A graph theoretic approach to determining the transfer functions of mechanical networks, towards efficient computation of their H_2 norms

Gareth H. Willetts and Timothy H. Hughes[1]

***Abstract*— An alternative method for the determination of a transfer function of a single-input single-output mechanical system is given, illustrated with a model of a train suspension system taken from [5], and its application in H_2 norm analysis following [7]. This example is demonstrated with a MATLAB workbook, which follows the example step-by-step, calculating the H_2 norm from this transfer function numerically and symbolically. Building on this, this method is then embedded within an interior point optimisation scheme within MATLAB, demonstrating how a workflow using these methods may look, and demonstrating a speed up factor of around 20 times. Computation time is given and compared to existing methods built in to MATLAB. The associated code to reproduce all results in this paper can be found on Github [6].**

I. Introduction

The transfer function is a widely used characteristic of a (linear) system. Transfer functions allow for frequency-domain perspectives, useful for analysing frequency responses, determining stability from the poles in the complex plane and designing control systems. They also determine the most commonly considered performance characteristics of the system, such as the H_2 and H_∞ norms.

It can often be useful to obtain the transfer function of a system given a graphical representation of the components in the system and their constituent behaviours, e.g., a schematic of a mechanical network or electric circuit. In particular, this then allows for calculation of performance characteristics using transfer function methods, such as the efficient method for computing the H_2 norm from [7] which we will further explore in this paper.

Given the range of methods available for analysis of first-order state-space realisations, it is common for analysis of transfer functions to be neglected in favour of state-space methods. In the context of H_2 norm computation, such methods amount to the solution of Lyapunov equations, yet [7] emphasised the inefficiency of this approach and introduced a more efficient transfer function method for computation of the H_2 norm. Furthermore, the construction of a state-space realisation for a given mechanical network or electric circuit from a schematic can be quite cumbersome [2]. Accordingly, in this paper, we introduce a method based on Kirchhoff's tree theorem to directly compute the transfer function of a mechanical network from its schematic, bypassing the need for a state-space realisation. The results are equally applicable to electric circuit analysis using the force-current analogy [4].

This method builds on the work of Percival [3], expanding the work on Kirchhoff's tree formula for electrical networks, whose edges have differential equation relationships/admittances, to mechanical networks. In Section II we provide a summary of some of the notation and results in [3] in the context of mechanical networks. Section III outlines the method to determine a transfer function of the mechanical network, once the initial graph G has been constructed following the definition in Section II.

The method is illustrated with a model of a train suspension system in Section IV taken from [5]. This example was used in [7] to determine the H_2 norm of a mechanical network from its transfer function. For comparison, a free body diagram approach is introduced in Section V. The benefits of the method introduced in this paper are outlined in Sections VI and VII, notably the simplification of determining the transfer function of a mechanical system bypassing the often cumbersome construction of a state-space realisation entirely. We combine this method with the method from [7] for symbolic computation of the H_2 norm from the coefficients of a transfer function, and we embed this within an interior point optimisation algorithm to provide an efficient method for the optimal design of a mechanical network. This is compared with an alternative optimisation approach using existing functionality in MATLAB, and is demonstrated to be around 20 times faster.

II. Notation

As mentioned, this paper extends the work of Percival [3]. We begin by providing a summary of the notation from the aforementioned paper, for mechanical networks:

1) G denotes an undirected graph, constructed using the components of the mechanical system. One node on the graph represents ground, and every other node represents a point of connection. The edges connecting the nodes in the graph represent components, with weights denoting the mechanical admittance of the components (i.e., the equal and opposite force applied at the two end points of the element and directed along the axis between these two end points divided by the component of the relative velocity of these two end points in the same direction, where positive values imply that the force is directed outwards and the two end points are moving apart, and where the force and velocity are in the Laplace domain and assumed to have zero initial conditions). Every mass component is

[1]Gareth H. Willetts and Timothy H. Hughes are with the Department of Earth and Environmental Sciences, University of Exeter, Penryn, Cornwall, ghw205@exeter.ac.uk, t.h.hughes@exeter.ac.uk.

represented by an edge between its point of connection with other components and the ground node[1];

2) $v_{ab,cd}$ denotes the component of the relative velocity between nodes c and d along the axis between these two nodes when either (i) the component of the relative velocity between nodes a and b along the axis between those two nodes is equal to v; or (ii) an equal and opposite force F is applied at nodes a and b along the axis between those two nodes (for any given nodes a, b, c and d on the graph G). Here, a positive value of v (resp. $v_{ab,cd}$) implies that nodes c and d (resp. a and b) are moving apart, while a positive value of F implies a force that is directed outwards along the axis between nodes a and b. The force F and velocities v and $v_{ab,cd}$ are in the Laplace domain and are assumed to have zero initial conditions;
3) T and $T_{a,b}$ are tree admittance products (to be defined in the next section) corresponding to the weighted graph G and the weighted graph $G_{a,b}$ obtained by connecting node a to node b on the graph G (for any distinct pair of nodes a, b on the graph G);
4) By definition, $T_{a,a} = 0$ for any given node a.

III. Method

Following [3], the method for obtaining transfer functions of passive electrical networks using mathematical trees has been adapted to obtain a transfer function of a mechanical network using a graphical approach as follows. First, an undirected graph G must be constructed following the approach in Section II of this paper. For any given graph, with associated edge weights (admittances) and for any given pair of distinct nodes a, b on the graph G, the tree admittance product T or $T_{a,b}$ (see point 3 in Section II) is obtained by:

1) finding every spanning tree on the relevant graph (G or $G_{a,b}$),
2) multiplying the admittances of all of the edges on each of these spanning trees,
3) summing all of these admittance products together.

The transfer function is then obtained from the following equations:

$$\frac{v_{ab,cd}}{v} = \frac{T_{a,d} + T_{b,c} - T_{a,c} - T_{b,d}}{2T_{a,b}}, \tag{1}$$

$$\frac{v_{ab,cd}}{F} = \frac{T_{a,d} + T_{b,c} - T_{a,c} - T_{b,d}}{2T}. \tag{2}$$

This follows from Eqns. (38) and (40) in [3], referring to the notation on the first page of the paper. Note that in [3], $T_{a,b}$ is instead characterised as 2-trees, which are two disjointed trees on the graph that cover all nodes, one of which contains node a and the other of which contains node b. Our equivalent characterisation of these using the connection of nodes on the graph of the network allows for their computation using standard algorithms for obtaining the spanning trees of a given graph. Further discussion of this equivalent characterisation and the proof of this graph theoretic method for obtaining the transfer function of a network will follow in an upcoming journal paper. This paper is only intended to outline the methodology.

To obtain the transfer function of the mechanical system, the following steps are executed in order.

- First, construct the graph of your mechanical network, following the definition of G in Section II,
- Next, combine edges and sum their corresponding weights when there is more than one edge connecting any pair of nodes, in order to simplify the graph and remove repeated edges,
- Combine pairs of nodes to form new graphs corresponding to those whose tree admittance products appear in Eqns. (1) or (2),
- In each of the graphs formed in the previous stage, sum the admittances of any repeated edges and remove any edges that connect a node to itself,
- Find all spanning trees on each of these graphs and sum their admittance products, as described at the start of Section III,
- Substitute these values into the Eqns. (1) or (2) to compute the transfer function.

After the first step, where the graph G is constructed, the rest of the steps can be automated using code; specifically, using programming languages which allow for the calculation of spanning trees of a simple, undirected graph with weighted edges. In the context of electric circuits, further work could look at automating the construction of the weighted graph G from a SPICE netlist generated from a circuit schematic drawn in a graphical user interface, e.g., using the programme LTspice. This approach will also be relevant to mechanical networks using the force-current analogy (where springs, dampers and inerters are represented by inductors, resistors and capacitors, respectively, and masses are represented by capacitors connected to ground). The authors are unaware of any user interfaces that can directly be used to create weighted graph descriptions from a mechanical network schematic.

The next section will feature an example using MATLAB and its Symbolic Toolbox, but equivalently can be carried out in Python using its NetworkX package and SymPy. This will be discussed later in Section VII.

The final two steps are potentially inefficient due to the number of computations involved, as each admittance product in the numerators of Eqns. (1) and (2) will be computed twice and some of these will cancel when those numerators are computed. We are in the process of developing an algorithm that determines which admittance products appear in those numerators prior to computation, which may prove to be more efficient.

IV. Example

To put this into context, consider the model of a train suspension system in [5]. Using this example, we have the following mechanical network:

[1]Note that admittance is sometimes defined as the inverse of this quantity. The definition that we choose is analogous to electrical admittance under the force current analogy, which allows us to directly apply Kirchhoff's tree theorem to the analysis of mechanical networks.

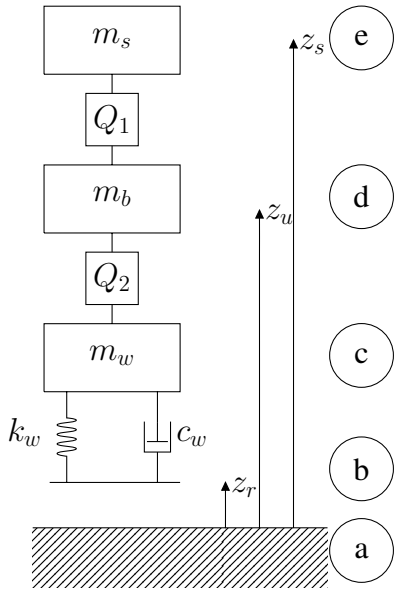

Fig. 1. Model of a train suspension system, with associated graph node labels and displacements. Here, m_s denotes the sprung mass, m_b the mass of the bogie, m_w the mass of the wheel, k_w the wheel stiffness, c_w the wheel damping coefficient, and Q_1 and Q_2 are suspension admittances to be designed (see [7]).

Using the nodes on Fig. 1 and the connections between the mechanical components, we obtain the following graph G.

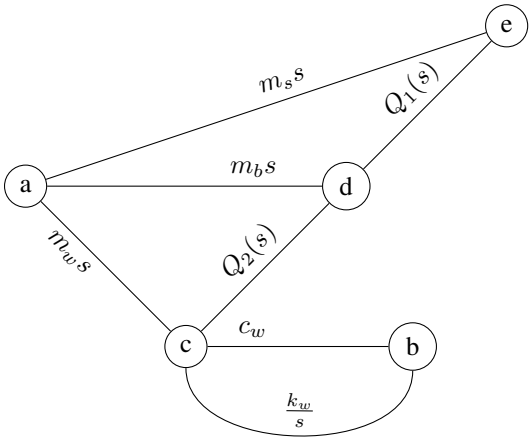

Fig. 2. Graph G, constructed from the mechanical network in Fig. 1 following the definition in Section II. Note that this is the original graph, before any repeated edges are combined (e.g., the edges between nodes b and c). This is shown in Fig. 1 of the MATLAB workbook [6].

As there are two edges between nodes b and c, representing the spring and damper connected in parallel, the edges are combined and the weights summed to produce the following simplified undirected graph:

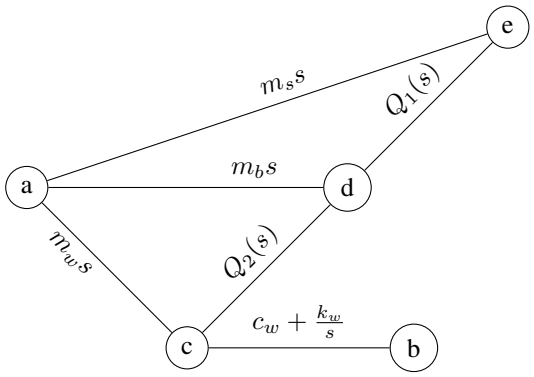

Fig. 3. Graph G after the edges between nodes b and c are combined by summing the weights of the edges. This is shown in Fig. 2 of the MATLAB workbook [6].

The desired transfer function is from the displacement of the train wheel (z_r in Fig. 1) to the velocity of the sprung mass (the time derivative of z_s in Fig. 1), i.e., $v_{ab,ae}/(v/s)$. The H_2 norm of this transfer function is a measure of passenger comfort. To obtain this, referring back to Eqn. (1), we require

$$\frac{v_{ab,ae}}{v/s} = \frac{s(T_{a,e} + T_{b,a} - T_{b,e})}{2T_{a,b}}, \tag{3}$$

as we know that $T_{a,a} = 0$.

Thus, we begin by combining nodes a and e to calculate $T_{a,e}$ as follows. Each spanning tree will be drawn for this first graph, to illustrate the method and to visualise what the code in the associated MATLAB workbook [6] is doing. This content is provided for tutorial purposes and includes some duplication of computation of admittance products which can be avoided as indicated at the end of the last section.

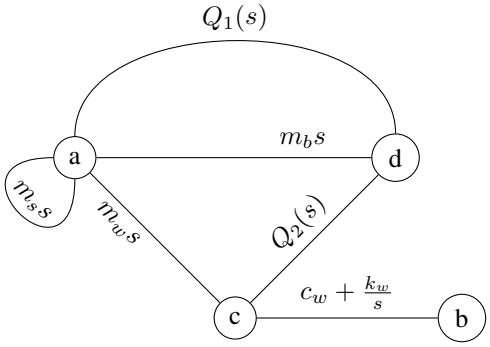

Fig. 4. The graph obtained by combining nodes a and e, in order to calculate $T_{a,e}$.

As can be seen above, in Fig. 4, combining nodes a and e creates a loop at node a, and there are two edges between a and d. The loop at a is removed per the method outlined in Section III, and edges between a and d combined by summing the weights and removing the repeated edge:

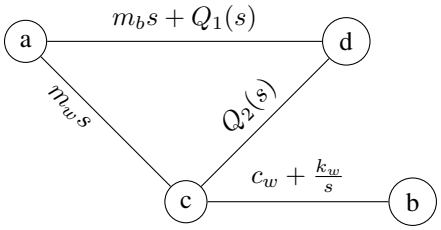

Fig. 5. The simplified graph, obtained by combining nodes a and e and combining repeated edges, as well as removing the loop at node a.

Next, we find all spanning trees of this graph and sum their admittance products. For this graph, there are 3 different spanning trees.

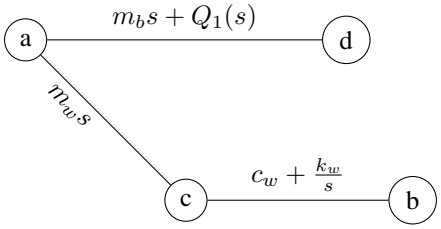

Fig. 6. The first spanning tree of the graph shown in Fig. 5, with admittance product $m_w s(m_b s + Q_1(s))(c_w + \frac{k_w}{s})$.

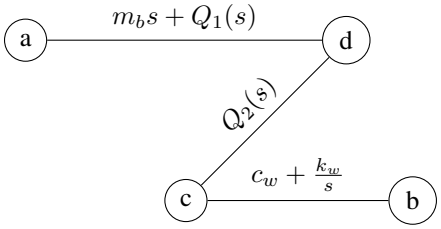

Fig. 7. The second spanning tree of the graph shown in Fig. 5, with admittance product $Q_2(s)(m_b s + Q_1(s))(c_w + \frac{k_w}{s})$.

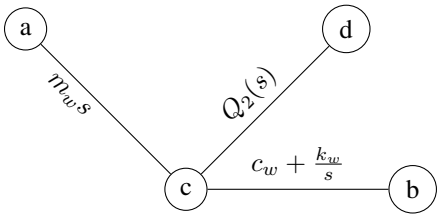

Fig. 8. The final spanning tree of the graph shown in Fig. 5, with admittance product $m_w s Q_2(s)(c_w + \frac{k_w}{s})$.

Summing the admittance products from Figs. 6, 7, 8 yields a tree admittance product of $((k_w + c_w s)(Q_1(s)Q_2(s) + m_b m_w s^2 + Q_2(s)m_b s + Q_1(s)m_w s + Q_2(s)m_w s))/s$. It's worth noting that the edge between nodes b and c appears in all three graphs, and thus is a factor in all three admittance products. There can be some efficiency gains in accounting for this, as there will be fewer computations if the tree admittance products of the graph omitting this edge is computed, and then multiplied by this edge weight. This is an example of one of the rules included in the Percival paper [3] to calculate tree admittance products. That paper considered how this could be done manually, and further work could consider how this could be automated. This is particularly relevant when computing the denominator terms $T_{a,b}$ or T in Eqns. (1) and (2). We expect that a different approach, described briefly at the end of Section III, will be more effective at improving the efficiency of the computation of the numerator terms in those equations.

We repeat this for node pairs b and a, and b and e (omitted for brevity), to calculate $T_{b,a}$ and $T_{b,e}$ respectively. Substituting these into the formula (3) returns the desired transfer function.

Taking this transfer function, we can follow the method in reference [7] to obtain the H_2 norm of this transfer function. This is shown in the MATLAB workbook [6], expanding on the method outlined in reference [7] by determining a symbolic expression for the H_2 norm and using an unconstrained minimisation technique to minimise the H_2 norm over two variables c_1 and c_2, where $Q_1(s) = \frac{k_s}{s} + c_1$ and $Q_2(s) = \frac{k_b}{s} + c_2$. This optimisation problem was considered in reference [5], and our results confirm theirs. Alternatively, the H_2 norm can be calculated numerically via the same method for specified values of c_1 and c_2.

V. Free Body Diagram Approach

For comparison, a free-body approach is applied in MATLAB through the following implementation. We isolate each of the masses in turn, drawing the corresponding free body diagram and writing out the relevant equations using Newton's Second Law. We consider the mechanical network to be originally at rest, and for a fixed but arbitrary non-zero input displacement z_r to be applied whose Laplace transform $z_r(s)$ exists. We then seek the relationship between this and the Laplace transforms $z_s(s), z_u(s)$ and $z_w(s)$ of the displacements z_s, z_u and z_w.

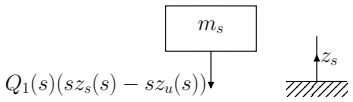

Fig. 9. Considering the mass of the train body, m_s.

First, considering the mass attributed to the train body m_s, we obtain the following equation:

$m_s s^2 z_s(s) = -Q_1(s)(sz_s(s) - sz_u(s))$.

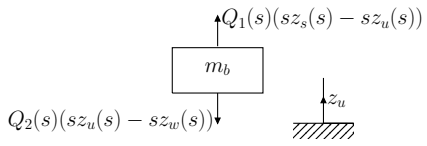

Fig. 10. Considering the mass of the train bogie, m_b.

Next, considering the mass attributed to the train bogie m_b, we obtain the following equation:

$m_b s^2 z_u(s) = Q_1(s)(sz_s(s) - sz_u(s)) - Q_2(s)(sz_u(s) - sz_w(s))$.

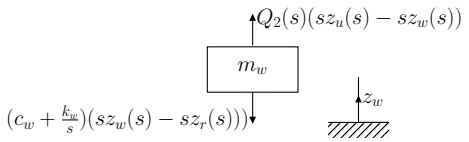

Fig. 11. Considering the mass of the train wheel, m_w.

Finally, considering the mass attributed to the train wheel m_w, we obtain the following equation:

$m_w s^2 z_w = Q_2(s)(sz_w(s) - sz_w(s)) - (c_w + \frac{k_w}{s})(sz_w(s) - sz_r(s)))$.

These three equations are arranged to form the matrix equation $Kx = L$, where

$$K = \begin{bmatrix} m_s s^2 + Q_1(s)s & -sQ_1(s) & 0 \\ -sQ_1(s) & s^2 m_b + s(Q_1(s) + Q_2(s)) & -sQ_2(s) \\ 0 & -sQ_2(s) & m_w s^2 + Q_2(s)s + c_w s + k_w \end{bmatrix}$$

$x = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^T = \begin{bmatrix} \frac{z_s(s)}{z_r(s)} & \frac{z_u(s)}{z_r(s)} & \frac{z_w(s)}{z_r(s)} \end{bmatrix}^T$, and $L = \begin{bmatrix} 0 \\ 0 \\ c_w s + k_w \end{bmatrix}$.

The desired transfer function is then given by

$$\frac{sz_s(s)}{z_r(s)} = \begin{bmatrix} s & 0 & 0 \end{bmatrix} x.$$

and thus substituting $K^{-1}L$ for x yields the transfer function. These polynomial matrix equations allow for the computation of the desired transfer function. In the MATLAB workbook, we proceed via calculating the inverse of the matrix K symbolically. Further work can investigate how the efficiency and accuracy can be improved through polynomial matrix algebra.

VI. Optimisation

The following table comprises the main results of this paper, summarising the timings for optimisation of the H_2 norm, starting from the schematic of the train suspension system shown in Fig. 1. This provides a comparison between the methods presented in this paper and in reference [7], and MATLAB's inbuilt methods.

In the approach titled "Graph theoretic approach and [7]", the relevant H_2 norm squared is computed symbolically using the graph theory approach to computing the transfer function described in Section IV and the approach for symbolic computation of the H_2 norm squared described in reference [7]. This is passed to an unconstrained minimisation scheme in MATLAB, to determine the optimal values for the decision variables c_1 and c_2 (the to-be-specified damping rates in the mechanical network). The function mapping the decision variables in the optimisation to the H_2 norm can also be used to compute the Gradient vector and Hessian matrices symbolically if required by the optimisation scheme, but this is not done here as it slows down the computation. Further work could explore this for different systems, to see if there are any potential efficiency gains.

In the approach titled "Free-body diagrams and `norm` function" the free-body diagram method outlined in Section V is combined with MATLAB's in-built `norm` function to construct a function that, given inputs corresponding to the decision variables c_1 and c_2, returns the desired H_2 norm. This function is then passed to the same unconstrained minimisation scheme.

	Timings (s)
Graph theoretic approach and [7]	0.1460
Free-body diagrams and `norm` function	2.8098

Fig. 12. Table showing computation times for determining the transfer function used in Section IV, and optimisation of its H_2 norm over c_1 and c_2.

The table in Fig. 12 shows a speed up factor of approximately 20 times using the methods presented here and in reference [7]. The approach considered in this paper is generalisable to H_2 norm optimisation of any single-input single-output mechanical network comprising an interconnection of linear springs, dampers, masses and inerters, subject to the usual limitations inherent in nonlinear optimisation. The extension to multi-input multi-output mechanical networks will be presented in a subsequent publication.

These MATLAB results are provisional, and investigations are ongoing into how the efficiency can be further improved.

VII. Discussion

The approach presented here involves a complete symbolic calculation of the square of the H_2 norm to obtain a function mapping the unknown parameter values (the decision variables in the optimisation) to the H_2 norm. In other words, the transfer function and H_2 norm computation is executed only once but symbolically in order to obtain a function to pass to an optimisation routine. Alternatively, the function passed to the optimisation routine can instead comprise the algorithm for computing the transfer function and its H_2 norm, which will then be evaluated numerically within the optimisation routine. Further work can consider the relative efficiency and accuracy of these two approaches when used alongside various optimisation routines.

In contrast, MATLAB's inbuilt functionality requires the symbolic expression of the transfer function to be evaluated numerically before being passed to the optimisation scheme and at each subsequent step in the optimisation, which slows down the method considerably.

Next, a comparison of the two methods for determining the transfer function symbolically (i.e., as an expression in terms of c_1 and c_2) will be shown.

	Timings (s)
Graph method for transfer function	0.0451
Free-body diagram for transfer function	0.0251

Fig. 13. Table showing computation times for determining the transfer function of the model train suspension system outlined in Section IV.

It's noted that the computation time for the graph method (described in Sections III and IV) is just under double the computation time of the free-body diagram method, as observed in Fig. 13, and thus a larger speed up factor could have been observed in Fig. 12 if we had used the free-body diagram approach alongside [7] to calculate the H_2 norm. The graph approach has the advantage of requiring less

manual intervention once the graph G has been constructed, and therefore is arguably easier to check and avoid errors in computation. The relative efficiency of both methods as the system grows in size is as yet unknown pending further investigation.

As discussed earlier, it is expected that the efficiency of the graph method can be improved considerably. In particular, each of the admittance products in the numerators of Eqns. (1) and (2) are being computed twice and further work can be done to adjust the method to avoid this double computation. This can be done through using logical operations on lists. Referring back to Eqn. (1), a list can be constructed of all of the trees that appear in both of the sets of trees featuring in the tree admittance products $T_{a,d}$ and $T_{b,c}$. Another list can be constructed of all of the trees that appear in both of the sets of trees featuring in the tree admittance products $T_{a,c}$ and $T_{b,d}$. The trees that appear in both lists can be removed, and the sum of the tree admittance products that remain in the first list less the sum of the tree admittance products in the second list will correspond to half of the value of the numerator in Eqn. (1). This avoids double computation, which is particularly computationally expensive when using symbolic admittances.

As also described earlier, it is expected that further efficiency gains can be obtained by automating the manual techniques for efficiently computing tree admittance products described in [3]. One technique, of relevance to the worked example in Section IV, involves decomposing the graph into biconnected components, whereupon the tree admittance product will be given by the tree admittance products for each of these biconnected components [1]. MATLAB and Python both offer functions to return the biconnected components of a graph.

A Python implementation is being developed using the NetworkX package which circumvents certain issues entirely, notably that MATLAB does not support symbolic weights for graphs and thus look-ups are performed between vectors currently. Python can support symbolic weights with SymPy, and these can be used alongside NumPy arrays, which are expected to increase performance.

VIII. Further Work

Journal papers are being prepared, consolidating the work of this paper and [7] with mathematical proofs of various stages, and performance metrics. These papers will also describe a new version of the algorithm described in [7] that is more tailored towards numerical computations as opposed to symbolic. These methods can also be applied in a multi-input multi-output setting, which is yet to be explored, but will feature in this future work.

Further specific examples could be explored and documented in order to showcase the advantages of this method over conventional Lyapunov approaches. A work package considering a general solution to the polynomial Diophantine equation in [7] could be completed, extending some of the techniques used in calculating the H_2 norm. This could form a toolbox of general polynomial tools, for use in common mathematical programming languages such as MATLAB and Python.

Finally, it is aimed for these techniques to be packaged up and shared as MATLAB and Python toolboxes, with associated documentation, for use in research and industry.

IX. Conclusion

A novel method for calculating a transfer function of a mechanical network has been shown, motivated by the need for efficient determination of the transfer function of a given system for H_2 norm analysis. This has been applied to a model of a train suspension system, taken from [5]. For this example, the graph-theoretic approach to determining the transfer function took 0.0451s, and it's noted that there is potential for this to be even faster by avoiding repetition in its computation. A full optimisation of the H_2 norm for the train suspension example takes 0.1460s, using the method outlined in this paper along with the method for calculating the H_2 norm symbolically from the coefficients from its transfer function [7]. In contrast, MATLAB's `norm` function, along with a free-body approach to determining the transfer function, takes 2.8098s, and is unable to compute a symbolic expression for the H_2 norm.

References

[1] J. Hopcroft and R. Tarjan. "Algorithm 447: Efficient algorithms for graph manipulation". In: *Communications of the ACM* 16.6 (June 1973), pp. 372–378. DOI: 10.1145/362248.362272.

[2] T.H. Hughes. "Passivity and Electric Circuits: A behavioral approach". In: *IFAC-PapersOnLine* 50.1 (2017), pp. 15500–15505. DOI: 10.1016/j.ifacol.2017.08.2117.

[3] W.S. Percival. "The solution of passive electrical networks by means of mathematical trees". In: *Journal of the Institution of Electrical Engineers* 1953.5 (1953), pp. 214–214. DOI: 10.1049/jiee-2.1953.0144.

[4] M.C. Smith. "Synthesis of mechanical networks: The inerter". In: *IEEE Transactions on Automatic Control* 47.10 (2002), pp. 1648–1662. DOI: 10.1109/tac.2002.803532.

[5] F.C. Wang et al. "The performance improvements of train suspension systems with mechanical networks employing inerters". In: *Vehicle System Dynamics* 47.7 (2009), pp. 805–830.

[6] G.H. Willetts. *Github Repository: GraphsUKACC2024*. Nov. 2023. URL: https://github.com/GarethWilletts/GraphsUKACC2024.

[7] G.H. Willetts and T.H. Hughes. "Efficient and symbolic computation of the H_2 norm via the polynomial Diophantine equation". In: *2022 UKACC 13th International Conference on Control (CONTROL)* (2022). DOI: 10.1109/control55989.2022.9781359.

X. Acknowledgement

This work was fully supported by the UK Engineering and Physical Sciences Research Council (EPSRC) grant EP/T518049/1 for the University of Exeter.