A New Method for Reconstructing Network Topology from Flux Measurements

Aravind Rajeswaran  Shankar Narasimhan*

Systems & Control Group, Department of Chemical Engineering,
Indian Institute of Technology Madras, Chennai - 600036, India

June 2, 2015

Abstract

We solve the problem of reconstructing network topology from steady state flux measurements. Specifically, given a data matrix $X$ where each column contains measurements corresponding to a distinct steady state configuration, we wish to estimate a model $A$ which captures the approximate linear relationships between the different variables comprising $X$ (i.e. of the form $AX \approx 0$) such that $A$ is full rank (highest possible) and consistent with a network incidence structure. Hence the network topology can be obtained from this model. The problem is solved through a sequence of steps including estimating approximate linear relationships between variables using Principal Component Analysis (PCA), obtaining fundamental cut-sets from the model estimated by PCA, and graph realization from f-cut-sets (or equivalently f-circuits). Each step and the overall process is polynomial time. The method is illustrated using an example of water network reconstruction from steady-state flow rate measurements. We also establish hard limits on what can be, and cannot be done with steady-state data.

Key words: Network reconstruction, Low rank approximation, PCA, Estimation, Graph realization

1 Introduction

Model identification from data is ubiquitous and of interest to many different scientific disciplines. Broadly, the problem is to develop a model which effectively predicts and explains the variability in the given data set. Various techniques based on regression analysis and machine learning have been developed to improve predictive capabilities, and to handle large data sets efficiently. However, a lot of work has not been directed towards identification of structured models, outside the realm of sparsity. In particular, techniques for incorporating partial structural knowledge of the model at the model building stage is an open problem. In this paper, we propose a technique to reconstruct networks from steady state flux data in polynomial time.

*Corresponding author, Email: naras@iitm.ac.in
In many engineering applications, models have particular structures which provide insights about the nature of process under question. These structures are largely dictated by the physics of the process. Examples include conservation laws like mass or energy balance, which take an even simpler structure for network models. For example, the mass balance for nodes in a network can be written as:

$$\dot{m}_i = \sum_{k \in \delta^-(i)} f_{ki} - \sum_{k \in \delta^+(i)} f_{ik}$$  \hspace{1cm} (1)

where the $\dot{m}_i$ term describes the accumulation (of say mass) in node $i$, and the $f_{ij}$ terms describe the fluxes or flow rates transferred between different nodes. The fluxes in general can be considered as some functions of the nodal values. For example, in case of reaction networks, fluxes may be related to the concentrations through mass action kinetics. Hence, the different states (nodes, and fluxes) are constrained to evolve in accordance with equation (1).

Given the network topology (i.e. connectivity), it is possible to write down balance equations for every node in terms of the edge fluxes. An interesting question is the inverse problem: given a large data set containing only these edge fluxes under steady state conditions, is it possible to reconstruct the network topology? It is to be noted here that the only information available are the edge flux values - we do not know the connectivity (whether two given edges are incident on the same node for example). For steady state networks, the only useful measurements are edge fluxes since nodal values shed no light on network structure since we are not tracking the temporal evolution.

Identifying network topology can provide many insights about the system. For example, degree distribution reveals the presence of any central or “important” hubs in the network; clustering coefficient can provide insight about redundancy and hence robustness of systems; and betweenness centrality can provide an insight about bottlenecks and flow traffic in the network, thereby giving us a good understanding of which variables are difficult to control. Further, reconstructing networks from flux data have wide range of applications like inferring connectivity from meter and flow measurements in distribution networks like water or power \[1\]; to improving constrain-based modeling approaches for metabolic networks. \[2\].

Previous work on network reconstruction have typically used time-series data \[3\]. Relevant examples include identifying plant topology or connectivity from time series measurements \[4\], reconstruction of chemical reaction networks from data \[5\] etc. These exercises involve fitting an appropriate time series model (like Vector Auto Regressive models), and inferring connectivity from the coefficients of fitted models. However, obtaining the same connectivity information from steady state data is a difficult problem. In part, this is due to a rotation ambiguity present in estimating models from steady-state data, unlike the case of time-series data. In this paper, we propose a method which works around this rotation ambiguity thereby enabling us to reconstruct topology of the network. We also answer closely related questions like: (i) under what conditions is a network realizable; (ii) how many distinct steady state measurements are required to do so; (iii) is the realized network unique and whether there are any fundamental limitations when using steady-state data.

More concretely, the problem we wish to solve is the following: Given only data matrix $X$ where each column corresponds to a different instance or steady state and each row is an edge flux, find a model $A$ such that,

$$AX \approx 0$$  \hspace{1cm} (2)

and such that the model, $A$, is full rank (highest possible) and consistent with a network structure. In other
words, just by looking at the estimated model $\mathbf{A}$, we should be able to draw the network structure that connects the different variables.

The organization of the paper is as follows: we will first provide a brief overview of the proposed method; following which we review basics of algebraic graph theory and Principal Component Analysis (PCA). Next, we will elucidate fundamental limitations posed by techniques such as PCA, which employ an adaptive choice of basis sets to represent the model. Following this, we shall propose a technique to transform the model basis to one that is suitable for dealing with networks. Finally, we will describe a method to recover the network topology from this representation. We will also establish that uniqueness of realized network cannot be guaranteed, and this is a fundamental limitation of steady-state data.

2 Overview of Method

The proposed method for reconstructing network topology involves multiple steps. First, we obtain an approximate linear model or constraint matrix using Principal Component Analysis (PCA). This model however will not correspond to a network structure directly, since the set of equations will be rotated (or transformed) so that they are orthogonal to each other. The following steps will transform these set of approximate linear relationships to a different but equivalent set of linear relationships that are consistent with network models, thereby enabling us to obtain the network topology just by looking at these transformed set of equations. We first obtain the fundamental cut-sets of the network from the PCA estimate. This transformation restores network identifiability. Next, we obtain the underlying undirected graph structure given a spanning tree (minimal) and associated cut-sets, which are obtained in the previous step. This is possible within 2-isomorphism, and the problem is generally called graph realization from cut-sets. Finally,

![Figure 1: Sequence of steps in the proposed method.](image-url)
the direction of each edge is obtained by performing linear regression on the structured model obtained using the structure of underlying undirected graph. The process is summarized in Fig. 1. The overall method is polynomial time, since each step of the method is polynomial in number of arithmetic operations.

3 Review of algebraic graph theory

A graph \( G = (N, E) \) contains a set of nodes \( (N) \) and associated edges \( (E) \). The number of nodes will be denoted by \( n \) and the number of edges by \( m \) throughout this paper. Unless explicitly stated, we are working with simple connected graphs. That is, self loops and multiple edges between same nodes are not allowed.

The incidence matrix \( (A) \) describes how the edges are incident on nodes. The dimensions of \( A \) are \( n \times m \) where the rows denote nodes, and columns, edges.

\[
A_{ik} = \begin{cases} 
+1 & \text{if edge } k \text{ enters node } i \\
-1 & \text{if edge } k \text{ leaves node } i \\
0 & \text{if edge } k \text{ is not incident on node } i 
\end{cases}
\]

Due to its particular structure, this matrix is most useful for network reconstruction. It is trivial to verify that under steady state,

\[
\sum_k A_{ik}x_k = \dot{m}_i = 0 \quad \forall i \in N
\]

In other words, the fluxes are constrained to lie in the nullspace of \( A \). Since \( A \) describes the network completely, the task is essentially to recover \( A \) given many instances of \( x \), concatenated to form \( X \). What is to be noted is the specific structure of \( A \). Each column of \( A \) will have exactly two non-zero entries (one +1 and one −1) since an edge can leave only one node and enter another. Hence the matrix is sparse and given this information, it is also easy to see that the rank of \( A \) is \( n - 1 \) when working with closed and connected networks, i.e. each edge must originate at a node and terminate at a node.

A closely related form of this matrix is the reduced incidence matrix. The idea here is that since \( A \) is one rank deficient, the entire information contained in \( A \), can be represented in just \( n - 1 \) rows. This is particularly useful when dealing with flow networks which may contain dangling edges. Other examples of the same include external flux edges in metabolic networks. In such cases these dangling edges are all assumed to be connected to one “environment node”. In the reduced representation, dangling edges are allowed, so that all the edges corresponding to external fluxes will be incident on only one node. Hence, the corresponding columns in \( A \) will have only one non-zero entry (either +1 or −1). This idea is illustrated in Fig. 2. The dimensions of this matrix are \( n \times m \) where \( n \) is the number of nodes in the system we are considering (excluding environment node). Hence it is still one rank lesser than the “true” graph (which also contains the environment, hence summing to \( n + 1 \)).

Other useful matrices for this paper are the fundamental circuit \( (B_f) \) and fundamental cut-set \( (C_f) \) matrices. If we consider any spanning tree, with respect to this tree, it is possible to construct a set of circuits and cut-sets which are said to be fundamental, since all other circuits and cut-sets can be represented as ring sums of these fundamental ones. The definitions of these matrices again are analogous to the incidence...
matrix. These matrices set up the relationships between the edges and fundamental circuits or cut-sets.

\[ B_{ij} = \begin{cases} 
+1 & \text{edge j is in circuit i, same direction} \\
-1 & \text{edge j is in circuit i, opposite direction} \\
0 & \text{otherwise} 
\end{cases} \]

and similarly for the cut-set matrix,

\[ C_{ij} = \begin{cases} 
+1 & \text{edge j is in cut-set i, same direction} \\
-1 & \text{edge j is in cut-set i, opposite direction} \\
0 & \text{otherwise} 
\end{cases} \]

Further, since these fundamental matrices are written with respect to a spanning tree in mind, a graph can have multiple fundamental circuit or cut-set matrices. It is also true that multiple graphs can have the same \( B_f \) and \( C_f \) matrices even when they are not isomorphic. Such examples are called 2-isomorphic graphs [6], and characterize the ambiguity (in fact, exactly) in network reconstruction. This will be discussed in detail in later sections. These fundamental matrices can be further decomposed as follows:

\[ B_f = [B_t | I_\mu] \]

\[ C_f = [I_{n-1} | C_c] \]

where \( \mu = m - (n - 1) \). The dimensions of \( B_f \) are \((\mu \times m)\) and that of \( C_f \) are \((n - 1 \times m)\). A standard result is that \( B_t = -C_c^T \) and that \( C_fB_t^T = 0 \) (mod 2). Hence, given one the other is trivial to find, and both the cut-set and circuit matrices describe a graph within 2-isomorphism. We refer the interested readers to Chapter 7 of Narsingh Deo’s book [6] for a detailed discussion.

4 Model Identification using PCA or SVD

PCA and Singular Value Decomposition (SVD) are among the most widely used multivariate statistical techniques. PCA or SVD provides the best low-rank approximation to a given matrix under Gaussian white noise assumption. Equivalently PCA provides the best approximate linear model between a set of variables assuming each is corrupted with Gaussian white noise [7]. However, the obtained model suffers from a rotation ambiguity.

Let \( X \) be the data matrix \((m \times d)\) with edge fluxes stacked along rows. Hence, each column of the data matrix describes a new steady state scenario. The measurements are of course corrupted with noise (in general, with unknown error co-variance). Consider the scaled co-variance matrix

\[ S_x = \frac{1}{d}XX^T \]

The SVD of the scaled data matrix can be written as

\[ \text{svd} \left( \frac{1}{\sqrt{d}}X \right) = U_1S_1V_1^T + U_2S_2V_2^T \]
where $\mathbf{U}_1$ are the set of orthonormal eigenvectors corresponding to the $p$ largest singular values of $\mathbf{S}_x$ while $\mathbf{U}_2$ are the set of orthonormal eigenvectors corresponding to the remaining $(m - p)$ smallest singular values of $\mathbf{S}_x$. The matrices $\mathbf{S}_1$ and $\mathbf{S}_2$ are diagonal, whose entries are the square root of the eigenvalues of $\mathbf{S}_x$. It is assumed that the singular values and corresponding principal components are sorted in decreasing order.

To compute the SVD of data matrix, we require $O(k_1m^2d + k_2d^3)$ operations [8]. Hence, computation of SVD is possible in polynomial time.

If the objective is to reduce dimensionality (or obtain a low rank approximation of $\mathbf{X}$), then it suffices to store only the hyperplane on which most of the data resides ($\mathbf{U}_1$) and the projection of data onto this hyperplane. Since the equation of the hyperplane and coordinates of the data points on this hyperplane (projections) are known, the original data can be recovered up to the level of hyperplane. It is assumed that the variations which deviated the data from this hyperplane are due to noise, and hence dimensionality reduction and filtering have been performed simultaneously. The only task that is left is to determine the number of PCs to be retained ($p$). In absence of any additional information other than data, $p$ is often heuristically chosen based on criteria like percentage variance explained or scree plot [7].

The same idea can be extended for obtaining an approximate linear model as well. In model identification, we are interested in finding the sub-space on which the data is constrained to lie, and this sub-space is the set of feasible or admissible region allowed by the model. To characterize this sub-space, we need to find a set of basis vectors orthogonal to this sub-space. This is readily obtained from the $(m - p)$ non-retained PCs. Hence, the model obtained is given by:

$$\mathbf{U}_2^T \mathbf{x} = 0$$  \hspace{1cm} (4)

It is worth noting the rotation ambiguity, that if we pre-multiply $\mathbf{U}_2$ by some invertible matrix $\mathbf{M}$ we have:

$$\mathbf{M}\mathbf{U}_2^T \mathbf{x} = \mathbf{M}0 = 0$$

Hence, the best that can be obtained from data is the row-space for the true constraint matrix, which in this case is the directed version of incidence matrix, which is what we wish to recover. If we do not impose any structural restrictions, the best that we can get is a basis, which in case of PCA is an orthogonal basis set chosen adaptively based on data input. Indeed, historically PCA was developed as a solution to the Total Least Squares problem. In this paper we shall discuss a method to recover the true constraint matrix (by exploiting its known structural properties) from the PCA estimate.

Let us denote the true constraint matrix, obeying the structural requirements, by $\mathbf{A}$. The objective is to recover this from a rotated form, which we denote as $\hat{\mathbf{A}}$. The first step is to get an estimate of $\hat{\mathbf{A}}$ denoted by $\hat{\mathbf{A}}_r$ using PCA. In order to do so, we have different options depending upon the available information. If the error-covariance matrix is known to be homoscedastic, i.e. of the form $\Sigma_e = \sigma^2_e \mathbf{I}$, an approximate linear model can be identified by performing SVD of the data matrix like in equation (3) and obtain the constraint matrix estimate as $\hat{\mathbf{A}}_r = \mathbf{U}_2$, which is also the BLU estimate. In case of heteroscedastic error covariance, if $\Sigma_e$ is known, the Maximum Likelihood estimate of the linear model can be obtained using MLPCA [9, 10]. If $\Sigma_e$ is not known, it was shown by Narasimhan and Shah [10, 11] that it is possible to recover both the model as well as $\Sigma_e$ by using a technique called Iterative PCA or IPCA.

At this stage, we introduce an example, which will serve to illustrate each step of the network reconstruction procedure.
**Example:** Consider the flow network shown in Fig. 2. We obtain measurements of all the flows labeled 1 through to 6, where each flow measurement is corrupted with some noise. We wish to recover the true connectivity from this flux data alone, without knowledge of the number of nodes in the network. The true constraint relationships are given by:

\[
A \mathbf{x} = 0
\]

\[
A = \begin{bmatrix}
1 & 1 & -1 & 0 & 0 & 0 \\
0 & 0 & 1 & -1 & 0 & 0 \\
0 & 0 & 0 & 1 & -1 & -1 \\
0 & -1 & 0 & 0 & 0 & 1 \\
\end{bmatrix}
\]

\[
\mathbf{x} = \begin{bmatrix}
x_1 \\
x_2 \\
x_3 \\
x_4 \\
x_5 \\
x_6 \\
\end{bmatrix}^T
\]

Data is generated using the following process:

1. \(x_1\) and \(x_2\) are chosen as independent variables, so that the rest of the flows can be obtained by solving a set of linear equations. The independent variables are perturbed around a base value (see Table 1).

2. True values of \(x_3\), \(x_4\), \(x_5\) and \(x_6\) are computed exactly, and a Gaussian noise is added to each of the six flows, after computing true values for all flows.

The base values for simulation and noise are given in Table 1. In this example, a diagonal co-variance matrix is used, hence each sensor error is independent of the other.

![Flow network example](image)

Figure 2: Flow network example. Both the reduced graph (a) and closed graph (b) are depicted.
Table 1: Base values, standard deviation of fluctuation (SDF), and standard deviations of error (SDE)

| Variable | Base Value | SDF | SDE  |
|----------|------------|-----|------|
| $x_1$    | 10         | 1   | 0.1  |
| $x_2$    | 10         | 2   | 0.08 |
| $x_3$    | solved     | computed | 0.15 |
| $x_4$    | solved     | computed | 0.2  |
| $x_5$    | solved     | computed | 0.18 |
| $x_6$    | solved     | computed | 0.1  |

When SVD is applied to the data matrix \( \left( \frac{1}{\sqrt{d}} X \right) \) from the example, we obtain the following singular values:

\[
\begin{bmatrix}
34.96 & 2.25 & 0.43 & 0.40 & 0.34 & 0.3
\end{bmatrix}
\]

This clearly indicates that there are 4 singular values close to 0, thereby the data is in fact present in a two dimensional subspace of \( \mathbb{R}^6 \). In other words the data is constrained to lie in the null space of a \( 4 \times 6 \) matrix. We get the constraint relationships from:

\[
\hat{A}_r x = U^T_2 x \approx 0
\]

We obtain \( \hat{A}_r = \)

\[
\begin{bmatrix}
0.099 & 0.107 & 0.612 & -0.773 & 0.068 & 0.047 \\
-0.318 & 0.213 & -0.315 & -0.173 & 0.806 & 0.275 \\
-0.748 & -0.253 & 0.432 & 0.199 & 0.110 & -0.372 \\
-0.045 & -0.737 & 0.077 & -0.009 & -0.023 & 0.669
\end{bmatrix}
\]

This estimate has been arrived at assuming no knowledge of the true error co-variance matrix, which is representative of the worst-case scenario. Also, we have used SVD instead of the more specialized techniques for heteroscedastic error co-variance to illustrate that the method works remarkably well even when only an approximate model is obtained - either from data alone or partially based on data, and partially based on process knowledge.

Clearly, the estimate is different from the true constraint matrix, and an element by element comparison is not possible. This is because the basis for the row-space (the rows) have been chosen adaptively to be orthonormal to each other. However, to check if the two constraint matrices do indeed share the same row-space, Narasimhan and Shah [10] have proposed two criteria: (i) the subspace angle between the row subspace of the estimated and true constraint matrices, and (ii) the sum of orthogonal distances of the row vectors of the estimated constraint matrix from the subspace defined by the rows of the true constraint matrix.

(i) The subspace angle measures the maximum possible angle between the row sub-spaces spanned by the two matrices. For this example, we have a subspace angle of 0.0091 degrees.

(ii) Here we project \( \hat{A}_r \) on to the row space of \( A \), and use Frobenius norm (sum-square of element wise difference) between the projection and \( \hat{A}_r \) for comparison. For the example, the Frobenius norm was calculated to be \( 8.21 \times 10^{-5} \).
The above values suggest that a good estimate of the constraint matrix was obtained by using PCA. Our experience with larger examples indicates that subspace angle is not a good criteria for comparison, because it does not clearly indicate the quality of the estimated constraint matrix, especially as the number of constraints increases [11]. The second criteria performs satisfactorily, but we propose another comparison method here, which in our experience performs very well, and is also the motivation for the network reconstruction procedure. For this purpose, the flow variables (x) are partitioned into a set of dependent (xD) and independent variables (xI). This partitioning could be on the basis of process knowledge, or it could be achieved in a completely data driven manner (this will be discussed shortly). Once this partitioning has been done, we have:

\[ Ax = A_D x_D + A_I x_I = 0 \]  

A successful partitioning into dependent and independent sets ensures that \( A_D \) is invertible, thus we have:

\[ x_D = -A_D^{-1} A_I x_I = R x_I \]  

It is easy to see that for any admissible partition, this matrix \( R \) is unique to that partition, since this operation de-convolutes any rotation by an invertible matrix. That is, if \( A_r = M A \) where \( M \) is invertible, then

\[ x_D = -A_r^{-1} A_r x_I = -(MA_D)^{-1} (MA_I) x_I \]

\[ = A_D^{-1} M^{-1} M A_I x_I = R x_I \]  

Hence, an element by element comparison between \( \hat{R} \) (obtained from PCA estimate) and \( R \) (obtained from true constraint) matrix is possible. Consider the example with \( x_1 \) and \( x_2 \) chosen to be the independent variables.

\[
R = \begin{bmatrix}
1 & 1 \\
1 & 1 \\
1 & 0 \\
0 & 1 \\
\end{bmatrix}
\]  

\[
\hat{R} = \begin{bmatrix}
1.007 & 0.995 \\
1.015 & 0.987 \\
1.005 & -0.005 \\
-0.001 & 1.001 \\
\end{bmatrix}
\]

Element wise comparison clearly shows us that PCA has recovered the correct row space. Small deviations from the true values are due to noise which we added.

4.1 Graph Theoretic Interpretation

A more useful and powerful interpretation of the previous set of operations comes from graph theory, especially when dealing with true constraint matrices that correspond to a network. The act of partitioning the set of process variables into dependent and independent variables, is exactly equivalent to determining a minimum-spanning tree of the graph, and the matrix \( R \) in fact represents the chords associated with the chosen spanning tree (xD). Notice that the row space of the incidence matrix or the constraint matrix is same as the row space of the f-cut-set matrix \( C_r \). The procedure to construct \( C_r \) from \( A \) will be discussed shortly, and is fairly straightforward. The reverse process however, is not trivial. Consider the partitioning of \( C_r \) as:

\[ C_r = [I_{n-1}] C_c \]  

The matrix $\mathbf{C}_c$ in fact exactly corresponds to $\mathbf{R}$ and the set $\mathbf{x}_D$ makes up the spanning tree. Thus we have reduced the problem to reconstructing the network, given a spanning tree and associated chords (or equivalently the f-cut-set matrix). This is an old problem in graph theory called graph realization from circuit or cut-set matrices, and is possible within 2-isomorphism.

5 Obtaining cut-sets from PCA estimate

At this stage, we are left with only two steps: a method to automatically generate $\mathbf{R}$ or $\mathbf{C}_f$ from data; and a method to reconstruct $\mathbf{A}$ from $\mathbf{C}_f$. Here we discuss the former step.

Reduced Row Ehelon Form (RREF): A matrix is in RREF if it satisfies the following conditions:

- It is in row echelon form.
- Every leading coefficient is 1 and is the only nonzero entry in its column.

The RREF of a fat full rank matrix (like the one obtained from PCA) can be computed using an algorithm like Gauss-Jordan method in $O(m^2d)$ arithmetic operations. Unlike the row echelon form, the reduced row echelon form of a matrix is unique and does not depend on the algorithm used to compute it. Further, RREF is a unique property of the row space, and hence is constant even if the matrix is pre-multiplied by an invertible matrix. Hence it is straightforward to compute $\mathbf{C}_f$ given $\mathbf{A}$ as:

$$\mathbf{C}_f = \text{rref}(\mathbf{A})$$  \hspace{1cm} (9)

It must be noted however that $\mathbf{A}$ and $\mathbf{C}_f$ might require column permutation for the above equation to hold. But once the RREF is computed, the columns can always be permuted such that the first block of the matrix is Identity. Different permutations correspond to different ways of ordering the edges or equivalently 1-isomorphism.

The same idea can be used for the estimated matrices as well. By doing so, we get

$$\hat{\mathbf{C}}_f = \text{rref}(\hat{\mathbf{A}}_r)$$  \hspace{1cm} (10)

In practice, we observed that the above computation suffers from numerical instabilities. This can however be mitigated by using both row and column pivoting. Knowing that element by element comparison between $\hat{\mathbf{C}}_f$ and $\mathbf{C}_f$ is possible, and that all entries of $\mathbf{C}_f$ are 0, 1, or $-1$; it is possible to get an exact recovery of $\mathbf{C}_f$ from $\hat{\mathbf{C}}_f$ by employing a simple rounding off procedure. As an example, we again look at the same flow process. From the estimated constraint matrix, we calculate (choosing $x_1$ to $x_4$ to be the spanning tree)

$$\hat{\mathbf{C}}_f = \begin{bmatrix}
1 & 0 & 0 & 0 & -0.995 & -0.005 \\
0 & 1 & 0 & 0 & -0.001 & -0.999 \\
0 & 0 & 1 & 0 & -1.003 & -0.999 \\
0 & 0 & 0 & 1 & -1.011 & -0.992 \\
\end{bmatrix}$$
The true cut-set matrix (i.e. RREF of $A$) is:

$$C_f = \begin{bmatrix}
1 & 0 & 0 & -1 & 0 \\
0 & 1 & 0 & 0 & -1 \\
0 & 0 & 1 & -1 & -1 \\
0 & 0 & 0 & 1 & -1 \\
\end{bmatrix}$$

There is a clear element-wise correspondence between the two, and hence exact recovery of $C_f$ from data is possible. Note that if the entries in $\hat{C}_f$ have large deviations from acceptable values of 0, 1, or $-1$, this is strong evidence that the original data generating process was not a network.

6 Graph Realization Problem

The only remaining step is to reconstruct $A$ from $C_f$.

**Definition:** Two graphs $G_1$ and $G_2$ are said to be 2-isomorphic if they share circuit (or equivalently cut-set) correspondence. This means that there is a one to one correspondence between edges such that, whenever a set of edges in $G_1$ forms a circuit (or cut-set), the corresponding edges in $G_2$ also form a circuit (or cut-set).

**Theorem:** Two graphs are 2-isomorphic if and only if the row space of their incidence matrices are identical, after a possible column permutation (1-isomorphism).

**Proof:** The ‘if’ part of the proof is straightforward. If they share the same row-space, then subject to column permutation (one to one correspondence between edges), they share the same $C_f$ matrix, since it is a property of the row-space. This means that there is a circuit correspondence, and hence by definition 2-isomorphic. The converse argument is also true. For two graphs to be 2-isomorphic, they should produce the same $C_f$ matrix subject to column permutation. If this is the case, then their incidence matrices have the same row-space (after the necessary permutation) due to the property of RREF.

The fundamental result of the above arguments is that uniqueness of reconstruction cannot be guaranteed by any procedure given only steady state data. This is straightforward to understand, since if we have a constraint matrix $A$ such that $Ax = 0$, then for any invertible matrix $M$, the following also holds true: $M(Ax) = 0$. If there exists another network whose incidence matrix $A'$ can be written as $A' = MA$ for some $M$, then it is not possible to distinguish between the two networks based on data alone. However, it is possible to recover at-least one network (if it actually exists). This problem is called the graph realization problem from fundamental cut-sets or circuits, and the realization is possible in almost linear time (in number of edges).

We propose to perform the realization procedure in two steps. In the first step, we realize the structure of the underlying undirected graph. Once this structure is determined, the direction of edges are ascertained by performing linear regression using the structured model. With reference to the flow process example, we first take the sparsity pattern of $C_f$ (i.e. we are ignoring directionality of edge at this stage). After obtaining
$\hat{C}_f$ from data, rounding off, and ignoring signs, we obtain $C_u$.

$$C_u = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

To reconstruct the incidence matrix, we first make use of the structural property of the same: Each column of $A_u$ has utmost two non-zero entries, which are equal to 1. We observe that the above form has more than two non-zero entries in columns 4 and 5. We hence perform the matrix operation: $R_4 \leftarrow R_3 + R_4$ and perform mod 2 arithmetic (i.e. set all elements with value 2 equal to 0) we get:

$$A^{(1)}_u = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

This is in an acceptable network form, and the underlying undirected structure has been recovered. Since the structure has now been obtained, it is trivial to perform linear regression with this particular structure to obtain the direction of edges. This can be done so by estimating a single equation corresponding to only the variables present in each row. The signs of elements in the first row are chosen arbitrarily, and this fixes the signs of all the rows since each column has exactly one positive and one negative entry (including the environment node). If the true data generating process was a network, we expect to get a consistent set of entries without any clashes. The reconstructed network is shown in Fig. 3 and the corresponding reduced incidence matrix is:

$$A_{\text{red}} = \begin{bmatrix} 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 & 0 & 0 \end{bmatrix}$$
We can see that this graph is different from the original one we started out with. However, the two are completely indistinguishable using flow conservation equations, and both could have generated the data in an equally plausible manner. It is also trivial to verify that the two networks are indeed 2-isomorphic. This can be done by obtaining the RREF of both $A$ and $A_{est}$, and they would be identical. On the other hand, if we had stacked the variables in an alternate manner (this corresponds to reordering or renaming the edges, i.e. 1-isomorphism) as

$$ x = \begin{bmatrix} x_1 & x_3 & x_5 & x_2 & x_4 & x_6 \end{bmatrix} $$

The corresponding undirected cut-set matrix we would obtain after going through the PCA and RREF sequence would be:

$$ C^{(2)}_u = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} $$

Performing $R_1 \leftarrow R_1 + R_2$ followed by $R_1 \leftarrow R_1 + R_4$, we have the following incidence matrix.

$$ A^{(2)}_u = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} $$

This corresponds to exactly the network we originally used, subject to the same column permutations (i.e. the two graphs are 1-isomorphic). This can be verifying by permuting the columns of the original incidence matrix accordingly and comparing element wise with $A^{(2)}_u$.

In general to automate this process, we need a scalable graph realization algorithm. Some such algorithms include ones developed by Fujishige[12], Bixby & Wagner[13], Parker[14], and Jianping[15]. These have been developed primarily for undirected graphs and use the fundamental circuit matrix as their inputs, whereas we are working with the f-cut-sets. It is however, very easy to obtain one from the other using the following relationships:

$$ B_f = [-C^T | I_{\mu}] $$

We of course use the undirected version as the input, and recover the directions later. This completes the sequence of steps needed to reconstruct the network.

7 Future Work

The problem of graph realization from circuits and cut-sets received much attention during the 1980s, but have remained dormant since then. Further work on this problem, and more intuitive and efficient algorithms for the same, would prove very useful, given the data deluge today. We have shown in this work that graph
realization is an indirect method for estimating network models from data, which is a very important problem today in a number of fields - from economics to biology. We also believe that the proposed method can be extended to a larger set of structured models, in particular the stoichiometry matrix reconstruction. This would provide an alternate method to cross-verify or reconstruct chemical reaction networks using only steady state data.

Also, it may be possible to ascertain the direction of edges after recovering the underlying undirected network topology by just looking at the structure of the signed circuit matrix, and performing a loop traversal. Work on this front could help in reducing one step in the proposed sequence.

8 Conclusions

A polynomial time algorithm for the problem of network reconstruction from steady state flux data was developed. The method involves a sequence of procedures summarized in Fig. 1. The major components are PCA (or SVD) followed by obtaining the cut-set matrix (in reduced row echelon form) and finally graph realization. The edge directions are ascertained using linear regression. This problem finds application in variety of fields like network topology reconstruction in distribution systems like water and power, and with further work in areas like economics and systems biology. The overall procedure is polynomial time, with the most expensive step being the singular valued decomposition of the data matrix, which can be obtained in $O(k_1 m^2 d + k_2 d^3)$ where $m$ is the number of edges (or fluxes) and $d$ is the number of distinct steady state data collected [8]. We revisit the questions raised in Section I and answer them:

(i) Network reconstruction is possible when entries in $\hat{C}_f$ are not far off from acceptable values of 0, 1 or $-1$, and the structure is compatible to a network. This is determined by graph realization algorithms [13].

(ii) We need at least $n$ steady states for reconstruction. Since the value of $n$ may not be known upfront, a safe estimate would be $m$ distinct steady states.

(iii) The realized network is unique only up-to 2-isomorphism. Any further resolvability is not possible from steady state data alone. Causality cannot be determined as expected - reversing the direction of every edge yields an equally plausible network.

Acknowledgments

The authors would like to thank Dr. Nirav Bhatt, IIT Madras, for his constructive criticisms and review which helped in development of the proposed method.
References

[1] V. Arya et al. “Inferring connectivity model from meter measurements in distribution networks,” In Proceedings of the Fourth International Conference on Future Energy Systems, New York, USA, 2013.

[2] S. Wiback, R. Mahadevan, and B. Palsson, “Using metabolic flux data to further constrain the metabolic solution space and predict internal flux patterns: the Escherichia Coli spectrum,” Biotechnology and Bioengineering, 2004.

[3] S. Gigi and A. K. Tangirala, “Quantitative analysis of directional strengths in jointly stationary linear multivariate processes,” Biological Cybernetics, 2010.

[4] S. Gigi and A. K. Tangirala, “Reconstructing Plant Connectivity using Directed Spectral Decomposition,” Proc. 8th IFAC Symposium on Advanced Control of Chemical Processes, 2012.

[5] Y. Yuan et al. “Robust dynamical network structure reconstruction,” Automatica, 2011.

[6] N. Deo, “Graph Theory with Applications to Engineering and Computer Science,” Prentice-Hall, Inc., 1974.

[7] I. Jolliffe, “Principal component analysis,” John Wiley & Sons, 2002.

[8] G. H. Golub and C. F. Van Loan. “Matrix computations,” Vol. 3, JHU Press, 2012.

[9] P. D. Wentzell et al. “Maximum likelihood principal component analysis,” Journal of Chemometrics, 1997.

[10] S. Narasimhan and S. L. Shah, “Model identification and error covariance matrix estimation from noisy data using PCA,” Control Engineering Practice, 2008.

[11] S. Narasimhan and N. Bhatt, “Deconstructing principal component analysis using a data reconciliation perspective,” Computers & Chemical Engineering, 2015.

[12] S. Fujishige, “An efficient PQ-graph algorithm for solving the graph-realization problem,” Journal of Computer and System Sciences, 1980.

[13] R. E. Bixby and D. K. Wagner, “An almost linear-time algorithm for graph realization,” Mathematics of Operations Research, 1988.

[14] S. Parker and H. Lohse, “A direct procedure for the synthesis of network graphs from a given fundamental loop or cutset matrix,” IEEE Transactions on Circuit Theory, 1969.

[15] J. QIAN and P. WOO, “Realization of the Linear Tree that Corresponds to a Fundamental Loop Matrix,” Wireless Sensor Network, 2010.