Eigendecomposition Parameterization of Penalty Matrices for Enhanced Control Design: Aerospace Applications

Nicholas P. Nurre^a, Ehsan Taheri^{a,*}

^aDepartment of Aerospace Engineering, Auburn University, 141 Engineering Dr, Auburn, AL 36849, USA

Abstract

Modern control algorithms require tuning of square weight/penalty matrices appearing in quadratic functions/costs to improve performance and/or stability output. Due to simplicity in gain-tuning and enforcing positive-definiteness, diagonal penalty matrices are used extensively in control methods such as linear quadratic regulator (LQR), model predictive control, and Lyapunov-based control. In this paper, we propose an eigendecomposition approach to parameterize penalty matrices, allowing positive-definiteness with non-zero off-diagonal entries to be implicitly satisfied, which not only offers notable computational and implementation advantages, but broadens the class of achievable controls. We solve three control problems: 1) a variation of Zermelo's navigation problem, 2) minimum-energy spacecraft attitude control using both LQR and Lyapunov-based methods, and 3) minimum-fuel and minimum-time Lyapunov-based low-thrust trajectory design. Particle swarm optimization is used to optimize the decision variables, which will parameterize the penalty matrices. The results demonstrate improvements of up to 65% in the performance objective in the example problems utilizing the proposed method.

Keywords: Gain tuning, Penalty matrix parameterization, Eigendecomposition, LQR, Attitude control, Low-thrust, Spacecraft, Trajectory, Lyapunov, Quadratic cost, MPC

1. Introduction

Modern control algorithms rely on tuning a set of gains to achieve a desired stability and/or performance output. For instance, positive-definite and/or positive

^{*}Corresponding author

Email address: ezt0028@auburn.edu (Ehsan Taheri)

semi-definite penalty matrices are used to construct quadratic costs or candidate quadratic control Lyapunov functions. Diagonal parameterization of the penalty (also referred to as the "weighting" or "parameter") matrices is used extensively in LQR controllers [1, 2], model-predictive control (MPC) [3, 4], sliding mode control [5], control Lyapunov functions (CLFs) [6], and control barrier function methods [7]. In this work, we consider quadratic CLFs and LQR controllers and investigate the impact of various parameterizations of the penalty matrices on control performance. Lyapunov control (LC) laws derived from CLFs can generate near-optimal control solutions and are used in many aerospace applications. In astrodynamics and space-craft trajectory optimization, some well-known low-thrust LC laws include Q-law [6] and the Chang-Chichka-Marsden (CCM) control law [8]. LC laws are also used for spacecraft attitude control [9, 10]. LQR is used extensively for solving spacecraft attitude control [9, 10].

Diagonal parameterization of the penalty matrices is quite common due to 1) implementation simplicity, 2) ease of enforcement of sign-definiteness by constraining the signs of the diagonal entries, and 3) making the gain-tuning intuitive since diagonal entries correspond to each state and control without any cross-coupling. However, diagonal matrices do not span the full solution space, as they neglect cross-coupled terms, leading to suboptimal gains (in the sense of some particular performance and/or stability criterion). Thus, it will be advantageous to consider penalty matrices with non-zero off-diagonal terms, which we refer to herein as "full" matrices. This raises a key question: How can a sign-definite full matrix be efficiently parameterized? A straightforward approach is eigendecomposition [11], as proposed in Ref. [9]. Sign-definiteness is trivially enforced through the signs of the eigenvalues. However, parameterizing the associated eigenvector rotation matrices remains less straightforward.

Multiple methods exist for parameterizing orthogonal matrices. Ref. [12] outlines four parameterizations that are generalizable to $N \times N$ dimensions and require the minimum number of design variables of M = N(N-1)/2. One of the first parameterizations discovered, the Cayley transform [13, 14, 15, 9, 12, 10], has been found useful in various practical engineering applications such as in attitude kinematic representations [15, 10] and efficient solution of the matrix Riccati differential equation [16, 14]. Ref. [17] proposes another parameterization of orthogonal matrices similar to the work of Ref. [18] requiring M decision variables and is based on successively projecting unit vectors (starting with an N-dimensional unit vector constructed with spherical coordinates [19]). Our goal is to parameterize square penalty matrices as efficiently as possible over the entire solution space, by decomposing the matrices through their 1) eigenvalues and 2) eigenvector matrices. Choosing how to parameterize the orthogonal eigenvector matrix can have a substantial impact on the penalty matrices and the search for more optimal control gains. While full penalty matrices can improve controller performance, a drawback is knowing how to choose the increased number of free variables (N for diagonal matrices versus $(N^2 + N)/2$ for full matrices) which also may not necessarily be physically meaningful.

Traditional gain-tuning techniques include the Ziegler-Nichols step response method [20, 21] for PID gains and Bryson's rule (stemming from Section 5.4 Example 2 of Ref. [22]) which serves as a starting point for manual trial-and-error gain tuning of LQR controllers. However, modern stochastic/evolutionary optimization techniques and computing technology have led to improve gain-tuning methods. For example, Ref. [23] uses a stochastic optimization algorithm that approximates the gradient of the performance metric to optimize LQR gains based on a performance metric evaluated from experiment. The same gains are improved further in Ref. [2] by using global Bayesian optimization which better utilizes the experimental data. Ref. [24] uses genetic algorithm and simulated annealing to find the best diagonal penalty matrix for Q-law to form a time-of-flight and propellant mass Pareto front for the same low-thrust transfer cases we solved in Ref. [17] and in this paper. Automated gain tuning will be necessary for optimizing the larger number of decision variables used to parameterize the full penalty matrices. We elect to use the metaheuristic optimization method particle swarm optimization (PSO) [25] to tune the gains in this work for its simplicity and because it is already available in the MATLAB Global Optimization Toolbox.

The main contribution of this work is to extend the work in Ref. [17] by considering additional examples and analyses for the purpose of investigating the improvement in performance of feedback control laws with full penalty matrices in a variety of control methodologies and aerospace applications. We parameterize the full penalty matrices with eigendecomposition, which allows for the straightforward enforcement of positive-definiteness. The Cayley transform [12], Givens rotation [12], and a parameterization based on generalized Euler angles and Gram-Schmidt process (GEAGSP) [17] are presented, but only the GEAGSP parametrization is used to obtain the results because of the existing code architecture. There may be certain benefits in using one orthogonal matrix parameterization over another due to factors such as computational efficiency [12]. The decision variables of the parameterization methods are optimized with the metaheuristic particle swarm optimization (PSO) algorithm [25]. This choice of optimization method was arbitrary as the optimization of the decision variables can be achieved by any meta-heuristic optimization method and the optimization algorithm is not the focus of this study. As such, parameters of the PSO algorithm such as swarm size were also arbitrarily chosen to achieve (what

we qualitatively perceive to be) optimal solutions. In the present work, we focus on investigating the advantages of using full penalty matrices for solving three control problems: 1) a variation of Zermelo's navigation problem, 2) minimum-energy spacecraft attitude control using both LQR and Lyapunov-based methods, and 3) minimum-fuel and minimum-time Lyapunov-based low-thrust trajectory design. To our knowledge, no work explores using a full penalty matrix in aerospace control applications for the sake of improving a certain performance objective such as propellant consumption, aside from our earlier papers [17, 26] (Ref. [27] considers full parameter matrices for the purpose of designing a flexible structure and its respective controller simultaneously based on eigenvalue placement). Consideration of the off-diagonal penalty terms, as it is shown herein, can enlarge the solution set and result in improved controller performance and reduced values of the cost function for each example problem. Our hope is that exemplifying and quantifying this improvement for a variety of aerospace control problems along with providing a way to efficiently parameterize full penalty matrices will potentially allow practitioners to explore improving their own control algorithms.

The paper is organized with Section 2 presenting a review of the parameterization methods of positive-definite matrices. Sections 3, 4, and 5 present the comparisons between the two types of penalty matrices for a variation of Zermelo's navigation problem, minimum-energy spacecraft attitude control using both LQR and Lyapunov-based methods, and minimum-fuel and minimum-time Lyapunov-based low-thrust trajectory design, respectively. Section 6 concludes the paper.

2. Penalty Matrix Parameterization

One approach to parameterize positive-definite penalty matrices is to consider a symmetric matrix and thus use $M = (N^2 + N)/2$ decision variables for parameterization. Let \mathbf{K} denote a generic penalty matrix and let $\mathbf{k} = (k_1, \ldots, k_N, \ldots, k_M)$, the matrix can be written as,

$$\boldsymbol{K} = \begin{bmatrix} k_1 & k_2 & k_3 & \cdots & k_N \\ k_2 & k_{N-1} & k_{N-2} & \cdots & k_{2N-1} \\ k_3 & k_{N-2} & k_{2N-1} & \cdots & k_{3N-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ k_N & k_{2N-1} & k_{3N-1} & \cdots & k_M \end{bmatrix},$$
(1)

where the diagonal parameters, \mathbf{k}_{diag} , and the off-diagonal parameters, $\mathbf{k}_{\text{off-diag}}$, are bounded as $\mathbf{k}_{\text{diag}} \in \mathbb{R}^+$ (leveraging the fact that positive-definite matrices always have positive diagonal elements) and $\mathbf{k}_{\text{off-diag}} \in \mathbb{R}$. Then at each iteration within an optimization process, we can check if \mathbf{K} is positive-definite in a number of ways. For example, we can calculate its eigenvalues, but this can become computationally demanding as the number of dimensions increases. While this simple parameterization of $\mathbf{K} \prec 0$ only requires the minimum amount of variables needed to parameterize a positive-definite matrix, M, the search process over \mathbf{k} can be made more efficient by only considering a search space that will always result in a positive-definite \mathbf{K} . This could be achieved by explicitly enforcing positive-definiteness through nonlinear constraints on the decision variables, \mathbf{k} . However, this method also has the limitations of optimization with nonlinear constraints, which won't scale appropriately to higher dimensions.

A preferred approach is to implicitly satisfy the sign-definiteness property of the penalty matrix by using eigendecomposition [9, 11]. Here, we can parameterize \boldsymbol{K} as,

$$\boldsymbol{K} = \boldsymbol{Q} \boldsymbol{\Lambda} \boldsymbol{Q}^{\top}, \qquad (2)$$

where $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_N)$ are the eigenvalues and $\mathbf{Q} \in O(N)$ denotes the orthogonal matrix of eigenvectors. The eigenvalues and eigenvectors can be thought of as representing the scale/size deformation and the rotational deformation, respectively, of \mathbf{K} [9]. Positive-definiteness is implicitly enforced by ensuring all eigenvalues are positive.

We seek to parameterize Q with the minimum number of parameters needed, $M = (N^2 - N)/2$, in order to span O(N), i.e., we desire $Q(\phi) \in O(N)$ where ϕ denotes the set of parameters with dim $(\phi) = M$. Because we use derivative-free optimization exclusively in this work, we do not need $\partial Q/\partial \phi$. However, gradient-based optimization may further improve results (see Ref. [12] for a detailed discussion on the computational cost of gradient calculations for each orthogonal-matrix parameterization presented).

We outline three (out of the many) possible parameterizations methods: Cayley transform (CT), Givens Rotation (GR), and one based on generalized Euler angles and Gram-Schmidt Process (GEAGSP). We note that only the GEAGSP parameterization was used in all the results in this paper. Through numerical simulations, we have found that the CT and GR parameterizations to be comparable in performance. However, there may be benefits of using one parameterization over another for reasons such as computational efficiency, properties of the inverse mapping if required, or nonlinearity of the search space. Thus, the best parameterization could be explored further in future work.

2.1. Cayley Transform

The Cayley transform can be used to parameterize the special orthogonal matrices, SO(N) [9, 12]. Letting \boldsymbol{X} denote an $N \times N$ skew-symmetric matrix, then the orthogonal matrix $\boldsymbol{Q}(\boldsymbol{X}(\boldsymbol{\phi}))$ is defined with the Cayley transform as,

$$\boldsymbol{Q} = (\boldsymbol{I} + \boldsymbol{X}) \left(\boldsymbol{I} - \boldsymbol{X} \right)^{-1}.$$
(3)

Please refer to Ref. [12] for more details such as the reverse transformation and its smoothness and continuity properties.

2.2. Givens Rotation

The Givens rotation parameterization from Ref. [12] can be thought of as a succession of plane transformations. Letting $G(i, j, \theta)$ denote each plane transformation defined as,

$$\mathbf{G}(i,j,\theta) = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & \cos\theta & \cdots & -\sin\theta & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & \sin\theta & \cdots & \cos\theta & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix},$$
(4)

where the $\sin \theta$ and $\cos \theta$ populate the elements intersected by the *i*-th column and *j*-th row. The diagonal elements are all 1 except when a trig function populates the diagonal. More formally, the nonzero elements are defined as,

$$G(i, j, \theta)_{ii} = G(i, j, \theta)_{jj} = \cos \theta,$$
(5)

$$G(i, j, \theta)_{ji} = -G(i, j, \theta)_{ij} = \sin \theta, \quad \text{for } i > j,$$
(6)

$$G(i, j, \theta)_{kk} = 1 \quad \text{for } k \notin \{i, j\}.$$

$$\tag{7}$$

Let $\boldsymbol{\theta} = [\theta_1, \cdots, \theta_M], \boldsymbol{Q}(\boldsymbol{G}(\boldsymbol{\theta}))$ is defined as,

$$\boldsymbol{Q} = \boldsymbol{G}_1^\top \dots \boldsymbol{G}_M^\top. \tag{8}$$

Remark 1. It is well known that these plane rotations are ambiguous at certain angles, since there are multiple sets of angles that can represent the same rotation matrix. In 3 dimensions, this is known as "gimbal lock." We note this introduces redundancies, however, we find this characteristic to be ignorable when using heuristic optimization to tune the parameters. This could also be avoided in the optimization algorithm by detecting when one angle reaches an ambiguous value and setting the rest of the angles to arbitrarily fixed values [12].

2.3. Generalized Euler Angles and Gram-Schmidt Process

Another approach to parameterize the Q matrix is to use a generalization of Euler angles derived in [19] and perform successive orthogonalization through the Gram-Schmidt process, as we proposed in Ref. [17]. This method is similar to the earlier work of [18]. Let $\theta_{ij} \forall i = 1, ..., N, j = 1, ..., N - 1$, denote the angle-like parameters of Q, this parameterization can be described by Algorithm 1 where

$$\boldsymbol{v}_{i}^{p} = \begin{bmatrix} \cos(\theta_{i1}) \\ \sin(\theta_{i1})\cos(\theta_{i2}) \\ \vdots \\ \prod_{j=1}^{N-i-1}\sin(\theta_{ij})\cos(\theta_{i(N-i)}) \\ \prod_{j=1}^{N-i}\sin(\theta_{ij}) \end{bmatrix},$$
(9)

where \boldsymbol{v}_i^p is a unit vector in a (N - i + 1)-dimensional space. The superscript p denotes that \boldsymbol{v}_i is not expressed in the original N-dimensional space and it should be projected back to the original dimension to construct the desired rotation matrix. All the constituent vectors in \boldsymbol{Q} should be mutually orthogonal, which equivalently means that the generated vectors should be in the null space of all previous vectors. In 2 dimensions, \boldsymbol{Q} can be written as,

$$\boldsymbol{Q} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix},\tag{10}$$

and in 3 dimensions, Q can be defined using any of the Euler sequence rotations [10]. MATLAB and Python codes are provided in a GitHub repository here¹.

3. Zermelo's Navigation Problem

We solve a variation of Zermelo's navigation problem with a Lyapunov-based control law. The low-dimensionality of the problem allows for visualization of the candidate quadratic Lyapunov functions and the impact of using a full penalty matrix. Letting the states be $\boldsymbol{x}^{\top} = [x, y]$ and the control be $\boldsymbol{u}^{\top} = [u_x, u_y]$, the goal is to drive the system from an initial nonzero state, $\boldsymbol{x}(0)$, to the origin, [0, 0] (i.e., the regularization problem). The dynamics are defined as,

$$\dot{\boldsymbol{x}} = \boldsymbol{u} + \boldsymbol{f}(\boldsymbol{x}), \qquad \qquad \boldsymbol{f}(\boldsymbol{x}) = -[x, y]^{\top} \cos y \sin x.$$
 (11)

¹https://github.com/saeidtafazzol/positive_definite_parameterization

Algorithm 1 N-dimensional Rotation Matrix Parameterization

1: Input: Angles θ_{ij} for i = 1, ..., N, j = 1, ..., N - i2: Output: an Orthonormal Matrix Q3: Initialize: $Q \leftarrow \{\}$ {The set of orthonormal vectors} 4: for i = 1 to N do 5: $S = \text{null-space}(Q^{\top})$ 6: B = Basis(S) {Gram–Schmidt process} 7: Construct v_i^p using $\theta_{i(1,...,N-i)}$ and Eq. (9) 8: $v_i = Bv_i^p$ {Project back to N-dimensional space} 9: $Q \leftarrow [Q, v_i]$ {Append v_i as a new column to Q} 10: end for

The control, \boldsymbol{u} , will be derived from a candidate quadratic Lyapunov function, which is written as,

$$V = \frac{1}{2} \boldsymbol{x}^{\top} \boldsymbol{K} \boldsymbol{x}, \qquad (12)$$

where $\mathbf{K} \in \mathbb{R}^{2 \times 2}$ is the positive-definite penalty matrix. The control law is derived such that the time derivative of Eq. (12),

$$\dot{V} = \frac{\partial V}{\partial \boldsymbol{x}} \dot{\boldsymbol{x}} = \boldsymbol{x}^{\top} \boldsymbol{K} (\boldsymbol{u} + \boldsymbol{f}(\boldsymbol{x})), \qquad (13)$$

is negative definite, which can be achieved by considering a control $\boldsymbol{u} = -\boldsymbol{f}(\boldsymbol{x}) - (\boldsymbol{x}^{\top}\boldsymbol{K})^{\top}$. Let $\boldsymbol{K}_1 = \text{diag}(k_1, k_2)$ denote a diagonal penalty matrix. We parameterize the full penalty matrix with eigendecomposition, \boldsymbol{K}_2 , as,

$$\boldsymbol{K}_2 = \boldsymbol{Q} \boldsymbol{\Lambda} \boldsymbol{Q}^{\top}, \qquad (14)$$

where $\mathbf{\Lambda} = \text{diag}(\lambda_1, \lambda_2)$ denotes the eigenvalue matrix and \mathbf{Q} is parametrized using Eq. (10). PSO is used to optimize the parameters $(k_1 \in [0, 10], k_2 \in [0, 10] \text{ and } \theta \in [0, 2\pi])$ to solve the optimization problem given in Eq. (15).

$$\min_{\boldsymbol{u}} \quad J = \frac{1}{2} \int_0^\infty \boldsymbol{u}^\top \boldsymbol{u} \, dt. \tag{15}$$

We consider $\boldsymbol{x}(0) = [-8, 6]^{\top}$ and the dynamics in Eq. (11) are integrated over a time horizon of 100 seconds. An event-detection feature is used to terminate integration when $\|\boldsymbol{x}\| \leq 10^{-3}$. The full and diagonal penalty matrices resulted in J = 24.83 and J = 26.52, respectively. The penalty matrices for each parameterization method were

$$\boldsymbol{K}_1 = \begin{bmatrix} 0.8094 & 0 \\ 0 & 0.1611 \end{bmatrix}, \ \boldsymbol{K}_2 = \begin{bmatrix} 1.7421 & 0.9560 \\ 0.9560 & 1.1414 \end{bmatrix}$$

Figure 1 shows the state space with trajectories from both solutions, along with the nonlinearities in the dynamics shown as a vector field. Figures 2 and 3 show the CLFs and their time derivatives, respectively, plotted as a function of states with the trajectory plotted on top. Additionally, Figure 4 shows the surface of the control Euclidean norm as a function of states with the state trajectory overlaid. The difference in the CLFs, CLF time derivative, and control norm surfaces as a function of states evidently provide some advantage leading to a reduced controleffort requirement.



Figure 1: Zermelo's problem: state space with the trajectories from each solution with the vector field of nonlinearities in the dynamics.

4. Spacecraft Attitude Control Problem

We proceed to compare the benefits of full penalty matrices in designing LQR and Lyapunov-based control laws for two arbitrarily chosen spacecraft attitude maneuvers: detumbling and rest-to-rest. The target state for both maneuvers is $(\psi(\infty), \theta(\infty), \phi(\infty)) =$



Figure 2: Zermelo's problem: Lyapunov functions vs. states.



Figure 3: Zermelo's problem: Lyapunov functions time-derivatives vs. states.

(0, 0, 0) deg and 0 deg/s angular velocity (note that $t = \infty$ is considered as the final time since the maneuvers are solved as infinite-time horizon problems). Both maneuvers start from an orientation of $(\psi(t_0), \theta(t_0), \phi(t_0)) = (60, 80, -60)$ deg and the detumbling maneuver has an initial angular velocity of $(p(t_0), q(t_0), r(t_0)) =$ (0.1, -0.1, 0.1) rad/s. The spacecraft has a moment of inertia matrix of $\mathbf{I} = \text{diag}(10, 15, 20)$ kg·m². These problem parameters were arbitrarily chosen and the coordinate representations (i.e., Euler angles and quaternions) are typically used in applications [10, 28].



Figure 4: Zermelo's problem: Euclidean norm of control function vs. states.

4.1. LQR Attitude Control Design

The spacecraft's orientation is parameterized using Euler angles, ψ , θ , and ϕ with angular velocity vector as $\boldsymbol{\omega}^{\top} = [p, q, r]$. Let $\boldsymbol{X}^{\top} = [\psi, \theta, \phi, \boldsymbol{\omega}^{\top}]$ denote the state vector. The attitude dynamics can be written as,

$$\dot{\boldsymbol{X}} = \begin{bmatrix} \frac{\boldsymbol{\omega}}{c(\theta)} \begin{bmatrix} 0 & s(\phi) & c(\phi) \\ 0 & c(\phi)c(\theta) & -s(\phi)c(\theta) \\ c(\theta) & s(\phi)s(\theta) & c(\phi)s(\theta) \end{bmatrix} \\ \boldsymbol{I}^{-1} \left(\boldsymbol{U} - \boldsymbol{\omega} \times \boldsymbol{I} \boldsymbol{\omega} \right) \end{bmatrix},$$
(16)

where $c(.) = \cos(.)$ and $s(.) = \sin(.)$. We linearize dynamics around the reference state and control vectors, $\mathbf{X}_0 = \mathbf{0}$ and $\mathbf{U}_0 = \mathbf{0}$, respectively. The state and control errors are denoted by $\mathbf{x} = \mathbf{X} - \mathbf{X}_0$ and $\mathbf{u} = \mathbf{U} - \mathbf{U}_0$, respectively. The quadratic cost functional whose minimization gives the optimal controller gain, i.e., \mathbf{k}_{LQR} appearing in \mathbf{U} (following the standard solution approach to LQR problems [22]), is defined as,

$$\min_{\boldsymbol{k}_{\text{LQR}}} \quad J_{\text{LQR}} = \frac{1}{2} \int_{t_0}^{\infty} \left(\boldsymbol{x}^{\top} \boldsymbol{Q} \boldsymbol{x} + \boldsymbol{u}^{\top} \boldsymbol{R} \boldsymbol{u} \right) dt,$$
(17)

where $Q_{6\times 6}$ and $R_{3\times 3}$ denote the penalty matrices. The control law used for propagating the nonlinear dynamics is $U = u = -k_{LQR}x$ and the MATLAB function lqr is used to find the optimal gain matrix, k.

Remark 2. There may be states that we do not want to be penalized. When Q is diagonal, we can simply set the element corresponding to the state we do not want

penalized to be 0. However, this is not as trivial when \mathbf{Q} is full. Rather than reducing the dimension of \mathbf{x} in Eq. (17) and thus possibly having to recode the LQR solution back into existing software architecture, a diagonal matrix \mathbf{N} with binary elements can be premultiplied with \mathbf{x} instead to choose which states are included in the cost functional. For an example, this is done with Q-law in Section 5.

PSO is used to tune the penalty matrices such that the maneuver is performed with minimal control effort (i.e., the same cost function given in Eq. (15)). Note that control takes the same form, $\boldsymbol{u} = -\boldsymbol{k}\boldsymbol{x}$, but is applied on the state, \boldsymbol{x} , that is obtained from the nonlinear dynamics. Other applications may call for minimizing the deviation from a reference trajectory, in which case the cost function that PSO minimizes might be state-dependent. The diagonal penalty matrices are denoted as \boldsymbol{Q}_1 and \boldsymbol{R}_1 and the full penalty matrices are denoted as \boldsymbol{Q}_2 and \boldsymbol{R}_2 . The upper and lower bounds for the diagonal elements of \boldsymbol{Q}_1 and \boldsymbol{R}_1 and for the eigenvalues of \boldsymbol{Q}_2 and \boldsymbol{R}_2 were 10 and 10^{-8} , respectively.

PSO was invoked 10 times for each case to account for stochastic variations across solutions. Default options were used except for a swarm size of 500 and a maximum number of iterations of 10⁴. These values were arbitrarily chosen such that solutions we deemed to be optimal for each respective form of penalty matrix could be obtained after a number of trials. While the theoretical time horizon considered is infinite, the maximum integration time is set to 100 seconds. The spacecraft attitude dynamics are propagated with MATLAB's ode113 with AbsTol and RelTol of 1.0×10^{-8} . The built-in event-detection feature was used to determine when the states were "close enough" to the target state, X_0 . The considered minimum-energy cost leads to solutions that use the entire time horizon (i.e., less aggressive control commands are prioritized).

Table 1 summarizes the results for both maneuvers. Figure 5 summarizes these results in a bar graph. A significant improvement in using the full penalty matrices was observed, with an average 65.4697% and 62.0296% decrease in cost value for the detumbling and rest-to-rest maneuvers, respectively. Figure 6 shows the most optimal solutions to the detumbling maneuver from both types of penalty matrices. The penalty matrices are given in Appendix A.

4.2. Lyapunov-Based Attitude Control Design

The Lyapunov-based controller uses quaternions, $\mathbf{q}^{\top} = [q_0, q_1, q_2, q_3]$. This quaternion attitude representation is considered in this example to demonstrates the generalization of the method of the proposed and that in practice attitude parameterization is achieved with quaternions. The spacecraft's orientation rate is described

	Minimum-Energy Cost $[kg \cdot m^2/s]$						
	Detur	nbling	Rest-t	o-rest			
$\operatorname{Run}\#$	$oldsymbol{Q}_1,oldsymbol{R}_1$	$oldsymbol{Q}_2,oldsymbol{R}_2$	$oldsymbol{Q}_1,oldsymbol{R}_1$	$oldsymbol{Q}_2,oldsymbol{R}_2$			
1	0.47437	0.14191	0.43308	0.06817			
2	0.47436	0.09139	0.43307	0.06420			
3	0.30796	0.14311	0.29337	0.20775			
4	0.29380	0.08996	0.43307	0.09513			
5	0.47437	0.08397	0.43308	0.07716			
6	0.47437	0.25393	0.24552	0.11512			
7	0.47437	0.10233	0.43308	0.18937			
8	0.30785	0.28289	0.43424	0.07248			
9	0.47436	0.19370	0.43309	0.38302			
10	0.47435	0.07756	0.43307	0.24820			
Mean	0.42302	0.14607	0.40047	0.15206			
Best	0.29380	0.07756	0.24552	0.06420			

Table 1: Detumbling and rest-to-rest maneuver results for LQR controllers with diagonal and full penalty matrices.

in terms of the angular velocity vector $\boldsymbol{\omega}^{\top} = [p, q, r]$. We propagate the spacecraft's attitude dynamics as

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{q} \\ \boldsymbol{\omega} \end{bmatrix}, \qquad \qquad \dot{\boldsymbol{X}} = \begin{bmatrix} \frac{1}{2} \boldsymbol{\Omega} \boldsymbol{q} \\ \boldsymbol{I}^{-1} \left(\boldsymbol{U} - \boldsymbol{\omega} \times \boldsymbol{I} \boldsymbol{\omega} \right) \end{bmatrix}, \qquad (18)$$

where Ω is a 4 × 4 skew-symmetric form of ω [10]. Using a nonlinear candidate quadratic Lyapunov function, the controller is derived in Ref. [10] and is defined as,

$$\boldsymbol{U} = -K_p \boldsymbol{q}_{\rm e} - K_d \boldsymbol{\omega},\tag{19}$$

where $\boldsymbol{q}_{e} \in \mathbb{R}^{3}$ denotes the quaternion error vector and both K_{p} and K_{d} denoting 3×3 control gain matrices (that are used in forming the Lyapunov function). Similar to the LQR problem, we denote the diagonal penalty matrices as $\boldsymbol{K}_{p,1}$ and $\boldsymbol{K}_{d,1}$ and the full penalty matrices as $\boldsymbol{K}_{p,2}$ and $\boldsymbol{K}_{d,2}$.

PSO is invoked 10 times for each case to account for stochastic variations across solutions and the penalty matrices are optimized to minimize the cost functional



Figure 5: Comparison of the cost values for the LQR and Lyapunov-based controller with diagonal and full penalty matrices.



Figure 6: Results from best detumbling solution for each type of penalty matrix with the LQR controller.

defined in Eq. (15). The upper and lower bounds for the diagonal elements of $K_{p,1}$ and $K_{d,1}$ and for the eigenvalues of $K_{p,2}$ and $K_{d,2}$ were 10 and 10^{-8} . The maximum integration time is set to 100 seconds to create an artificial finite time horizon. The built-in event-detection feature of MATLAB's ode113 was used to determine when the states were "close enough" to the target states. The spacecraft attitude dynamics are propagated with MATLAB's ode113 with the same tolerances and PSO was run with the same options used in the LQR problem.

	Minimum-Energy Cost $[kg \cdot m^2/s]$							
	Detur	nbling	Rest-to-rest					
$\operatorname{Run}\#$	$oldsymbol{K}_{p,1},\ oldsymbol{K}_{d,1}$	$oldsymbol{K}_{p,2},oldsymbol{K}_{d,2}$	$oldsymbol{K}_{p,1},oldsymbol{K}_{d,1}$	$oldsymbol{K}_{p,2},oldsymbol{K}_{d,2}$				
1	0.09266	0.07841	0.08399	0.05507				
2	0.09665	0.10420	0.10394	0.05854				
3	0.09683	0.07002	0.08414	0.07439				
4	0.09362	0.09401	0.08073	0.07305				
5	0.10960	0.06194	0.10907	0.07109				
6	0.08541	0.05594	0.07993	0.08840				
7	0.08806	0.10104	0.08311	0.07015				
8	0.13964	0.05885	0.11482	0.05463				
9	0.16088	0.06611	0.08597	0.06427				
10	0.10820	0.06923	0.08034	0.06610				
Mean	0.10716	0.07598	0.09061	0.06757				
Best	0.08541	0.05594	0.07993	0.05463				

Table 2: Detumbling and rest-to-rest maneuver results for the Lyapunov-based controllers with diagonal and full penalty matrices.



Figure 7: Results from best detumbling solution for each type of penalty matrix with the Lyapunovbased controller.

Table 2 shows the results for both maneuvers and with each type of penalty matrix. Figure 5 shows a bar graph comparison of the values of costs. Again, improvement in using the full penalty matrices can be observed. However, the reduction in cost is not as significant as with the LQR controller. This difference is attributed

to the significant improvement the nonlinear Lyapunov-based controller brings as opposed to the LQR controller which is based on dynamics linearized around the target state. Figure 7 shows the solution for the most optimal solution to the detumbling maneuver from both types of penalty matrices. The penalty matrices are given in Appendix A.

5. Low-Thrust Trajectory Optimization Problem

To demonstrate the impact of using a full penalty matrix for Lyapunov-based low-thrust trajectory optimization, a generic LC law and the well-known Q-law [6] are used to solve a variety of low-thrust transfer trajectories. Both control laws are used with a diagonal penalty matrix, \mathbf{K}_1 , and with a full penalty matrix, \mathbf{K}_2 . The full penalty matrix, \mathbf{K}_2 , is generated using Algorithm 1.

In the considered geocentric (and one Vesta-centered) low-thrust maneuvers, the goal is to transfer the spacecraft starting from a fully defined state to an orbit in minimum-time and, with Q-law only, in minimum-fuel. The boundary conditions and parameters for the five considered transfer scenarios are summarized in Table 3. These are a set of transfer cases originally defined in Ref. [6] and have been solved as benchmark problems in a number of other studies, including Refs. [24] and [29]. Note that the Case E* defines the boundary conditions that were used with Q-law. These set of boundary conditions, only differing in the inclination (as a result of a rotation of the inertial frame by 30° about the x-axis), was required to achieve convergence because we found that convergence was not possible with the implementation of Q-law as outlined in Ref. [6] for the original Case E boundary conditions.

Cases	Orbit	$a \; [deg]$	e [-]	$i \; [deg]$	Ω [deg]	ω [deg]	Thrust [N]	Initial Mass [kg]	Specific Impulse [s]	Central Body	PSO Swarm Size	PSO Maxi- mum Itera- tions
А	Initial	7000	0.01	0.05	0	0	. 1	300	3100	Earth	50	50
11	Target	42000	0.01	free	free	free	T					
В	Initial	24505.9	0.725	7.05	0	0	0.35	2000	2000	Earth	50	50
	Target	42165	0.001	0.05	free	free	0.00					
С	Initial	9222.7	0.2	0.573	0	0	03	300	3100	Earth	50	50
	Target	30000	0.7	free	free	free	5.0					
D	Initial	944.64	0.015	90.06	-24.60	156.90	0.045	950	30/15	Vosta	50	50
D	Target	401.72	0.012	90.01	-40.73	free	0.040	550	0040	vesta	50	50
E	Initial	24505.9	0.725	0.06	0	0	- 2 2000		2000 2000	Earth	200	300
	IIIItiai	(24505.9)	(0.7)	(30.06)	(180)	(180)		2000				
(E^*)	Tangat	26500	0.7	116	180	180		2000				
	Target	(26500)	(0.7)	(86)	(180)	(180)						

Table 3: Low-thrust orbit transfer cases.

Earth and Vesta gravitational parameters are 398600.49 km³/s² and 17.8 km³/s², respectively. A canonical scaling is performed on all Cartesian states, such that μ (the central body's gravitational parameter) is 1 DU³/TU². For the geocentric problems, the distance unit (DU) is 6378.1366 km and the time unit (TU) is 806.8110 seconds. For the Vesta-centered problem, DU is 289 km and TU is 1164.4927 seconds.

The state of the spacecraft, $\boldsymbol{x} \in \mathbb{R}^7$, consists of Cartesian position \boldsymbol{r} and velocity \boldsymbol{v} vectors, defined in an inertial frame centered at the central body. All acceleration vectors are expressed with respect to the inertial frame. An additional state, m, is used to track the spacecraft's mass. The spacecraft equations of motion are written as,

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{r} \\ \boldsymbol{v} \\ m \end{bmatrix}, \qquad \dot{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{v} \\ -\frac{\mu}{r^3} \boldsymbol{r} + \frac{T_{\max}}{m} \hat{\boldsymbol{\alpha}} \delta \\ -\frac{T_{\max}}{I_{sp}g_0} \delta \end{bmatrix}, \qquad (20)$$

where $r = ||\mathbf{r}||$ is the spacecraft's distance from the central body, T_{max} is the spacecraft's maximum thrust, I_{sp} is the spacecraft's specific impulse, g_0 is the acceleration of gravity on Earth at sea level defined as 9.80665 m/s², and $\hat{\alpha}$ is the thrust steering unit vector (i.e., $\hat{\alpha}^{\top}\hat{\alpha} = 1$) assumed to freely orient in space. Lastly, $\delta \in [0, 1]$ is the thruster throttle magnitude control, which for the minimum-time transfers is 1 during the entire transfer, but is allowed to vary between 0 and 1 for the minimum-fuel transfers solved with Q-law.

The diagonal penalty matrix, K_1 , is trivially constructed and the full penalty matrix, K_2 , is calculated according to Algorithm 1. MATLAB's PSO algorithm is used to optimize the parameters for each of the control laws. Because PSO is a stochastic meta-heuristic optimization algorithm, it is invoked 5 times for each control law and both penalty matrices. The swarm sizes and the number of iterations are summarized in Table 3. These values are chosen such that the resulting solutions appear to be close to the optimal one. The diagonal elements of K_1 and the eigenvalues of K_2 are bounded arbitrarily between 0 and 100.

5.1. Generic Lyapunov Control Law

Generic LC laws are defined based on classical orbital elements [30]. These results were originally presented in Ref. [17] (except for cost function values that were reported in terms of mass rather than time of flight). We define an error vector, \boldsymbol{w} , based on each transfer case in Table 3 to be used in deriving the control law. This error vector is defined such that it becomes **0** when the final boundary conditions in Table 3 are satisfied. We define the error vector for Case E, $\boldsymbol{w}_{\rm E}$, first. This error vector is defined in terms of the specific angular momentum vector, $\mathbf{h}^{\top} = [h_x, h_y, h_z]$, and the eccentricity vector, \mathbf{e} . Definitions of each are as follows [30]:

$$\boldsymbol{h} = \boldsymbol{r} \times \boldsymbol{v}, \qquad \boldsymbol{e} = \frac{\left(v^2 - \frac{\mu}{r}\right)\boldsymbol{r} - \left(\boldsymbol{r}^\top \boldsymbol{v}\right)\boldsymbol{v}}{\mu}. \tag{21}$$

The error vector for Case E, $\boldsymbol{w}_{\rm E}$, is defined as,

$$\boldsymbol{w}_{\mathrm{E}} = \begin{bmatrix} \boldsymbol{h} - \boldsymbol{h}_{\mathrm{T}} \\ \boldsymbol{e} - \boldsymbol{e}_{\mathrm{T}} \end{bmatrix} \in \mathbb{R}^{6},$$
 (22)

where the subscript 'T' corresponds to the target orbit values.

Remark 3. The considered choice of the error vector is entirely arbitrary for each maneuver. In fact, for Case E, a vector of only 5 dimensions is needed since only 5 orbital elements are being targeted at the final boundary condition due to transfer-type of the considered maneuvers.

For Cases A through D, the boundary conditions are defined in terms of a combination of the orbital elements consisting of the specific angular momentum magnitude h, eccentricity e, inclination i, and right ascension of the ascending node Ω . Definitions of the orbital elements are as follows [30]:

$$h = \|\boldsymbol{h}\|, \qquad e = \|\boldsymbol{e}\|, \qquad (23)$$

$$i = \cos^{-1}\left(\frac{h_z}{h}\right), \qquad \Omega = \cos^{-1}\left(\frac{\hat{\boldsymbol{x}} \cdot \boldsymbol{n}}{\|\hat{\boldsymbol{x}}\|\|\boldsymbol{n}\|}\right), \qquad (24)$$

where $\boldsymbol{n} = \hat{\boldsymbol{z}} \times \boldsymbol{h} = [n_x, n_y, n_z]^{\top}$ is the line of nodes vector and a quadrant check is performed on Ω such that if $n_y < 0$ then $\Omega = 360^\circ - \Omega$.

The error vectors for Cases A through D are defined as,

$$\boldsymbol{w}_{\mathrm{A}} = \begin{bmatrix} h - h_{\mathrm{T}} \\ e - e_{\mathrm{T}} \end{bmatrix} \in \mathbb{R}^{2}, \qquad \boldsymbol{w}_{\mathrm{B}} = \begin{bmatrix} h - h_{\mathrm{T}} \\ e - e_{\mathrm{T}} \\ i - i_{\mathrm{T}} \end{bmatrix} \in \mathbb{R}^{3}, \qquad (25)$$

$$\boldsymbol{w}_{\mathrm{C}} = \begin{bmatrix} h - h_{\mathrm{T}} \\ e - e_{\mathrm{T}} \end{bmatrix} \in \mathbb{R}^{2}, \qquad \boldsymbol{w}_{\mathrm{D}} = \begin{bmatrix} h - h_{\mathrm{T}} \\ e - e_{\mathrm{T}} \\ i - i_{\mathrm{T}} \\ \Omega - \Omega_{\mathrm{T}} \end{bmatrix} \in \mathbb{R}^{4}.$$
(26)

The CLF for each of the transfer problems is defined as,

$$V_{j,i} = \frac{1}{2} \boldsymbol{w}_j^\top \boldsymbol{K}_i \boldsymbol{w}_j, \qquad (27)$$

Г,

for $i = \{1, 2\}$ and $j = \{A, B, C, D, E\}$. The control law is derived to make the total time derivative of Eq. (27) (note that indices j and i are not written for brevity),

$$\dot{V} = \frac{\partial V}{\partial \boldsymbol{r}} \boldsymbol{v} + \frac{\partial V}{\partial \boldsymbol{v}} \left(-\frac{\mu}{r^3} \boldsymbol{r} + \frac{T_{\max}}{m} \hat{\boldsymbol{\alpha}} \right), \qquad (28)$$

becomes negative, leading to the control law, in Eq. (29), as,

$$\hat{\boldsymbol{\alpha}}^* = -\left(\frac{\partial V}{\partial \boldsymbol{v}}\right)^\top / \left\|\frac{\partial V}{\partial \boldsymbol{v}}\right\|.$$
(29)

We used automatic differentiation capabilities of CasADi [31] to derive $\hat{\alpha}^*$. MAT-LAB's variable-step variable-order explicit differential equation solver ode113 is used to integrate Eq. (20) with the control law in Eq. (29) substituted in for $\hat{\alpha}$ with absolute and relative integration tolerances of 10^{-10} . The event-detection capability of ode113 is used to determine when the solution is close enough to the target orbit, since LC laws will not make the system converge in finite time.

Cases	Penalty Matrix	Objective Value, $J(K_{1,2}) = t_f$ [days]							
		Run 1	Run 2	Run 3	Run 4	Run 5	Mean	[days]	
А	K_1	14.5705	14.5702	14.57	14.5703	14.5701	14.5702	_0 0052252	
	K_2	14.4751	14.4748	14.4749	14.4754	14.4748	14.475	-0.0352262	
В	K_1	142.229	142.229	142.229	142.229	142.229	142.229	3 18558	
	K_2	139.03	139.067	139.02	139.074	139.023	139.043		
С	K_1	1.5102	1.5102	1.5102	1.5102	1.5102	1.5102	-0.018357	
	K_2	1.49184	1.49184	1.49184	1.49184	1.49184	1.49184	-0.010331	
D	K_1	24.9903	24.9903	24.9903	24.9903	24.9903	24.9903	-0 153639	
	K_2	24.6992	24.7059	24.9456	24.9456	24.8868	24.8366	-0.100000	
Е	K_1	93.581	80.9959	83.4887	89.0044	106.926	90.7993	2 2/03	
	K_2	109.751	89.6992	85.8899	79.7213	77.6889	88.55	-2.2435	

Table 4: Low-thrust results for generic Lyapunov-based control laws.

Table 4 summarizes the results for both K_1 and K_2 . It can be observed that considering K_2 in the CLF improves the time of flight in each transfer case, with the improvement being more evident for maneuvers occurring over a longer time horizon like Case B or that are more complicated like Case E. The global minimum is achievable for the shorter duration transfers, especially Cases A and C. This is substantiated by PSO being able to find roughly the same solution for all 5 runs of each penalty matrix. For these cases, the new parameterization of the penalty matrix improved the solution optimality given the improvement in the global minimum.

The results indicate that multiple local minima are found for the other transfer cases. In particular, Case E demonstrates the existence of many local minima, with the times of flight ranging broadly and with the worst solution achieved being found by the full penalty matrix, whereas in the other cases the worst solution was always found by the diagonal penalty matrix. On the other hand, the best solution is found by the full matrix representation in all transfer cases and the average times of flights for the full penalty matrix, \mathbf{K}_2 , are lower than those from the diagonal penalty matrix, \mathbf{K}_1 , in every transfer case. Figure 8 summarizes the results in Table 4 as a bar graph. The average times of flight for the 5 runs of PSO in each case. This error bar further illustrates the few or many local minima existing in each case, with Case E having the widest distribution.

Figure 9 also shows the time histories of the orbital elements for the best Case E solution for \mathbf{K}_1 and \mathbf{K}_2 . Note that the oscillations followed by a settling to a particular value in Ω and ω are due to the orbit being nearly equatorial at the beginning of the transfer, leading to Ω and ω not being properly defined. Figure 10 shows the trajectory for these solutions. The full weighting matrix, \mathbf{K}_2 , drives the orbital elements to their target values differently (and likely more efficiently based on the improved times of flight) than the diagonal penalty matrix \mathbf{K}_1 . Inspecting the \mathbf{K}_1 solution in red Figure 9, the semi-major axis has first increased to beyond the semi-major axis of the target orbit, but then, it is reduced noticeably below the target value. On the other hand, the eccentricity is first decreased and then gradually increased and maintained more or less until the end of the maneuver. The eccentricity, however, has changed in a sinusoidal manner. The \mathbf{K}_2 parameterization taking into account the cross-coupling errors of the orbital elements is likely what results in a solution with a better time of flight than the diagonal matrix \mathbf{K}_1 .



Figure 8: Average time-of-flight t_f for each case and penalty matrix for the low-thrust results for the generic Lyapunov-based control laws and Q-law, denoted "GLC" and "QL" in the legend, respectively.



Figure 9: Case E orbital elements: minimum-time generic control law with K_1 and K_2 .



(a) Case E trajectory: minimum-time generic control law (b) Case E trajectory: minimum-time generic control law with K_1 .

Figure 10: Case E trajectory: minimum-fuel Q-law with K_2 .

5.2. Q-law Low-Thrust Trajectory Design Method

Q-law is used to solve the same cases in Table 3. The Q-law from Ref. [6] is used, but with some modifications to make it amenable to a full penalty matrix implementation. The proximity quotient with both K_1 and K_2 , $Q_{1,2}$, is defined as,

$$Q_{1,2} = (1 + W_P P) \left(\boldsymbol{SND} \right)^{\top} \boldsymbol{K}_{1,2} \left(\boldsymbol{ND} \right), \qquad (30)$$

where D denotes the error vector defined as,

$$\boldsymbol{D} = \left(\frac{\boldsymbol{\omega} - \boldsymbol{\omega}_{\mathrm{T}}}{\dot{\boldsymbol{\omega}}_{\mathrm{xx}}}\right), \qquad \qquad \text{for } \boldsymbol{\omega} = a, e, i, \omega, \Omega. \tag{31}$$

In Eq. (31), $\dot{\alpha}_{xx}$ denotes the maximum rate of change of the respective orbital element over thrust direction angles and over true anomaly on the osculating orbit. Analytical relations for these rates are given in Ref. [6]. Note that the difference between instantaneous and target ω and Ω values is taken inside inverse cosine and cosine functions in Ref. [6]. We found that a singularity in the derivative of Eq. (30) with respect to state (which is required to derive Q-law [32]) was encountered when the instantaneous element, α , equals the target element, α_T . For instance, for ω , we have

$$\frac{d}{d\omega} \left(\cos^{-1} \left(\cos \left(\omega - \omega_{\rm T} \right) \right) \right) = \frac{\sin \left(\omega - \omega_{\rm T} \right)}{\sqrt{1 - \cos^2 \left(\omega - \omega_{\rm T} \right)}}.$$
(32)

We found for some transfer cases that this singularity can be encountered frequently before the transfer is complete.

The scaling matrix function, \boldsymbol{S} in Eq. (30), is defined as,

$$\boldsymbol{S} = \operatorname{diag}\left(S_a, S_e, S_i, S_\omega, S_\Omega\right),\tag{33}$$

where the elements of S are defined according to Ref. [6] as,

$$S_{\infty} = \begin{cases} \left[1 + \left(\frac{a - a_T}{m \ a_T} \right)^n \right]^{\frac{1}{r}}, & \text{for } \varpi = a, \\ 1, & \text{for } \varpi = e, i, \omega, \Omega, \end{cases}$$
(34)

where m, n, and r are scalars with nominal values set to 3, 4, and 2, respectively. In Ref. [6], if certain elements weren't desired to be targeted, then their penalty factor, W_{α} , could simply be set equal to 0. When \mathbf{K}_1 is used, the same step can be performed by setting the desired elements of \mathbf{K}_1 to 0 to avoid targeting those elements. However, when \mathbf{K}_2 is used, the matrix \mathbf{N} must be introduced and is defined as,

$$\boldsymbol{N} = \operatorname{diag}\left(N_a, N_e, N_i, N_\omega, N_\Omega\right), \text{ with }$$
(35)

$$N_{\infty} = \begin{cases} 1 & \text{if } \varpi \text{ is targeted} \\ 0 & \text{if } \varpi \text{ is not targeted} \end{cases} \text{ for } \varpi = a, e, i, \omega, \Omega.$$
(36)

This follows the procedure described in Remark 2, which can be useful when control laws must be hard-coded into vehicle hardware, or when it is cumbersome to rederive the control law depending on the problem being considered. Alternatively, Eq. (30) could be rederived for a different set of α depending on what elements are desired to be targeted. However, the present approach allows for a more user-friendly implementation for solving a wider variety of problems.

The penalty function, P, serves to enforce a minimum-periapsis-radius constraint and is defined as,

$$P = \exp\left(k\left(1 - \frac{r_p}{r_{p,\min}}\right)\right),\tag{37}$$

where the value of k is set to 4. The periapsis radius can be found with $r_p = a(1-e^2)$ and $r_{p,\min}$ is the prescribed minimum-periapsis radius and is set arbitrarily to 10 km in this work for all problems. In Eq. (30), the penalty factor, W_P , is either 1 or 0 to activate 0r deactivate the penalty function P. In this work, it is set to 1. The thrust steering control law, $\hat{\alpha}$, is parameterized by two angles, which are derived similarly to Ref. [32]. For minimum-time solutions, the thrust magnitude control δ is always equal to 1. Ref. [33] describes a coasting mechanism, which determines the thrust magnitude when minimum-fuel transfers are desired which is defined as,

$$\delta = \begin{cases} 0 & \eta_r \le \eta_{\text{cut}}, \\ 1 & \eta_r > \eta_{\text{cut}}, \end{cases}$$
(38)

where $\eta_{\text{cut}} \in [0, 1]$ is a user-defined cut-off value, which determines the trade-off between time-of-flight and propellant savings and η_r is the so-called relative effectivity, which is a measure of how effective it is to thrust on an orbit and is defined as,

$$\eta_r = \frac{\dot{Q}_n - \dot{Q}_{nx}}{\dot{Q}_{nn} - \dot{Q}_{nx}},\tag{39}$$

where \dot{Q}_n is the time-derivative of Q minimized over the steering control, \dot{Q}_{nn} is \dot{Q}_n minimized over the orbit's true anomaly, and \dot{Q}_{nx} is \dot{Q}_n maximized over the orbit's true anomaly. Please refer to Ref. [33] for further details. The spacecraft's dynamics are propagated in classical orbital elements (COEs) using MATLAB's ode45 with both absolute and relative tolerances set to 10^{-10} . This choice of numerical integrator was different from the one used in the previous sections due to existing code architecture of the authors. Checks are performed during integration to ensure that the singularities at circular and planar orbits are not encountered. Due to the chattering that can result from the Q-law [34, 35], the control input is assumed constant over 1-minute intervals.

The results are summarized in Table 5. The last column shows the difference in the mean objective value across the 5 runs for each penalty matrix. Because time-of-flight is being minimized and final fuel mass is being maximized, the full weighting matrix shows improvement in every case, with the difference being negative for minimum-time problems and positive for minimum-fuel problems. Most cases show little improvement in objective values, indicating that Q-law might be less dependent on the penalty matrix and more dependent on the maximum orbital element rates-of-change, $\dot{\mathbf{e}}_{xx}$. However, Case E shows a significantly larger improvement with a 10.5 day improvement in time-of-flight and a nearly 90 kg improvement in final fuel mass. Figure 8 summarizes the minimum-time results (along with the minimum-time results from the generic Lyapunov control laws) where the average time-of-flight represents the bars and the error bars represent the distribution in solutions. The lower distribution visible in Case E further shows how Q-law might be less dependent on the value of the penalty matrix than the generic Lyapunov control law. Figure 11 shows the results from the minimum-fuel solutions summarized as a bar graph.

Case	Objective,	Penalty		$\frac{1}{\max(J(K_2)) - \max(J(K_2))}$						
	$J(K_{1,2})$	Matrix	Run 1	Run 2	Run 3	Run 4	Run 5	Mean	$\begin{bmatrix} \text{days or } \text{kg} \end{bmatrix}$	
A	t [dava]	t [dava]	K_1	14.815	14.815	14.815	14.815	14.815	14.815	0 175079
	if [uays]	K_2	14.626	14.626	14.625	14.694	14.626	14.639	-0.113912	
	<i>m</i> . [kg]	K_1	259.807	259.807	259.809	259.809	259.807	259.808	0 496174	
	m _f [kg]	K_2	260.304	260.346	260.306	260.239	260.324	260.304	- 0.430174	
	te [dave]	K_1	137.31	137.309	137.31	137.31	137.31	137.31	0.0606944	
В	if [uays]	K_2	137.182	137.309	137.189	137.256	137.309	137.249	-0.0000344	
	m e [kg]	K_1	1809.67	1809.67	1809.66	1809.67	1809.71	1809.67	- 0.0409008	
	m _f [kg]	K_2	1809.71	1809.67	1810	1809.73	1809.46	1809.72	- 0.0405000	
С	t c [davs]	K_1	1.4007	1.4007	1.4007	1.4007	1.4007	1.4007		
	<i>v_f</i> [ddy5]	K_2	1.3944	1.3958	1.39514	1.3958	1.3958	1.3954		
	m . [kg]	K_1	269.769	269.88	269.769	269.769	269.769	269.792	1_86853	
	mf [K8]	K_2	271.66	271.66	271.66	271.66	271.66	271.66	1.00000	
	t c [davs]	K_1	25.756	25.744	25.746	25.735	25.754	25.747		
D	if [uays]	K_2	25.527	24.706	25.422	24.817	26.026	25.300	-0.447001	
D	m e [kg]	K_1	946.697	946.697	946.697	946.696	946.697	946.697	0 029531	
	mf [K8]	K_2	946.701	946.719	946.761	946.7	946.75	946.726	0.023001	
F	te [dave]	K_1	101.206	104.549	110.029	102.395	109.906	105.617	-10 5368	
	<i>if</i> [uays]	K_2	99.3076	91.6944	100.107	92.2472	92.0444	95.0801	-10.0000	
Ц	m e [kg]	K_1	1147.83	1145.12	1136.5	1130.6	1125.07	1137.02	89 5596	
	mf [rg]	K_2	1210.86	1238.72	1241.05	1235.96	1206.32	1226.58	09.0090	

Table 5: Q-law low-thrust transfer results.



Figure 11: Average minimum-fuel objective values for the Q-law solutions.

Figures 12, 13, 14, and 15 show the orbital elements time histories for the best Case E minimum-time and minimum-fuel Q-law solutions for each penalty matrix type. Figures 16, 17, and 18 show the trajectories for these solutions. Note that the rotated Case E* boundary conditions are plotted in red in the orbital element plots along with the original Case E orbital elements in black. The problem is solved in the Case E* frame and the solution is simply rotated back into the Case E frame with single rotation about the x-axis. The trajectories show the Case E trajectory solution. Similar to the results for the generic Lyapunov control law in the previous section, the full penalty matrix \mathbf{K}_2 changes orbital elements over time quite differently than with the diagonal penalty matrix, \mathbf{K}_1 . Note that any oscillations or sharp jumps in Ω and ω between 0° and 360° are due to the orbit being nearly circular and/or equatorial leading to ambiguity in Ω and ω .

6. Conclusion

Parameterization and optimization of positive-definite penalty matrices has widespread applications in modern control and optimization theories. Diagonal parameterization is a common approach, but it represents only a subset of the full solution space, as it neglects cross-coupled terms. We propose an efficient parameterization that is based



Figure 12: Case E orbital elements: minimum-time Q-law with K_1 .

on eigendecomposition of symmetric matrices, in which the set of parameters consists of eigenvalues and variables used to parameterize the orthogonal eigenvector matrices. Considering a full penalty matrix can broaden the solution space by including cross-coupling error terms in quadratic costs that are used extensively for designing modern controls, such as a linear quadratic regulator (LQR) or Lyapunov-based nonlinear controllers, as we have demonstrated through the results.

Several control problems were solved with Lyapunov-based control laws and LQR for the standard diagonal and the proposed full penalty matrix representations. We clarify that solutions are by no means optimal due to limited runs of PSO and limited computational resources. Additionally, other global optimization algorithms may even perform better than PSO. For instance, Ref. [24] finds more optimal transfer solutions using Q-law by performing extensive parameter searches with genetic algorithm over multiple days of CPU time. The matter of optimizing the parameters should be investigated further. However, our comparative results demonstrate that improved solutions can be obtained with the full penalty matrix with respect to the considered optimality criteria of each problem. The improvement is especially evident for classes of more nonlinear and long-duration problems for which optimization



Figure 13: Case E orbital elements: minimum-time Q-law with K_2 .

of the off-diagonal terms broadens the optimality of the solutions (e.g., large changes in orbital elements for the low-thrust trajectories). In the most extreme case, the improvement in performance was 65%. Our hope is that the simplicity of the demonstrated methods allows researchers/practitioners in many fields of control theory and optimization to potentially obtain more optimal solutions to their practical control problems.

Appendix A. Numerical Values of the Penalty Matrices

For reproducibility, we provide the penalty matrices for some of the solutions we obtained in this work. Eqs. (A.1) and (A.2) show the penalty matrices that were found for the best detumbling attitude control solution for LQR and Lyapunov-based control, respectively, corresponding with the solutions shown in Figures 6 and 7. Eqs. (A.3), (A.4), (A.5) show the penalty matrices for the Case E for the minimum-time generic LC solution, minimum-time Q-law solution, and minimum-fuel Q-law solution, respectively.



Figure 14: Case E orbital elements: minimum-fuel Q-law with K_1 .

$$\begin{aligned} \boldsymbol{Q}_{1} &= \text{diag}\left(1.05309, 0.00299126, 0.68307, 4.5293, 9.87489, 0.0452255\right) & (A.1a) \\ \boldsymbol{R}_{1} &= \text{diag}\left(8.36281, 4.40636, 9.23266\right) & (A.1b) \\ \boldsymbol{Q}_{2} &= \begin{bmatrix} 1.185 & -0.613646 & 0.311778 & 0.48939 & 0.343032 & 0.0979317 \\ -0.613646 & 0.698828 & 0.572306 & -1.82684 & -0.0230107 & -0.0140387 \\ 0.311778 & 0.572306 & 1.84127 & -3.45008 & 0.448065 & 0.110059 \\ 0.48939 & -1.82684 & -3.45008 & 7.57063 & -0.592071 & -0.132696 \\ 0.343032 & -0.0230107 & 0.448065 & -0.592071 & 0.172537 & 0.0455803 \\ 0.0979317 & -0.0140387 & 0.110059 & -0.132696 & 0.0455803 & 0.0121875 \end{bmatrix} & (A.1c) \\ \boldsymbol{R}_{2} &= \begin{bmatrix} 9.57947 & 1.63521 & 0.963686 \\ 1.63521 & 3.39122 & -4.30809 \\ 0.963686 & -4.30809 & 6.53455 \end{bmatrix} & (A.1d) \end{aligned}$$



Figure 15: Case E orbital elements: minimum-fuel Q-law with K_2 .

$$\boldsymbol{K}_{p,1} = \text{diag}\left(0.192456, 0.167666, 0.272458\right) \tag{A.2a}$$

$$\boldsymbol{K}_{d,1} = \text{diag}\left(1.82655, 2.25069, 3.04384\right) \tag{A.2b}$$

$$\boldsymbol{K}_{p,2} = \begin{bmatrix} 2.2675 & -0.295543 & 1.97134 \\ -0.295543 & 0.915225 & -0.761434 \\ 1.97134 & -0.761434 & 2.35022 \end{bmatrix}$$
(A.2c)
$$\boldsymbol{K}_{d,2} = \begin{bmatrix} 7.03549 & 0.367325 & 1.74591 \\ 0.367325 & 7.73374 & -2.65179 \\ 1.74591 & -2.65179 & 2.34927 \end{bmatrix}$$
(A.2d)



Figure 16: Case E trajectory: minimum-time Q-law with K_1 and K_2 .



Figure 17: Case E trajectory: minimum-fuel Q-law with K_1 .

$$\boldsymbol{K}_{1} = \text{diag}\left(6.5225, 98.1494, 8.9658, 10.2037, 97.3815, 20.7142\right)$$
(A.3a)

$$\boldsymbol{K}_{2} = \begin{bmatrix} 39.4746 & 17.5941 & -2.0538 & -3.3242 & 0.1723 & -0.3125 \\ 17.5941 & 68.1822 & -3.7857_{34} - 6.9744 & 0.1840 & -0.5589 \\ -2.0538 & -3.7857 & 4.6930 & 0.5193 & 2.9905 & 0.2195 \\ -3.3242 & -6.9744 & 0.5193 & 11.5512 & 0.4138 & -5.6421 \\ 0.1723 & 0.1840 & 2.9905 & 0.4138 & 77.1079 & 1.5671 \\ -0.3125 & -0.5589 & 0.2195 & -5.6421 & 1.5671 & 81.3064 \end{bmatrix}$$
(A.3b)



Figure 18: Case E trajectory: minimum-fuel Q-law with K_2 .

$$\boldsymbol{K}_{1} = \text{diag}\left(2.54742, 0.00530434, 0.242459, 9.64791, 7.76675\right)$$
(A.4a)

$$\boldsymbol{K}_{2} = \begin{bmatrix} 9.61437 & 0.59816 & 0.727462 & 0.0886329 & 0.0288422 \\ 0.59816 & 5.65613 & -4.0757 & -1.62804 & -1.24825 \\ 0.727462 & -4.0757 & 3.52475 & 0.90911 & 1.25853 \\ 0.0886329 & -1.62804 & 0.90911 & 4.76366 & 0.652616 \\ 0.0288422 & -1.24825 & 1.25853 & 0.652616 & 2.68927 \end{bmatrix}$$
(A.4b)

$$\boldsymbol{K}_{1} = \text{diag} \left(8.65871, 2.22226, 4.21207, 4.84089, 8.3779\right)$$
(A.5a)
$$\boldsymbol{K}_{2} = \begin{bmatrix} 8.25151 & 3.14031 & 1.22609 & 0.337485 & 0.0941679 \\ 3.14031 & 3.62918 & -1.34696 & 0.0247984 & 0.066369 \\ 1.22609 & -1.34696 & 4.13089 & 0.256997 & -0.0302524 \end{bmatrix}$$
(A.5b)

$$\boldsymbol{K}_{2} = \begin{bmatrix} 8.25151 & 3.14031 & 1.22009 & 0.337485 & 0.0941679 \\ 3.14031 & 3.62918 & -1.34696 & 0.0247984 & 0.066369 \\ 1.22609 & -1.34696 & 4.13089 & 0.256997 & -0.0302524 \\ 0.337485 & 0.0247984 & 0.256997 & 8.95998 & -0.429254 \\ 0.0941679 & 0.066369 & -0.0302524 & -0.429254 & 7.19688 \end{bmatrix}$$
(A.5k)

References

[1] Z. Zhou, R. Colgren, A non-linear spacecraft attitude tracking controller for large non-constant rate commands, International Journal of Control 78 (5) (2005) 311-325. doi:10.1080/00207170500079779.

- [2] A. Marco, P. Hennig, J. Bohg, S. Schaal, S. Trimpe, Automatic LQR tuning based on Gaussian process global optimization, in: 2016 IEEE International Conference on Robotics and Automation (ICRA), IEEE, Stockholm, 2016, pp. 270–277. doi:10.1109/ICRA.2016.7487144.
- [3] S. Di Cairano, H. Park, I. Kolmanovsky, Model Predictive Control approach for guidance of spacecraft rendezvous and proximity maneuvering, International Journal of Robust and Nonlinear Control 22 (12) (2012) 1398–1427. doi:10. 1002/rnc.2827.
- [4] K. Nguyen, S. Schoedel, A. Alavilli, B. Plancher, Z. Manchester, TinyMPC: Model-Predictive Control on Resource-Constrained Microcontrollers, in: 2024 IEEE International Conference on Robotics and Automation (ICRA), IEEE, Yokohama, Japan, 2024, pp. 1–7. doi:10.1109/ICRA57147.2024.10610987.
- [5] J. L. Crassidis, F. L. Markley, Sliding mode control using modified Rodrigues parameters, Journal of Guidance, Control, and Dynamics 19 (6) (1996) 1381– 1383. doi:10.2514/3.21798.
- [6] A. Petropoulos, Low-Thrust Orbit Transfers Using Candidate Lyapunov Functions with a Mechanism for Coasting, in: AIAA/AAS Astrodynamics Specialist Conference and Exhibit, American Institute of Aeronautics and Astronautics, Providence, Rhode Island, 2004. doi:10.2514/6.2004-5089.
- [7] A. D. Ames, X. Xu, J. W. Grizzle, P. Tabuada, Control Barrier Function Based Quadratic Programs for Safety Critical Systems, IEEE Transactions on Automatic Control 62 (8) (2017) 3861–3876. doi:10.1109/TAC.2016.2638961.
- [8] D. Eui Chang, D. F. Chichka, J. E. Marsden, ,Control and Dynamical Systems 107-81, California Institute of Technology, Pasadena, CA 91125, Lyapunovbased transfer between elliptic Keplerian orbits, Discrete & Continuous Dynamical Systems - B 2 (1) (2002) 57–67. doi:10.3934/dcdsb.2002.2.57.
- [9] J. L. Junkins, Y. Kim, Introduction to Dynamics and Control of Flexible Structures, American Institute of Aeronautics and Astronautics, Reston, 2000.
- [10] H. Schaub, J. L. Junkins, Analytical mechanics of space systems, fourth edition Edition, AIAA education series, American Institute of Aeronautics and Astronautics, Inc(AIAA), Reston, VA, 2018.

- [11] G. Strang, Introduction to linear algebra, 6th Edition, Wellesley-Cambridge press, Wellesley, Mass, 2023.
- [12] R. Shepard, S. R. Brozell, G. Gidofalvi, The Representation and Parametrization of Orthogonal Matrices, The Journal of Physical Chemistry A 119 (28) (2015) 7924-7939. doi:10.1021/acs.jpca.5b02015.
- [13] A. Cayley, On the Motion of Rotation of a Solid Body, Cambridge Mathematics Journal 3 (1843) 224–232.
- [14] I. Bar-Itzhack, F. Markley, Minimal parameter solution of the orthogonal matrix differential equation, IEEE Transactions on Automatic Control 35 (3) (1990) 314–317. doi:10.1109/9.50344.
- [15] H. Schaub, P. Tsiotras, J. L. Junkins, Principal rotation representations of proper N × N orthogonal matrices, International Journal of Engineering Science 33 (15) (1995) 2277–2295. doi:10.1016/0020-7225(95)00070-E.
- [16] Y. Oshman, I. Bar-Itzhack, Eigenfactor solution of the matrix Riccati equation– A continuous square root algorithm, IEEE Transactions on Automatic Control 30 (10) (1985) 971–978. doi:10.1109/TAC.1985.1103823.
- [17] N. P. Nurre, S. Tafazzol, E. Taheri, Expanding the Class of Quadratic Control-Lyapunov Functions for Low-Thrust Trajectory Optimization, arXiv:2408.14412 [math] (Aug. 2024).
- [18] D. K. Hoffman, R. C. Raffenetti, K. Ruedenberg, Generalization of Euler Angles to N -Dimensional Orthogonal Matrices, Journal of Mathematical Physics 13 (4) (1972) 528–533. doi:10.1063/1.1666011.
- [19] L. E. Blumenson, A Derivation of n-Dimensional Spherical Coordinates, The American Mathematical Monthly 67 (1) (1960) 63. doi:10.2307/2308932.
- [20] J. G. Ziegler, N. B. Nichols, Optimum Settings for Automatic Controllers, Journal of Fluids Engineering 64 (8) (1942) 759–765. doi:10.1115/1.4019264.
- [21] K. Åström, T. Hägglund, Revisiting the Ziegler-Nichols step response method for PID control, Journal of Process Control 14 (6) (2004) 635-650. doi:10. 1016/j.jprocont.2004.01.002.
- [22] A. E. Bryson, Y.-C. Ho, Applied optimal control: optimization, estimation, and control, rev. print Edition, Cambridge University Press, Cambridge, 1975.

- [23] S. Trimpe, A. Millane, S. Doessegger, R. D'Andrea, A Self-Tuning LQR Approach Demonstrated on an Inverted Pendulum, IFAC Proceedings Volumes 47 (3) (2014) 11281–11287. doi:10.3182/20140824-6-ZA-1003.01455.
- [24] S. Lee, P. Von Ailmen, W. Fink, A. Petropoulos, R. Terrile, Design and optimization of low-thrust orbit transfers, in: 2005 IEEE Aerospace Conference, IEEE, Big Sky, MT, USA, 2005, pp. 855–869. doi:10.1109/AER0.2005.1559377.
- [25] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95 - International Conference on Neural Networks, Vol. 4, IEEE, Perth, WA, Australia, 1995, pp. 1942–1948. doi:10.1109/ICNN.1995.488968.
- [26] N. P. Nurre, E. Taheri, End-to-end lyapunov-based eclipse-feasible low-thrust transfer trajectories to NRHO, arXiv preprint arXiv:2409.03196 (2024).
- [27] J. L. Junkins, D. W. Rew, A Simultaneous Structure/Controller Design Iteration Method, in: 1985 American Control Conference, 1985, pp. 1642–1647. doi: 10.23919/ACC.1985.4788878.
- [28] J. L. Junkins, J. D. Turner, Optimal spacecraft rotational maneuvers, no. 3 in Studies in astronautics, Elsevier Scientific, Amsterdam, 1986.
- [29] G. I. Varga, J. M. S. Perez, Many-Revolution Low-Thrust Orbit Transfer Computation using Equinoctial Q-Law Including J2 and Eclipse Effects, in: 6th International Conference on Astrodynamics Tools and Techniques (ICATT), Darmstadt, Germany, 2016.
- [30] D. A. Vallado, Fundamentals of astrodynamics and applications, fifth edition, first printing Edition, no. 21 in Space technology library, Microcosm Press, Torrance, CA, 2022.
- [31] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, M. Diehl, CasADi: a software framework for nonlinear optimization and optimal control, Mathematical Programming Computation 11 (1) (2019) 1–36. doi:10.1007/ s12532-018-0139-4.
- [32] J. L. Shannon, M. T. Ozimek, J. A. Atchison, C. M. Hartzell, Q-Law Aided Direct Trajectory Optimization of Many-Revolution Low-Thrust Transfers, Journal of Spacecraft and Rockets 57 (4) (2020) 672–682. doi:10.2514/1.A34586.

- [33] A. E. Petropoulos, Refinements to the Q-law for low-thrust orbit transfers, in: 15th AAS/AIAA Space Flight Mechanics Conference, AAS 05-162, Copper Mountain, Colorado, 2005.
- [34] Noble Ariel Hatten, A Critical Evaluation of Modern Low-Thrust, Feedback-Driven Spacecraft Control Laws, Ph.D. thesis, The University of Texas at Austin, Austin, Texas (Dec. 2012).
- [35] G. R. Hecht, E. M. Botta, Q-Law Control With Sun-Angle Constraint for Solar Electric Propulsion, IEEE Transactions on Aerospace and Electronic Systems (2024) 1–13doi:10.1109/TAES.2024.3424429.