A Matrix Inverse Algorithm based on Bi-Directional Graph Search

Gabriel M. Perry, Micah H. Olson, David Grimsman, and Sean Warnick

***Abstract*— We present an algorithm to estimate a single entry of the inverse of a matrix; it is derived using a bidirectional search on a flow graph. This method has immediate application in analyzing dynamical system destabilization attacks, where previous research quantifies system node-to-node vulnerability in terms of a matrix inverse. We present evidence that the expected ∞-norm error, the number of floating point operations, and the number of unique entries used to complete the computation are comparable to existing row or column approximation methods.**

I. Introduction

This work is focused on computing a single entry from the inverse of a matrix. Such computations are useful in systems analysis, especially in signal gain between specific nodes; it is also necessary for analyzing system vulnerability to destabilization attacks [1]. In most matrix applications (as when solving systems of equations), at least an entire row or column is needed, so few existing methods improve the case where only a single entry is needed from the inverse matrix. To approach this issue, we utilize an existing bridge between the matrix inverse problem and graph theory to frame matrix inversion as a graph search. We also address the issue of some matrices being much too large to read or perform operations on in a reasonable time, specifically taking advantage of sparsity to achieve this. We then apply the existing bidirectional search algorithm on the graph search instance, which by reduction produces the desired matrix inverse entry.

In Section I-A, we review existing matrix inverse methods and their relation to graph theory. We specifically focus on the Taylor series, which is closely related to our method. In Sec. III, we present our method, which uses a bidirectional search over a graph to compute a single entry of a matrix's inverse. In Sec. IV, we provide bounds on the complexity, error, and memory usage of our algorithm. Memory usage is particularly noteworthy; as the matrix/data in question is assumed to be too large for practical use, we attempted to reduce the scope of the computation to a relatively small subset of the original matrix. We suggest that trade-offs exist that utilize the memory hierarchy in more efficient ways, and this question motivated our method.

We implemented our algorithm along with a few other graph-based algorithms mentioned in the following section; our code is available in [2]. We exerted little effort to optimize our implementations and appreciate academic feedback on improving our implementations.

All authors are associated with the Computer Science Department, Brigham Young University, Provo, UT, USA. The corresponding author can be reached at gmp99@byu.edu. This work is supported by DOE Grant #SC0021693.

A. Related Work

Some existing algorithms that have interpretations over signal-flow graphs similar to our method include:

1) Gaussian Elimination: Gaussian elimination is a fundamental method and is often introduced as a naïve method of entire matrix inversion. It runs in $O(n^3)$ time and is often stable enough for reasonable instances, though, in general, it is not stable. It is equivalent to the Floyd-Warshall Algorithm, a versatile algorithm that computes net flow over a given algebra. This is why it computes $(I-Q)^{-1}$ when using the standard plus-times algebra, as this is the net flow across Q. Other applications include finding shortest paths in a graph (min-Tropical algebra), transitive closures (Boolean algebra), and regular expressions (Finite Automata) [3].

2) Taylor Series: If Q is convergent, the power or Taylor series approximation is:

$$G = M^{-1} = (I-Q)^{-1} = I + Q + Q^2 + Q^3 + ...$$

Or, if we wish only to compute the jth column of M^{-1},

$$M^{-1}_{\cdot j} = \vec{e}_j + Q\vec{e}_j + Q^2\vec{e}_j + Q^3\vec{e}_j + ... \tag{1}$$

where $\vec{e}_j$ is the jth column of the identity matrix. A similar equation can be used to compute a row of the inverse. A physical interpretation of these series is that each term Q^k measures the flow of all the paths of length k, and the sum of all these contributions approaches the net flow through the graph. This is a breadth-first computation since we perform all computations of a single depth before moving on to the next term.

3) Mason's Gain Formula and Cramer's Rule: Mason's gain formula computes a single entry of M^{-1} by considering individual signal-flow paths in Q and summing them together [4], [5], [6]. This search explores the signal-flow graph using a depth-first search; this is equivalent to Cramer's Rule. Using dynamic programming to reduce redundant path computations reproduces Gaussian Elimination.

We approach the problem of single-entry calculation within the context of this connection to signal-flow graph searches and design a novel algorithm based on bidirectional search [7].

II. Definitions

A. Flow Graphs

Consider a simple directed graph $H = (V, E, L)$ with sets of vertices V, edges $E \subseteq V \times V$, and edge labels $L \in \mathbb{F}^{|E|}$ for some field $\mathbb{F}$. We denote the matrix Q as the adjacency matrix of H: for edge $(i,j) \in E$ with label l, $q_{ij} := l$, and $q_{ij} = 0$ when $(i,j) \notin E$. A *path* in the graph is a (possibly

infinite) sequence of edges $r = (e_1, e_2, \dots)$ such that if $e_k = (a_k, b_k)$ and $e_{k+1} = (a_{k+1}, b_{k+1})$, then $b_k = a_{k+1}$. The *flow* along the path r is the product of the edges labels in r, that is, $\prod_{k=1}^{|r|} l_k$, where l_k is the edge label e_k.

Definition 1: For a graph $H = (V, E, L)$ the *net flow* or net transfer g_{ij} from node j to node i is the sum of the flows across every path in H that begins at j and ends at i. More precisely stated,

$$g_{ij} := \sum_{r \in R_{ij}} \prod_{k=1}^{|r|} l_k$$

where R_{ij} is the set of all paths that begin at j and end at i.

Denote by $G = [g_{ij}]$ a matrix of net flows on graph H. The previous work in [8] has established the relationship $G = (I - Q)^{-1}$, where the matrix Q is the adjacency matrix for the signal-flow graph H (though over $\mathbb{R}$, we refer to it simply as a flow graph). Net flow is closely related to the transfer function of LTI systems theory, where $\mathbb{F}$ is the set of rational polynomials. In that case, the transfer function G represents the system's input-output behavior.

The key insight is that if we understand M to be $I - Q$ for some corresponding flow graph H, then computing M^{-1} reduces to computing the net flows for each pair of nodes, i.e., by computing G. This sum of flows across all paths is equivalent to the transfer function in LTI systems theory, and H is analogously a signal-flow graph (or, over $\mathbb{R}$, a flow graph). Net flow is also defined in [8] as *net effect*.

B. Convergent Matrices

Let $M \in \mathbb{F}^{n \times n}$ be the matrix we desire to invert. We restrict our consideration to a specific set of matrices $\mathscr{M}$ such that $\forall M \in \mathscr{M}$, the eigenvalues of M all lie within an open half-plane not containing zero.

Remark 1: Using a precomputation scale factor αM to achieve the convergence criteria for $Q = I - \alpha M$, it is necessary and sufficient that the eigenvalues of M be strictly contained in some open half of the complex plane not also containing the origin. Noting that $(I - Q) = \alpha M$, so $M^{-1} = \alpha (I - Q)^{-1}$, we see that such a scale factor α has the effect of shrinking and rotating the eigenvalues down to the open disk centered at 1. As $\rho(I - \alpha M) = \rho(Q) < 1$, Q is convergent. Necessity is obvious as no scaling or rotation can place all eigenvalues in an open half-plane unless they began in one. We note that this precomputation scaling requires us to know *a priori* an upper bound on the eigenvalues of M, as well as the open half-plane containing them. Also, scaling affects the numerical error of inversion algorithms.

The above remark implies that $\mathscr{M}$ includes positive and negative definite matrices, as well as Hurwitz matrices (for stable CT LTI systems). Convergence is accelerated if the eigenvalues of Q are close to zero, corresponding to αM being close to the identity matrix. This includes strongly diagonally dominant matrices with diagonal entries relatively close to one.

C. Epsilon Horizons

A natural method for approximation is to round insignificant terms to zero. For example, one could approximate (1) by rounding each term in the sum before continuing. In the context of flow graphs, this is equivalent to limiting the search to paths with flow above some tolerance $\varepsilon > 0$.

Definition 2: For a node $j \in V$, the outgoing *ε-horizon* is the subset of nodes i such that $g_{ij} > \varepsilon$. The incoming ε-horizon is the subset of nodes i such that $g_{ji} > \varepsilon$.

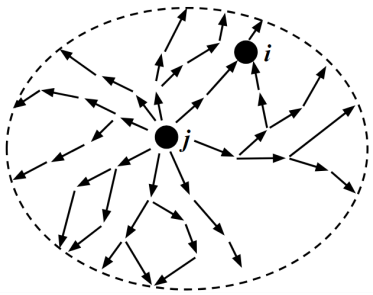

Fig. 1. The ε-horizon of a companion graph H with adjacency matrix $Q = I - M$. Beginning at source j, we follow the network flow out to magnitude ε, and consider only nodes within this horizon as computationally pertinent. If our target node i is not in the horizon, $M_{ij}^{-1} \approx 0$.

This truncation minimizes the exploration of paths, allowing us to focus our view within some ε-horizon—ideally, a set much smaller than all nodes (see Fig. 1). Using this constraint provides the following bound on the number of entries read from the matrix, assuming sparsity; a proof of this bound is given in Appendix-A.

Theorem 1: The number of nodes p within the ε-horizon of node j is bounded[1] by:

$$p \in O\left(\exp\left(\frac{\ln(s)\ln(\varepsilon)}{\ln(\rho)}\right)\right) \tag{2}$$

where

- $\exp(x)$ is the exponential function e^x (note that the bound is not exponential; this notation is for brevity)
- $\varepsilon \geq 0$ is the tolerance or approximation factor
- $s \geq 1$ is the maximum number of nonzero entries in any column of Q, or the maximum branching factor of the flow graph H (or $s = \|Q\|_0$, the induced zero "norm")
- $\rho = \|Q\|_1$ is the induced one norm of Q (this provides an upper bound on the spectral radius $\rho(Q) \leq \rho$)

A high-level understanding of (2) is that the worst-case size p of the ε-horizon shrinks when:

- ε increases (more path truncation)
- s decreases (less branching)
- ρ decreases (paths decay quickly)

We note that correctly rejecting low-flow paths greedily (no path later becomes significant) requires the entries of Q to have magnitudes < 1, which is implied when $\rho < 1$.

[1]This bound is conservative and assumes that every branch produces a new node; in real-world data, branches tend to stay within communities of nodes, either due to limitations of physical distance or other factors, which reduces p and thus the number of entries we must read from M.

Example 1: To illustrate this idea, consider the following matrix $M \in \mathbb{R}^{4\times4}$, which we desire to invert:

$$M = \begin{bmatrix} 1 & -c & 0 & 0 \\ 0 & 1 & -b & 0 \\ 0 & -e & 1 & -a \\ -d & 0 & 0 & 1 \end{bmatrix}$$

The corresponding signal-flow graph H (Fig. 2) has adjacency matrix $Q = I - M$:

$$Q = \begin{bmatrix} 0 & c & 0 & 0 \\ 0 & 0 & b & 0 \\ 0 & e & 0 & a \\ d & 0 & 0 & 0 \end{bmatrix} \tag{3}$$

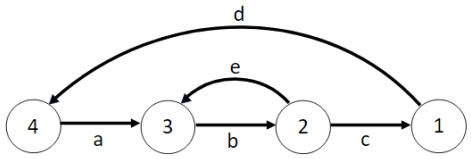

Fig. 2. The flow graph H given in (3). The edge labels a, b, c, d, e are elements of some field (rational functions in LTI systems, $\mathbb{R}$ in this work).

The inverse of interest M^{-1} is:

$$\frac{1}{\det(M)} \begin{bmatrix} (1-be) & c & bc & abc \\ dab & 1 & b & ab \\ da & e+cda & 1 & a \\ d(1-eb) & cd & bcd & (1-be) \end{bmatrix} \tag{4}$$

Where $\det(M) = 1 - be - abcd$, and the matrix is $\text{adj}(M)$, the adjugate or classical adjoint of M.

We use a Taylor series (1) to illustrate ε-horizons. Suppose we wish to estimate the column $M_{\cdot 2}^{-1}$. We start by using the estimate $\vec{u} = \vec{e}_2 = (0,1,0,0)^\top$; this encodes all path flows of length zero (the source node 2 starting at itself). We then multiply by Q on the left, producing $Q\vec{e}_2 = (c,0,e,0)^\top$, which encodes flow paths of length 1 starting from source 2 (c to 1, and e to 3).

Suppose that $c < \varepsilon$; then we would round this path to zero, removing node 1 from the ε-horizon and updating our estimate $Q\vec{e}_2 \approx (0,0,e,0)^\top$. Our new estimate is the running sum, $\vec{u} = (0,1,e,0)^\top$. We repeat this process, giving $Q^2\vec{e}_2 \approx (0,be,0,0)^\top$, $Q^3\vec{e}_2 \approx (0,0,be^2,0)^\top$, etc. In the limit, these terms sum to:

$$\vec{u} \to \begin{bmatrix} 0 \\ 1+be+(be)^2+... \\ e+be^2+b^2e^3+... \\ 0 \end{bmatrix} = \frac{1}{1-be} \begin{bmatrix} 0 \\ 1 \\ e \\ 0 \end{bmatrix}$$

This is exactly the estimate of $M_{\cdot 2}^{-1}$ we desire (as $c < \varepsilon$, all terms involving c are dropped). Of course, we will not compute this exactly, as we will terminate the sums once the working term $Q^i\vec{e}_j$ has magnitude $< \varepsilon$.

Algorithm 1 Bidirectional Matrix Inversion

Require: A matrix M satisfying the mentioned assumptions
indices of the target entry (j,i) of M^{-1},
cutoff tolerance $\varepsilon > 0$.
procedure BIDIRECTIONALINVERSE($\mathrm{M}, j, i, \varepsilon$)
  Set $0 < \varepsilon_{sink}, \varepsilon_{source} < \varepsilon$ s.t. $\varepsilon_{sink}\varepsilon_{source} = \varepsilon$
  $\vec{s}, \vec{w} \leftarrow \vec{e}_i$
  $inHoriz \leftarrow$ set of indices
  loop while $\|\vec{w}\| > \varepsilon_{sink}$ ▷ Find the sink horizon
    $\vec{w} \leftarrow (I - M)^\top \vec{w}$
    $\vec{s} \leftarrow \vec{s} + \vec{w}$
    round entries of $\vec{w}$ below ε to 0
  end loop
  $inHoriz \leftarrow$ non-zero indices of $\vec{s}$
  $\vec{u}, \vec{w} \leftarrow \vec{e}_j$
  loop while $\|\vec{w}\| > \varepsilon_{source}$ ▷ Find the source horiz.
    $\vec{w} \leftarrow (I - M)\vec{w}$
    $\vec{u} \leftarrow \vec{u} + \vec{w}$
    round entries of $\vec{w}$ below ε to 0
  end loop
  loop while $\|\vec{w}\| > \varepsilon$ ▷ Continue in the sink horizon
    $\vec{w} \leftarrow (I - M)\vec{w}$
    $\vec{u} \leftarrow \vec{u} + \vec{w}$
    round indices of $\vec{w}$ not in $inHoriz$ to 0
    round entries of $\vec{w}$ below ε to 0
  end loopreturn $\vec{u}_i$
end procedure

III. Bi-Directional Search

In this section, we present our main result: the Bi-Directional Inverse (Alg. 1) for computing a single entry of a matrix inverse. We begin with an example; see Fig. 3 and its caption for an illustration.

Example 2: We use the same matrix M from Sec. II-C. Suppose that we wish to estimate row 3, column 2 of M^{-1} (the term $(e + cda)/\det(M)$). We begin at node 3 and execute a priority BFS (rounded power series) on the reverse graph (using the different ε values as noted earlier), producing an estimate of row 3. We start with $\vec{w} = (0,0,1,0)^\top$, iteratively multiplying by $Q^\top$, adding the new term to $\vec{s}$, and rounding $\vec{w}$'s entries. We produce the row vector:

$$\vec{s} = (0, [e + ebe + e(be)^2 + ...], [1 + be + (be)^2 + ...], 0)$$

from which c is absent due to our construction $c < \varepsilon$ used in Example 1, and (for illustrating this search) we now assume $a < \varepsilon$, so a is also omitted. From this estimate, we see that only nodes 2 and 3 will be within the in-horizon of the destination node 3 (we otherwise ignore the sum's values).

We then execute the same search flowing forward from node 2, starting with $\vec{u} = \vec{e}_2$ as in Example 1. We flow until $\|\vec{w}\|$ reaches the modified threshold ε_{source}, at which point we round all values *except at nodes 2 and 3* to zero (as these nodes are in the in-horizon of the sink node 3). We continue the computation of the forward flow, rounding not

only the entries below ε but also all the entries not in node 3's in-horizon. For this small example, we end with the same estimate as in Example 1, but in general, the estimates may differ.

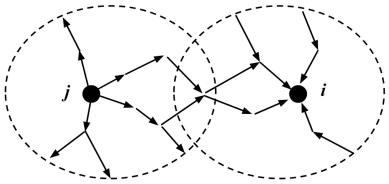

Fig. 3. An illustration of the two ε-horizons of Bidirectional Inverse. The stages (loops) of the algorithm are: 1) we find all nodes within the in-horizon of the sink i (right oval), 2) we explore and compute all paths within the out-horizon of source j (left oval), and 3) once we reach the edge of that horizon, we only continue computations within the in-horizon of i.

To clarify, when computing M_{ij}^{-1}, we first approximate the ith row of M^{-1} using the relatively large ε_{sink} as our vector-term-rounding threshold; we still use ε for intermediate rounding. The value $\varepsilon_{sink} = \varepsilon_{source} = \sqrt{\varepsilon}$ minimizes the expected number of nodes in the union of both horizons. We mark all non-zero entries of the row i estimate as "within" i's in-horizon by saving them in a set.

We then perform the second loop, a power series estimate for the jth column of the forward flow network starting from the source j, using the final threshold ε for intermediate entry rounding, but ε_{source} on the current vector term to break.

The third loop begins once the current current vector term $\vec{w}$ reaches magnitude ε_{source}. We truncate all entries of $\vec{u}$ not in the ε_{sink} in-horizon of node i We only continue computations within the in-horizon, as only these paths will reach the sink with a value greater than ε.

IV. Complexity and Error

We ran our method in a Python script [2] for matrices of various sizes and sparsities. We generated matrices using a spectral radius $\rho(Q) = 0.7$ and sparsities $s = 2, 5, 20, 500$. We used the optimal $\varepsilon_{source} = \varepsilon_{sink} = \sqrt{\varepsilon}$. Note that the standard power series for columns is the edge case $\varepsilon_{source} = \varepsilon$, and for rows it is $\varepsilon_{sink} = \varepsilon$.

A. Error

Theorem 2: Assuming balanced source and sink εs, the error of Alg. 1 is bounded by (using the induced 1-norm):

$$\|err\|_1 \leq \|M^{-1}\|_1 \cdot \left(\frac{\ln \varepsilon}{\ln \rho} s\varepsilon\sqrt{p} + \frac{\ln \varepsilon}{2 \ln \rho} \rho\sqrt{\varepsilon} + \rho\varepsilon \right)$$

where the actual column $M_{\cdot j}^{-1} = \vec{u} + err$. Note that $\|M^{-1}\|_1$ is highly sensitive to a high conditioning number. Also note the difference between the horizon sizes p and the parameter ρ from (2). See Appendix-B for the proof. See also Fig. 4.

B. Runtime

For $s > 3$, the bidirectional inverse consistently uses twice the number of FLOPs used in the power series with ε-horizon rounding. Both approach a maximum number of required

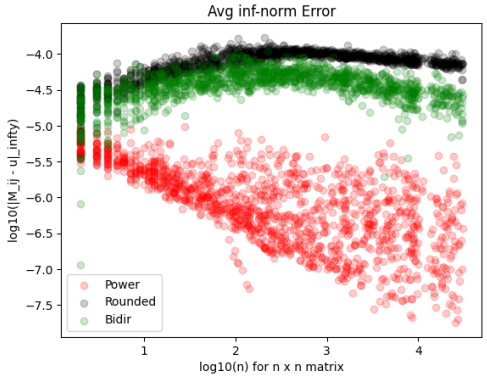

Fig. 4. Our method consistently has on average half the absolute error compared to a strictly forward computation, though these simulations suggest that this may improve with larger n (we were unable to invert matrices larger than this in a reasonable amount of time). Note that a simple power series has the best accuracy, as it uses less rounding. Note also that our method only measures the error of the single entry $|M_{ij}^{-1} - \vec{u}_i|$, while the others measure the maximum error of the entire column $\|M_{\cdot j}^{-1} - \vec{u}\|_\infty$.

FLOPs for a given s, whereas the power series without an ε-horizon requires FLOPs proportional to n. See Fig. 5.

The maximum number of nodes within the union of the horizons is $\in O(\sqrt{p})$, where p is the edge-case ε-horizon bound in (2). This bound is easily derived by substituting $\sqrt{\varepsilon}$ into (2) for each horizon, using exponent and logarithmic identities to move the $1/2$ from the ε out to surround the entire expression, and noting that the two horizons are at worst disjoint. We can use (2) in this generalization because in the worst case, bidirectional inverse algorithms do not utilize intermediate rounding (which is always ε), so using only the terminal rounding (at $\sqrt{\varepsilon}$) gives the desired bound.

When the convergence criteria are met, the power series with rounding achieves a time complexity of $O(slp)$, where l, the number of iterations, $\leq \frac{\ln \varepsilon}{\ln \rho}$ (see Appendix-A), and s and p are consistent with (2). Each of the l matrix-vector multiplication takes $O(sp)$ time using adjacency lists (since $\vec{u}$ has at most p non-zero entries and each column of $Q = I - M$ has at most $s+1$ non-zero entries). This implies that the space complexity of the bidirectional search is $O(sl\sqrt{p})$.

C. Memory Hierarchy

Decreasing the number of unique columns requested from memory could provide additional computational speed, due to the memory hierarchy in modern computers; this is due to more opportunities for cache hits, requiring fewer slow disk retrievals. This improvement in space complexity may reduce the number of entries of M we need to read from memory, which could lead to more cache hits and fewer disk fetches during execution. See Fig. 6.

V. Conclusion

We used an existing relationship between matrix inversion and network flow to transform a bidirectional search into an approximate algorithm producing a single entry of a matrix inverse. This uncovered a subset of the corresponding flow graph, an epsilon horizon, that contains the most important

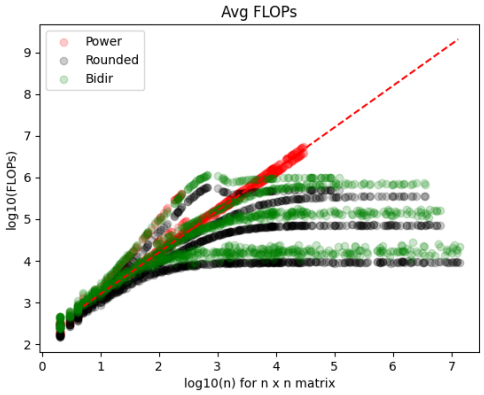

Fig. 5. From the bottom, FLOPs used when $s = 2, 5, 20, 500$. Note the behavior of $s = 500$, achieving its maximum near $n = 500$. Our method consistently uses twice the number of FLOPs for each n, s combination, with the standard deviation decreasing as s increases. Actual runtime and FLOPs were correlated with $r^2 = 0.99578$.

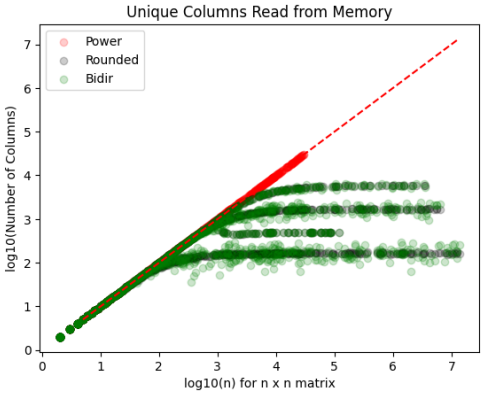

Fig. 6. The number of columns actually requested from memory to execute Alg. 1, not including duplicates. Sparcities shown are, from the bottom, $s = 2, 500, 5, 20$. Note that $s = 500$ requested fewer columns than the others. Also, note that the power series (BFS) without rounding used all columns throughout this size range.

entries for the inverse computation (according to some tolerance ε). We showed that our algorithm generally requires reading fewer entries of the original matrix to perform its computations, which may improve real-world performance by reducing cache misses during execution.

A. Future Work

We wish to analyze various matrix preconditioning, such as using D-scaling (similar to the structured singular value bound in [9], [10]). This would immediately extend the set $\mathscr{M}$ to include all strictly diagonally dominant matrices, perhaps more.

We are also interested in using the precomputed net flow from each node in the sink-horizon to the sink. Currently, these values (from loop one) are thrown out, and only the set of nodes is used to continue the computation, but one could imagine a scheme using these known net-flow-to-sink values as soon as any forward flow reaches the node.

We could consider an extension computing multiple entries, where all source and sink nodes contribute a $\sqrt{p}$ ε-horizon, and computations are performed efficiently for all such flow paths within the union of all these horizons.

There are also other algorithms for traversing flow graphs; for example, it may be useful to consider the necessary conditions in certain data for H to be planar or obey the triangle inequality. This would open the possibility of using some heuristic in an A^* search [11] of the flow graph to prioritize only the most significant flow paths in H, providing a worse but faster estimate of the inverse entry.

A practical extension of bidirectional inversion is in estimating signal transfer in dynamical systems, which is one focus of our research lab. This would use matrices over the field of rational functions of a single variable, instead of the real numbers. Other algebraic semirings may be of interest to other disciplines; these connections would be similar to those explored in [3], and we are interested in physical interpretations of these flow computations.

References

[1] V. Chetty, N. Woodbury, E. Vaziripour, and S. Warnick, "Vulnerability analysis for distributed and coordinated destabilization attacks," in *53rd IEEE Conference on Decision and Control*. IEEE, 2014, pp. 511–516.

[2] G. Perry and M. Olson, "Colaboratory," 2024, verified: 2024-02-01. [Online]. Available: https://colab.research.google.com/drive/1jiLIK1T8VcDwLznHHlevgNggpsePX_fS

[3] D. J. Lehmann, "Algebraic structures for transitive closure," *Theoretical Computer Science*, vol. 4, no. 1, pp. 59–76, 1977.

[4] C. Coates, "Flow-graph solutions of linear algebraic equations," *IRE Transactions on circuit theory*, vol. 6, no. 2, pp. 170–187, 1959.

[5] G. Samuel, M. Pollatschek, and E. Kehat, "Inversion of sparse matrices by a method based on graph theory," *Computers & chemical engineering*, vol. 11, no. 6, pp. 763–768, 1987.

[6] S.-L. Jeng, R. Roy, and W.-H. Chieng, "A matrix approach for analyzing signal flow graph," *Information*, vol. 11, no. 12, p. 562, 2020.

[7] I. Pohl, "Bi-directional search, machine intelligence 6," *Edinburgh University Press, Edinburgh*, vol. 127, p. 14, 1971.

[8] C. A. Johnson, "Net path and net effect of linear networks with dynamics," Ph.D. dissertation, Brigham Young University, 2017.

[9] J. Doyle, "Analysis of feedback systems with structured uncertainties," *IEE Proceedings D Control Theory and Applications*, vol. 129, no. 6, p. 242–250, Nov 1982, funding by Honeywell Incorporated.

[10] A. Packard and J. Doyle, "The complex structured singular value," *Automatica*, vol. 29, no. 1, pp. 71–109, 1993. [Online]. Available: https://www.sciencedirect.com/science/article/pii/000510989390175S

[11] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

Appendix

A. Proof of Theorem 1

We first find a bound on the length l of the longest path from a node in Q to its ε-horizon. We observe that the longest possible path occurs when the flow does not spilt (i.e. columns have one nonzero entry), and ends after l edges when the path flow is strictly below ε. This means that all paths having a flow above ε (contained within the ε-horizon) must have a bounded path length (recall $\rho < 1$):

$$\rho^l > \varepsilon \implies l < \frac{\ln \varepsilon}{\ln \rho} \tag{5}$$

We also know that the maximum number of nodes in the ε-horizon is bounded by an exponentially branching flow

using all s outgoing edges:

$$p \leq 1 + s + ... + s^l \in O(s^l) \implies p \in O(s^{\frac{\ln \varepsilon}{\ln \rho}}) \quad (6)$$

as $O(s^l)$ is a subset of (6) due to the bound (5). Using some exponent and logarithm identities gives (2). This bound is highly conservative, as l will decrease when paths branch. This path length l is the number of iterations of Alg. 1. ■

B. Proof of Theorem 2

We begin by partitioning the exact solution (1) (noting $M = I - Q$) into three stages of error-introduction:

$$\begin{aligned} M^{-1}\vec{e}_j &= \vec{e}_j + Q\vec{e}_j + Q^2\vec{e}_j + ... \\ &= \vec{e}_j + \sum_{k=1}^{j} Q^k\vec{e}_j + \sum_{k=1}^{i} Q^{k+j}\vec{e}_j + \sum_{k=1}^{\infty} Q^{k+j+i}\vec{e}_j \end{aligned}$$

The three sums correspond with the second and third loops and remainder after termination of Alg. 1, respectively. We overload j to denote the longest path from the source j to its out-horizon (i.e. j is the number of iterations of loop 2). We also overload i in this proof to mean the number of iterations performed in loop 3. Throughout this proof, we are careful to distinguish vectors (lowercase) from matrices (uppercase).

We will rewrite the exact partitioned power series according to the terms kept and those thrown out due to rounding and path truncation. We define the following recurrence relations:

$$\begin{aligned} q^{(0,0)} &= \vec{e}_j,\ \delta^{(0,0)} = \vec{0} \\ Qq^{(0,k-1)} &= q^{(0,k)} + \delta^{(0,k)} \quad (7) \\ Qq^{(k-1,j)} &= q^{(k,j)} + \delta^{(k,j)} + \tilde{Q}q^{(k-1,j)} \quad (8) \end{aligned}$$

Where δ contains all the entries $< \varepsilon$, which are rounded after multiplying any q by Q on the left at each iteration. Additionally, $\tilde{Q}$ is the complement of the matrix Q restricted to the sink horizon (i.e. the entries $Q_{ab}\ \forall a, b \in$ the sink-horizon are set to zero, or $Q = Q|_{sink} + \tilde{Q}$). This allows us to rewrite each term as:

$$Q^k\vec{e}_j = q^{(0,k)} + \sum_{n=1}^{k} Q^{k-n}\delta^{(0,n)} \quad (9)$$

$$\begin{aligned} Q^{k+j}\vec{e}_j &= q^{(k,j)} + \sum_{n=1}^{j} Q^{k+j-n}\delta^{(0,n)} \\ &\quad + \sum_{n=1}^{k} Q^{k-n}\delta^{(n,j)} + \sum_{n=1}^{k} Q^{k-n}\tilde{Q}q^{(n-1,j)} \quad (10) \end{aligned}$$

$$\begin{aligned} Q^{k+j+i}\vec{e}_j &= Q^k q^{(i,j)} + \sum_{n=1}^{j} Q^{k+j+i-n}\delta^{(0,n)} \\ &\quad + \sum_{n=1}^{i} Q^{k+i-n}\delta^{(n,j)} + \sum_{n=1}^{i} Q^{k+i-n}\tilde{Q}q^{(n-1,j)} \quad (11) \end{aligned}$$

Where the q terms will become part of the estimate $\vec{u}$, the δs represent the rounded error, and the terms involving $\tilde{Q}$ represent a truncation of paths outside the sink horizon. This equivalence is easily shown by induction via multiplying the kth term on the left by Q and expanding each expression using (7) for (9), and using (8) for (10). The final set of terms (11) is simply the final term in (10) ($k = i$) multiplied by Q^k on the left. The estimate $\vec{u}$ with error err is expressed by collecting like terms (we reindex to combine term families):

$$\begin{aligned} \vec{u} &= q^{(0,0)} + \sum_{k=1}^{j} q^{(0,k)} + \sum_{k=1}^{i} q^{(k,j)} \\ err &= \sum_{m=0}^{\infty}\sum_{n=1}^{j} Q^m\delta^{(0,n)} + \sum_{m=0}^{\infty}\sum_{n=1}^{i} Q^m\delta^{(n,j)} \\ &\quad + \sum_{m=0}^{\infty}\sum_{n=1}^{i} Q^m\tilde{Q}q^{(n-1,j)} + \sum_{n=1}^{\infty} Q^n q^{(i,j)} = \\ &M^{-1}\left(\sum_{n=1}^{j}\delta^{(0,n)} + \sum_{n=1}^{i}\delta^{(n,j)} + \tilde{Q}\sum_{n=1}^{i} q^{(n-1,j)} + Qq^{(i,j)}\right) \end{aligned}$$

Separating indices and using any sub-multiplicative (induced) norm, we can bound the magnitude of the error:

$$\begin{aligned} \|err\| &\leq \|M^{-1}\| \cdot (\|\vec{\mathbf{w}}\| + \|\vec{\mathbf{x}}\| + \|\vec{\mathbf{y}}\| + \|\vec{\mathbf{z}}\|) \\ \|\vec{\mathbf{w}}\| &\leq \sum_{n=1}^{j}\|\delta^{(0,n)}\|, \quad \|\vec{\mathbf{x}}\| \leq \sum_{n=1}^{i}\|\delta_{|}^{(n,j)}\| \\ \|\vec{\mathbf{y}}\| &\leq \|\tilde{Q}\|\sum_{n=1}^{i}\|q^{(n-1,j)}\|,\ \|\vec{\mathbf{z}}\| \leq \|Q\| \cdot \|q^{(i,j)}\| \end{aligned}$$

Where $\|M^{-1}\|$ is the norm of the inverse of M, equal to the condition number κ divided by $\|M\|$. The rounded δ terms are defined to only have entries smaller than ε. We use Thm. 1 to compute p and the size of each horizon according to their respective εs ($\varepsilon_{source} = \varepsilon^q$, $\varepsilon_{sink} = \varepsilon^r$, where $q, r \geq 0, q + r = 1$). The source-horizon has at most p^q non-zero entries and the sink has at most p^r. As each member of the horizon connects to at most s nodes, and the rounded terms are external to the horizon, $\vec{\mathbf{w}}$ and $\vec{\mathbf{x}}$ are bounded by:

$$\|\vec{\mathbf{w}}\|_1 \leq jp^q s\varepsilon, \quad \|\vec{\mathbf{x}}\|_1 \leq ip^r s\varepsilon$$

For $\vec{\mathbf{y}}$, note that $\tilde{Q}$ contains a strict subset of the entries of Q. As $\rho(Q) \leq \rho = \|Q\|_1 < 1$ (to ensure convergence), the entries of Q in each column must have magnitudes summing to no more than $\rho < 1$. This fact is also true for any sub-matrix of Q; in particular, $\|\tilde{Q}\|_1 \leq \rho$. Additionally, by the definition of the transition from loop 2 to loop 3, $\|q^{(n,j)}\| \leq \epsilon^q\ \forall n \geq 1$. Similarly, $\|q^{(i,j)}\| \leq \varepsilon$. This provides a final bound:

$$\|\vec{\mathbf{y}}\|_1 \leq i\rho\varepsilon^q,\ \|\vec{\mathbf{z}}\|_1 \leq \rho\varepsilon$$

Combining these gives the general bound. Note that i and j represent the lengths of paths used in loops 2 and 3 of the algorithm, so we can bound them using l in (5) ($j = ql, i = rl$). If we also assume that $q = r = 0.5$, the error bound simplifies to Thm. 2. ■

As we see, the accuracy of the algorithm depends on the conditioning of M (equivalently, when any eigenvalue of Q is close to one). Also note that this is a bound on the error of the entire column $\|\vec{u} - M_{\cdot j}^{-1}\|_1$, and not only on the entry $|\vec{u}_i - M_{ij}^{-1}|$, so it is conservative for $s > 1$.