Addendum

Subashish Datta

This document is addendum to the following article:

1) S. Datta and H. Mathur, "Feedback Control for Structured Descriptor Systems with Minimum Free-entry Pattern Gain Vectors", *Automatica*, Accepted, 2022.

I. Discussion

Due to page limit constraint in the above stated article, we could not provide some of the relevant materials, which we believe, are use helpful for a reader. These materials are provided here. The algorithms are also explained elaborately.

II. Digraph Representation of a Structured Matrix and Related Definitions

For a given structured system $\overline{E}\dot{x} = \overline{A}x + \overline{b}u$ with feedback control $u = \overline{f}x$, following square structured matrices of size $(n+1)\times(n+1)$ are defined:

$$\overline{H}_c := \begin{bmatrix} s\overline{E}-\overline{A} & \overline{b} \\ \overline{f}^T & 1 \end{bmatrix}, \quad \overline{H}_o := \begin{bmatrix} s\overline{E}-\overline{A} & \overline{b} \\ \overline{f}^T & 0 \end{bmatrix}. \tag{1}$$

Considering the system matrices as follows:

$$\overline{E} = \begin{bmatrix} \star & 0 & \star \\ 0 & 0 & \star \\ 0 & 0 & \star \end{bmatrix}, \ \overline{A} = \begin{bmatrix} 0 & \star & 0 \\ \star & 0 & 0 \\ 0 & 0 & \star \end{bmatrix}, \ \overline{b} = \begin{bmatrix} 0 \\ \star \\ 0 \end{bmatrix}, \ f = \begin{bmatrix} \star \\ \star \\ \star \end{bmatrix}, \tag{2a}$$

the digraphs $\mathcal{G}(\overline{H}_c)$ and $\mathcal{G}(s\overline{E}-\overline{A})$ are depicted in Figure 1-A and Figure 1-B, respectively.

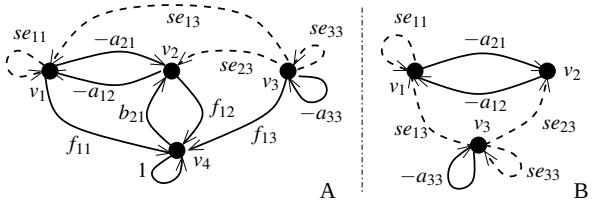

Fig. 1. Digraphs $\mathcal{G}(\overline{H}_c)$ and $\mathcal{G}(s\overline{E}-\overline{A})$ are in A and B, respectively.

For a given structured matrix $\overline{M}$ of size $n\times m$, we associate a bipartite graph, denoted as $\mathcal{G}_b(\overline{M})$, on the set of $n+m$ vertices where n vertices, denoted as: $r_1, r_2, \cdots, r_n$ correspond to n rows of $\overline{M}$, and m vertices: $c_1, c_2, \cdots, c_m$ correspond to m columns of $\overline{M}$. In $\mathcal{G}_b(\overline{M})$, there is an edge from vertex c_j to vertex r_i with weight m_{ij}, if $m_{ij} \neq 0$. A *matching* in $\mathcal{G}_b(\overline{M})$ refers to a subset of edges of $\mathcal{G}_b(\overline{M})$ such that none of the two edges in the subset have a common vertex. A matching in $\mathcal{G}_b(\overline{M})$ having maximum number of edges is referred to as *maximum matching*. For a given square structured matrix $\overline{M}$, one can obtain digraph $\mathcal{G}(\overline{M})$ from its corresponding bipartite graph $\mathcal{G}_b(\overline{M})$, by combining the vertices c_i and r_i of $\mathcal{G}_b(\overline{M})$ and forming a single new vertex v_i for $\mathcal{G}(\overline{M})$. With this, an edge between the vertices c_i and r_i in $\mathcal{G}_b(\overline{M})$ becomes a self loop in $\mathcal{G}(\overline{M})$ at vertex v_i. Similarly, an edge between the vertices c_i and r_j in $\mathcal{G}_b(\overline{M})$ becomes an edge in $\mathcal{G}(\overline{M})$ between the vertices v_i and v_j. This is illustrated in Figure 2 (A and B).

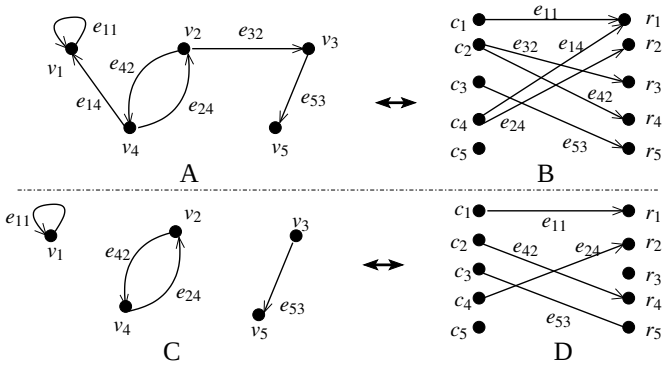

Fig. 2. Directed graph A corresponds to the bipartite graph B. The vertex disjoint self loop, cycle and path in C forms a matching in the corresponding bipartite graph D.

A. Definition for 1 - Connection of a digraph

For a digraph $\mathcal{G}(\overline{M})$, an 1-*connection*, from vertex v_j to vertex v_i, denoted as L_{ij}, is a *spanning subdigraph* of $\mathcal{G}(\overline{M})$ having following properties:

1) *for* $i \neq j$:
   - at vertex v_i, exactly one edge enters and no edge leaves,
   - at vertex v_j, exactly one edge leaves, but no edge enters, and
   - for vertex $v_k \neq v_i, v_j$, exactly one edge enters and exactly one edge leaves,
2) *for* $i = j$:
   - no edges enter or leave at vertex v_i, and
   - for vertex $v_k \neq v_i$, exactly one edge enters and exactly one edge leaves.

Note that there is a close relation between the spanning cycle families of the digraph $\mathcal{G}(\overline{M})$ and its 1-connections. If we include an edge e_{ji} (if exists) in an 1-connection L_{ij} from vertex v_i to vertex v_j, then it will become a spanning cycle family of $\mathcal{G}(\overline{M})$. Hence, if we denote δ as the number of vertex disjoint cycles in a spanning cycle family of $\mathcal{G}(\overline{M})$, and $\bar{\delta}$ as the number of vertex disjoint cycles in a 1-connection of $\mathcal{G}(\overline{M})$, then they satisfy the following relation: $\delta = \bar{\delta} + 1$. For

illustration, the 1-connections of $\mathscr{G}(s\overline{E}-\overline{A})$, corresponding to the system matrices $\overline{E}$ and $\overline{A}$ as in (2), to compute $d_{ij}(s)$, where $d_{ij}(s)$ as an element of matrix $D(s)=\text{adj}\left(s\overline{E}-\overline{A}\right)$, are depicted in Figure 3.

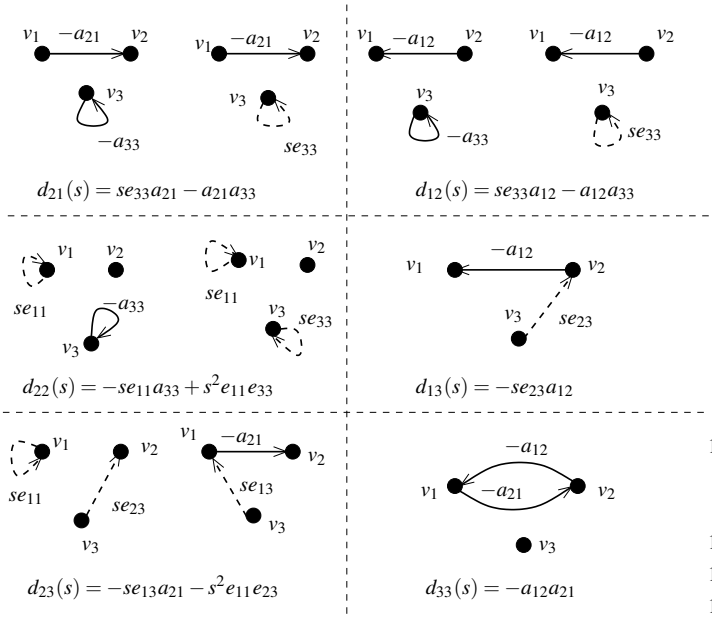

Fig. 3. The 1-connections of $\mathscr{G}(s\overline{E}-\overline{A})$ corresponding to the elements $d_{ij}(s)$, where $\overline{E}$ and $\overline{A}$ are as in (2). The elements: $d_{11}(s)=d_{31}(s)=d_{32}(s)=0$, since the corresponding 1-connections do not exist.

III. Algorithms and Their Explanations

A. Algorithm 1 and its Explanation

An algorithm to compute the SCFs of digraph $\mathscr{G}$, which has n vertices: $\{v_1,v_2,\cdots,v_n\}$ and m edges. Define the following quantities: $s=[\cdots\ i\ \cdots]$, $t=[\cdots\ j\ \cdots]$, $s_l=[\cdots\ k\ \cdots]$, $t_l=[\cdots\ k\ \cdots]$, where i and j are the indices of initial and final vertices of an edge (v_i,v_j), and k is the index of the vertex v_k, where self-loop appears.

Steps 1-4: The indices of starting and ending vertices of the edges in $\mathscr{G}$ are stored in the vectors: $\bar{s}=[s\ \ s_l]$ and $\bar{t}=[t\ \ t_l]$, respectively, in Step-1. Since the objective is to find the SCFs of a digraph $\mathscr{G}$, we assume that there are no isolated vertices in $\mathscr{G}$. Hence, the maximum of all the elements in $\bar{s}$ gives the number of vertices in $\mathscr{G}$ (Step-2), and the number of elements in $\bar{s}$ gives the number of edges (including self-loops) in $\mathscr{G}$ (Step-3). The digraph $\mathscr{G}$ may contain parallel edges. For instance, in the digraph $\mathscr{G}(\overline{H}_o)$, where $\overline{H}_o$ is as in (1), if e_{ij} and a_{ij} are free-entries of $\overline{E}$ and $\overline{A}$, then there would be parallel edges from vertex v_j to v_k, having one with edge weight $-a_{ij}$, and another with edge weight se_{ij}. Hence, to distinguish these two edges, a vector ω_t is defined in Step-4 to assign a UIN to each edge of $\mathscr{G}$. For $i=1,2,\cdots,m$, the i^{th} element of ω_t is defined as: $\omega_t(i)=n+i+\kappa$, where κ is any positive number, which, without loss of generality, is chosen as 2. Then, $\omega_t(i)$ is the UIN for the edge $(v_{\bar{s}(i)},v_{\bar{t}(i)})$.

Steps: 5-6: The incidence matrix I_g of $\mathscr{G}$, without considering the self-loops, is constructed in Step-5. Then, I_{ga} is constructed as in Proposition 12 (of above state article) by including the self-loops. Note that each column of I_{ga} corresponds to an edge of $\mathscr{G}$. Hence, the matrix I_{gw} is defined, by stacking ω_t with I_{ga}, so that each edge of $\mathscr{G}$ will get assigned with its UIN. To compute all the possible SCFs of $\mathscr{G}$, we take all possible combinations of n columns of I_{ga}, and then verify if the conditions of Proposition 11 and Proposition 12 (of above state article) are satisfied. For this, a matrix C_{mb} is constructed in Step-6, whose rows contains all possible combinations of the numbers from 1 to m, taken n at a time.

Steps: 7-12: In these steps, we obtain all possible SCFs of $\mathscr{G}$, which do not have self-loops. In Step-8, at i^{th} iteration,

Algorithm 1 Pseudocode for computing the spanning cycle families of a digraph $\mathscr{G}$.

Input: The vectors: s, t s_l and t_l.
Output: The matrices: Θ_{sw} and Θ_{tw}.

1: Define: $\bar{s}=[s\ \ s_l]$ and $\bar{t}=[t\ \ t_l]$.
2: $n\leftarrow$ the maximum of all the elements appear in $\bar{s}$.
3: $m\leftarrow$ total number of elements in $\bar{s}$.
4: $\omega_t=[\omega_{t_1}\ \omega_{t_2}\ \cdots\ \omega_{t_m}]$, where $\omega_{t_i}=n+i+2$.
5: Construct the incidence matrix I_g of $\mathscr{G}$ by removing its self-loops. Define the matrices: I_{ga} as in Proposition 12 (of above state article) and $I_{gw}:=\begin{bmatrix}I_{ga}\\ \omega_t\end{bmatrix}$.
6: Define $\nu=[1\ 2\ \cdots\ m]$. Construct a matrix C_{mb} of size $(p\times n)$, where $p=\frac{m!}{(m-n)!n!}$, by considering all possible combinations of the elements of ν, taken n at a time.
7: **for** $i=1$ to p **do**
8: $\quad M\leftarrow I_{ga}(:,C_{mb}(i,:))$ and $M_w\leftarrow I_{gw}(:,C_{mb}(i,:))$,
9: $\quad$ **If** M satisfies condition-2 and condition-3 of Proposition 11 (of above state article), **then** $S\leftarrow M$ and $S_w\leftarrow M_w$.
10: $\quad\quad$ Identify the edges from the columns of S, and store the initial and final vertices of those edges in vectors c_s and c_t, respectively.
11: $\quad\quad T_{sw}(i,:)\leftarrow[c_s\ \ S_w(n+1,:)]$
12: $\quad\quad T_{tw}(i,:)\leftarrow[c_t\ \ S_w(n+1,:)]$
13: $\quad$ **else**
14: $\quad\quad$ **If** there is at least one zero column in M, and M satisfies condition-(2) and condition-(3) of Proposition 12 (of above state article), **then** $H_l\leftarrow M$ and $H_{lw}\leftarrow M_w$.
15: $\quad\quad\quad$ **If** there are self-loops at the vertices of $\mathscr{G}$ corresponding to the zero rows of H_l, and the UINs in H_{lw} correctly refer to the vertices, corresponding to the zero rows of H_l, **then**
16: $\quad\quad\quad\quad S_l\leftarrow H_l$ and $S_{lw}\leftarrow H_{lw}$.
17: $\quad\quad\quad\quad$ Identify the edges from the non-zero columns of S_l, and store the initial and final vertices of those edges in c_{sl} and c_{tl}, respectively. Identify the position of zero rows in S_l, and store them in s_{pl}.
18: $\quad\quad\quad\quad$ Construct $l_{sl}=[c_{sl}\ \ s_{pl}]$ and $l_{tl}=[c_{tl}\ \ s_{pl}]$.
19: $\quad\quad\quad\quad T_{swl}(i,:)\leftarrow[l_{sl}\ \ S_{wl}(n+1,:)]$
20: $\quad\quad\quad\quad T_{twl}(i,:)\leftarrow[l_{tl}\ \ S_{wl}(n+1,:)]$
21: $\quad\quad\quad$ **end if**
22: $\quad\quad$ **end if**
23: $\quad$ **end if**
24: **end for**
25: Set $\Theta_{sw}=\begin{bmatrix}T_{sw}\\ T_{swl}\end{bmatrix}$ and $\Theta_{tw}=\begin{bmatrix}T_{tw}\\ T_{twl}\end{bmatrix}$.

the matrices M and M_w are constructed by taking a particular combination (determined by i^{th} row of C_{mb} ($C_{mb}(i,:)$)) of the columns of matrices I_{ga} and I_{gw}, respectively. The matrix M_w is constructed so as to keep track of the UINs of the edges, associated with columns of M. In Step-9, the matrix M, which satisfies the conditions of Proposition 11, is stored in S, and the corresponding matrix M_w is stored in S_w. According to Proposition 11, the edges, determined by the columns of S, form a SCF of $\mathscr{G}$. Depending upon the position of 1 and -1 in a column of S, the edges of $\mathscr{G}$ are identified at Step-10, and the initial and final vertices of these edges are stored in vectors c_s and c_t, respectively. Note that the UINs of the edges, associated with constructed SCF, are available in the last row of S_w. Hence, the i^{th} row of matrices T_{sw} and T_{tw}, are constructed as: $T_{sw}(i,:) = [c_s \;\; S_w(n+1,:)]$ (Step-11) and $T_{tw}(i,:) = [c_t \;\; S_w(n+1,:)]$ (Step-12), respectively.

Steps: 14-25: In these steps, we obtain all possible SCFs of $\mathscr{G}$, which are having at least one self-loop. If M does not satisfy the conditions of Proposition 11, then at Step-14, it is verified if the conditions of Proposition 12 are satisfied. If M satisfies the conditions of Proposition 12, then it is stored in H_l, and the corresponding matrix M_w is stored in H_{lw}. Note that H_l has equal number of zero rows and columns. If there are self-loops at the vertices of $\mathscr{G}$, corresponding to the zero rows of H_l, then it follows from Proposition 12 that the edges characterized by the non-zero columns and zero rows of H_l, will form a SCF. This condition is verified at Step-15, and the matrix H_l, which satisfies this condition, is stored in S_l (Step-16). Further, the corresponding matrix H_{lw} is stored in S_{lw} by verifying that H_{lw} contains the correct UINs, corresponding to the zero rows. This is required, since all the columns of I_{gw}, corresponding to the self-loops, are only zero columns. For instance, assume that in H_l, 3^{rd} and 4^{th} rows are zero rows. Then, it is required to verify if the UINs assigned to the zero columns of H_{lw} correspond to the self-loops at vertices v_3 and v_4. The edges, corresponding to the non-zero columns of S_l, are identified at Step-17, and the initial and final vertices of these edges are stored in c_{sl} and c_{tl}, respectively. Further, the position of zero rows in S_l are identified, and stored them in s_{pl}. Following vectors: $l_{sl} = [c_{sl} \;\; s_{pl}]$ and $l_{tl} = [c_{tl} \;\; s_{pl}]$ are constructed in Step-18. Further, using the UINs of the edges, available in last row of S_{lw}, the i^{th} row of following two matrices T_{swl} and T_{twl}, are constructed: $T_{swl}(i,:) = [l_{sl} \;\; S_{wl}(n+1,:)]$ (Step-19) and $T_{twl}(i,:) = [l_{tl} \;\; S_{wl}(n+1,:)]$ (Step-20), respectively. Finally, at Step-25, $\Theta_{sw} = \begin{bmatrix} T_{sw} \\ T_{swl} \end{bmatrix}$ and $\Theta_{tw} = \begin{bmatrix} T_{tw} \\ T_{twl} \end{bmatrix}$ are constructed using T_{sw}, T_{tw}, T_{swl} and T_{twl}. Note that $\Theta_{sw}(i,1:n)$ and $\Theta_{tw}(i,1:n)$ correspond to the initial and final vertices of the edges, respectively, involved in the i^{th} SCF of $\mathscr{G}$. Further, the entries in $\Theta_{sw}(i,n+1:2n)$ and $\Theta_{tw}(i,n+1:2n)$ are equal, and they represent the UINs of the edges involved in i^{th} SCF of digraph $\mathscr{G}$.

Algorithm 2 Pseudocode for computing the vertex disjoint cycles in a spanning cycle family.

Input: The vectors: v_{cs} and v_{ct}, where the elements of v_{cs} and v_{ct} are the indices of initial and final vertices, respectively, of the edges in a SCF.

Output: The number of vertex disjoint cycles δ in the SCF.

1: Compute the number of self-loops (n_{sl}) in the SCF, by verifying $v_{cs}(i) = v_{ct}(i)$, for $i = 1,2,\cdots,n$, where n is the total number of edges in a SCF, which is equal to the number of vertices.
2: Construct $\bar{v}_{cs}$ and $\bar{v}_{ct}$ from v_{cs} and v_{ct}, respectively, by removing the self-loop edges. Set $V_l = \begin{bmatrix} \bar{v}_{cs} \\ \bar{v}_{ct} \end{bmatrix}$.
3: Set $l_p = 1$, $n_l = 0$ and $\bar{l} =$ number of columns in V_l.
4: **for** $i = 1$ to $\bar{l}$ **do**
5: Starting from $h_l = V_l(:,l_p)$, find the set of edges from V_l such that the initial vertex of succeeding edge is equal to the final vertex of preceding edge. Continue this process till $h_l(1,1)$ becomes the final vertex of an edge in V_l. Store the index of all those columns in x_p.
6: Set $n_l = n_l + 1$.
7: Remove the columns of V_l, indexed by x_p. Set $\bar{l} =$ number of remaining columns in V_l, and $l_p = 1$.
8: **if** V_l is empty, **then**
9: $\delta = n_l + n_{sl}$.
10: **stop**.
11: **end if**
12: **end for**

B. Algorithm 2 and its Explanation

In this algorithm, we compute the number of vertex disjoint cycles in a SCF. Two vectors: v_{cs} and v_{ct}, where the elements of v_{cs} are the indices of initial vertices and v_{ct} are the indices of final vertices of the edges in a SCF, are given as inputs to the algorithm. In Step-1, the number of self-loops, denoted as n_{sl}, are computed by verifying whether $v_{cs}(i) = v_{ct}(i)$, for $i = 1,2,\cdots,m$. In Step-2, new vectors $\bar{v}_{cs}$ and $\bar{v}_{ct}$ are constructed by removing the self-loop edges $(v_{cs}(i), v_{ct}(i))$ from v_{cs} and v_{ct}. Using $\bar{v}_{cs}$ and $\bar{v}_{ct}$, a new matrix V_l is defined, where the j^{th} column of V_l represents an edge $(V_l(1,j), V_l(2,j))$ of a cycle in the given SCF. In Step-5, the first column of V_l is considered as first edge, and then, a set of edges are searched in V_l such that the initial vertex of succeeding edge is equal to the final vertex of preceding edge. This process is continued until the initial vertex of first edge of this set $(V_l(1,1))$ becomes the final vertex of an edge in V_l. Then, according to the definition, these set of edges will form a cycle. Once such a set of edges are found out from V_l, the number of cycles n_l in the given SCF is updated by $n_l = n_l + 1$ in Step-6. Then, the set of edges (indexed by x_p), corresponding to the obtained cycle, are removed from V_l in Step-7. This process is continued until V_l becomes empty. By this time, we have counted all the cycles n_l (excluding self-loops) in the SCF. Hence, the total number of vertex disjoint cycles in the given SCF is $\delta = n_l + n_{sl}$, where n_{sl} is the number of self-loops in the SCF (Step-9). The algorithm is then terminated at Step-10.

C. Algorithm 3 and its Explanation

We assume that the structured matrices: $\overline{E}$, $\overline{A}$ and $\bar{b}$ of structured descriptor system $\overline{E}\dot{x} = \overline{A}x + \bar{b}u$ are given as input to the algorithm. To implement the algorithm in mathematical

Algorithm 3 Pseudocode for computing the structured gain vector $\bar{f}^*$ and $f^* \in \bar{f}^*$, which has minimum free-entries.

Input: 1) The structured matrices $\overline{E}$, $\overline{A}$, $\bar{b}$ and 2) the set of desired closed loop finite poles $\Lambda := \{\lambda_1, \lambda_2, \cdots, \lambda_r\}$.
Output: The structured gain vector $\bar{f}^*$ and $f^* \in \bar{f}^*$.

1: Set $f_{1j} = 1$, for all $j = 1, 2, \cdots, n$.
2: For the given $\overline{E}$, $\overline{A}$, $\bar{b}$, and $\bar{f}$, construct the matrix $\overline{H}_0$. Then, construct vectors: $\bar{s}$ and $\bar{t}$ by stacking the starting and ending vertices, respectively, of E-edges, A-edges, b-edges and f-edges in $\mathscr{G}(\overline{H}_0)$.
3: Construct ω_{eg} by stacking the respective edge weights.
4: Using Algorithm 1, compute Θ_{sw} and Θ_{tw}.
5: Compute the structural rank of $\overline{E}$.
6: Set $\overline{Z} = \mathbf{0}_{n\times(r+1)}$ and $Z = \mathbf{0}_{n\times(r+1)}$.
7: **for** $i = 1:1:l$, where l is the number of rows in Θ_{sw}, **do**
8: Compute the number of E-edges appeared in i^{th} SCF, and denote it as k. Determine the index j of f_{1j} that appears in i^{th} SCF.
9: Update $\overline{Z}(j, k+1) = \overline{Z}(j, k+1) + 1$
10: Compute the product of weights of the edges associated with i^{th} SCF, and denote it as p_w.
11: Set $v_{cs} = \Theta_{sw}(i, 1 : n+1)$, $v_{ct} = \Theta_{tw}(i, 1 : n+1)$. Compute the vertex disjoint cycles δ in i^{th} SCF using Algorithm 2.
12: Compute $p_{sw} = -1^{(\delta+n)} p_w$
13: Update $Z(j, k+1) = Z(j, k+1) + p_{sw}$
14: **end for**
15: Compute s-rank$(\overline{Z})$
16: **if** s-rank$(\overline{Z}) \neq r+1$, **then**
17: **Display** 'Numerical realizations of given SDS can not be made impulse-free, and place their finite poles at any desired location', and **Stop**.
18: **else**
19: **if** s-rank$(\overline{Z}) = r+1$, **then**
20: Using Algorithm 4, compute $\overline{Z}_{st} = \overline{Z}_s^T$.
21: Set $Z_m = \mathbf{0}_{n\times(r+1)}$ and $\bar{f}_{01}^* = \mathbf{0}_{1\times n}$.
22: **for** $j = 1:1:n$ **do**
23: **if** $\overline{Z}_{st}(:, j) \neq 0$, **then**
24: $Z_m^T(:, j) = Z^T(:, j)$ and $\bar{f}_{01}^*(1, j) = 1$
25: **end if**
26: **end for**
27: Construct $\alpha \in \bar{\alpha}$ from E, A, and $\sigma \in \bar{\sigma}$ from Λ.
28: Solve $Z_m^T f^* = \alpha - \sigma$ for f^*.
29: **end if**
30: **end if**

software, we assign some randomly chosen non-zero real numbers to the free-entries of $\overline{E}$, $\overline{A}$ and $\bar{b}$, so that their zero/free-entry patterns can be preserved. If the exact numerical entries of $\overline{E}$, $\overline{A}$ and $\bar{b}$ are known, then those numerical values need to assigned instead of random numbers to the free entries of $\overline{E}$, $\overline{A}$ and $\bar{b}$. In addition, a set of complex numbers (including their conjugates): $\Lambda := \{\lambda_1, \lambda_2, \cdots, \lambda_r\}$, which are the desired closed-loop finite poles (may be randomly chosen), are also given input to the algorithm. The algorithm will produce a gain vector $f^* \in \bar{f}^*$, which has minimum number of free-entries, as an output. In addition, one can also extract the zero/free-entry pattern gain vector $\bar{f}^*$ from the algorithm. The step-wise explanations of Algorithm 3 is discussed next.

Steps 1-5: Since the exact structure of the feedback gain vector $\bar{f}$ is not known a priori, it is assumed that all of its elements f_{1j} are free-entries. Hence, similar to the assignment of non-zero real numbers to the free-entries of $\overline{E}$, $\overline{A}$ and $\bar{b}$ (to preserve their zero/free-entry patterns), the free-entries of $\bar{f}$ are set to 1 (Step-1). The choice of assigning 1 to the free-entries of $\bar{f}$ will help us to use Lemma 8 of the above stated article in computing $\overline{Z}$. In Step-2, the vectors $\bar{s}$ and $\bar{t}$ are constructed by stacking the indices of starting and ending vertices, respectively, of E-edges, A-edges, b-edges and f-edges in $\mathscr{G}(\overline{H}_0)$ (refer to Section 3.1 of the above stated article). The weights of E-edges, A-edges, b-edges and f-edges (the randomly chosen non-zero real numbers assigned to the free-entries) are stored in the vector ω_{eg} (Step-3). Then, in Step-4, the matrices Θ_{sw} and Θ_{tw}, associated with the SCFs of $\mathscr{G}(H_o)$, are computed using Algorithm 1 (see Appendix - C). The i^{th} rows of Θ_{sw} and Θ_{tw} gives the following information: i) $\Theta_{sw}(i, 1 : n+1)$, $\Theta_{tw}(i, 1 : n+1)$ are the starting and ending vertices, respectively, of the edges associated with i^{th} SCF, and ii) $\Theta_{sw}(i, n+2 : \text{end})$, $\Theta_{tw}(i, n+2 : \text{end})$ are the unique identification numbers (UINs) used for the edges involved in i^{th} SCF. The structural rank of $\overline{E}$ is computed in Step-5 (one may use the MATLAB command *sprank*).

Steps 6-14: Two matrices $\overline{Z}$ and Z, having zero entries with size $n \times (r+1)$, are defined at Step-6, and the entries are updated in the subsequent steps. According to Lemma 8, the entries of $\overline{Z}$ are completely determined by the number of E-edges, and the f-edge used in a SCF. Hence, at Step-8, the total number of E-edges used in i^{th} SCF is computed, and is denoted as k. Further, the f-edge, that is, the index j of f_{1j} that appears in i^{th} SCF is determined. The above two quantities are obtained using the UINs assigned to each edge of i^{th} SCF. Since all the f-edges enter to the vertex v_{n+1}, every SCF of $\mathscr{G}(H_o)$ can have only one f-edge (according to the definition, at each vertex of a SCF at most one edge can enter and one edge can leave). To determine the structural rank of $\overline{Z}$, we only need its zero/free-entry pattern, and not its exact entries. Hence, the zero/free-entry pattern of $\overline{Z}$ is stored in $\overline{Z}$ matrix, by updating its entries as follows (Step-9): $\overline{Z}(j, k+1) = \overline{Z}(j, k+1) + 1$, if i^{th} SCF contains k E-edges and a f-edge with edge weight f_{1j}. This step directly follows from Lemma 8. When the exact numerical entries of system matrices $\overline{E}$, $\overline{A}$ and $\bar{b}$ are specified, in terms of the edge weights, then a numerical realization Z of $\overline{Z}$ can be obtained as follows. The product of weights of the edges associated with i^{th} SCF is computed in Step-10 using the elements of ω_{eg}, and stored it in p_w. Further, the number of vertex disjoint cycles δ in i^{th} SCF is computed using Algorithm 2 in Step-11. Then, in Steps: 12-13, the entries of matrix Z are updated as follows: $Z(j, k+1) = Z(j, k+1) + p_{sw}$, where $p_{sw} = -1^{(\delta+n)} p_w$, which follows from Lemma 8.

Steps 15-30: The structural rank of $\overline{Z}$ is computed in Step-15. According to Theorem 9, if s-rank$(\overline{Z}) \neq r+1$, the numerical realizations of given SDS can not be made impulse-free and place their finite poles at desired locations, and hence,

the algorithm will terminate at Step-17. If s-rank$(\overline{Z}) = r+1$, then using Algorithm 4, the structured matrix $\overline{Z}_{st} = \overline{Z}_s^T$ is constructed from $\overline{Z}$ in Step-20. Note that none of the free entries of $\overline{Z}_{st}$ lie in a same line (rows or columns). In the subsequent steps, we follow the discussion mentioned in the proof of second statement of Theorem 9. At Step-21, a zero matrix Z_m of size $n \times (r+1)$ is defined, whose j^{th} column is updated as follows: $Z_m^T(:,j) = Z^T(:,j)$, if $\overline{Z}_{st}(:,j) \neq 0$ (Step-22). In addition, a zero vector $\bar{f}_{01}^*$ is defined at Step-21, and then its entries: $\bar{f}_{01}^*(1,j)$ are set to 1, if $\overline{Z}_{st}(:,j) \neq 0$ at Step-22. Note that the resulting vector $\bar{f}_{01}^*$, after the end of *for* loop, gives the zero/free-entries pattern of gain vector $\bar{f}^*$. The 1's in $\bar{f}_{01}^*$ indicate the free-entries. The exact value of free-entries of $\bar{f}_{01}^*$, that is, $f^* \in \bar{f}_{01}^*$ is determined by: $E \in \overline{E}$, $A \in \overline{A}$, $b \in \bar{b}$. Recall that $\overline{E}$, $\overline{A}$ and $\bar{b}$ are given input to the algorithm, by assigning some randomly chosen non-zero real numbers to their free-entries. Assume that these matrices describe a particular numerical realization of structured descriptor system. Then, using these input matrices, the coefficient vector $\alpha \in \bar{\alpha}$ is constructed. Further, σ is constructed from the given set of desired closed-loop finite poles Λ (Step-27). Finally, the gain vector $f^* \in \bar{f}_{01}^*$ is obtained by solving a set of linear equations: $Z_m^T f^* = \alpha - \sigma$ at Step-28 (for this, in MATLAB *linsolve* function is used).

D. Algorithm 4 and its Explanation

Algorithm 4 Pseudocode for computing the structured matrix $\overline{Z}_s$, whose none of the entries present in a same line (row or column).

Input: 1) The structured matrix $\overline{Z}$.
Output: The structured matrix $\overline{Z}_s$.

1: Given a matrix $\overline{Z}$, represent it as a bipartite graph, as stated in the above article.
2: Represent the bipartite graph as a network flow graph as follows: i) add two extra vertices: source and sink, ii) connect all the c_i vertices of the bipartite graph with source vertex having direction from source vertex to c_i, and iii) connect all r_i vertices to sink vertex with edges having direction from r_i to sink vertex.
3: Relabel the vertices, and put the weight of all the edges of the network flow graph as 1.
4: Use the MATLAB command *maxflow* to extract the edges associated with a maximum matching of the bipartite graph.
5: Construct the matrix $\overline{Z}_s$ using the edges associated with obtained maximum matching of the bipartite graph.

Steps 1-4: For a given structured matrix $\overline{Z}$, obtained at the end of Step 14 of Algorithm 3, a bipartite graph is first constructed. Since we are representing the bipartite graph as a network flow graph, by adding two extra vertices: source and sink in Step 2, the vertices of bipartite graph are labeled carefully. In a bipartite graph, there is an edge from vertex c_1 to r_3 if $z_{31} \neq 0$ in $\overline{Z}$. Since we need to add a new vertex *source* in the network flow graph, the vertices c_i of the bipartite graph are started with index 2, and source vertex is given as v_1, as shown in Figure 4. The vertex set r_i of the bipartite graph are started with $v_k + 1$, where index k is the last vertex index of c_i in the network flow graph. Similarly, the sink vertex is index as the last vertex of the network flow graph. Then, the starting and ending vertices of all the edges in the network flow graph are stored in two vectors. By assigning the weight of all the edges as 1, the edges associated with a *maximum matching* of the bipartite graph is extracted using the MATLAB command *maxflow*.

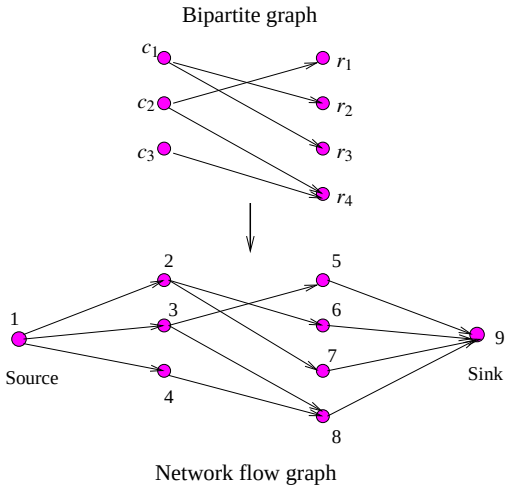

Fig. 4. A bipartite graph is converted to a network flow graph, with relabeling the vertices, which is used for Algorithm 4.

Step 5: Once the edges associated with a maximum matching of the bipartite graph is obtained, the vertices are reassigned with the correct vertices of the bipartite graph, that is, in terms of c_i and r_i. Then, the elements of matrix $\overline{Z}_s$ are updated as follows: $\overline{Z}_s(i,j) = \overline{Z}(i,j)$ if an edge c_j to r_i is present in the computed maximum matching.

IV. Demonstrative Example

Example 1: Consider a structured descriptor system where $\overline{E}$, $\overline{A}$ be as in (2), and $\bar{b} = \begin{bmatrix} 0 & b_{21} & b_{31} \end{bmatrix}^T$. The digraph $\mathscr{G}(\overline{H}_0)$ and the associated spanning cycle families are shown in Figure 5. To implement the algorithm in MATLAB, the free-entries of $\overline{E}$, $\overline{A}$ and $\bar{b}$ are assigned with randomly chosen real numbers as follows: $e_{11} = 0.8147$, $e_{13} = 0.9058$, $e_{23} = 0.1270$, $e_{33} = 0.9134$, $a_{12} = 0.6324$, $a_{21} = 0.0975$, $a_{33} = 0.2785$, $b_{21} = 0.5469$ and $b_{31} = 0.9575$. Then, the SCFs, as shown in Figure 6, and obtained from Algorithm 3 (from Step - 4) are:

$$\Theta_{sw} = \begin{bmatrix} 3&1&4&2&7&10&12&14 \\ 3&2&4&1&8&9&12&13 \\ 2&1&4&3&9&10&12&15 \\ 3&4&2&1&8&12&14&16 \\ 2&4&1&3&9&11&13&17 \\ 2&4&1&3&9&11&13&18 \\ 4&2&1&3&11&14&16&17 \\ 4&2&1&3&11&14&16&18 \end{bmatrix}, \quad \Theta_{tw} = \begin{bmatrix} 1&2&3&4&7&10&12&14 \\ 2&1&3&4&8&9&12&13 \\ 1&2&3&4&9&10&12&15 \\ 2&3&4&1&8&12&14&16 \\ 1&2&4&3&9&11&13&17 \\ 1&2&4&3&9&11&13&18 \\ 2&4&1&3&11&14&16&17 \\ 2&4&1&3&11&14&16&18 \end{bmatrix}.$$

Using Lemma 8 and the SCFs, as in Figure 6, the matrix $\overline{Z}$ is obtained as follows:

$$\overline{Z} = \begin{bmatrix} -a_{12}a_{33}b_{21} & a_{12}e_{33}b_{21} - a_{12}e_{23}b_{31} & 0 \\ 0 & -a_{33}e_{11}b_{21} - a_{21}e_{13}b_{21} & z_{23} \\ -a_{12}a_{21}b_{31} & 0 & 0 \end{bmatrix},$$

where $z_{23} = e_{11}e_{33}b_{21} - e_{11}e_{23}b_{31}$. Moreover, following matrices:

$$\overline{Z} = \begin{bmatrix} 1 & 2 & 0 \\ 0 & 2 & 2 \\ 1 & 0 & 0 \end{bmatrix}, \quad \overline{Z}_{st} = \begin{bmatrix} 0 & 0 & 1 \\ 2 & 0 & 0 \\ 0 & 2 & 0 \end{bmatrix},$$

$\bar{f}^* = [1\ 1\ 1]^T$ are obtained from Algorithm 3. For the choice of closed loop finite poles -3 ± 1, following gain vector is obtained $f^* = [-27.6883\ \ -3.1967\ 214.2639]^T$.

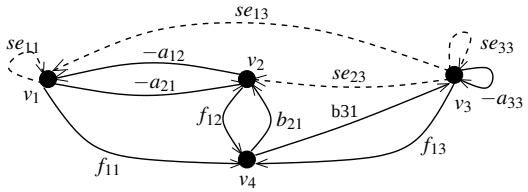

Fig. 5. Digraph $\mathscr{G}(\overline{H}_0)$, corresponding to the matrices: $\overline{E}$, $\overline{A}$, $\overline{f}$ as in (2), and $\overline{b} = [0\ b_{21}\ b_{31}]$.

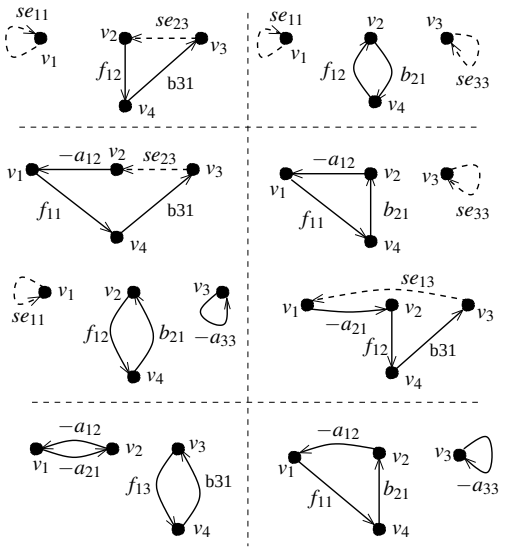

Fig. 6. SCFs of $\overline{H}_0$ for matrices: $\overline{E}$, $\overline{A}$, $\overline{f}$ as in (2), and $\overline{b} = [0\ b_{21}\ b_{31}]$.