16. MathWorks. (2020b). *Shortest path between two single nodes - MATLAB shortestpath*. Retrieved 10/15/20 from <https://www.mathworks.com/help/matlab/ref/graph.shortestpath.html>
17. MathWorks. (2022a). *Cusum test for structural change - MATLAB cusumtest*. Retrieved 2/1/23 from <https://www.mathworks.com/help/econ/cusumtest.html>
18. MathWorks. (2022b). *MATLAB lsqminnorm*. The MathWorks Inc. <https://www.mathworks.com/help/matlab/ref/lsqminnorm.html>
19. McCabe, T. (1976). A complexity measure. *IEEE Transactions on Software Engineering, SE-2*(13). <https://doi.org/10.1109/TSE.1976.233837>
20. McLaughlin, D., & Townley, L. R. (1996). A reassessment of the groundwater inverse problem. *Water Resources Research, 32*(5), 1131-1161. <https://doi.org/10.1029/96wr00160>
21. Menke, W. (2012). *Geophysical data analysis: Discrete inverse theory*. Academic Press.
22. Sedgewick, R., & Wayne, K. (2020). *Shortest paths algorithms*, 4th Edition. Retrieved 10/5/2020 from <https://algs4.cs.princeton.edu/44sp/>
23. Tarantola, A. (2005). *Inverse problem theory and methods for model parameter estimation*. Society for Industrial and Applied Mathematics.
24. Vatinlen, B., Chauvet, F., Chrétienne, P., & Mahey, P. (2006). Simple bounds and greedy algorithms for decomposing a flow into a minimal set of paths. *European Journal of Operational Research* (185), 1390-1401. <https://doi.org/10.1016/j.ejor.2006.05.043>
25. Watson, A. H., Wallace, D. R., & McCabe, T. J. (1996). *Structured testing: A testing methodology using the cyclomatic complexity metric*. NIST Special Publication 500-235, National Institute of Standards and Technology, Gaithersburg, MD, USA. <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication500-235.pdf>
26. Weisstein, E. W. (2022). *Maximally linearly independent*. Retrieved from <https://mathworld.wolfram.com/MaximallyLinearlyIndependent.html>
27. Zijl, W., De Smedt, F., El-Rawy, M., & Batelaan, O. (2018). *The double constraint inversion methodology*. Springer Nature. <https://doi.org/10.1007/978-3-319-71342-7>.