Embedding Desired Eigenstates into Active and Passive Dynamics of a Linear, Under-actuated Feedback System

Saunders, F., and Rife, J.
Tufts Univ., Medford, MA, USA

Submitted to ASME J. of Mechanical Design
This is a pre-print.
Final Version available at <TBA>
Embedding Desired Eigenstates into Active and Passive Dynamics of a Linear, Under-actuated Feedback System

Frank Saunders  
Mechanical Engineering Dept., Tufts University  
Medford, Ma, USA

Jason Rife  
Mechanical Engineering Dept., Tufts University  
Medford, Ma, USA

ABSTRACT

No method currently exists to select feedback and physical parameters to instantiate desired Eigenvalues for an under-actuated linear time-invariant system. This paper introduces a novel methodology that allows specification of a system’s Eigenvalues (Eigenvalues and their corresponding Eigenstates) by tuning physical and control parameters concurrently. The method requires the solution of a dual-domain Eigenstate factorization problem, in which it is necessary to compute certain unknown elements of a matrix and of its Eigenvectors at the same time.

Key words: Co-Design, Eigenstructure Assignment, Under-actuation

1 Introduction

Feedback control is often employed to drive mechanical systems to follow reference trajectories. Since the dynamics of open and closed-loop linear systems are determined by their Eigenstates (Eigenvalues and Eigenvectors) [1-3], leveraging Eigenstates to create desired motion patterns is an alternative to conventional reference-trajectory tracking. This alternative is compelling, particularly because Eigenstates can be tuned to harness resonance, thereby reducing energy requirements and actuator count especially for systems possessing a great number of degrees of freedom.

This document proposes an Eigenstate-centered approach for mapping desired motion patterns into two sets of tunable design parameters: feedback-control parameters and passive dynamic parameters. Tuning both physical and control parameters simultaneously is a key characteristic of a number of recent dynamic-system design methods that are sometimes referred to as co-design [4-7], embodiment [8], or template-based design [9, 10]. Our method contrasts with existing co-design and related methodologies in that we employ Eigenstates as an abstraction for representing desired trajectories.

Our use of Eigenstates in co-design offers not only operational benefits (harnessing resonance) but also design benefits (streamlining the design problem). One such design benefit is the capability to modify reference trajectories to be more compatible with natural system dynamics. Periodic desired trajectories can typically be approximated with a small number of Eigenstates; mapping trajectories to Eigenstates eliminates errors introduced when reference trajectories are not explicitly designed to be realizable for a physical system [11]. A second design benefit is that our approach eliminates redundancies between tunable feedback and physical parameters. For mechanical systems, desired Eigenstates can generally be obtained from many combinations of physical and feedback parameters; our approach eliminates these redundancies to simplify the design problem (resulting in fewer parameters to tune) without otherwise restricting the design space.

Importantly, our proposed method applies to under-actuated systems, including under-actuated flexible and continuum systems. By under-actuated, we mean systems in which not every degree of freedom is controlled by a separate actuator [12-14]. Existing methods for Eigenstructure assignment assume either full actuation (i.e., all degrees of freedom actuated) [15,16] or purely passive design (i.e., no actuation) [17]. Our method aims to bridge the gap between these two classes of existing design methods, at least in designing the simplest class of under-actuated system: under-actuated linear time-invariant (LTI) systems.

To solve the problem of embedding desired motion trajectories into the dynamics of under-actuated LTI systems, we introduce a generalization of conventional Eigenstate factorization [18]. We refer to this generalization as the Balanced- Domain Eigenstate Factorization (DDEF) problem. By solving the DDEF equations, it is possible to relate desired periodic motion trajectories directly to system parameter values. Our solution approach contrasts with iterative search or optimization methods [19], which must integrate dynamic simulations repeatedly until an acceptable system configuration is obtained.

The remainder of the document consists of three sections. These sections (1) characterize the DDEF problem, (2) introduce a solution algorithm to solve the DDEF problem to embed a desired motion pattern in an under-actuated LTI system, and (3) provide a case study to verify this approach. A brief summary concludes the paper.
2 Dual-Domain Eigenstate Factorization

2.1 Generalization of Conventional Eigenstate Factorization

The section defines the DDEF problem, a non-standard Eigenstate factorization problem that arises in framing the novel co-design method introduced in this paper.

The standard Eigenstate factorization problem involves the decomposition of a matrix \( A \) into a diagonal matrix \( \Lambda \) (with the Eigenvalues of \( A \) on its diagonal) and the transform matrix \( X \) (whose columns are the Eigenvectors of \( A \)) according to the following formula:

\[
A = X \Lambda X^{-1}.
\]  

The standard Eigenstate factorization starts with the left side of (1) and computes the right. Many numerical algorithms have been developed to compute this decomposition efficiently [20].

A complementary problem starts with the right side of (1) and computes the left. This complementary problem is trivial, since \( A \) can be obtained by simple matrix multiplication.

DDEF is a hybrid problem that unifies Eigenstate factorization with the complementary multiplication problem (see Figure 1). In DDEF, a subset of elements on both the left and right sides of (1) are unknown. To solve the DDEF problem it is necessary to compute these unknown elements in \( A, X \) and \( \Lambda \). Both the conventional Eigenstate factorization problem and the complementary multiplication problem may be viewed as special cases of DDEF. The conventional Eigenstate factorization problem is the special case in which all elements of \( A \) are known, and of \( X \) and \( \Lambda \) are unknown. The complementary multiplication problem is the special case in which all elements of \( X \) and \( \Lambda \) are known, and of \( A \) are unknown.

A necessary condition for a unique solution to the DDEF problem is that the number of independent scalar must equal the number of unknowns. A subtlety of this condition is that the two special cases mentioned above each seemingly involve a different number of unknown elements. For \( A \in \mathbb{R}^{N \times N} \), the Eigenstate factorization problem solves for \( N^2 + N \) values (namely, the elements of \( X \) and the diagonal elements of \( \Lambda \)). The complementary multiplication problem solves for only \( N^2 \) elements (elements of \( A \)). The seeming discrepancy is resolved by noting that the magnitudes of the Eigenvectors in \( X \) are arbitrary. To resolve this ambiguity, Eigenvectors are often reported with their norm scaled to a particular value (e.g. to unity). Taking this into account, the number of unknowns in both directions can be matched by introducing a magnitude value \( \gamma \) for each eigenvector \( x \), such that

\[
\|x\| = \gamma_j \quad \forall j.
\]  

The complete quantitative statement of the DDEF problem combines together equations (1) through (3), as follows.

\[
\Lambda = XAX^{-1}
\]

\[
\|x\| = \gamma_j \quad \forall j
\]

\[
g(A, \gamma, \Lambda, X) = 0
\]  

Solving this set of equations provides values for all the elements in the solution matrices \( A, \gamma, \Lambda, \) and \( X \). In all, this set consists of \( 2(N^2 + N) \) equations, including \( N^2 \) Eigenstate parameters.
equations from (1), $N$ magnitude equations from (2), and $N^2 + N$ constraint equations from (3).

### 2.2 Application of DDEF to LTI System Design

The DDEF problem arises, for example, in designing an LTI system to produce desired motion patterns by specifying a subset of modal parameters (i.e. elements of $A$ and $X$) given constraints on time-domain parameters (i.e. elements of $A$).

In the case of an LTI system, the $A$ matrix is a function of both physical and feedback parameters. Consider the following state-space expression for an LTI system.

$$
\dot{z} = \hat{A}z + \hat{B}u
$$

(5)

Here the vector of system states is $z$, the vector of actuator inputs is $u$, and the mappings of these vectors into the state derivatives are labeled $\hat{A}$ and $\hat{B}$. It is further assumed that linear, full-state feedback is applied, with

$$
u = -Kz,
$$

(6)

where $K$ is a constant gain matrix. The Eigenstates of the closed-loop system are the Eigenstates of the matrix $A$:

$$A = \hat{A} - BK.
$$

(7)

In modeling under-actuated mechanical systems, some elements of the $A$ matrix are typically fixed (e.g. with values of one or zero). Such fixed values can be modeled as constraints on the $A$ matrix. These constraints are compiled in the vector $g_A(A)$, which is a special case of (3).

Additional constraints are needed to specify desired Eigenstates. In this paper, desired Eigenvalues are labeled $\tilde{\lambda}_{ii}$ and desired Eigenvector elements are labeled $\tilde{x}_{ij}$. The subscripts indicate the row and column positions of each element in $A$ and $X$, respectively. Note that only a subset of Eigenstates can be specified when time-domain parameters (in $A$) are constrained. As such, a set $\Omega_\lambda$ is defined to contain indices for all specified Eigenvalues. An associated Eigenvalue constraint vector $g_\lambda(A)$ is

$$g_\lambda(A) = \{\lambda_{ij} - \tilde{\lambda}_{ii} | (i,j) \in \Omega_\lambda\} = 0.
$$

(8)

Similarly, a set of Eigenvector constraints, for all index pairs in a set $\Omega_x$, can be compiled as a column vector $g_x$.

$$g_x(X) = \{x_{ij} - \tilde{x}_{ij} | (i,j) \in \Omega_x\} = 0
$$

(9)

The magnitude $\gamma_j$ is variable for any Eigenvector $j$ with one or more elements unspecified by (9). To resolve this ambiguity, the magnitude should be defined for these vectors. Magnitude constraints are compiled in a vector $g_\gamma$. Combining this constraint vector with the time-domain and modal constraints into a concatenated constraint vector $g$ gives:

$$g(A, \gamma, A, X) = \begin{bmatrix}
g_\lambda(A) \\
g_\gamma(\gamma) \\
g_x(X) \\
g_A(A)
\end{bmatrix}.
$$

(10)

The problem of specifying a subset of modal properties for a physical system of form (5) is thus a DDEF problem as described by (4), where the constraint vector $g$ is given by (10).

### 3 DDEF Solution Algorithm

This section describes an algorithm to solve the DDEF problem that arises when specifying Eigenstates for an LTI system. In other words, the algorithm solves (4) given a constraint vector of the form (10).

The guiding principle to the proposed solution algorithm is to match the number of independent equations to the number of unknowns. Matching equations to unknowns is nontrivial given the nonlinear structure of the DDEF problem described by (4). To ensure a solution, the algorithm assumes that the designer provides a full set of Eigenstate specifications and, furthermore, rank orders the Eigenstates by their importance. In this application, rank ordering is trivial if the Eigenvalues are specified as harmonics of a base frequency [21], as lower harmonics typically provide a greater contribution to the desired motion pattern, and therefore merit a ranking of more importance. Using ranking information provided by the designer, the algorithm matches the size of the constraint vector (10) to the number of unknowns and solves.

#### 3.1 Initialization

The algorithm’s first step involves the construction of a constraint vector $g(A, \gamma, A, X)$. According to (10), this constraint vector concatenates independent constraint vectors $g_\lambda(A)$, for the time-domain parameters; $g_\gamma(\gamma)$, for the Eigenvector magnitudes; and $g_A(A)$ and $g_x(X)$, for the Eigenvalues and Eigenvector parameters, respectively.

Time-domain constraints are considered first, as these are highly dependent on the structure of the LTI system model (5). Most typically, constraints on the time-domain parameter matrix $A$ are of the form $A_{ij} = \overline{c}_{ij}$, where $\overline{c}_{ij}$ is a constant. The total number of time-domain parameter constraints is labeled $L$.

Eigenvalue constraints are considered second. The algorithm sets all $N$ eigenvalues. This step implicitly assumes that the system model is controllable in the sense of co-design, where controlable means that all Eigenvalues can be set by tuning either feedback or physical parameters. System stability is guaranteed (so long as the user-specified eigenvalues $\tilde{\lambda}_{ii}$ are all placed in the left-half plane).
Eigenvector magnitudes are considered third. The algorithm introduces constraints to set all $N$ magnitude values $\gamma_j$. By default, magnitudes are set to unity.

Eigenvector element values are considered last. Only a subset of the $N^2$ Eigenvector elements are specified. The reason is that the total number of independent constraints in the $g$ vector is limited to $N^2 + N$, in order to match the number of independent equations in (4) to the number of unknowns. Considering the number of constraints introduced above, the number of remaining independent Eigenvector constraints desired is $N^2 - N - L$. A separate constraint is required to set the imaginary and the real component of each Eigenvector element (for each complex-conjugate pair). It should be noted that the last imaginary or real component of each Eigenvector specification provides no new information (since this last component can be inferred from the Eigenvector magnitude $\gamma_j$, constrained above). In other words, the total number of independent constraints introduced by specifying a full complex-conjugate Eigenvector pair is $2N - 1$ constraints. Any constraint on the last real or imaginary component of the Eigenvector is redundant and does not change the total number of independent constraints. It is convenient to introduce these redundant equations for the purposes of simplifying bookkeeping, however. Assuming that Eigenvectors are assigned in rank order (with Eigenvector elements assigned in row order, imaginary values before real), then the algorithm introduces an extra number $B$ of redundant Eigenvector constraints.

$$B = \text{floor}\left(\frac{N^2 - N - L}{2N - 1}\right)$$

Taking this redundancy into account, the total number of constraints on Eigenvector elements is $N^2 - N - L + B$.

### 3.2 Verify Necessary Condition for Solution

Conceptually, the initialization step introduces a constraint vector $g$ which consists of the correct number of constraints to obtain a unique solution to the DDEF problem (4). Because of the nonlinearity of (4), however, it is challenging to prove that the constraint set is fully independent at every point in the solution space. As a second step, the algorithm thus checks the linearized equations, to ensure that a local solution is at least possible. To do so, the algorithm evaluates the linearized equations’ condition number, which is a numerically robust metric for assessing whether or not a system is solvable [22].

To obtain the linearization, a solution vector $y \in \mathbb{R}^{N^2 + N}$ is formed from the elements of the solution matrices $A \in \mathbb{R}^{N \times N}$, $\gamma \in \mathbb{R}^N$, $\text{diag}(A) \in \mathbb{R}^N$, and $X \in \mathbb{R}^{N \times N}$. Care must be taken when solving for the complex elements of $A$ and $X$. The real and imaginary components of each complex number must be computed separately; because each oscillatory mode consists of pairs of complex-conjugate Eigenvalues and Eigenvectors, the value of the second member of the pair can be computed from the first. Hence only the first of each complex-conjugate pair (the odd Eigenvalues and Eigenvectors) is included in $y$. The full Eigenvector $y$ is constructed as follows.

$$y = [a_1^T a_2^T \ldots y_1^T \text{re} \lambda_1^T \text{im} \lambda_1^T \ldots \text{re} x_1^T \text{im} x_1^T \text{re} x_2^T \text{im} x_2^T \ldots]^T$$

The vector $y$ concatenates each column of $A$ (the vectors $a_i$) with the Eigenvector magnitudes $\gamma$, the real and imaginary parts of each odd Eigenvalue ($\text{re} \lambda_1^T$ and $\text{im} \lambda_1^T$), and the real and imaginary parts of each odd Eigenvector ($\text{re} x_1^T$ and $\text{im} x_1^T$). Similarly, the vector equations of (4) are concatenated into a single function of $y$:

$$h(y) = 0,$$

where

$$h(y) = \begin{bmatrix} A x_1 - \lambda_{11} x_1 \\ \vdots \\ A x_N - \lambda_{NN} x_N \\ \|x_1\| - y_1 \\ \vdots \\ \|x_N\| - y_N \\ g^T(A, X, A) \end{bmatrix}.$$  

The Jacobian matrix $H$ is computed by linearizing $h(y)$.

$$H_{ij} = \frac{d^2 h(y)}{d y_i d y_j}$$

If the rank of $H$ is full, or more precisely if the condition number of $H$ is less than a cutoff threshold, then a local solution is possible. In this case the algorithm proceeds to solve the linearized system of equations (see step 3 below).

If $H$ is not full rank, or more precisely if the condition number of $H$ exceeds a threshold, then the system is augmented with an additional equation. Specifically, the next desired Eigenvector component value is added to $g(X)$. Subsequently, step 2 is repeated to ensure the solution is well conditioned. This loop is guaranteed to eventually converge, since in the extreme case, all of the Eigenvector components become specified, and the problem becomes, simply, the complementary multiplication problem identified in Figure 1.

In a practical implementation, it should be noted there is no reason to solve for solution variables that are directly set by a constraint in $g$. For numerical efficiency, it is advantageous to treat these values as known. Known elements can be removed from $y$, and the corresponding constraints removed from $g$. Consequently the lengths of $y$ and $h(y)$ are typically smaller than $2(N^2 + N)$ in practice. For clarity, the truncated versions of these vectors are labeled $\tilde{y}$ and $\tilde{h}(\tilde{y})$, and the truncated form of the Jacobian is labeled $\tilde{H}$.

### 3.3 Iterative Solution of DDEF

Once a full set of constraints has been defined, the third step of the algorithm is to solve the linearized DDEF equations.
Many numerical solutions can be employed, such as Richardson’s method or Newton’s method (20). In our implementation, we apply the Newton-Raphson method, which updates the following equations iteratively until convergence has occurred.

\[ \tilde{y} \rightarrow \tilde{y} - \mathbf{H}^{-1} \mathbf{h}(\tilde{y}) \] (16)

Because, at each iteration, the solution vector \( \tilde{y} \) is updated with a new value, it is prudent to check the local condition number (step 2) before computing the next Newton-Raphson iteration (step 3).

### 4 Case Study

In this section, a mass-spring-damper system is used to demonstrate the application of the proposed DDEF solution algorithm.

#### 4.1 System Design

The mass-spring-damper system consists of three masses connected sequentially by tunable linear springs and dampers. The model system is illustrated in Figure 2. It is assumed that the three masses slide on a frictionless surface. Two linear actuators apply forces to the first and third masses directly. So the overall system can be categorized as “under-actuated,” no direct actuation is applied to the second mass.

The two linear actuators are configured to provide feedback as described by (6). The parameters of the feedback law are labeled \( K_1 \) through \( K_{12} \) as follows.

\[
\mathbf{u} = - \begin{bmatrix} K_1 & K_2 & K_3 & K_4 & K_5 & K_6 \\ K_7 & K_8 & K_9 & K_{10} & K_{11} & K_{12} \end{bmatrix} \mathbf{z} \] (17)

The first actuator also applies a sinusoidal forcing input \( \mathbf{u}_{\text{ex}} \) to excite the dynamic modes of the full system, which is described by the state-space model (18).

The values \( k_i \) are spring constants, \( c_i \) are damper constants, and \( m_i \) are the masses, as identified in Figure 2.

\[
\mathbf{z} = \mathbf{A}\mathbf{z} + \mathbf{B}\mathbf{u}_{\text{ex}}
\]

\[
\mathbf{A} = \begin{bmatrix}
-c_1-c_2-K_1 & c_2-K_2 & -K_3 & -K_1-K_2-K_4 & k_2 & -K_6 \\
-m_1 & m_1 & m_1 & m_1 & k_2 & m_2 \\
-m_2 & m_2 & m_2 & m_2 & k_2 & m_2 \\
-K_7 & m_3 & K_2-K_8 & -K_3 & K_2-k_5 & -K_6 \\
-k_2-K_3 & m_3 & -K_10 & K_10 & K_3-K_{11} & -K_2-K_{12} \\
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0
\end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \] (18)

Figure 2: Mass-Spring-Damper Configuration; with tunable passive dynamical parameters (mass \( m_i \), linear damper \( c_i \), and linear spring \( k_i \)) and feedback-controlled forces acting on the center of mass of mass one and three (\( u_i \)). For this model it is assumed that contact between blocks does not occur.

A set of desired Eigenstates for the system was selected in order to test the proposed DDEF algorithm. These target Eigenvalues and Eigenvectors approximated a traveling wave pattern, which is illustrated in Figure 3. The traveling wave had a \( 3\pi \) sec period, with all displacement occurring within the first \( \pi \) sec and an extended pause occurring over the remainder of the period. This pattern was approximated using the Fourier-Series based decomposition method described in[21], which also specifies the magnitude and phase of the sinusoidal forcing \( u_{\text{ex}} \).

The desired trajectory illustrated in Figure 3 includes only the three position states (for each mass) and no velocity states. The three velocity states are algebraically coupled to the position states by kinematics, and hence the desired trajectories of the velocity states cannot be determined independently from the desired trajectories of the position states.
The mass-spring-damper system was simulated to verify the DDEF solution methodology. First, a solution to the DDEF problem was obtained for this system. Next, the DDEF solution was simulated and the time-history for each state compared to that for an ideal system, one based on the full set of desired Eigenstates. For this system there are 20 constraints on the $A$ matrix, leaving only 16 of 36 complex Eigenvector elements to be specified. As indicated by (18), the 20 constraints on $A$ include the 2nd and 5th elements of the second row (which are linear combinations of other parameters in the same row), and all elements of the last 3 rows of $A$.

Although the existence of constraints on the elements of $A$ implies that the DDEF problem statement can only specify a subset (of 16) Eigenvector components, the match between the time responses of the desired Eigensystem and the DDEF solution is good. The DDEF solution provides a close match to the desired response because of importance ranking, which assigned greater importance to lower frequency modes. The comparison is illustrated in Figure 4. In the figure, the response expected for the full set of desired Eigenstates is shown as a solid (blue) curve; the response for the DDEF solution is shown as a dashed (red) curve. The plot shows the displacement of each mass as a function of time over many periods, starting from zero initial conditions. Near the right side of the plot, the system has converged toward its periodic steady-state motion pattern. From Figure 4 it is evident that the frequency, phasing, amplitudes, and general shape of the DDEF solution are close to those of the response for the full set of desired Eigenstates. Note that no attempt is made to match the DC component of the trajectory specified in Figure 3; hence the values of states are slightly displaced from zero during the roughly six-second “rest” phase for each motion period.

An important detail in implementing the proposed algorithm is that the system parameters (masses $m$, spring constants $k$, damper constants $c$, or matrix of control gains $K$) are not directly determined by the DDEF solution. Rather, the solution computes the entries of the $A$ matrix, which are themselves functions of the physical and feedback parameters. The 16 configurable elements of $A$ are functions of 21 feedback and physical parameters, as described by (18). Thus, multiple combinations of feedback and physical parameters can be used to obtain the $A$ matrix generated by solving the DDEF problem. One valid solution, obtained by hand calculations, is presented in Table 1.

### Table 1: Physical and Control Parameter Values for DDEF Solution

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mass, $m$</td>
<td>0.25, 0.15, 0.45</td>
</tr>
<tr>
<td>Spring Constant, $k$</td>
<td>0.26, 0.15, 0.087</td>
</tr>
<tr>
<td>Damper Constant, $c$</td>
<td>0.67, 0.94, 0.52</td>
</tr>
</tbody>
</table>

### 4.2 Discussion

Insights drawn from this case study have broader implications for development of co-design methods in the future. A first observation is that it is possible and useful to leverage modal dynamics in some co-design problems. For the most part, existing co-design methods have solved directly for time-domain parameters [4-7]. Using a modal analysis makes it possible to leverage resonance phenomena, however. For instance, the redundancy of the feedback and physical parameters in Table 1 might be exploited to minimize actuator
energy requirements, without any modification to the motion patterns exhibited by the system.

A second observation is that co-design provides a useful tool set for expanding design options for under-actuated systems. The case study in the previous section considered an example in which only two actuators were used to control a system with three modes. By tuning a combination of physical parameters and feedback parameters, it was possible to shape 16 elements of the A matrix. This tuning capability nearly matches what would be possible were three actuators available (forcing all three masses), in which case 18 elements of the A matrix defined in (18) would have been tunable. (By contrast, in the absence of tunable physical parameters, it would only have been possible to modify only 12 elements of the A matrix with two actuators). In short, the co-design approach permits the under-actuated system to access nearly the entire design space accessible by a fully actuated system. A corollary implication is that co-design could enable the identification and removal of non-essential actuators, allowing for highly articulated mechanical systems with lower cost, weight and complexity [23].

More work remains in the future to extend the method discussed in this paper. Useful generalizations of the DDEF methodology include modeling inequality constraints and designing for systems which are not controllable (e.g. in which not all of the Eigenvalues can be specified). Preliminary concepts for such extensions to DDEF are discussed in [24].

5 Summary

A co-design method was presented for specifying modal dynamics of an under-actuated system by tuning its physical and feedback parameters. In order to match system parameters to modal specifications, a generalized Eigenstate Factorization problem, which we call Dual-Domain Eigenstate Factorization (DDEF), was formulated. An algorithm to solve the DDEF problem was introduced and verified using a simple case study. The case study demonstrated that the proposed method is effective in under-actuated system applications and suggests that the DDEF methodology might be extended for the purpose of eliminating actuators in complex mechanical systems, to reduce system cost, weight and complexity.

Acknowledgments

Many thanks to the institutions and agencies that have provided financial support for this work, including Tufts University and the National Science Foundation. This material is based upon work supported by the National Science Foundation under Grant No. 110045.

References