Single- and Multi-Degree of Freedom Servo Trajectory Generation: An Optimization Framework, Implementation, and Examples

Layne Clemen[1] and Cory J. Rupp[2,*]

***Abstract*— Single-axis motion profiles are typically generated by an engineer through adjusting acceleration, jerk, and velocities at the bounds of system specifications to find a trajectory that meets the system performance requirements. This is an often trite task that does not allow many secondary system performance options such as energy usage to be considered. In multi-degree of freedom systems, secondary considerations are typically brushed aside even more as the task complexity requires more focus on the coordinate motion of multiple stages. This paper presents an optimization framework to generate optimal trajectories offline for single- and multi-axis systems based entirely on basic kinematic relationships. The framework, implemented in the Python package Pyomo, consists of setting a minimization target and appropriate norm in conjunction of a set of operational constraints. Trajectory generation results for minimal power, jerk, acceleration, and velocity considerations using $\mathcal{L}_1$, $\mathcal{L}_2$, and $\mathcal{L}_\infty$ norms are compared. A nonlinear optimization scheme to minimize move time is also examined. Results show that trapezoidal/S-curve profiles embody minimization of velocity norms as well as move time, while other optimal profiles can be utilized in cases where system specifications require different performance options. Normalized peak power and total system energy cost functions are also developed. This paper provides practicing engineers with an easily implementable tool to facilitate system component selection and performance optimization.**

I. INTRODUCTION

Trajectory generation is a well studied area with extensive literature describing solutions for a myriad of applications [1], [2]. Many applications consider multiple degree of freedom systems such as robotic arms [1] and multi-axis machines [3], typically utilizing offline or online optimization techniques with the dynamics of the actuators and controlled system as constraints. An often overlooked application is trajectory generation for 1-dimensional motion control systems such as basic linear motion stages. There have been some recent developments for these simpler systems but not much that is readily usable by the practicing engineering community [4].

Two types of movement profiles are typically implemented in commercial of the shelf (COTS) motor controllers: a trapezoidal profile or an S-curve. Parameters for these profiles are commonly generated online by tweaking the motion parameters until the engineer achieves acceptable performance, usually focusing on one aspect (e.g., velocity or acceleration) while disregarding other factors. While these profiles work well for general applications, there exists a need to optimize systems for additional performance metrics such as reducing controller and mechanical energy usage and wear. Trajectory generation for multi-degree for freedom platforms rarely considers such secondary as energy usage, although their application are equally needed.

This paper presents a simple offline framework to generate motion trajectories optimized under various performance objectives. This framework casts trajectory generation as a continuous optimization problem with constraints relating the kinematics of the system. The theoretical development is implemented in Pyomo [5], an algebraic modeling language (AML) package for Python. Results show that trapezoidal/S-curves embody trajectories that minimize of velocity norms or move duration. Other cost functions and their implementations are presented that can be utilized in cases where system specifications require different performance options. Extension to multi degree of freedom trajectory optimization is straightforward and a simple application is demonstrated using the framework.

This paper is organized as follows, Section 2 presents a basic problem formulation around a generic cost function with kinematic constraints. Section 3 discusses implementation in the Pyomo package. Section 4 discusses the results of studies of a single axis system, including exploration of a multi-objective formulation. In section 5, the framework is extended to a two-degree of freedom system and demonstrated with an example. Section 6 discusses conclusions and future extensions of the work.

II. PROBLEM FORMULATION

Trajectory motion consists of multiple kinematic variables, including displacement d, velocity v, acceleration a, and jerk j, all as a function of time and related through the kinematic equations

$$\begin{aligned} \dot{d} &= v \\ \dot{v} &= a \\ \dot{a} &= j \end{aligned} \tag{1}$$

These variables are typically subject to a variety of constraints on their possible values

$$\begin{aligned} \underline{v} \leq v \leq \overline{v} \\ \underline{a} \leq a \leq \overline{a} \\ \underline{j} \leq j \leq \overline{j} \end{aligned} \tag{2}$$

where upper $\overline{\square}$ and lower $\underline{\square}$ limits are determined through electromechanical system requirements and specifications.

[1]Elexity, Bend, OR, USA layne@elexity.io
[2]ATA Engineering, Inc., Lakewood, CO, USA cory.rupp@ata-e.com
*Corresponding author

The trajectory optimization problem can be formulated as a standard optimization problem of the form

$$\begin{aligned} &\min_{\mathbf{x}} \ J \\ s.t. \quad &\mathbf{A}\mathbf{x} = \mathbf{b} \\ &\mathbf{H}\mathbf{x} \leq \mathbf{k} \end{aligned} \tag{3}$$

where J is the cost or objective function, $\mathbf{x}$ is the vector of decision variables, and $\mathbf{A}$, $\mathbf{b}$, $\mathbf{H}$, and $\mathbf{k}$ are appropriately sized matrices and vectors describing the problem constraints. The kinematic motion of the system 1 can be discretized and used as constraint equations between the decision variables

$$\begin{aligned} d_{k+1} &= d_k + v_k \Delta t \\ v_{k+1} &= v_k + a_k \Delta t \\ a_{k+1} &= a_k + j_k \Delta t \end{aligned} \tag{4}$$

These constraints, the decision variable box constraints, and boundary conditions for a motion starting and stopping at rest[1] can be used to formulate the optimization problem as:

$$\begin{aligned} &\min_{\mathbf{x}} \ J \\ s.t. \quad &\left.\begin{aligned} &x_0 = 0 \\ &v_0 = 0 \\ &a_0 = 0 \\ &d_{k+1} = d_k + v_k \Delta t \\ &v_{k+1} = v_k + a_k \Delta t \\ &a_{k+1} = a_k + j_k \Delta t \\ &d_N = D \\ &v_N = 0 \\ &a_N = 0 \\ &0 \leq v_k \leq \overline{v} \\ &\underline{a} \leq a_k \leq \overline{a} \\ &\underline{j} \leq j_k \leq \overline{j} \end{aligned}\right\} \mathcal{C} \end{aligned} \tag{5}$$

where D is the total distance to move, Δt is the discretization period, and $\mathcal{C}$ is the entire set of constraints for the problem. The kinematic variables are discretized into $k \in [1,\ 2,\ \ldots\ ,\ N-1,\ N]$ indexes totaling N segments, where

$$N = \frac{t_f}{\Delta t} \tag{6}$$

Depending on the cost function, the decision variables $\mathbf{x}$ will consist of either the kinematic variables[2] $\mathbf{d}$, $\mathbf{v}$, $\mathbf{a}$, $\mathbf{j}$ or the total move time t_f.

A. Minimizing Kinematic Variables

The cost function when using kinematic variables is formulated in terms of $\mathcal{L}_1$, $\mathcal{L}_2$, or $\mathcal{L}_\infty$ norms of the vectors of the decision variables. These norms represent various aspects of the profile performance, they are[3]:

$$\begin{aligned} \|\mathbf{x}\|_1 &= \sum_{i=1}^{n} |x_i| && \text{(Absolute or Taxicab Norm)} \\ \|\mathbf{x}\|_2 &= \left(\sum_{i=1}^{n} x_i^2 \right)^{1/2} && \text{(RMS or Euclidean Norm)} \\ \|\mathbf{x}\|_\infty &= \max_{1 \leq i \leq n} |x_i| && \text{(Peak or Inifinty Norm)} \end{aligned}$$

B. Minimizing Power and Energy

For a linear mechanical system with a constant mass, the power that the load experiences is

$$P_k = F_k v_k \tag{7}$$

where F_k is the force on the system at step k. The time-vector of forces, acceleration, and velocity are

$$\begin{aligned} \mathbf{F} &= [F_1, F_2, \ldots, F_N]^T \\ \mathbf{a} &= [a_1, a_2, \ldots, a_N]^T \\ \mathbf{v} &= [v_1, v_2, \ldots, v_N]^T \end{aligned} \tag{8}$$

From Newton's 2nd law $\mathbf{F} = m\mathbf{a}$, the instantaneous power delivered to the load is the Hadamard product of $\mathbf{F}$ and $\mathbf{v}$

$$\mathbf{P} = \mathbf{F} \circ \mathbf{v} = (m\mathbf{a}) \circ \mathbf{v} \tag{9}$$

Because norms have the property with constant c

$$\|c\mathbf{x}\|_p = c\|\mathbf{x}\|_p \tag{10}$$

and mass is constant, the objective to minimize the norm of the power can be reduced to

$$J = \|\mathbf{P}\|_p = m\|\mathbf{a} \circ \mathbf{v}\|_p \tag{11}$$

Therefore, combining the optimization problem (5) with (11) will yield minimization of the RMS power for $p = 2$ and the peak power for $p = \infty$. This can be used to meet system requirements, such as motor driver specification. It is noted that using the $\mathcal{L}_1$-norm (i.e., $p = 1$) of power as the cost function will also minimize the total energy of the movement. This can be shown by writing the total energy as the absolute sum of the instantaneous power multiplied by the constant time interval Δt (assuming constant power between intervals) and then using the definition of the $\mathcal{L}_1$-norm:

$$E_{total} = \sum_{i=1}^{n} |P_i \Delta t| = \sum_{i=1}^{n} |P_i| \Delta t = \|\mathbf{P}\|_1 \Delta t = m\|\mathbf{a} \circ \mathbf{v}\|_1 \Delta t \tag{12}$$

Remark 1: These costs functions can easily be extended to a time-varying mass if the mass-time profile is known *a priori*. The time-varying mass would effectively weight the entries of the power vector.

Remark 2: The minimization of power produces a lower bound on the power the actuator must inject into the system. In a real system, one would also have to account for the system friction. To do this, it can be assumed that the friction has the form

$$\mathbf{F}_{friction} = f(\mathbf{v}) \tag{13}$$

The total force to move the mass would be

$$\mathbf{F} = f(\mathbf{v}) + m\mathbf{a} \tag{14}$$

[1] Different boundary conditions can be used without loss of generality.

[2] The variables here represent a linear mechanical system but can be changed to any energy domain without loss of generality [6].

[3] The 2-norm only varies from the RMS value by a constant of $1/\sqrt{n}$ and therefore behaves the same under optimization.

whereby trajectory optimization would now reference the cost function

$$J = \|(f(\mathbf{v}) + m\mathbf{a}) \circ \mathbf{v}\|_p \tag{15}$$

to account for friction. Further discussion of this approach is out of the scope of this paper.

C. Minimizing Move Time

In order to minimize the move time step, Δt is added as a decision variable. The cost function becomes

$$J = N\Delta t \tag{16}$$

where N is again the number of time steps. It is important here to ensure that the time discretization factor N is large enough to accurately capture the system dynamics throughout the range of Δt encountered while solving the optimization problem. An easy way to do this would be to add side constraints to the permissible of Δt into (4) and selecting a value of N that is sufficient for the worst case.

In this cost function formulation, the distance constraint of (5) must be reformulated as

$$\sum_{k=1}^{N} v_k \Delta t = D \tag{17}$$

which happens to be the product of two decision variables. As a consequence, a side effect of this formulation is the introduction of nonlinear constraint functions (the kinematic constraints (4) also become nonlinear). These nonlinearities have downstream implications on the selection of appropriate optimization algorithms for solving the move time minimization problem.

III. IMPLEMENTATION NOTES

The use of algebraic modeling languages makes trajectory optimization programs such as those presented much more accessible to practicing engineers with limited knowledge of optimization solvers and nomenclature. With that said that are two aspects to note when implementing these programs in an AML package. For this paper, the above problems were implemented using Pyomo, for which the range of math functions that are available for the solvers is limited. For example, norms are not supported so additional decision variables, constraints, and cost function changes were required to complete the problem formulation. These adaptation are described here. The optimization code used in this paper is provided on GitHub at `TBDsite`.

A. Absolute Norm ($\mathcal{L}_1$)

Introduce a new variable X_k for each element in $\mathbf{x}$. For each new variable, implement the two-sided constraint

$$-x_k \leq X_k \leq x_k \tag{18}$$

The cost function becomes

$$J = \sum_{k=1}^{N} X_k \tag{19}$$

Using these relationships in (5) results in the following optimization program

$$\begin{aligned} \min_{\mathbf{d},\mathbf{v},\mathbf{a},\mathbf{j},\mathbf{X}} \quad & \sum_{k=1}^{\infty} X_k \\ s.t. \quad & \mathcal{C} \\ & X_k \leq x_k \\ & X_k \geq -x_k \end{aligned} \tag{20}$$

where $\mathbf{x}$ may be $\mathbf{v}$, $\mathbf{a}$, $\mathbf{j}$, or $\mathbf{P}$.

B. RMS Norm ($\mathcal{L}_2$)

Pyomo does not allow use of the square root function so a straight minimization of an RMS value is not possible. It can be shown, however, that minimizing the sum of squares is equivalent to minimizing the RMS value. This results in the following optimization program

$$\begin{aligned} \min_{\mathbf{d},\mathbf{v},\mathbf{a},\mathbf{j}} \quad & \sum_{k=1}^{\infty} x_k^2 \\ s.t. \quad & \mathcal{C} \end{aligned} \tag{21}$$

where $\mathbf{x}$ may be $\mathbf{v}$, $\mathbf{a}$, $\mathbf{j}$, or $\mathbf{P}$.

C. Peak Norm ($\mathcal{L}_\infty$)

Introduce the same variables and constraints as in (18) as well as a new variable $\bar{X}$. Add the following constraint for each variable, X_k

$$X_k \leq \bar{X} \tag{22}$$

and the cost function becomes

$$J = \bar{X} \tag{23}$$

The resulting optimization program is

$$\begin{aligned} \min_{\mathbf{d},\mathbf{v},\mathbf{a},\mathbf{j},\bar{X}} \quad & \bar{X} \\ s.t. \quad & \mathcal{C} \\ & X_k \leq \bar{X} \\ & X_k \leq x_k \\ & X_k \geq -x_k \end{aligned} \tag{24}$$

where $\mathbf{x}$ may be $\mathbf{v}$, $\mathbf{a}$, $\mathbf{j}$, or $\mathbf{P}$.

D. Minimum Total Energy Formulation

To minimize the total energy of the move, (12) is used for the cost function. Since the mass m and time step Δt are considered constant here, this results in the following optimization program

$$\begin{aligned} \min_{\mathbf{v},\,\mathbf{a},\,\mathbf{j}} \quad & m\Delta t \mathbf{a}^T \mathbf{v} \\ s.t. \quad & \mathcal{C} \end{aligned} \tag{25}$$

IV. SINGLE-AXIS EXAMPLE AND RESULTS

To demonstrate the above trajectory optimization formulations, a realistic problem for a small single axis motion system is used. The set of system parameters and constraints are provided in Table I. Trajectory optimization results are shown for minimizing velocity, acceleration, jerk, power, and time. In the case of velocity minimization, two different bounds on the jerk were used to show the effects of the amount of jerk in the system. As explained previously, minimizing total energy is equivalent to minimizing the $\mathcal{L}_1$-norm of power, so this case is redundant. Peak power and energy usage minimization are compared between the different cases before exploring the relationship between the two in a multi-objective minimization problem. In all simulations the Generalized Disjunctive Programming optimizer was used (referenced as gdpopt within Pyomo [7]), although other optimizers also worked well on these problems.

TABLE I: System parameters

mass (m)	10 kg
Sampling frequency (Δt)	0.005 s
Final Time (t_f)	1 s
Maximum Velocity ($\bar{v}$)	0.5 m/s
Maximum Acceleration ($\bar{a}$)	4.905 m/s^2
Maximum Jerk ($\bar{j}$)	98.1 m/s^3
Maximum Power (P)	100 W

A. Minimal Velocity

Fig. 1 shows the minimum velocity trajectories for all norms studied using the nominal $\bar{j}$ and when $\bar{j}$ is 10 times higher than listed in Table I. For the nominal jerk allowable, the familiar S-curve trajectory results from the minimization of the $\mathcal{L}_2$ and $\mathcal{L}_\infty$ norms. As the allowable jerk increase, these curves morph into the familiar trapezoidal profile, with indistinguishable results when using the $\mathcal{L}_2$ and $\mathcal{L}_\infty$ norms. Another result of interest here is that the trajectory produced by minimizing $\|v\|_1$ uses less peak power.

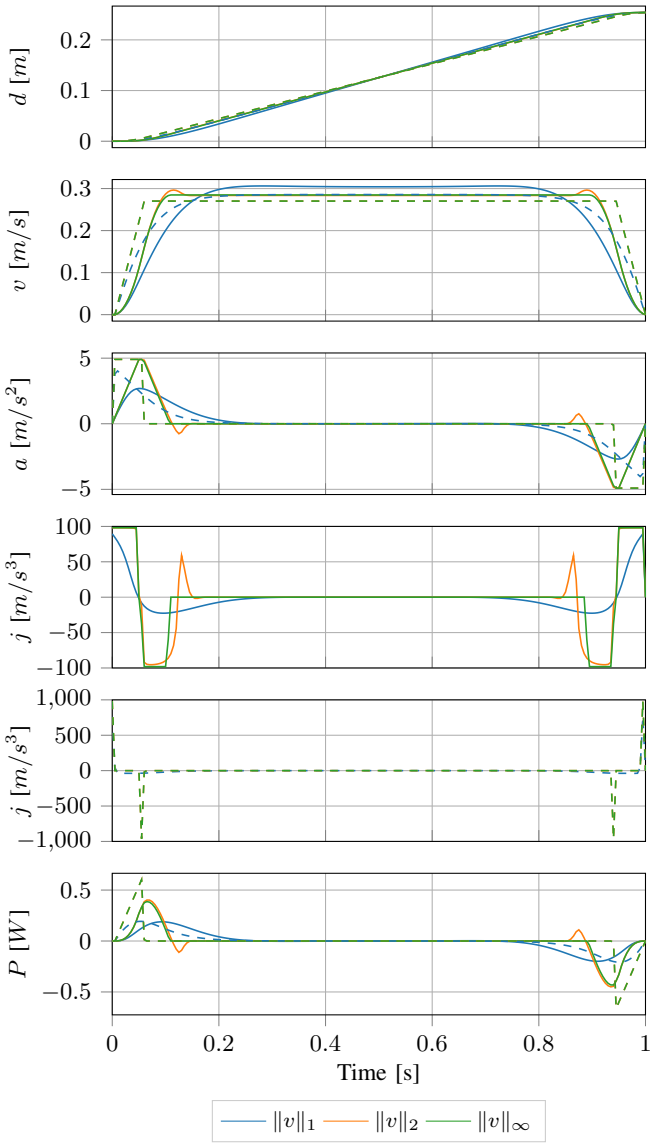

Fig. 1: Minimum velocity trajectories - nominal $\bar{j}$ (solid) and 10x $\bar{j}$ (dashed)

B. Minimal Acceleration and Jerk

Minimizing the acceleration across the 3 norms results in significantly different trajectory profiles. . The results shown in Fig. **??** demonstrate a parabolic velocity profile for minimization of the $\mathcal{L}_2$ norm and a triangular profile for minimization of the $\mathcal{L}_\infty$ norm. Minimization of the $\mathcal{L}_1$ norm of acceleration is similar to minimization of the $\mathcal{L}_\infty$ norm of velocity, which is expected from the time derivative relationship between these variables. Fig. **??** shows minimization of jerk using the three norms. In this the velocity profiles develop a range of bell shapes, nearing a triangle shape in the case of the $\mathcal{L}_1$ norm. of the system jerk and results in bell shaped velocity profiles.

C. Minimal Power

Minimizing power results (Fig. **??**) in velocity profiles that are largely similar to those of acceleration minimization, while the other profiles are accentuated at the beginning and end of the motion. Interestingly, the overall peak power using the $\mathcal{L}_1$ norm is higher than that of several of the other cases, indicating that this cost function is not appropriate for power minimization. Note, however, that the $\mathcal{L}_1$ norm this is also the minimum total energy solution, as was shown via (12).

D. Minimal Time

The minimal time trajectory results in an S-curve velocity profile. In separate tests not shown, it was found that as the final time was manually reduced for any of the previously described optimization scenarios, the results converged to this same minimum time S-curve solution. This result can potentially be used to justify the specification of a different actuator that can meet requirements but will also be able to lower to total energy used in the motion.

E. Total Energy and Peak Power Comparison

The generated trajectory profiles for the various optimization problems considered will naturally exhibit different total

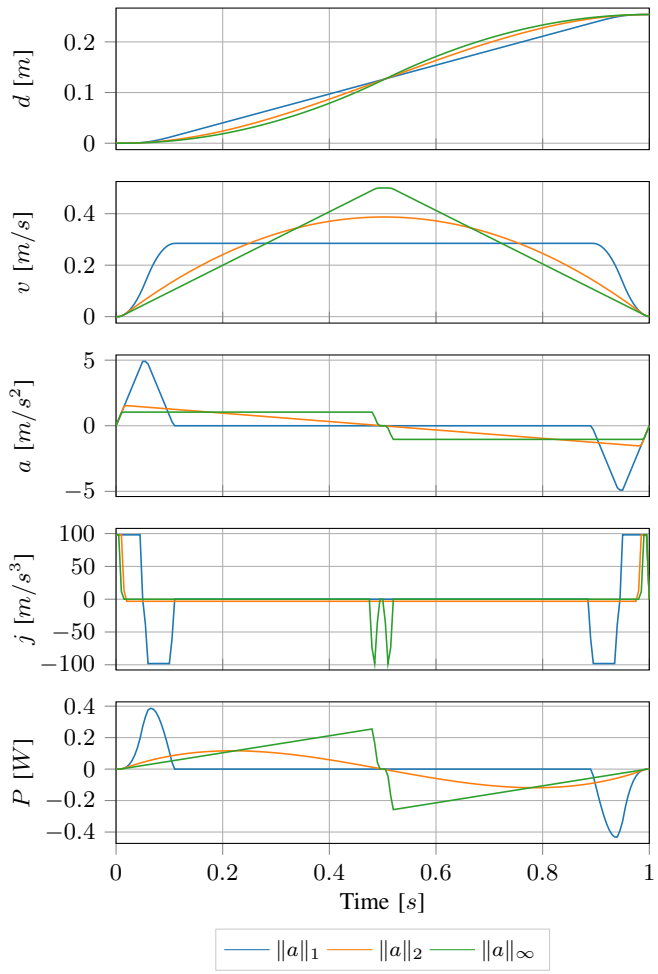

Fig. 2: Minimum acceleration trajectories

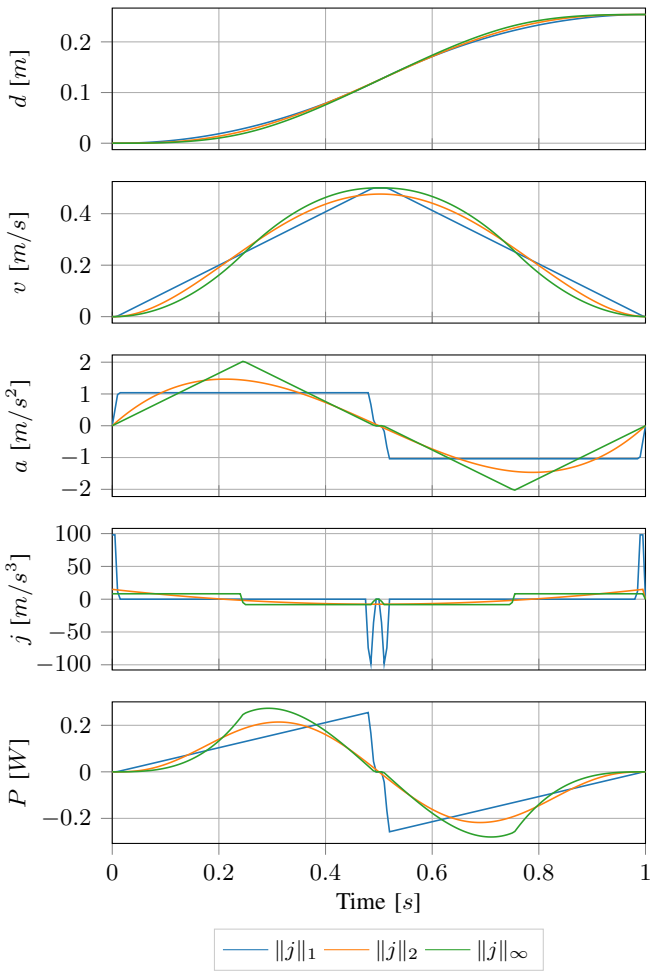

Fig. 3: Minimum jerk trajectories

TABLE II: Total energy relative to best case

	v	a	j	P	E_{total}	t_f
$\|\cdot\|_1$	1.17	1.00	3.08	1.00	1.00	3.08
$\|\cdot\|_2$	1.17	1.85	2.80	1.39		
$\|\cdot\|_\infty$	1.00	3.08	3.08	1.96		

TABLE III: Peak power relative to best case

	v	a	j	P	E_{total}	t_f
$\|\cdot\|_1$	2.37	5.13	3.06	5.13	5.13	11.2
$\|\cdot\|_2$	5.45	1.41	2.59	1.77		
$\|\cdot\|_\infty$	5.13	3.06	3.33	1.00		

energy and peak power consumption characteristics that may further aid in the selection of the proper trajectory. A relative comparison of these values for the above cases is provided in Tables II and III. The total energy was found via trapezoidal integration of the absolute value of the power. The results were divided by the best case over all combinations. It can be seen that, in general, minimizing the velocity resulted in the lowest energy required for the move, which makes sense as kinetic energy is proportional to the velocity squared. As expected per previous discussion, the minimization of the $\mathcal{L}_1$ norm of Power similarly achieves the minimum total energy mark. Minimal peak power, on the other hand, is generally reduced by minimizing the jerk or more directly so by minimizing the $\mathcal{L}_\infty$ norm of the power. Simultaneous comparison of these tables illustrates the well-known tradeoff that must be considered between system specifications and performance requirements.

F. Multi-Objective Trajectory Optimization

Extension of the trajectory optimization framework to mixed optimization problems can be easily handled by modifying the cost function in (5) with the weighted multi-objective formulation

$$\begin{aligned} &\min_{\mathbf{x}} \sum_i w_i J_i \\ s.t. \quad &\mathcal{C} \\ &\sum_i w_i = 1 \end{aligned} \tag{26}$$

where w_i are weights for each of the J_i cost functions. Equal weighting of total energy and peak power cost functions results in the motion profiles shown in Fig. 6, which exhibits a total energy 1.50 times greater and peak power 1.08 times greater than their respective optimal cases. Varying the cost function weights provides a means to extracting the multi-

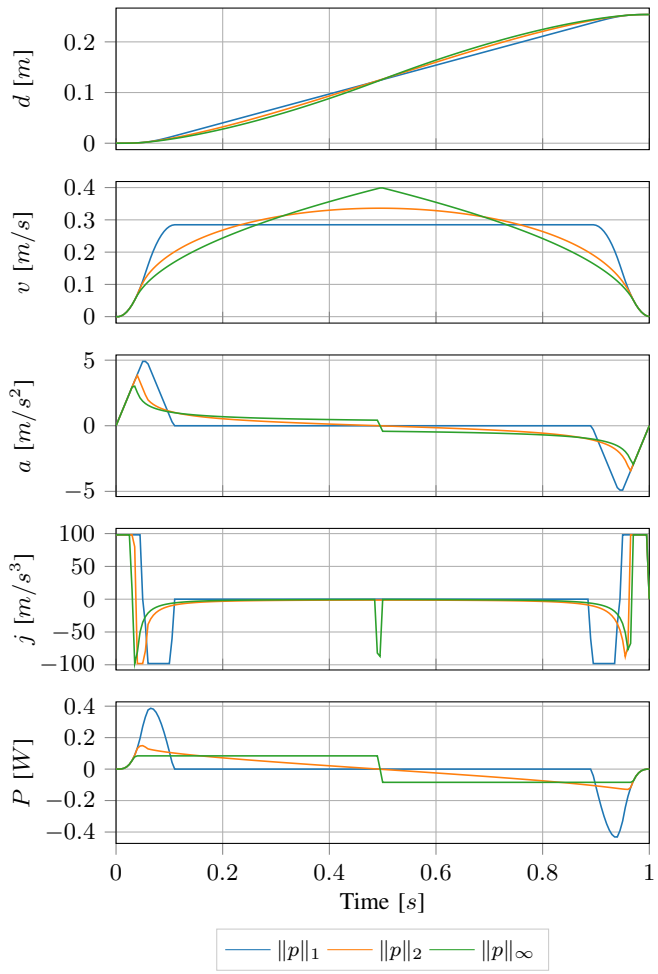

Fig. 4: Minimum power trajectories

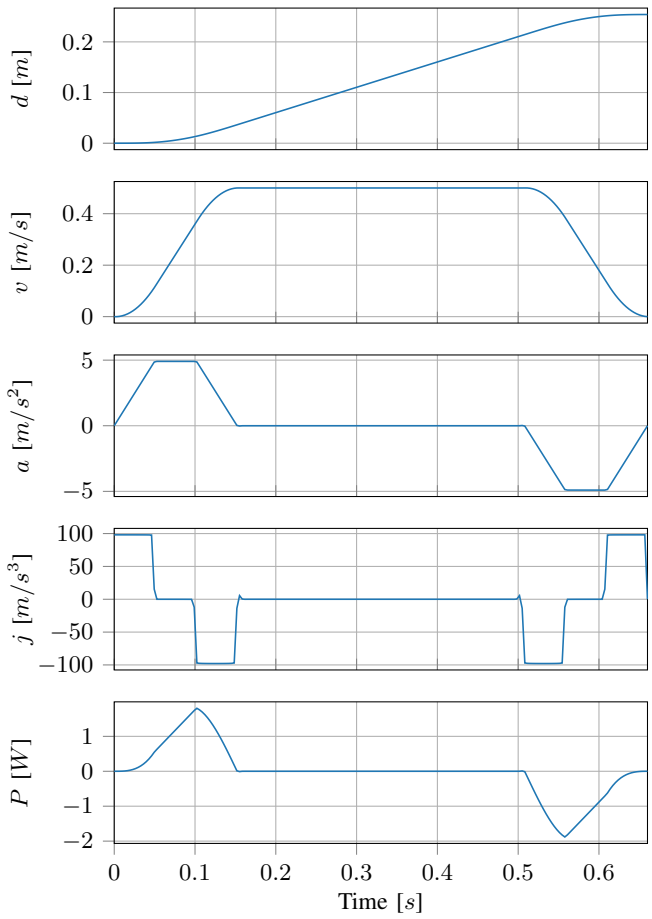

Fig. 5: Minimum time trajectory

objective Pareto front, as shown in Fig. 7 in which weights were varied from 0 to 1. Representing the range of possible solutions to this problem, at the extremes the Pareto front replicates the best and worst energy and power case results shown in Tables II and III. In between lies a trade space between the two cost functions where an engineer can find a motion profile that appropriately balances the needs of their particular system.

V. MULTI-AXIS TRAJECTORY OPTIMIZATION EXAMPLE

The trajectory optimization framework can be generalized to multi-axis problems by extension of the problem formulation (5) with new kinematic variables and associated kinematic constraints (4) for each moving body. Also necessary is the reformulation of the cost and objective function variables to include coupled effects of the system (e.g., staged motion and accompanying dynamics).

Here we demonstrate such an extension through and example of a two-link Selective Compliance Articulated Robotic Arm (SCARA). A SCARA consists of a fixed rotary joint rotating a link at the end of which another rotary joint/link pair is situated. In this example, it is clear that the kinematics of the endpoint of link 2, as well as the joint torque and power requirements, are coupled and dependent on the relative motion of the two joints. Derivation of these relationships is straightforward using Lagrange's equations and will not be reproduced here.

To exercise this example, we define a trajectory generation problem for the SCARA starting at rest such that the tip velocity $\left\langle v_x^{tip}, v_y^{tip} \right\rangle$ of the second link is equal to $\left\langle 0, v_f^{tip} \right\rangle$ at t_f (as if it were throwing a baseball). We choose to formulate the tip velocities as constraints while the cost function minimizes the equally weighted peak power ($\mathcal{L}_\infty$ norm) of the two joint actuators. In this case we only care about hitting the velocity target at the end of motion, so the final position and acceleration constraints d_N and a_N are removed from the problem formulation. Note that this does not dictate the position or pose of the SCARA at the end of motion, only the tip velocity, so there may be many ways in which the end condition may manifest. An alternative formulation to this problem could instead add new constraint equations to enforce the tip velocity at an intermediate point in time.

The solution of this trajectory generation problem is shown through the following figures. Fig. 8 shows the initial and final positions of the SCARA as well as traces of the tip

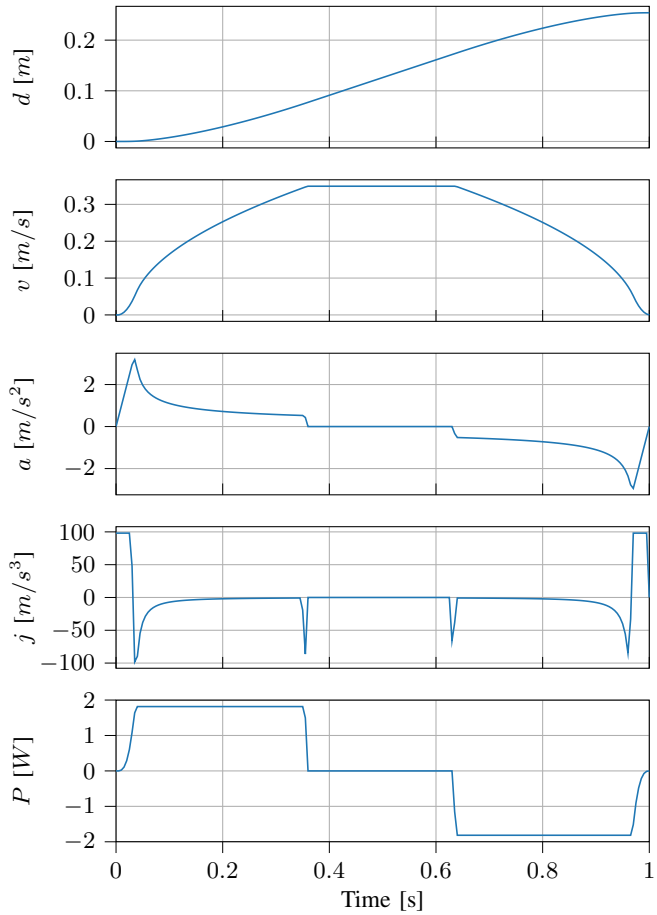

Fig. 6: Minimum mixed total energy-peak power objective trajectories – equal weighting

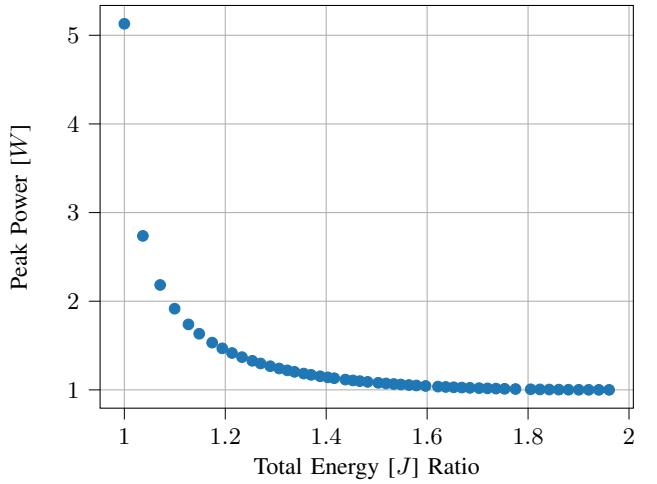

Fig. 7: Minimum total energy-peak power Pareto front – normalized to axes minima

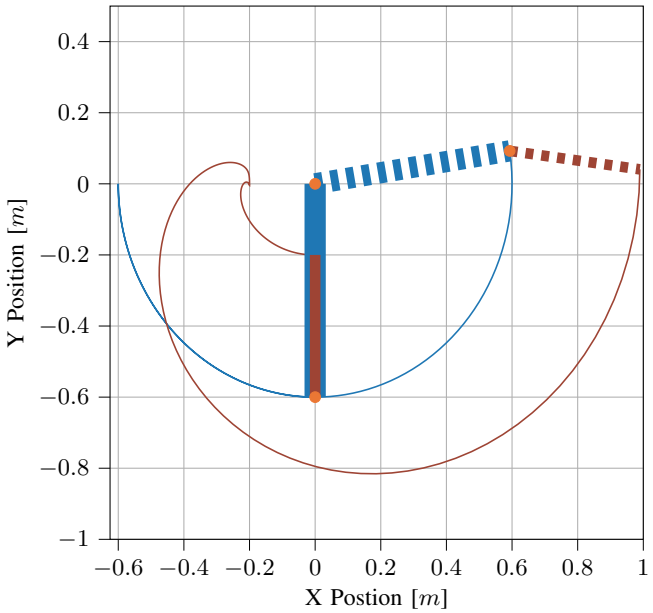

Fig. 8: SCARA in-plane motion with starting (solid) and ending (dashed) poses

positons, Fig. 9 shows the motion profiles of the two joint actuators, and Fig. 10 shows the trajectories the link tips. From the motion profiles it is clear that the position (limited to $\pm\pi$ radians) and jerk side constraints are being enforced at various points through the motion, while the peak power cost function manifests as a flattening of the actuator power curves. An interesting outcome of this example is that the optimizer plans "wind-up" motion for the arm (much like a pitcher throwing a baseball), which it turns out is required in order to reach the tip velocities within the given joint position constraints (i.e., it is not possible to reach the tip speeds if only forward motion is allowed). The optimization code used for this example, along with specific link dimensions and properties, is provided on GitHub at `TBDsite`. Readers are encouraged to explore this example to better understand the relationship between the problem formulation, variables, and constraints.

VI. CONCLUSIONS AND FUTURE WORK

This paper provides a straight forward analysis tool for optimizing servo movement trajectories that motion systems engineers can use to tailor the performance of their systems. Multiple objectives were proposed to minimize different aspects of the trajectory profile. Results showed that minimization of the $\mathcal{L}_2$ *and* $\mathcal{L}_\infty$-norms resulted in the traditional trapezoidal and S-curve profiles that are used in many off-the-shelf motion control systems. It was shown that trapezoidal profiles are the natural extension of S-curves when the allowable jerk in the system in increased. Also noted is that a trapezoidal profile is both the solution of the minimal time trajectory under motion rate constraints as well as for all other optimization problems considered when under a time constraint that approaches the minimal

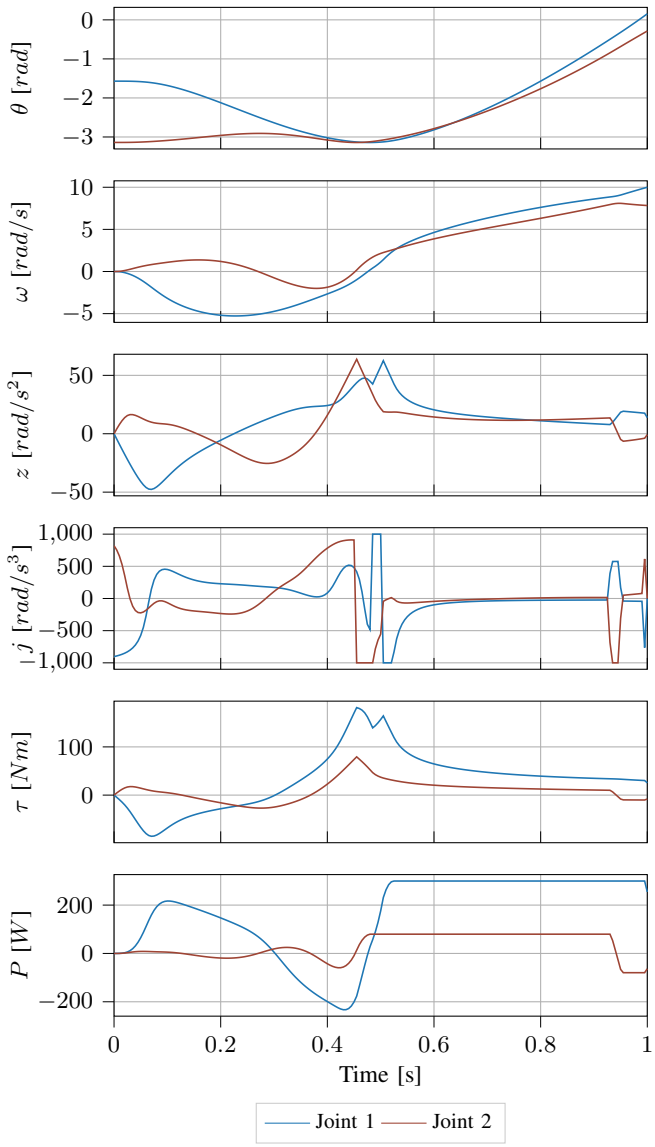

Fig. 9: SCARA motion profiles

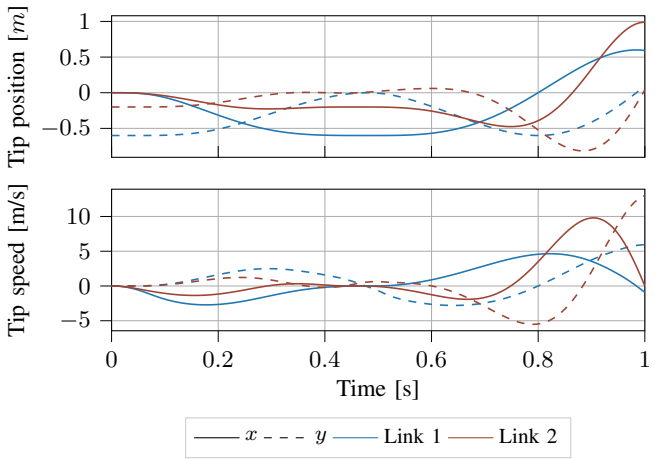

Fig. 10: SCARA tip trajectories

time. This suggests that the trapezoidal velocity profile is equivalent to the minimum move time profile given specific bounds. Further comparison of total energy and peak power between the results clearly demonstrate the trade off that must be considered with selection a motion profile.

A key distinction between the trajectory optimization formulation presented here and other optimization problem formulations of dynamic systems is the independence of the kinematic variables. Whereas in most other formulations the kinematics are constrained implicitly through the use of a time integration solver or algorithm (e.g., Runge-Kutta or forward Euler), here the kinematic variables are allowed to fall out of dynamic equilibrium within the context of the optimization program (i.e., the kinematic constraints may not be satisfied for a given optimization iteration). This unique feature affords the present formulation with additional freedom to explore the optimal solution landscape by not being required to strictly follow the laws of physics every step of the way. It is only within the scope of the final solution that physical laws must be upheld, so it can be envisioned that this relaxation may bring algorithmic benefits in terms of improved solution convergence rate and avoidance of local minima, although these suppositions are not investigated here.

There exist many extensions and/or other use cases of this work, including:

- Adding friction - Including friction within trajectory optimization will enable improved power and energy estimates as well as the ability to derive system friction requirements from system performance specifications.
- Multi-objective/norm regularization - Exploring other cost function formulations that allow for more tailored trajectories to balance the trade offs between power, energy, and actuator limits.
- Complex profiles - Creating trajectories with varying bounds across the trajectory, thereby enabling optimization of complex profiles with specific movement requirements.

REFERENCES

[1] T. Kröger, "Literature survey: Trajectory generation in and control of robotic systems," in *On-Line Trajectory Generation in Robotic Systems*. Springer, 2010, pp. 11–31.

[2] A. V. Rao, "Trajectory optimization: a survey," in *Optimization and optimal control in automotive systems*. Springer, 2014, pp. 3–21.

[3] K. Erkorkmaz, "Optimal trajectory generation and precision tracking control for multi-axis machines," Ph.D. dissertation, University of British Columbia, 2004.

[4] Y. Wang, Y. Zhao, S. A. Bortoff, and K. Ueda, "A real-time energy-optimal trajectory generation method for a servomotor system," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 2, pp. 1175–1188, 2014.

[5] W. E. Hart, C. D. Laird, J.-P. Watson, D. L. Woodruff, G. A. Hackebeil, B. L. Nicholson, and J. D. Siirola, *Pyomo - Optimization Modeling in Python*. Springer, 2017.

[6] D. C. Karnopp, D. L. Margolis, and R. C. Rosenberg, *System dynamics: modeling, simulation, and control of mechatronic systems*. John Wiley & Sons, 2012.

[7] Sandia National Laboratories, "Generalized disjunctive programming - pyomo documentation," [Accessed 29-January-2023]. [Online]. Available: https://pyomo.readthedocs.io/en/stable/modeling_extensions/gdp/index.html