Efficient optimal equation formulation in lumped power-conserving systems

J.D. Lamb *

Canterbury Business School, University of Kent at Canterbury, Canterbury, Kent, CT2 7PE, UK

Received 19 August 1996; received in revised form 16 February 1998; accepted 23 February 1998

Abstract

Sets of inputs and outputs are defined for lumped power-conserving systems, and a set of inputs is defined to be consistent if the corresponding set of outputs can be written in terms of it. To find a set of state equations, one needs a consistent set of inputs. Given one consistent set of inputs it is shown (1) how to test whether any other set of inputs is consistent, and (2) given a preference ordering on all sets of inputs with certain additional properties, how to find an optimal set. The algorithm for (2) is shown to be $O(m^3)$-time, where $m$ is the number of external elements of the system. Its application is to finding optimal sets of state equations.

Keywords: Equation formulation; Lumped system; Tellegen's theorem; Δ-matroid

1. Introduction

Lumped power-conserving systems are very common in engineering modelling. Electrical networks, for example, are lumped power-conserving systems. So are bond graphs and (usually) block diagrams and signal flow graphs, all of which are described in [8]. Lumped power-conserving systems also occur in physics (for example, as models of simple harmonic oscillators) and have been described in areas as diverse as biology, chemistry and economics (see [16]).

Lumped (as opposed to distributed) means that a system is modelled as a number of external elements with ideal connections between them. That is, only the properties of the external elements and the combinatorial structure of the interconnections matter. We need not worry about, for example, the speed of energy transfer, or capacitance effects in the interconnections. We will only consider one-port external elements in this paper. However, the reader can check that the results of Section 3 can easily be extended to multiport elements, while those of Section 5 can be extended to some multiport elements. Since transformers and gyrators are power-conserving, we will assume

* Fax. +44 1227 827932, e-mail: j.d.lamb@ukc.ac.uk.

0166-218X/98/$19.00 © 1998 Elsevier Science B.V. All rights reserved

PHYSICA 0166-218X(98)00037-7
that any in the system are modelled as part of the interconnections between external
elements that is, as internal elements and not as external elements. We will take
power-conserving to mean that a system satisfies the following form of Tellegen’s the-
orem. We will assume that the system has \( m \) external elements labelled \( b_1, \ldots, b_m \), and
we will assume that each external element \( b_i \) has two variables \( e_i \) and \( f_i \) associated
with it, representing the energy and coenergy variables in the external element, and
called the effort and flow, respectively. Then we say that the system satisfies Tellegen’s
theorem if

\[
\sum_{i=1}^{m} \sigma_i e_i f_i = 0,
\]

where \( \sigma_i = +1 \) or \( -1 \) according as the effort and flow are or are not taken as having
the same direction through or across \( b_i \). (For electrical circuits it is usual to take
the effort and flow directions to be the same for all external elements so that each
\( \sigma_i = 1 \).) This form of Tellegen’s theorem is proven for electrical networks in [12].
The analogous result for bond graphs is proven in [10]. Tellegen’s original result
[17] is more general in form, but restricted to electrical circuits which do not contain
transformers or gyrators. (The more general version of his result does not hold for
networks with gyrators.)

Shortly, we will make further (reasonable) assumptions about the systems we will
deal with so that we can derive the results of the next three sections. However, in
general, they can be linear or nonlinear, constant or time-varying, continuous or dis-
continuous, deterministic or stochastic.

In Section 2 we will see how sets of state equations can be formulated, and Sec-
ction 3 will show how the inputs for a system can be rearranged for different equation
formulations. Section 4 will show how to construct a preference ordering on the inputs
of a system. Section 5 will show that this ordering has certain properties that allow us
to construct a polynomial algorithm to find an optimal set of state equations.

2. Formulation of state equations

We need two further assumptions to be able to formulate state equations. We first
need some definitions. A set of inputs \( x = (x_1, \ldots, x_m)^\top \) and the corresponding set of
outputs \( y = (y_1, \ldots, y_m)^\top \) are column vectors such that for each \( i \) \((1 < i < m)\) either
\( x_i = e_i \) and \( y_i = f_i \) or \( x_i = f_i \) and \( y_i = e_i \). The operators \( \partial, \partial^{-1} \) and \( t \) are defined by

\[
\partial x_i = \frac{dx_i}{dt}, \quad \partial^{-1} x_i = \int_{0}^{t} x_i(y) \, dy, \quad \text{and} \quad t x_i = x_i,
\]

where \( t \) and \( y \) are variables representing time.

The first assumption we make is that, for some set of inputs \( x \), with corresponding
outputs \( y \), we can write an equation of the form

\[
Dx = f(y, t),
\]
where $D$ is a diagonal matrix each of whose entries is one of the operators $\partial$, $\partial^{-1}$ or $i$, and $f$ is a function of the outputs $y$ and time $t$.

The second assumption we make is that, for some set of inputs $x'$ with corresponding outputs $y'$, we can write

$$y' = Qx',$$

where $Q$ is a matrix with real entries. We will call a set of inputs $x'$ consistent if we can write a set of equations of this form for it; otherwise we will call $x'$ inconsistent.

Much work on systems theory has been devoted to finding equations of the form of (3). For electrical circuits without transformers or gyrators, these can be found from Kirchhoff's laws using the methods described, for example, in [14]. For electrical networks with transformers and gyrators the problem is much harder. It is well known [18, 9] that the problem can be solved if a normal spanning forest can be found for the network. Recski [18] shows that finding a normal spanning forest is a special case of the linear two-polymatroid matching problem, which has been solved by Lovász [13] and more efficiently and more recently by Gabow and Stallmann [6]. For bond graphs, Lamb et al. [10, 11] have described algorithms for finding equations of the form of (3). This paper will not consider further how to find such a set of equations, but only how to manipulate it once it is found.

Notice that if the sets of inputs of Eqs. (2) and (3) are the same (i.e. if $x' = x$), then we can write a set of equations

$$Dx = g(x, t),$$

where $g(x, t) = f(Qx, t)$. This is a set of state equations for the system.

It is usually easy to write down equations of the form of (2). We can see this in the following example, which we will use later to illustrate some of the problems involved in formulating Eq. (4) above. Suppose voltage is represented by effort and current by flow in an electrical system containing a voltage source $v_1$, a linear capacitor $v_2$, a linear resistor $v_3$ and a diode $v_4$, represented by the equations

$$e_1 = V_0 \sin(\omega t), \quad e_2 = Rf_2,$$

$$e_3 = \frac{1}{C} \int_0^t f_3(y) \, dy, \quad \text{and} \quad f_4 = \begin{cases} 0 & \text{if } -V_b < e_4 < V_f, \\ De_4 & \text{otherwise.} \end{cases}$$

We can represent the same system by replacing the second and third equations with

$$f_2 = \frac{1}{R}e_2, \quad \text{and} \quad f_3 = C \frac{de_3}{dt}.$$

Note that we cannot rearrange the equation associated with the voltage source or with the diode. One form of the capacitor's equation is likely to be preferred over the other, but the resistor is equally well expressed in either form.
There are at least three possible approaches to formulating a set of state equations.

(i) The usual approach is to choose a set of inputs $x$ for which we know we can write a set of equations of the form of (2), then to try to write a set of equations of the form of (3) with $x' = x$. The problem with this approach is that it fails if $x$ is inconsistent. Then the best we can do is to start again with a different $x$ for which we can write down equations of the form of (2).

(ii) An alternative approach is to look for sets of equations of the forms of (2) and (3) without regard to whether $x = x'$, and then to try to rearrange eqs. (3) so that the vector $y$ of Eq. 2 appears on the left. This approach will be an improvement on the first if it is easier to rearrange Eq. (3) than to formulate them. Section 3 will describe how we can try to rearrange them.

(iii) It is usually the case that we can write many sets of equations of the form of (2), but that we prefer some over others. For example, we might prefer one form because, as in the example above, it contains only differential rather than some integral equations. Or we might prefer one form because it gives better numerical accuracy in some simulation of the system. In general, we may suppose that there is some preference ordering on choices of inputs for the system, so that we wish to find an optimal set of inputs that is consistent. Section 4 will show how we might find, in practice, a preference ordering that has an additional property that we can use in Section 5 to devise an efficient algorithm to find an optimal set of state equations.

3. Rearranging inputs

Suppose we can write a set of equations of the form of (3). Then $x'$ is a consistent set of inputs. Now, suppose $x$ is another set of inputs. Then we can suppose, without loss of generality, that

$$
x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \quad y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}, \quad x' = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}, \quad \text{and} \quad y' = \begin{pmatrix} x_1 \\ y_2 \end{pmatrix}.
$$

\[ (\begin{pmatrix} y_1 \\ y_2 \end{pmatrix}) = \begin{bmatrix} Q_1 & Q_2 \\ Q_3 & Q_4 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}. \]

\[ \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \mathcal{Q}' \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \]

Theorem 1. The equation

\[ \begin{pmatrix} x_1 \\ y_2 \end{pmatrix} = \begin{bmatrix} Q_1 & Q_2 \\ Q_3 & Q_4 \end{bmatrix} \begin{pmatrix} y_1 \\ x_2 \end{pmatrix} \]

can be expressed in the form
if and only if $Q_1$ is nonsingular. Moreover, if $Q_1$ is nonsingular, then

$$Q' = \begin{bmatrix}
Q_1^{-1} & \frac{-Q_1^{-1}Q_2}{Q_1 - Q_1Q_1^{-1}Q_2}
\end{bmatrix}.$$

Proof. Suppose Eq. (6) can be expressed in the form

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{bmatrix}
Q_1' & Q_2' \\ Q_3' & Q_4'
\end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

(7)

for some $Q_1'$, $Q_2'$, $Q_3'$ and $Q_4'$. Put $x_2 = 0$. Then, from Eq. (6), $x_1 = Q_1y_1$ and, from Eq. (7), $y_1 = Q_1'x_1$ for all $x_1$ and $y_1$. So $x_1 = Q_1'x_1$ for all values of $x_1$. Hence, $Q_1Q_1' = I$ and so $Q_1$ is nonsingular.

Conversely, suppose that $Q_1$ is nonsingular. Then, from Eq. (6), $x_1 = Q_1y_1 + Q_2x_2$. So $y_1 = Q_1^{-1}x_1 - Q_1^{-1}Q_2x_2$. So $y_2 = Q_3Q_1^{-1}x_1 + (Q_4 - Q_5Q_1^{-1}Q_2)x_2$.

Thus, Eq. (2) can be expressed as

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{bmatrix}
Q_1^{-1} & \frac{-Q_1^{-1}Q_2}{Q_1 - Q_1Q_1^{-1}Q_2}
\end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

as required. □

How can we use this to try to formulate a set of state equations? First, we need a consistent set of inputs $x'$ and a set of equations of the form of Eq. (3). Then we need a set $x$ of inputs corresponding to a set of equations of the form of Eq. (2). We can assume, without loss of generality, that $x$ and $x'$ are written in the form of Eq. (5) because we can reorder the sets of equations if they are not. Thus, we can apply Theorem 1, and if $x$ is shown to be consistent then we can use the theorem to write

$$x = Q'y$$

(8)

and combine these with Eq. (2) to get a set of state equations.

Two comments should be made at this point. First, the computational complexity of this method is clearly the same as that of inverting a matrix of size at most $m \times m$, which (see [15]) is at most $O(m^3)$. Since, for an electric circuit without transformers or gyrators, approach (i) can be implemented in at most $O(m^2)$-time, the method I have just described is not the most efficient for such a circuit. However, it is more efficient than the repeated searches for normal spanning forests required for more general electrical networks. It is also at least as efficient as the causality methods for bond graphs described in [11].

The second comment is that, since we have not used Tellegen's Theorem in this section, the method is applicable to nonpower-conserving systems.
4. Optimal equation formulation

This section will introduce two assumptions about the systems we deal with. It will use the example of Section 2 to illustrate why, regardless of how the external elements are interconnected, the assumptions will be reasonable for many systems. The assumptions will allow us to construct a weight function \( w \) on the consistent sets of inputs that has two important properties. The first, that \( w(x_1) \geq w(x_2) \) if \( x_1 \) is at least as good a choice of inputs as \( x_2 \), is obviously useful. The second is less obviously useful. Section 5 will derive it in Proposition 2, then show why it is useful.

The first assumption is that, for each \( b_i (1 \leq i \leq m) \), we can define \( u_i \) to be one of \( e_i, f_i \) and \( v_i \) to be the other so that \( u_i \) is always at least as good a choice of input as \( v_i \). In the example of Section 2 we must choose as inputs the voltage across the voltage source and the current through the diode (unless we are prepared to go back and model these elements differently). Then (usually) we prefer the current through the capacitor to be an input, so that we get a differential equation in Eq. (2). Finally, we can choose either the voltage or the current to be the input for the resistor. The first assumption is usually reasonable given our assumption that all external elements are one-ports. One possible exception might be that if some element \( b_i \) were a switch, we might wish the better choice of inputs to change when the position of the switch is changed. If we allow multiport elements, then this assumption will often remain reasonable – the most obvious exceptions are transformers and gyrators, which we assume to be modelled not as external elements but as part of the interconnections of our system.

The second assumption is that, for some integer \( k \geq 0 \), each external element \( b_i (1 \leq i \leq m) \) can be assigned a priority \( P(b_i) \) in the range \( \{0, \ldots, k\} \) such that the following hold.

(i) We always choose inputs of higher priority before those of lower priority. Thus, having one more input of priority \( j \) is better than having any number more of priority less than \( j \).

(ii) If \( P(b_i) = 0 \) there is no preference for \( u_i \) over \( v_i \) as an input.

Again, we can illustrate this with our example. We can choose \( v_1 = e_1, v_2 = e_2, v_3 = f_3 \) and \( v_4 = f_4 \), and clearly we should assign priorities as \( P(b_1) = P(b_4) = 2, P(b_3) = 1 \) and \( P(b_2) = 0 \).

We now define rank functions on the external elements and on the set of sets of inputs as follows. For each \( j (1 \leq j \leq k) \), let \( n_j \) be the number of external elements \( b_i \) with \( P(b_i) = j \). Define a function \( \rho \) on \( \{0, \ldots, k\} \) recursively by putting \( \rho(0) = 0 \) and

\[
\rho(s) = 1 + \sum_{j=0}^{s-1} n_j \rho(j) \quad (1 \leq s \leq k).
\]

We can define the rank of each external element \( b_i \) by \( r(b_i) = \rho(s) \) if \( P(b_i) = s \). Then we define the rank of a set \( x \) of inputs by

\[
r(x) = \sum_{u_i} r(b_i) - \sum_{v_i} r(b_i),
\]  

(9)
where the first sum is over all \( b_i \) such that \( u_i \) is an element of \( x \) and the second sum is over all \( b_i \) such that \( v_i \) is an element of \( x \). In our example we get \( \rho(b_2) = 0 \), \( \rho(b_3) = 1 \) and \( \rho(b_4) = 2 \). Thus, \( r(e_1, e_2, f_3, f_4)^T = r(e_1, f_2, f_3, f_4)^T = 5 \), \( r(e_1, e_2, e_3, f_4)^T = r(e_1, f_2, e_3, f_4)^T = 3 \), \( r(f_1, e_2, f_3, f_4)^T = r(f_1, f_2, f_3, f_4)^T = 5 \), \( r(e_1, e_2, f_3, e_4)^T = r(e_1, f_2, f_3, e_4)^T = 1 \), etc. This tells us that the best choice of inputs is \( (e_1, e_2, f_3, f_4)^T \) or \( (e_1, f_2, f_3, f_4)^T \). If neither of these turns out to be consistent, the next best choice is \( (e_1, e_2, e_3, f_4)^T \) or \( (e_1, f_2, e_3, f_4)^T \), etc.

The problem, in general, is to find a consistent set of inputs of largest rank, or equivalently a set of inputs of largest weight where the weight is defined by

\[
w(x) = \begin{cases} 
    r(x) & \text{if } x \text{ is consistent,} \\
    -\infty & \text{otherwise.} 
\end{cases}
\]

\[ (10) \]

5. An algorithm for optimal equation formulation

Recall from Eq. (1) that

\[
\sum_{i=1}^{m} \sigma_i e_i f_i = 0,
\]

where each \( \sigma_i = \pm 1 \). Now, define \( S = \text{diag}(\sigma_1, \ldots, \sigma_m) \). Then clearly \( S^2 = I \), the identity matrix, and so \( S \) is nonsingular. Suppose, \( x' = (x_1, \ldots, x_m)^T \) is a consistent set of inputs and \( y' = (y_1, \ldots, y_m)^T \) is the corresponding set of outputs. Put \( A = [a_{ij}]_{m \times m} = SQ \) where \( Q \) is the matrix of Eq. (3). Then, for all values of \( x' \),

\[
\sum_{i=1}^{m} \sum_{j=1}^{m} a_{ij} x_i x_j = (A x') \cdot x' = (S Q x') \cdot x' = (S y') \cdot x' = \sum_{i=1}^{m} \sigma_i e_i f_i = 0.
\]

Thus, the coefficient of \( x_i x_j \) in the sum on the left is zero for all \( i \) and \( j \). But if \( i = j \) this coefficient is \( a_{ii} \), and if \( i \neq j \), this coefficient is \( a_{ij} + a_{ji} \). It follows that, for all \( i, j, a_{ii} = 0 \) and \( a_{ij} = -a_{ji} \), and so \( A \) is skew-symmetric. Now, suppose \( Q_1 \) is a principal submatrix of \( Q \): that is, a submatrix obtained by deleting similarly indexed rows and columns of \( Q \). And suppose \( A_1 \) and \( S_1 \) are the corresponding principal submatrices of \( A \) and \( S \). Then \( A_1 \) is skew-symmetric and

\[
\det(Q_1) = \det(S_1 Q_1) = \det(S_1 A_1) = \det(S_1) \det(A_1) = \pm \det(A_1).
\]

Hence, \( Q_1 \) is nonsingular if and only if \( A_1 \) is nonsingular.

What has all this to do with finding an optimal consistent set of inputs? Put \( M = \{1, \ldots, m\} \) and, for \( J \subseteq M \) let \( x_J = (z_1, \ldots, z_m)^T \) where \( z_i = y_i \) if \( i \in J \) and \( z_i = x_i \) if \( i \in M \setminus J \). (In particular, \( x_\emptyset = x' \).) Define also, for \( J \subseteq M \),

\[
w(J) = w(x_J).
\]

(11)

Then one (unsuccessful) approach to finding an optimal set of inputs would be the following (greedy) algorithm. Starting with \( J = \emptyset \), add successive elements \( j \in M \setminus J \)
to \( J \), choosing at each step the element \( j \) that maximises \( w(J \cup \{j\}) \) and stopping if there is no \( j \) such that \( w(J \cup \{j\}) \geq w(J) \). However, in Section 3 we saw that \( x_J \) is consistent (and hence \( w(J) \neq -\infty \)) if and only if some matrix \( Q_1 \) is nonsingular, and we have just seen that \( Q_1 \) is nonsingular if and only if a certain skew-symmetric matrix of order \(|J|\) is nonsingular. Since (see [4]) every skew-symmetric matrix of odd order is singular, it follows that this method will fail at the first step.

Now, suppose that, instead of trying to add just one element \( j \) to \( J \) in each step, we try to add the best pair of elements \( i, j \). Then, it turns out that we obtain an algorithm that finds an optimal set of inputs. It is this algorithm that we will shortly describe more formally. We can show that the algorithm works by using combinatorial properties of skew-symmetric matrices that are readily obtainable by extending the results of Brill [1, 2] and Heymans [7]. However, it is convenient to use more recent independent work on \( \triangle \)-matroids.

To describe the algorithm more formally, we will need some more notation. If \( A \) and \( B \) are sets, we define \( A \triangle B = (A \cup B) \setminus (A \cap B) \). This operation is sometimes called the Boolean sum or symmetric difference of \( A \) and \( B \). Note that \( \triangle \) is associative and commutative, and that \( A \triangle \{x\} \triangle \{x\} = A \).

The following algorithm constructs a set \( J \). The set of inputs \( x_J \) is the required optimal consistent set. Note that \( i \) is in the range of possible values for \( j \) in step 4 and so \( w(J) \) never decreases. Note also that we can find \( w(J \triangle \{i\} \triangle \{j\}) \) in step 4 from Eq. (11) using the test for consistency of \( x_{J \triangle \{i\} \triangle \{j\}} \) developed in Section 3.

**Algorithm 1**

1. \( J = \emptyset; N = M \).
2. while \( N \neq \emptyset \).
3. \( \text{choose } i \in N. \)
4. \( \text{choose } j \in N \text{ with } w(J \triangle \{i\} \triangle \{j\}) \text{ as large as possible.} \)
5. \( J = J \triangle \{i\} \triangle \{j\}. \)
6. \( N = N \setminus \{j\}. \)
7. end while.

It remains to prove that \( x_J \) constructed by Algorithm 1 really is optimal. To do this we need another result. Bouchet [3, Section 4] makes essentially the following definitions. Suppose \( A \) is an \( m \times m \) matrix and \( M = \{1, \ldots, m\} \). For \( X \subseteq M \), let \( A(X) \) be the principal submatrix of \( A \) indexed by \( X \). Let \( \mathcal{F}(A) = \{X: X \subseteq M \text{ and } A(X) \text{ is non-singular}\} \). Using a result of Bouchet [3, 4.1], Wenzel [19, Example (ii)] shows that if \( A \) is skew-symmetric, \( \mathcal{F} = \mathcal{F}(A) \) satisfies the following strong exchange axiom.

\((\text{SEA})\) For \( F_1, F_2 \in \mathcal{F} \) and \( e \in F_1 \triangle F_2 \), there exists \( f \in (F_1 \triangle F_2) \setminus \{e\} \) such that \( F_1 \triangle \{e, f\} \in \mathcal{F} \) and \( F_2 \triangle \{e, f\} \in \mathcal{F} \).

Since \( \mathcal{F}(Q) = \mathcal{F}(A) \), it follows immediately that \( \mathcal{F} = \mathcal{F}(Q) \) also satisfies (SEA).
Proposition 2. Let $M = \{1, \ldots, m\}$ and let $w : M \rightarrow \mathbb{R} \cup \{-\infty\}$ be the function defined in Eq. (11). Then the following hold.

(VA1) There exists some $J \subseteq M$ with $w(J) \neq -\infty$.

(VA2) For $J_1, J_2 \subseteq M$ with $w(J_1) \neq -\infty$ and $w(J_2) \neq -\infty$, and for every $i \in J_1 \triangle J_2$, there exists some $j \in (J_1 \triangle J_2) \setminus \{i\}$ such that

$$w(J_1) + w(J_2) \leq w(J_1 \triangle \{i\} \triangle \{j\}) + w(J_2 \triangle \{i\} \triangle \{j\}).$$

Proof. Since $x_0 = x'$ is consistent, $w(\emptyset) \neq -\infty$; so (VA1) holds.

Suppose, $J_1, J_2 \subseteq M$ with $w(J_1) \neq -\infty$ and $w(J_2) \neq -\infty$. Then $x_{J_1}$ and $x_{J_2}$ are consistent, and so $J_1 \in \mathcal{F}(Q)$ and $J_2 \in \mathcal{F}(Q)$. Suppose further that $i \in J_1 \triangle J_2$. Then, by (SEA), there exists $j \in (J_1 \triangle J_2) \setminus \{i\}$ such that $J_1 \triangle \{i\} \triangle \{j\} \in \mathcal{F}(Q)$ and $J_2 \triangle \{i\} \triangle \{j\} \in \mathcal{F}(Q)$. Now, for some $\tau_i = \pm 1$ and $\tau_j = \pm 1$,

$$w(J_1 \triangle \{i\} \triangle \{j\}) = w(J_1) + \tau_i r(i) + \tau_j r(j).$$

It is easy to check also that

$$w(J_2 \triangle \{i\} \triangle \{j\}) = w(J_2) - \tau_i r(i) - \tau_j r(j).$$

Hence,

$$w(J_1 \triangle \{i\} \triangle \{j\}) + w(J_2 \triangle \{i\} \triangle \{j\}) = w(J_1) + w(J_2).$$

Thus (VA2) holds (with equality in Eq. (12)).

To show that Algorithm 1 finds an optimal set of inputs we observe that it is essentially the same as the optimisation procedure described by Dress and Wenzel [5, (OP)], which is shown to find a set $J$ maximising $w(J)$ provided (VA1) and (VA2) hold for $M$ and $w$. And $w$ has been defined so that if $J \subseteq M$ maximises $w(J)$, then $x_J$ is a consistent set of inputs of maximum rank, as required.

We end this section by showing that Algorithm 1 is polynomial. We can calculate $r(b_i)$ for all $i$ ($1 \leq i \leq m$) in $O(m)$-time before we implement the algorithm. It is easy to check that, for $J \subseteq M$, we can calculate $r(x_J)$ in $O(m)$-time and check that $x_J$ is consistent in $O(|J|^3)$ time by determining if some matrix $Q_1$ is nonsingular as in Section 3. Thus, for $J \subseteq M$ and $i, j \in M$, we can find $w(x_{J \triangle \{i\} \triangle \{j\}})$ in $O(m^3)$-time. Clearly, the while-loop in Algorithm 1 takes $m$ iterations for completion, and in each iteration we must calculate $w(x_{J \triangle \{i\} \triangle \{j\}})$ for $m$ values of $j$. Thus, the algorithm is polynomial and can be implemented in $O(m^5)$-time.

6. Conclusion

We have seen how to generate an optimal set of state equations for a lumped power-conserving system given a consistent set of inputs and a weight function $w$ satisfying
Eq. (12). This paper has not considered here how to find a consistent set of inputs. That problem is explored for electrical networks in [18], and for bond graphs in [10, 11].

Section 4 showed how if we make certain assumptions about the system, we can define a rank function, which gives a weight function \( w \) defined in Section 5. This leaves open the question of how to deal with systems in which we cannot make these assumptions. Can we deal with such systems in the same way? Or can we modify the algorithm for them? One of the assumptions we made was that all the external elements were one-ports. Is it possible to generalise the results easily to multiports? Can some multiports be dealt with by ‘decomposing’ them into two-port transformers, two-port gyrators and multiport elements that can be dealt with in a similar manner to the one-port elements dealt with here?

Another question is how to implement the algorithm in practice. Given that it is \( O(m^2) \), it is most practical to implement it as a computer program, combined with some algorithm to generate an initial consistent set of inputs. It may still be useful to try to make the algorithm more efficient. In many cases there may be many ‘good’ sets of state equations and it may be sufficient to find just one of them. Can the algorithm be replaced by a more efficient heuristic that finds such a ‘good’ set of state equations?

References

