Force-Controlled Pose Optimization and Trajectory Planning for Chained Stewart Platforms

Benjamin Beach [*], William Chapin [†], Samantha Glassner [1‡], Robert Hildebrand [§], and Erik Komendera [¶]

August 24, 2022

Abstract

We study optimization methods applied to minimizing forces for poses and movements of chained Stewart platforms (SPs) that we call an "Assembler" Robot. These chained SPs are parallel mechanisms that are stronger, stiffer, and more precise, on average, than their serial counterparts at the cost of a smaller range of motion. Linking these units in a series overcomes their individual limitations and yet maintains their trusslike rigidity, enabling their potential use for a variety of purposes. The assembler robot will be used in concert with several other types of robots to perform complex space missions. We develop algorithms and optimization models that can efficiently decide on favorable positions and movements that reduce forces loads on the robot and hence reducing wear on this machine. The objective of this research is to maneuver the interior plates of the Assembler such that the load forces distributed through the legs of each constituent SP are more even, allowing for a larger workspace and a greater overall payload capacity. The Assembler is designed to function concert with several other robots for a variety of space missions. Our computations focus on assemblers with four chained SPs, but our methods apply to an arbitrary number of SPs, and can be extended to general over-actuated truss-like robot architectures.

Keywords: Optimization, Nonlinear Programming, Robotics, Kinematics, Stewart Platform, Modular, Forces

[*]Grado Department of Industrial and Systems Engineering, Virginia Tech
[†]FASER Lab, Virginia Tech, Mechanical Engineering, Blacksburg, Virginia, wchapin@vt.edu
[‡]FASER Lab, Virginia Tech, Mechanical Engineering, Blacksburg, Virginia
[§]Grado Department of Industrial and Systems Engineering, Virginia Tech
[¶]FASER Lab, Virginia Tech, Mechanical Engineering, Blacksburg, Virginia

1 Introduction

Robotic space missions are complex, expensive endeavors, often resulting in multiple mission extensions or scope expansions in order to maximize the robotic system's potential over its useful lifespan. Ensuring that a system is precise and robust enough to accomplish its mission in addition to unknown future mission requirements drives up development and deployment cost significantly, especially due to one-off hardware design. The overall cost of system deployment can be driven down by mass production of smaller, modular robots that can join together to enhance their capabilities. Specifically, we seek to use parallel mechanisms called SPs, which are typically stiffer, more precise, and more robust to vibration than their serial counterparts of a similar quality, at the cost of a smaller range of motion [22]. Linking these units in a series overcomes their individual limitations and yet maintains their trusslike rigidity, enabling their potential use for a variety of purposes. We designate such a configuration as an "Assembler" robot. While the number of SPs in a stack is arbitrary, the experimentation discussed in this article concerns stacks of four. A stack of SPs has the property that it is over-actuated, which implies that there is a continuum of internal plate poses for any pair of end plate poses. This allows internal plate poses be chosen to satisfy constraints and optimize a variety of metrics. The objective of this research is to maneuver the interior plates of the Assembler such that the load forces distributed through the legs of each constituent SP are balanced, allowing for a larger workspace and a greater overall payload capacity.

We propose to use these chained SPs as an alternative to robot serial arms, for eventual use in fully automated robot assembly on the moon or in space. Each SP consists of a pair of plates with six legs in between. The structure is actuated simply by extending or contracting the linear actuator legs. An example of the structure is shown in Figure 1.

Stacks of SPs, like the Assembler, combine the strength and stability of truss-like SPs, with the reach that serially linked robots such as Universal Robotics UR-10 can provide without the mass inherent to large actuators responsible for driving serial mechanisms. The construction of a SP of the type presented in this paper - six linear actuators separating two parallel plates presents precision and extreme stiffness [20] as linear actuators can not be easily backdriven (and hold their positions with no power applied). Position holding lends itself to optional application of the SP-Stack as semi-permanent long-term reconfigurable support truss structures. The high stiffness also means higher fundamental frequency and lower oscillation amplitude (i.e. cantilevering less of an issue). The stacking of the individual platforms to make a larger robot extends the reach by multiplying the workspace configurations, yielding a 24 total DOF robot (or more, if necessary). This overactuation opens up optimization, as near-infinite internal configurations can yield the same end effector pose. In addition, the

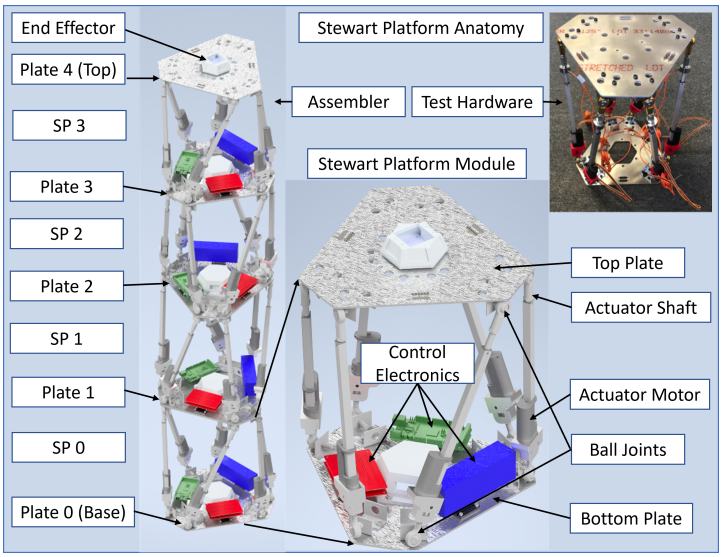


Figure 1: Anatomy of an Assembler Robot. The robot is comprised of four chained six degrees of freedom (DOF) SPs, giving it a larger range of motion and total 24 DOF. Each SP is comprised of six linear actuators mounted between two plates, since the stacked SPs share plates the Assembler has a total of 5 plates that serve as the upper and lower boundaries of each SP. The legs are connected via ball joints that are mounted below and above each plate. Thus, the leg connections do not lie in the same plane as the plates.

extreme overactuation present in the system allows for redundancy. If any individual actuator fails, the robot is capable of compensating for the failure, increasing the robustness of the system. The methods shown in this paper are applicable to a range of similarly configured truss-like robots, not limited to this particular 24DOF configuration. SP mechanisms are not perfect however, having poor passive compliance, limited velocity, and limited angular range of motion. The poor passive compliance can be compensated for with active compliance, but limited angular range of motion is a limiting factor which must be considered in choosing applications for this system, though it can be partially compensated for with a spherical wrist at the end effector, or a off-axis turntable system at the base.

We propose an optimization approach to reduce structural forces in poses and throughout motion. The complexity of the problem stems from the serial combination of parallel kinematic structures (each SP), which can render difficult the problem of even finding feasible poses for a given end effector position and load. From an optimization perspective, the primary source of complexity stems from nonlinearity and nonconvexity of the feasible space, resulting from the trigonometric functions required to model the physical kinematic transformations throughout the structure, along with the many other bilinear and quadratic terms. These include coordinate distance computations to determine leg lengths, wrench and force computations, and coordinate

transformations after computation of the transformation matrices.

There have been many works in the literature exploring design optimization of individual SPs, including [2, 6, 19, 29, 32, 34, 40]. These works typically optimize the design of a single SP with respect to parameters such as stiffness, manipulability, and accuracy. In particular, [42] designs and implements a variant of SP with passive control of leg forces. However, we found the literature on stacked SPs to be incredibly sparse, and were unable to find any prior works optimizing poses for stacked SP chains based on leg forces. There is a related field studying more general actuated truss structures, such as in [8, 24, 39, 41]. Among other structures, these works study variable geometry truss (VGT) structures, which are similar to SP's, except that each plate is replaced with a triangle of three actuated legs, and the legs between 'plates' are not actuated. [39, 41] also study a version of an SP where the plates are replaced with triangles of stiff legs. However, none of these works study the problem of controlling forces while maneuvering these devices.

For trajectory planning, state-of-the-art approaches include the random tree search-based method RRT [17] and variants RRT-Connect [16], RRT* [15], RRT*-SMART [14], among others. Other state-of-the-art approaches include CHOMP [43], TrajOpt [31], and ROMP [28]. A number of works have explored trajectory planning for individual SP's, including [7, 10, 11, 25, 26, 38]. Several such works, such as [7, 10], rely on variants of RRT. In one example, [7] even applies this method to an obstacle-avoiding trajectory planning problem for a 4-stack of SPs. [1] performs stiffness optimization on a 2-D version of the stacked platform structure. Aside from RRT variants, some obstacle-avoiding trajectory planning approaches applied primarily to simple structures such as serial arms are introduced in [27, 31, 43]. Prior works involving trust-region based approaches for trajectory planning, as used in this work, include [9, 13, 28] for serial arms, and [30, 33, 36] for more difficult robots involving closed kinematic chains.

We were unable to find any works in the literature performing any sort of force-controlled trajectory planning of stacked SPs in three dimensions, with very few performing any sort of trajectory planning for stacked SPs. Such optimization can significantly improve maximum leg forces throughout the structure, thereby increasing the workspace of an SP under large loads.

Contributions. We present an optimization model and approach to solve these complex kinematic optimization and trajectory planning problems. We are not aware of prior models for chained Stewart platforms, though for a single SP, [3] gives a dynamic model. We first introduce a near-instantaneous spline-based heuristic to find a (near-)feasible initial pose for the structure given a target end effector position and load. We then construct an optimized solution by modeling the structure as nonlinear program (NLP), then solving it locally using

IPOPT and modeling it with Pyomo. To obtain a good initial solution for the optimizer, we construct a simple closed-form initialization approach for which each SP has the same pose. The optimization process typically takes just over 1 second on average for a single pose, for an Assembler consisting of four chained SPs. We computationally demonstrate the effectiveness of the approach, and demonstrate the effectiveness of the method in improving the force valid range of motion of the assembler compared to the spline-based heuristic.

In Section 2 we describe the structure of the Assembler and its abilities in terms of movement and positions. We then introduce the forward and inverse kinematics problems for single SP's and the full Assembler, and mathematically define a kinematically valid SP. In Section 3.3, we propose a spline-based construction heuristic (SIK) to produce valid poses, and a simpler same-SP heuristic to produce starting solutions for the optimizer. In Section 3.4, we propose an optimization model and simple initial-solution scheme (OPT) to directly generate locally force-optimal poses. In Section 4.2, we propose a trust region approach for trajectory planning based on the optimization model. In Section 5.2, we compare the effectiveness of the SIK and OPT schemes to quickly generate workable poses for the Assembler. In Section 5.3, we compare the trust region trajectory planning scheme with a naive approach based on linearly interpolating leg lengths between the initial and target poses. Finally, in Section 5.4, we demonstrate the effectiveness of the optimizer in improving the achievable range of motion of the Assembler.

2 Assembler Robot, Notation, and Transformations

A *Stewart platform* consists of a pair of plates linked by six linear actuators, constituting a parallel mechanism with six degrees of freedom (DOF). The actuators for each platform are connected via ball joints on either end. We refer to the linear actuators as legs. The motor of each actuator is situated on the leg bottom (LB), while the shaft of the actuator is on the leg top (LT). We define the *pose* of a plate as its position and orientation in a given reference frame, and we characterize the pose of an Assembler via the poses of its plates in the Assembler's reference frame, also referred to as the *global reference frame*. That is, a pose is a list of plate positions $\mathbf{p}_i^{\mathcal{G},P}$ and rotations $\mathbf{r}_i^{\mathcal{G},P}$ for each plate i with $i = 0, \ldots, N_P$. See Figure 2(b). The pose of each SP can be expressed in terms of the pose of its top plate in the reference frame of its bottom plate. We primarily use this local 'SP-frame' to mathematically define a kinematically valid SP. See Figure 2(c) for examples of local reference frame variables.

Note that, crucially, a top plate pose is not uniquely defined by only the leg lengths [5, 18].

An important pose is the *resting pose*. We define this as the pose where all the legs are set to be 50% of their total possible length, as a heuristic estimate of the center of a SP's

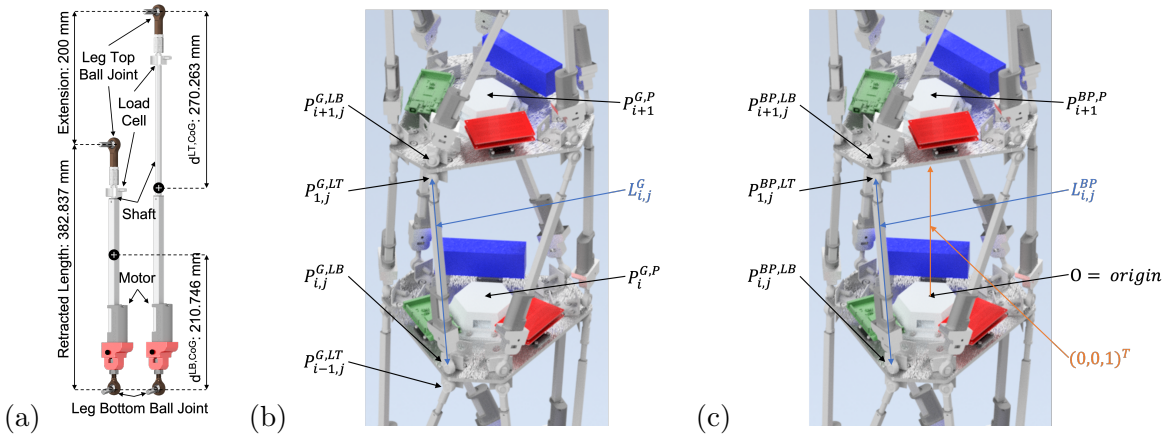


Figure 2: (a) Image of a linear actuator in two positions. The legs of the assembler are made from these linear actuators. Linear actuator has two parts: the motor (on the leg bottom) and the shaft (on the leg top). The actuator works by extending and contracting the shaft. For force computations, we record as data the distance from the bottom of the leg to the center of gravity of the motor, and from the top of the leg to the center of gravity of the shaft. (b) A section of the Assembler with several variable locations labeled in the global reference frame. Note that the center of gravity of plate i is $\mathbf{p}_i^{\mathcal{G},P}$. Variables in the global reference frame are denoted that a superscript $\mathcal{G}$. (c) The same pose, but rotated to look from the reference frame $\mathcal{BP}$ of SP i with respect to the bottom plate. We also occasionally use $\mathcal{TP}$ to denote the reference frame with respect to the top plate of the SP.

workspace. We define parameters $\boldsymbol{L}^{\text{rest}}$ for each leg as the vector that leg follows when in resting pose. Note that, in the resting pose, due to the symmetry of the SP design used here, each plate is translated vertically with no rotation, so that $\mathbf{r}_i = \mathbb{0}$ and $\mathbf{p}_{i,[1:2]} = \mathbb{0}$.

We use $\mathbb{0}$ to denote a vector or matrix of zeros, and we leave size of this vector/matrix to be deduced by context.

Given a goal position for the end effector of the Assembler, the goal of this work is to quickly find kinematically valid poses and motions for a given Assembler while controlling the worst-case forces on the legs. We consider only mass-related forces and external forces applied to the end-effector of the Assembler, and neglect any external forces applied to other parts of the Assembler by e.g. wind.

To resolve torques resulting from the masses of each leg, we must compute the center of gravity (CoG) for each part the leg. Crucially, the CoG for a leg is not in the center of each leg; rather, each leg is a linear actuator consisting of two connected parts, a motor and a shaft. See Figure 2(a). The CoG of each part is a fixed distance from its attachment location. This fixed-distance property gives the model for the CoG a measure of mathematical complexity, as one cannot simply multiply the vector defining the leg's position by a fixed number to obtain the position of the center of gravity for each part. Rather, the unit vector defining the direction of the leg is multiplied by the fixed distance from the attachment location of a part to its CoG. In this work, the motor of each leg is connected to the bottom plate of its SP, while the shaft is connected to the top plate.

In the remainder of this section, we first define our notation and mathematics for coordinate transformations. We then define the forward kinematics and inverse kinematics problems for an Assembler and for a single SP, define a kinematically valid SP, and give some reasoning for the use of four SPs for the Assembler in this work.

2.1 Notation

In this section, we introduce some useful notation and abbreviations to be used in the remainder of this work.

Shorthand: Here, we summarize our shorthand notation. TAA, SP, and KVC are used in text descriptions, while the rest are used in variable superscripts as object specifiers.

TAA : Translation-Axis-Angle
SP : Stewart Platform
KVC : Kinematic Validity Constraint
CoG : Center of Gravity
BP : Bottom Plate of SP
TP : Top Plate of SP
P : Plate
LB : Leg Bottom
LT : Leg Top

For parameter and variable definitions, we use the following notation,

$$VarType^{\text{RefFrame(Optional),DesiredObject,specifier}(Optional)}_{VarIndex,[ValIndex]}, \tag{1}$$

where RefFrame is the reference frame, $VarIndex$ gives indices for the related object, and $ValIndex$ gives vector and matrix indices. We implicitly define the reference frame of a plate by its leg attachment locations. The reference frame for each plate of an SP has its origin at the center of its exterior surface, the surface opposite its leg joints. In the Assembler stack, the top plate of each SP joins the bottom plate of the next at the plate centers, so that so that the two plates share the same origin, with the top SP rotated by 30 degrees. To simplify calculations, we treat the top bottom plate of a SP and the top plate of the preceding SP as a single plate, providing only a single reference frame. This is accomplished by rotating the leg attachment locations for the bottom SP by 30 degrees about the z-axis to obtain the leg attachment locations for the top SP.

Reference Frames These three reference frames are used throughout the paper.

$\mathcal{BP}$: 'SP' frame, the reference frame of the Bottom Plate of an SP
$\mathcal{TP}$: Reference frame of the Top Plate of an SP
$\mathcal{G}$: 'Global' assembler frame, the reference frame of the bottom plate of the bottom SP

Miscellaneous Notation We aggregate some useful miscellaneous notation here for reference.

$\mathbb{0}$ A vector or matrix of zeros.
$\mathbb{I}^3$ The 3×3 identity matrix.
$\diamond$ The operator to apply a coordinate transformation T to a point $\mathbf{p}$.
$\hat{\boldsymbol{e}}_k$ The standard unit vectors for $k = 1, 2, 3$, such that $\hat{\boldsymbol{e}}_{k,[k]} = 1$ and all other $\hat{\boldsymbol{e}}_{k,[j]} = 0$.

2.2 Coordinate Transformations

We use $\|\cdot\|$ to denote the 2-norm (Euclidean norm). For a vector $\boldsymbol{v}$, we use $\hat{\boldsymbol{v}}$ to denote $\boldsymbol{v}/\|\boldsymbol{v}\|$, that is, the unit direction of $\boldsymbol{v}$. We define $\hat{\boldsymbol{e}}_k \in \mathbb{R}^3$ for $k = 1, 2, 3$ as the standard unit vectors

with indices starting at 1. That is, $\hat{\boldsymbol{e}}_1 = [1,0,0]^\top$, $\hat{\boldsymbol{e}}_2 = [0,1,0]^\top$, and $\hat{\boldsymbol{e}}_3 = [0,0,1]^\top$.

We use $\boldsymbol{\Upsilon}$ to denote coordinate transformations via a vector containing the Translation-Axis-Angle representation (TAA)

$$\boldsymbol{\Upsilon} = \begin{pmatrix} \mathbf{p} \\ \mathbf{r} \end{pmatrix} = \begin{pmatrix} \mathbf{p} \\ \|\mathbf{r}\|\hat{\mathbf{r}} \end{pmatrix} = \begin{pmatrix} \mathbf{p} \\ \theta\hat{\mathbf{r}} \end{pmatrix}. \tag{2}$$

The TAA format is used to represent position $\mathbf{p}$ and orientation data $\mathbf{r}$ in place of transformation matrices because total translational and rotational errors can be found by taking the norm of the top 3 and the bottom 3 components, respectively.

The related rotation matrix can be defined via Rodrigues' formula as

$$\mathbf{R} := e^{[\mathbf{r}]} = I + \sin(\|\mathbf{r}\|)\frac{[\mathbf{r}]}{\|\mathbf{r}\|} + (1 - \cos(\|\mathbf{r}\|))\frac{([\mathbf{r}])^2}{\|\mathbf{r}\|^2}, \tag{3}$$

where $[\mathbf{r}]$ is the vector cross-product operator for $\mathbf{r}$

$$[\mathbf{r}] := \begin{bmatrix} 0 & -r_3 & r_2 \\ r_3 & 0 & -r_1 \\ -r_2 & r_1 & 0 \end{bmatrix}$$

so that, for any vector $\mathbf{v}$, we have $[\mathbf{r}]\mathbf{v} = \mathbf{r} \times \mathbf{v}$. We use the TAA scheme to represent reference frames in space due to its compactness, with the theoretically minimal six degrees of freedom, combined with the mathematical ease of converting the translational and rotational components to matrix transformation form. The matrix $\mathbf{R}$ is used to define transformation matrices via

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ \mathbb{0} & 1 \end{bmatrix}. \tag{4}$$

To apply the coordinate transformation defined by transformation matrix $\mathbf{T}$ to a point $\tilde{\mathbf{p}}$, we use

$$\mathbf{T} \diamond \tilde{\mathbf{p}} := \left(\mathbf{T} \begin{pmatrix} \tilde{\mathbf{p}} \\ 1 \end{pmatrix} \right)_{[1:3]} = \mathbf{R}\tilde{\mathbf{p}} + \mathbf{p}. \tag{5}$$

Where, for a vector $\boldsymbol{v}$, we use a bracketed MATLAB style subscript to denote components, for example, we define $\boldsymbol{v}_{[1:3]} = [v_1, v_2, v_3]^\top$.

2.3 Forward and Inverse Kinematics

There are two primary problems of interest to solve for a general robot with an end-effector: the forward kinematics (FK) and inverse kinematics (IK) problems. The FK problem is to

compute the end-effector pose given the actuator lengths. Conversely, the IK problem is to compute the actuator values given the end-effector pose.

For a single SP, the FK problem corresponds to computing the pose of the top plate w.r.t. the bottom plate given a vector of leg lengths. However, in general, the FK problem is difficult: it has multiple reachable solutions [5, 18], most of which cannot be reversed, such as 'pretzeled' poses for which the SP has twisted excessively, and spirals downwards until the legs collide. On the other hand, the IK problem corresponds to computing the leg lengths of the SP given the pose of the top plate w.r.t. the bottom plate. This computation is straightforward, with a simple closed-form solution, as described below. In this section, we describe all computations in the reference frame of the SP.

Define the pose of the top plate of an SP as $\mathbf{T}^{\mathrm{TP}}$, and let J be the set of legs for the SP. Then, for each leg of an SP, there are corresponding rest positions for the leg's joint attachment locations to the top and bottom plates. For a leg $j \in J$, these are $\mathbf{p}_j^{\mathcal{TP},\mathrm{LT,rest}}$ for the top plate and $\mathbf{p}_j^{\mathrm{LB}}$ for the bottom plate. These locations are defined in the reference frame of the corresponding plate, and are required for leg-vector and leg-length computations. We compute the coordinates of the top-plate leg joint locations via

$$\mathbf{p}_j^{\mathrm{LT}} = \mathbf{T}^{\mathrm{TP}} \diamond \mathbf{p}_j^{\mathcal{TP},\mathrm{LT,rest}} \qquad j \in J. \tag{6}$$

Note that the bottom-plate joint attachment locations $\mathbf{p}^{\mathrm{LB}}$ are known constants. Once joint locations have been determined, it is straightforward to calculate the resulting leg lengths by taking the magnitude of the positional displacement between corresponding leg joint pairs, as

$$\boldsymbol{L}_j^{\mathrm{len}} = \|\mathbf{p}_j^{\mathrm{LT}} - \mathbf{p}_j^{\mathrm{LB}}\| \quad j \in J. \tag{7}$$

The vector of leg lengths $\left(\boldsymbol{L}_j^{\mathrm{len}}\right)_{j\in J}$ is then the solution to the single-SP IK problem.

Considering that IK on an SP is a single step mathematical computation [21], it lends itself to rapid successive iteration. Solutions to the FK problem for SPs are based in this principle, with most algorithms performing calls to IK in order to numerically approximate the true position. However, as FK is not an integral component to the Assembler IK methodologies described in this paper, it is not described here in full. For further information, consult [23].

For an Assembler consisting of a serial chain of stacked SP's, which are parallel kinematic structures, neither the FK or IK problems are straightforward to solve. For the IK problem, there is generally a continuous space (or set of disjoint continuous spaces) of feasible plate positions for a given end-effector position. Moreover, this space is difficult to characterize, and can contain many bad solutions, such as those with extreme SP poses or very high leg forces. As such, before computing the leg lengths, one must first choose a 'good' solution for the plate

positions from the space of feasible solutions. On the other hand, to solve the FK problem for an Assembler, one must individually solve the difficult FK problem for each SP.

In this work, we focus on solving the IK problem for an Assembler. We then extend our approach via a trust region method to solve point-to-point trajectory planning.

2.4 Kinematic Validity Constraints

In this section, we define what it means for an SP to have a valid pose. In this section, we will define constraints in the reference frame of the base plate of SP, so that the origin is the center of the bottom of the bottom plate of the SP, with no rotation. In effect, we omit the reference frame superscript $\mathcal{BP}$, as defined in the following section.

Leg Length Bounds: The legs have minimum and maximum lengths that can be attained. For the robots we consider here, the legs all have uniform length bounds $L^{\min}$ and $L^{\max}$, but this is easily changed in our model if more general situations are of interest. Formally, for a particular leg, let $\mathbf{p}^{\mathrm{LT}}$ and $\mathbf{p}^{\mathrm{LB}}$ be the coordinates of the top and bottom connections of the leg. Then the vector $\boldsymbol{L} = \mathbf{p}^{\mathrm{LT}} - \mathbf{p}^{\mathrm{LB}}$ describes the direction and magnitude of the leg. This vector is then bounded in magnitude as

$$L^{\min} \leq \|\boldsymbol{L}\| \leq L^{\max}. \tag{8}$$

Note that, in the reference frame of the SP, the bottom-plate attachment locations $\mathbf{p}^{\mathrm{LB}}$ are known constants.

Leg Angular Deviation: Each leg must not exceed a certain angular deviation from its normal resting pose, as this could break the ball joints in physical hardware. We must constrain the leg lengths in both the top and bottom pose.

Formally, for a particular leg, let $\mathbf{p}^{\mathrm{LT,rest}}$ and $\mathbf{p}^{\mathrm{LB,rest}}$ be the rest coordinates of the top and bottom connections of the leg. Then the vector $\boldsymbol{L}^{\mathrm{rest}} = \mathbf{p}^{\mathrm{LT,rest}} - \mathbf{p}^{\mathrm{LB,rest}}$ describes the direction and magnitude of the leg in the resting pose.

The bottom-plate constraint for leg angle deviation is

$$\cos(\theta^{\max}) \leq \frac{\boldsymbol{L}}{\|\boldsymbol{L}\|} \cdot \frac{\boldsymbol{L}^{\mathrm{rest}}}{\|\boldsymbol{L}^{\mathrm{rest}}\|}. \tag{9}$$

Similarly, we require the same constraints for the top plate angles. Let $\mathbf{R}$ be the rotation matrix defining the orientation of the top plate w.r.t. the bottom plate. Then the top-plate angles are defined as in (9), except that the rest coordinates are pre-multiplied by $\mathbf{R}$ to move

them to the top plate, yielding

$$\cos(\theta^{\max}) \leq \frac{\boldsymbol{L}}{\|\boldsymbol{L}\|} \cdot \frac{\mathbf{R}\boldsymbol{L}^{\text{rest}}}{\|\mathbf{R}\boldsymbol{L}^{\text{rest}}\|}.$$

Since rotation operations are distance-invariant, we have $\|\mathbf{R}\boldsymbol{L}^{\text{rest}}\| = \|\boldsymbol{L}^{\text{rest}}\|$, yielding

$$\cos(\theta^{\max}) \leq \frac{\boldsymbol{L}}{\|\boldsymbol{L}\|} \cdot \frac{\mathbf{R}\boldsymbol{L}^{\text{rest}}}{\|\boldsymbol{L}^{\text{rest}}\|}. \tag{10}$$

Legs Point Up: To prevent legs from colliding with the base plate of an SP, we require that each leg is pointing 'up', so that the z-component $\boldsymbol{L}_{[3]}$ of $\boldsymbol{L}$ is nonnegative:

$$\boldsymbol{L}_{[3]} \geq 0. \tag{11}$$

Extreme Pose Prevention: We wish to prevent extreme and difficult-to-reach poses for each SP, particularly 'pretzeling,' a phenomenon where the top plate of an SP over-rotates in the z-direction, causing it to collapse, spinning down until the legs of the SP collide. See e.g. [5, 42] for more information on singularities, such as pretzeling, that can be encountered with some SP designs. To help prevent such extreme poses, we set a limit of $\theta^{\mathbf{R},\max}$ for the action of the plate rotation matrix $\mathbf{R}$ on any principle unit vector $\hat{\boldsymbol{e}}_k$, where $k \in \{1, 2, 3\}$. This corresponds to enforcing that each deviation angle $\theta_k^{\mathbf{R}} \leq \theta^{\mathbf{R},\max}$, which is equivalent to

$$\|\hat{\boldsymbol{e}}_k\|\|\mathbf{R}\hat{\boldsymbol{e}}_k\| \cos(\theta_k^{\mathbf{R}}) = \hat{\boldsymbol{e}}_k^\top \mathbf{R}\hat{\boldsymbol{e}}_k \geq \|\hat{\boldsymbol{e}}_k\|\|\mathbf{R}\hat{\boldsymbol{e}}_k\| \cos(\theta^{\mathbf{R},\max}).$$

Now, since $\|\hat{\boldsymbol{e}}_k\| = \|\mathbf{R}\hat{\boldsymbol{e}}_k\| = 1$, and $\hat{\boldsymbol{e}}_k^\top \mathbf{R}\hat{\boldsymbol{e}}_k = \mathbf{R}_{[k,k]}$, the kth diagonal element of $\mathbf{R}$, this simplifies to

$$\mathbf{R}_{[k,k]} \geq \cos(\theta^{\mathbf{R},\max}). \tag{12}$$

In this work, we use a maximum rotation of 60 degrees, or in radians, $\theta^{\mathbf{R},\max} = \frac{\pi}{3}$.

2.5 Problem Difficulty

For most cases, if forces are ignored, a feasible IK solution for the Assembler can be found very quickly. For example, with N=4, and using the SP parameters and computers specified in the numerical results section, IPOPT will typically converge (or report a locally infeasible solution) within 0.1s, so that a feasible solution to reach goal poses in the workspace can usually be obtained via local optimization from several different initial solutions.

However, finding an globally optimal solution to the IK problem, in terms of e.g. minimizing the maximal leg forces, is far more difficult in general. Due to the nonconvexity of both the

objective and constraints, there are often multiple locally optimal solutions. Moreover, these locally optimal solutions can differ greatly in solution quality, as seen in Figure 3. When attempting a direct global solve of our QCQP model in Section 3.4 with Gurobi 9.1.1, applied to the Assembler with parameters defined in Section 5, even directly optimizing an Assembler with only 2 SP's requires an inordinate amount of computational time (over an hour), despite the fact that the original problem has only six DOF, the pose of the middle plate, as the poses of the top and bottom plates are fixed.

Moreover, to demonstrate the potentially high number of locally optimal solutions, we optimized the 2-SP Assembler with a goal pose of $\mathbf{p} = [0, 0, 0.65]^\top$ and $\mathbf{r} = [0, 0, 0]^\top$, so that a vertical pose is impossible due to the leg length lower-bounds. Using 100 randomized starting poses for the middle plate drawn from a normal distribution, and minimizing the maximal leg forces via IPOPT, we obtained 22 different locally optimal solutions, with maximum leg forces ranging from 457N-519N.

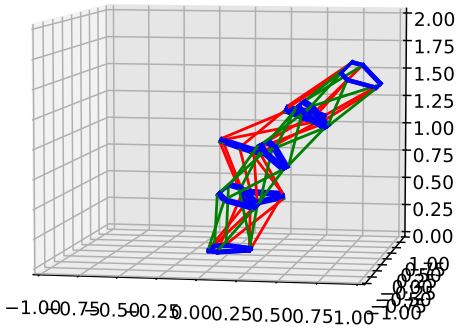


Figure 3: An example of a goal pose with two very different locally optimal solutions in IPOPT, for the Assembler defined in Section 5 with a 5kg weight on the top plate. The red pose has a maximum leg force of 703N, while the green pose has a maximum leg force of only 282N. Axes units are in meters.

2.6 Justification of Four SPs in a Stack

We choose the number of chained SP's for an Assembler so that it can reach a 'bent-over' pose, with the end effector is in-plane with the bottom plate, facing downwards. This enables the assembler roughly a hemisphere of motion, allowing a reasonably large workspace while keeping internal forces under control. While adding additional SP's would increase the kinematic flexibility of motion in a zero-gravity environment, on Earth, it would also increase the loads

on the legs particularly for the bottom SP, thereby shrinking the workspace of the Assembler due to excessive forces.

For the specifications of the SP used in this work, due to leg length and joint motion limitations, a minimum of four chained SP's are required to reach this bent-over pose. Thus, we use four chained SP's for the computational tests in this work.

3 Assembler IK

3.1 Definitions

In this section, we formally define the notation used for variables and parameters needed for this work.

Sets

$i \in I \quad := 0, 1, \ldots, N_P$: The plates.

$i \in I^{\mathrm{P}} \quad := 0, 1, \ldots, N_P - 1$: The SPs. Platform i connects plates i and $i+1$.

$j \in J \quad := 1, 2, \ldots, 6$: The legs for each SP.

Parameters

To reduce notation definitions, we implicitly define some coordinate variables $\mathbf{p}$, rotational variables $\mathbf{r}$, and matrix variables $\mathbf{R}$ and $\mathbf{T}$ from $\boldsymbol{\Upsilon}$ via (2).

Parameter	Size	Description
$\boldsymbol{\Upsilon}^{\mathcal{G},\mathrm{goal}}$	$\mathbb{R}^6$	: Target global-frame TAA pose for the end effector
$\boldsymbol{\Upsilon}_i^{\mathcal{BP},\mathrm{TP},\mathrm{rest}}$	$\mathbb{R}^6$	: SP-frame rest pose of the top plate for SP $i \in I^P$
$\mathbf{p}_{i,j}^{\mathcal{BP},\mathrm{LB}}$	$\mathbb{R}^3$	: SP-frame position of bottom joint $j \in J$ for SP $i \in I^P$
$\mathbf{p}_{i,j}^{\mathcal{TP},\mathrm{LT},\mathrm{rest}}$	$\mathbb{R}^3$	: Rest position of top joint $j \in J$ w.r.t. top plate for SP $i \in I^P$
$\boldsymbol{L}_{i,j}^{\mathcal{BP},\mathrm{rest}}$		$:= \mathbf{p}_{i,j}^{\mathcal{BP},\mathrm{LT},\mathrm{rest}} - \mathbf{p}_{i,j}^{\mathcal{BP},\mathrm{LB}}$: SP-frame rest-position leg vector for $j \in J$ for SP $i \in I^P$
$\theta^{\max}$	$\mathbb{R}$	: Maximal angle deviation from rest position for any leg
$\theta^{\mathbf{R},\max}$	$\mathbb{R}$	: Maximal angle between $\mathbf{R}_i^{\mathrm{BP},\mathrm{TP}}\hat{\boldsymbol{e}}_k$ and $\hat{\boldsymbol{e}}_k$ for plate $i \in I^P$ and $k \in [1:3]$
$(L^{\min}, L^{\max})$	$\mathbb{R}$	: Bounds on the length of any leg
$f^{\max}$	$\mathbb{R}$	: Upper bound on the compressive and tensile force on a leg
m_i^{P}	$\mathbb{R}$	: Mass of plate $i \in I$
m^{LT}	$\mathbb{R}$	: Mass of a leg motor
m^{LB}	$\mathbb{R}$	: Mass of a leg shaft
$d^{\mathrm{LT},\mathrm{CoG}}$	$\mathbb{R}$	: Distance from the top joint to the CoG for a leg shaft
$d^{\mathrm{LB},\mathrm{CoG}}$	$\mathbb{R}$	: Distance from the bottom joint to the CoG for a leg motor
g	$\mathbb{R}$	: Gravitational constant. For this work, we use Earth gravity, $g \approx 9.81 \frac{m}{s^2}$.

Note that, as the Assembler studied in this work consists of N_P identical stacked SPs,

there are really 2 plates between consecutive sets of legs, so that $m_i^P = 2m_0^P = 2m_{N_P}^P$ for $i = 1, 2, \ldots, N_P - 1$.

Variables: We color variables in blue to help readability of formulas.

Variable	Size	Description
$\boldsymbol{\Upsilon}_i^{\mathcal{BP},\mathrm{TP}}$	$\mathbb{R}^6$	: SP-frame pose of top plate for SP $i \in I^P$
$\boldsymbol{\Upsilon}_i^{\mathcal{G},\mathrm{P}}$	$\mathbb{R}^6$	: Global-frame pose of plate $i \in I$, where $\boldsymbol{\Upsilon}_{N_P}^{\mathcal{G},\mathrm{P}} = \boldsymbol{\Upsilon}^{\mathcal{G},\mathrm{goal}}$
$\mathbf{p}_{i,j}^{\mathcal{BP},\mathrm{LT}}$	$\mathbb{R}^3$	: SP-frame position of top joint $j \in J$ for SP $i \in I^P$
$\mathbf{p}_{i,j}^{\mathcal{G},\mathrm{LB}}$	$\mathbb{R}^3$	: Global-frame position of bottom joint $j \in J$ for SP $i \in I^P$
$\mathbf{p}_{i,j}^{\mathcal{G},\mathrm{LT}}$	$\mathbb{R}^3$	: Global-frame position of top joint $j \in J$ for SP $i \in I^P$
$\mathbf{p}_{i,j}^{\mathcal{G},\mathrm{LB,CoG}}$	$\mathbb{R}^3$	: Global-frame CoG position of motor for leg $j \in J$ for SP $i \in I^P$
$\mathbf{p}_{i,j}^{\mathcal{G},\mathrm{LT,CoG}}$	$\mathbb{R}^3$	: Global-frame CoG position of shaft for leg $j \in J$ for SP $i \in I^P$
$\boldsymbol{L}_{i,j}^{\mathcal{G}}$		$:= \mathbf{p}_{i,j}^{\mathcal{G},\mathrm{LT}} - \mathbf{p}_{i,j}^{\mathcal{G},\mathrm{LB}}$; Global-frame leg vector for leg $j \in J$, SP $i \in I^P$
$\boldsymbol{L}_{i,j}^{\mathcal{BP}}$		$:= \mathbf{p}_{i,j}^{\mathcal{BP},\mathrm{LT}} - \mathbf{p}_{i,j}^{\mathcal{BP},\mathrm{LB}}$; SP-frame leg vector for leg $j \in J$, SP $i \in I^P$
$\boldsymbol{L}_{i,j}^{\mathrm{len}}$		$:= \|\boldsymbol{L}^{\mathcal{BP}}\|$; Leg length for leg $j \in J$, SP $i \in I^P$
$\tau_{i,j}$	$\mathbb{R}$	: Gravitational force on leg $j \in J$ for SP $i \in I^P$
$\tau^{\max}$	$\mathbb{R}$	: Maximum force on any leg of the assembler
W_i	$\mathbb{R}^6$	: Global-frame wrench forces on the top plate of SP $i \in I^P$
		$W_i = [W_i^R, W_i^P]^\top$, where W_i^R relates to the torque and W_i^P is the force.
${J^{\mathrm{s}}}_i$	$\mathbb{R}^{6\times 6}$	: Spatial Jacobian related to leg-force computations for SP $i \in I^P$
${J^{\mathrm{t}}}_i$		$:= ({J^{\mathrm{s}}}_i)^{-\top}$

We assume that the Assembler is oriented so that gravity pulls directly downward in the reference frame of the Assembler, i.e. $\boldsymbol{g} = [0, 0, -g]^\top$. To handle different orientations of the Assembler, one needs only to redefine the gravity vector $\boldsymbol{g}$. Note that each $\boldsymbol{\Upsilon}$ transformation term implicitly defines corresponding TAA-form terms $\mathbf{p}$ and $\mathbf{r}$, and matrix-form terms $\mathbf{R}$ and $\mathbf{T}$. Finally, as the bottom plate is positioned at the origin of the global reference frame, we enforce that $\boldsymbol{\Upsilon}_0^{\mathcal{G},\mathrm{P}} = \boldsymbol{\Upsilon}_0^{\mathcal{G},\mathrm{P,rest}} = \mathbb{0}$.

In order to define a kinematically valid Assembler pose, we enforce the SP-frame kinematic validity constraints constraints in Section 2.4 for each SP, combined with the definition $\boldsymbol{\Upsilon}_{N_P}^{\mathcal{G},\mathrm{P}} = \boldsymbol{\Upsilon}^{\mathcal{G},\mathrm{goal}}$, which ensures that the end-effector is where it should be.

This constraint is enforced with a small implicit tolerance within the nonlinear solver.

3.2 Force Calculation

Force analysis follows the procedures set out in [21] with a few additional considerations.

For SP $i \in I^P$, we can define the Jacobian with the relationship

$$J^{\mathrm{s}-1}_i = \begin{bmatrix} \mathbf{p}^{\mathcal{G},\mathrm{LB}}_{i,1} \times \hat{\boldsymbol{L}}^{\mathcal{G}}_{i,1} & \dots & \mathbf{p}^{\mathcal{G},\mathrm{LB}}_{i,6} \times \hat{\boldsymbol{L}}^{\mathcal{G}}_{i,6} \\ \hat{\boldsymbol{L}}^{\mathcal{G}}_{i,1} & \dots & \hat{\boldsymbol{L}}^{\mathcal{G}}_{i,6} \end{bmatrix}^\top \tag{13}$$

where $\hat{\boldsymbol{L}}^{\mathcal{G}}_{i,j}$ is the unit vector for $\boldsymbol{L}^{\mathcal{G}}_{i,j}$.

Given a wrench W defined in the global frame and acting on the SP end effector, the resultant forces on each of the SP's legs can be determined by the relation:

$$\boldsymbol{\tau}_{i,[1:6]} = ({J^{\mathrm{s}}}_i)^\top (\mathrm{Adj}(\mathbf{T}^{\mathcal{G},\mathrm{TP}}_i))^\top W_i \tag{14}$$

where, as in [21], we define the matrix adjoint for a transformation matrix $\mathbf{T}$ (see (4)) as

$$\mathrm{Adj}(\mathbf{T}) := \begin{bmatrix} \mathbf{R} & \mathbb{0} \\ [\mathbf{p}]\mathbf{R} & \mathbf{R} \end{bmatrix} \tag{15}$$

For wrench computations, we first define the center of gravity for the motor and shaft of each leg $j \in J$ for SP $i \in I^P$ as

$$\mathbf{p}^{\mathcal{G},\mathrm{LB,CoG}}_{i,j} = d^{\mathrm{LB,CoG}} \frac{\boldsymbol{L}^{\mathcal{G},\mathrm{LT}}_{i,j}}{\|\boldsymbol{L}^{\mathcal{G},\mathrm{LT}}_{i,j}\|} + \mathbf{p}^{\mathcal{G},\mathrm{LB}}_{i,j} \quad \text{and} \quad \mathbf{p}^{\mathcal{G},\mathrm{LT,CoG}}_{i,j} = d^{\mathrm{LT,CoG}} \frac{-\boldsymbol{L}^{\mathcal{G},\mathrm{LT}}_{i,j}}{\|\boldsymbol{L}^{\mathcal{G},\mathrm{LT}}_{i,j}\|} + \mathbf{p}^{\mathcal{G},\mathrm{LT}}_{i,j}. \tag{16}$$

Then, to compute the global-frame wrenches W_i for each SP, we sum all the wrenches from forces applied on or above the platform by leg masses, platform masses, and the end effector load. For $i = 0, \dots, N_P - 2$, this is computed as

$$W_i = W_{i+1} + m^P_{i+1} \begin{bmatrix} [\mathbf{p}^{\mathcal{G},\mathrm{P}}_{i+1}]\mathbf{g} \\ \mathbf{g} \end{bmatrix} + \sum_j \left(m^{\mathrm{LT}} \begin{bmatrix} [\mathbf{p}^{\mathcal{G},\mathrm{LT,CoG}}_{i+1,j}]\mathbf{g} \\ \mathbf{g} \end{bmatrix} + m^{\mathrm{LB}} \begin{bmatrix} [\mathbf{p}^{\mathcal{G},\mathrm{LB,CoG}}_{i+1,j}]\mathbf{g} \\ \mathbf{g} \end{bmatrix} \right). \tag{17}$$

For the last platform $i = N_P - 1$, this is computed as

$$W_{N_P-1} = W^{\mathrm{EE}} + m^P_{N_P} \begin{bmatrix} [\mathbf{p}^{\mathcal{G},T}_{N_P}]\mathbf{g} \\ \mathbf{g} \end{bmatrix}. \tag{18}$$

Note that, as $\mathbf{g} = (0, 0, -g)^\top$, for any vector $\mathbf{p} \in \mathbb{R}^3$, we have

$$[\mathbf{p}]\mathbf{g} = g \begin{bmatrix} -\mathbf{p}_{[1]} \\ \mathbf{p}_{[2]} \\ 0 \end{bmatrix}.$$

Consequently, as W^{EE} is a parameter, and since the 3rd component of $[\mathbf{p}]\boldsymbol{g}$ is zero, the 3rd-6th components of the wrench are known constants, while the first two depend linearly on the plate and leg locations.

3.3 IK Heuristics

In this section, we introduce two heuristics for the initialization of SP poses.

3.3.1 Spline-Based IK

In this section, we introduce a splined kinematic approach, denoting a cubic spline originating at the platform base plate and ending at the end effector position. Define the following positions as helper points $\mathbf{p}^{\mathcal{G},1}$ and $\mathbf{p}^{\mathcal{G},2}$ from the formulas:

$$\mathbf{T}^{\mathcal{G},1} := \mathbf{T}^{\mathrm{goal}}\left(-\tfrac{2}{3}\mathbf{T}_0^{\mathcal{TP},\mathrm{BP,rest}}\right), \quad \mathbf{T}^{\mathcal{G},2} := \mathbf{T}^{\mathrm{Base}}\left(\tfrac{2}{3}\mathbf{T}_{N_P}^{\mathcal{TP},\mathrm{BP,rest}}\right). \tag{19}$$

Define the B-spline $p\colon [0,1] \to \mathbb{R}^3$ as

$$p(x) := \mathrm{Spline}\left(\mathbf{p}^{\mathcal{G},\mathrm{Base}}, \mathbf{p}^{\mathcal{G},1}, \mathbf{p}^{\mathcal{G},2}, \mathbf{p}^{\mathcal{G},\mathrm{goal}}\right). \tag{20}$$

We use SciPy's B-spline interpolation [35], which requires a minimum of four points to produce the spline curve, these two helper points serve a dual purpose: ensure that spline function has enough input to produce the expected curve, and also to ensure that interior plates are placed behind the plane of the goal end effector, and above the plane of the base plate.

The resulting spline function provides the positions of the interior $N_P - 1$ plates via interpolation at $\mathbf{p}_i^{\mathcal{G},\mathrm{P}} = p\left(\frac{i}{N_P}\right)$, for $i = 0, 1, \ldots, N_P$. We then recursively determine the rotation of the middle-most plate, and when there are an even number of plates left in the queue, we consider the two middle-most plates. The rotation of the middle-most plate(s) is then determined by an average of the rotation between the beginning and end plates, computed in TAA form in the reference frame of the beginning plate. Once the middle plate(s) rotation is determined, we recurse and determine rotation of the next middle plate(s).

Within this heuristic, we consider a pose to be valid if all kinematic validity constraints, including the end-effector position, are satisfied within some small tolerance. If a pose fails kinematically, we (try the recourse stuff depending on what failed). In the event of failure (such as a calculated leg being too long), all legs are re-scaled such that the legs are within bounds, maintaining orientation, and the end effector position is recalculated accordingly.

Validation of the SP proceeds in steps, described in the following algorithm.

3.3.2 Same-SP Initialization

We derive a simple initial guess for the Assembler pose by assuming that each SP has the same SP-frame pose $\boldsymbol{\Upsilon}_i^{\mathcal{BP},\mathrm{TP}} = \boldsymbol{\Upsilon}$. It is then trivial to compute the SP-frame rotation matrices, as since all share the same axis of rotation, we have for two rotations $\mathbf{r}_i^{BP,TP}$ and $\mathbf{r}_j^{BP,TP}$ that

$$\mathbf{R}_i^{\mathrm{BP,TP}}\mathbf{R}_j^{\mathrm{BP,TP}} = e^{[\mathbf{r}_i^{BP,TP}]}e^{[\mathbf{r}_j^{BP,TP}]} = e^{[\mathbf{r}_i^{BP,TP}+\mathbf{r}_j^{BP,TP}]}.$$

Thus, if $\mathbf{R}_i^{\mathrm{BP,TP}} = \mathbf{R}$ for all $i \in I^P$, we obtain

$$\mathbf{R}_{NP}^{\mathcal{G},\mathrm{P}} = e^{[\mathbf{r}_{N_P}^{\mathcal{G},P}]} = e^{[N_P\mathbf{r}]} = \mathbf{R}^{NP}.$$

Thus, we simply compute $\mathbf{r}$ as

$$\mathbf{r} = \tfrac{1}{N_P}\mathbf{r}_{N_P}^{\mathcal{G},P}.$$

Next, to compute the shared translation vector $\mathbf{p}$, we first note that

$$\mathbf{R}_i^{\mathcal{G},\mathrm{P}} = \mathbf{R}^i$$

and

$$\begin{aligned}
\mathbf{p}_1^{\mathcal{G},\mathrm{P}} &= \mathbf{p} \\
\mathbf{p}_2^{\mathcal{G},\mathrm{P}} &= \mathbf{p}_1^{\mathcal{G},\mathrm{P}} + \mathbf{R}_1^{\mathcal{G},\mathrm{P}}\mathbf{p} = (\mathbb{I}^3 + \mathbf{R})\mathbf{p} \\
\mathbf{p}_3^{\mathcal{G},\mathrm{P}} &= \mathbf{p}_2^{\mathcal{G},\mathrm{P}} + \mathbf{R}_2^{\mathcal{G},\mathrm{P}}\mathbf{p} = (\mathbb{I}^3 + \mathbf{R} + \mathbf{R}^2)\mathbf{p} \\
\mathbf{p}_i^{\mathcal{G},\mathrm{P}} &= \left(\sum_{l=0}^{i-1}\mathbf{R}^l\right)\mathbf{p} \qquad\qquad i \in I,
\end{aligned}$$

where $\mathbb{I}^3$ is the 3×3 identity matrix, and notable that $\mathbf{R}^0 = \mathbb{I}^3$ for any invertible 3×3 matrix $\mathbf{R}$. We then obtain $\mathbf{p}$ via a single cheap linear solve. To summarize, we compute $\mathbf{r}$ and $\mathbf{p}$, and the resulting $\mathbf{R}_i^{\mathcal{G},\mathrm{P}}$ and $\mathbf{p}_i^{\mathcal{G},\mathrm{P}}$'s, via

$$\begin{aligned}
\mathbf{r} &= \tfrac{1}{N_P}\mathbf{r}_{N_P}^{\mathcal{G},P} \\
\mathbf{R} &= e^{[\mathbf{r}]} \\
\mathbf{p}^{\mathcal{G},\mathrm{P,goal}} &= \left(\sum_{i=0}^{N_P-1}\mathbf{R}^i\right)\mathbf{p} \\
\mathbf{R}_i^{\mathcal{G},\mathrm{P}} &= \mathbf{R}^i \qquad\qquad i \in I \\
\mathbf{p}_i^{\mathcal{G},\mathrm{P}} &= \left(\sum_{l=0}^{i-1}\mathbf{R}^l\right)\mathbf{p} \qquad i \in I
\end{aligned} \tag{21}$$

where, again, $\mathbf{p}$ is computed via a linear solve in the third equation. Note that this initialization can correspond to multiple solutions, as e.g. a 180 degree z-rotation can be achieved via four 45-degree or four -45-degree z-rotations. Some examples of this phenomenon are shown in Figure 4. As such, we try up to two solutions in TAA form. Given $\boldsymbol{\Upsilon}_0^{\mathcal{G},\text{goal}}$, we first normalize $\|\mathbf{r}_0^{\mathcal{G},\text{goal}}\|$ to ensure $\|\mathbf{r}_0^{\mathcal{G},\text{goal}}\| < 2\pi$. To this end, if $\|\mathbf{r}_0^{\mathcal{G},\text{goal}}\| \in [2k\pi, 2(k+1)\pi)$ for some integer $k \geq 1$, we apply

$$\mathbf{r}_0^{\mathcal{G},\text{goal}} \leftarrow \mathbf{r}_0^{\mathcal{G},\text{goal}} \frac{\|\mathbf{r}_0^{\mathcal{G},\text{goal}}\| - 2k\pi}{\|\mathbf{r}_0^{\mathcal{G},\text{goal}}\|}.$$

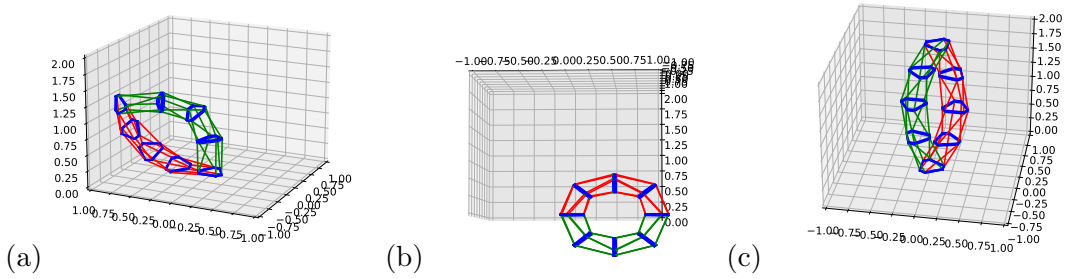
Then, if we find that the resulting SP-frame translation vector $\mathbf{p}^{\mathcal{BP},\text{TP}}$ is too far (more than 60 degrees) from vertical, which is true if and only if

$$\mathbf{p}_{[2]} \leq \|\mathbf{p}\| \cos(\tfrac{\pi}{3}),$$

we reject the solution, then try a second one. We obtain this second solution by reflecting the rotation $\|\mathbf{r}_0^{\mathcal{G},\text{goal}}\|$ about π to achieve the same rotation from the opposite direction, via

$$\mathbf{r}_0^{\mathcal{G},\text{goal}} \leftarrow \mathbf{r}_0^{\mathcal{G},\text{goal}} \frac{2\pi - \|\mathbf{r}_0^{\mathcal{G},\text{goal}}\|}{\|\mathbf{r}_0^{\mathcal{G},\text{goal}}\|}.$$

We have found computationally that this procedure consistently yields useful initial guesses for the optimizer, though the guesses are often kinematically invalid.



(a) (b) (c)

Figure 4: Some examples of pose initialization results. The red poses are the originals, while the green poses are reflected. (a) First try yields extreme angles and flattened SP's; (b) reflected try yields inverted pose with legs clipping through the plates; (c) initializations yield mirrored poses. Axes units are in meters.

3.4 Force-Based IK Optimization

We formulate the optimization problem for a stacked SP Assembler as a quadratically constrained quadratic program (QCQP). In this model, $\mathbf{R}_i^{\mathcal{G},\text{P}}$ and $\mathbf{p}_i^{\mathcal{G},\text{P}}$ are the driving decision variables, in that they uniquely specify a pose. The rest of the variables are derived directly

from these. Note that, due to our choice of reference frame, we have $\mathbf{R}_0^{\mathcal{G},\mathrm{P}} = e^{[\mathbb{0}]} = \mathbb{I}^3$, where $\mathbb{I}^3$ is the 3×3 identity matrix.

Constraints related to kinematic validity are denoted via **(KVC)**; all other constraints establish definitions of intermediate variables.

Equations in this section are either labeled as $(M\#)$ to denote that they are used explicitly in the model, or as $(\#)$, to denote that this is just a calculation useful to deriving the model equations.

3.4.1 Additional Definitions

For the optimization model, we use all variables and parameters in Section 3.1 except J_i^{s}, including only $\mathbf{R}$ and $\mathbf{p}$ for each $\boldsymbol{\Upsilon}$ variable or parameter, and add the following additional variables. For clearer distinction between variables and parameters within this model, we will represent all variables using blue text.

3.4.2 Constraints

Vector Norms: To model the two-norm $\|\boldsymbol{L}_{i,j}^{\mathcal{BP}}\|$ of the leg length vector (which is equivalent to $\|\boldsymbol{L}_{i,j}^{\mathcal{G}}\|$), we introduce intermediate variable $\boldsymbol{L}_{i,j}^{\mathrm{len}}$, then add the constraint

$$\boldsymbol{L}_{i,j}^{\mathrm{len}^2} = \sum_{k=1}^{3} \boldsymbol{L}_{i,j,[k]}^2 \quad \forall i \in I^P, j \in J. \tag{M1}$$

Moreover, any expressions of the form $\|\boldsymbol{v}\|^2$, $\boldsymbol{v} \in \mathbb{R}^m$, or $\|\mathbf{R}\|_F^2$, $\mathbf{R} \in \mathbb{R}^{m \times m}$, as in (M4), (T4), and (T1), are substituted explicitly with the defining expressions

$$\|\boldsymbol{v}\|^2 = \sum_{k=1}^{m} \boldsymbol{v}_{[k]}^2, \tag{22a}$$

$$\|\mathbf{R}\|_F^2 = \sum_{k=1}^{m} \sum_{l=1}^{m} \mathbf{R}_{[k,l]}^2. \tag{22b}$$

Reference Frame Computations: The constraints in this section are needed to define $\mathbf{R}_i^{\mathrm{BP,TP}}$'s and $\mathbf{R}_i^{\mathcal{G},\mathrm{P}}$'s. First, we express the global rotation matrices via their columns as

$$\mathbf{R}_i^{\mathcal{G},\mathrm{P}} = \left[\begin{array}{c|c|c} \mathbf{R}_{i,[:,1]}^{\mathcal{G},\mathrm{P}} & \mathbf{R}_{i,[:,2]}^{\mathcal{G},\mathrm{P}} & \mathbf{R}_{i,[:,3]}^{\mathcal{G},\mathrm{P}} \end{array} \right] \quad i \in I. \tag{M2}$$

Note that, as rotation matrices define an orthonormal right-handed coordinate system, it is

sufficient to consider $\mathbf{R}_{i,[:,1]}^{\mathcal{G},\mathrm{P}}$ and $\mathbf{R}_{i,[:,2]}^{\mathcal{G},\mathrm{P}}$ as the driving variables, then compute $\mathbf{R}_{i,[:,3]}^{\mathcal{G},\mathrm{P}}$ as

$$\mathbf{R}_{i,[:,3]}^{\mathcal{G},\mathrm{P}} = \mathbf{R}_{i,[:,1]}^{\mathcal{G},\mathrm{P}} \times \mathbf{R}_{i,[:,2]}^{\mathcal{G},\mathrm{P}}. \tag{M3}$$

We then ensure the orthonormality of each $\mathbf{R}_{i,[:,1]}^{\mathcal{G},\mathrm{P}}$ and $\mathbf{R}_{i,[:,2]}^{\mathcal{G},\mathrm{P}}$ via

$$\begin{aligned} \left\| \mathbf{R}_{i,[:,1]}^{\mathcal{G},\mathrm{P}} \right\|^2 &= 1, \\ \left\| \mathbf{R}_{i,[:,2]}^{\mathcal{G},\mathrm{P}} \right\|^2 &= 1, \\ \mathbf{R}_{i,[:,1]}^{\mathcal{G},\mathrm{P}} \cdot \mathbf{R}_{i,[:,2]}^{\mathcal{G},\mathrm{P}} &= 0. \end{aligned} \tag{M4}$$

Next, to obtain the translation portion of each plate transformation in the SP frame, we inverse transform the global-frame point $\mathbf{p}_i^{\mathcal{G}}$ by $\mathbf{T}_i^{\mathcal{G}}$ by solving the forward transformation $\mathbf{p}_i^{\mathcal{G}} = \mathbf{T}_i^{\mathcal{G}} \diamond \mathbf{p}_i^{\mathcal{BP},\mathrm{TP}}$, as defined in (5), for the pre-transformed point $\mathbf{p}_i^{\mathcal{BP},\mathrm{TP}}$, yielding

$$\mathbf{p}_i^{\mathcal{BP},\mathrm{TP}} = (\mathbf{R}_i^{\mathcal{G},\mathrm{P}})^\top (\mathbf{p}_{i+1}^{\mathcal{G},\mathrm{P}} - \mathbf{p}_i^{\mathcal{G},\mathrm{P}}) \quad i \in I^P. \tag{M5}$$

We constrain that the bottom plate is at the origin in the global Assembler frame via

$$\begin{aligned} \mathbf{p}_0^{\mathcal{G},\mathrm{P}} &= \mathbb{0} \\ \mathbf{R}_0^{\mathcal{G},\mathrm{P}} &= \mathbb{I}^3. \end{aligned} \tag{M6}$$

Finally, we define the SP-frame rotation matrices $\mathbf{R}_i^{\mathrm{BP,TP}}$ via $\mathbf{R}_i^{\mathcal{G},\mathrm{P}} \mathbf{R}_i^{\mathrm{BP,TP}} = \mathbf{R}_{i+1}^{\mathcal{G},\mathrm{P}}$, yielding

$$\mathbf{R}_i^{\mathrm{BP,TP}} = \left(\mathbf{R}_i^{\mathcal{G},\mathrm{P}} \right)^\top \mathbf{R}_{i+1}^{\mathcal{G},\mathrm{P}} \quad i \in I^P. \tag{M7}$$

Leg Attachment Locations: In (M8), we define the SP-frame and global-frame locations of the leg attachments on the top/bottom plates for each SP according to (6). Note that the position of the bottom leg attachments are known constants in local space. See also Figure 2.

$$\begin{aligned} \mathbf{p}_{i,j}^{\mathcal{BP},\mathrm{LT}} &= \mathbf{R}_i^{\mathrm{BP,TP}} \mathbf{p}_{i,j}^{\mathcal{TP},\mathrm{LT,rest}} + \mathbf{p}_i^{\mathcal{BP},\mathrm{TP}} && \forall i \in I^P, j \in J \\ \mathbf{p}_{0,j}^{\mathcal{G},\mathrm{LB}} &= \mathbf{p}_{0,j}^{\mathcal{G},B,\mathrm{rest}} && \forall j \in J \\ \mathbf{p}_{i,j}^{\mathcal{G},\mathrm{LB}} &= \mathbf{R}_i^{\mathcal{G},\mathrm{P}} \mathbf{p}_{i,j}^{\mathcal{BP},\mathrm{LB}} + \mathbf{p}_i^{\mathcal{G}} && \forall i \in I^P, i \geq 1, j \in J \\ \mathbf{p}_{i,j}^{\mathcal{G},\mathrm{LT}} &= \mathbf{R}_{i+1}^{\mathcal{G},\mathrm{P}} \mathbf{p}_{i,j}^{\mathcal{TP},\mathrm{LT,rest}} + \mathbf{p}_{i+1}^{\mathcal{G}} && \forall i \in I^P, j \in J. \end{aligned} \tag{M8}$$

Leg Centers of Gravity: To define the center of gravity for each leg $j \in J$ for SP $i \in I^P$, we

start with (16), multiply through by denominators, and rearrange, yielding

$$\begin{aligned} \boldsymbol{L}^{\text{len}}_{i,j}(\mathbf{p}^{\mathcal{G},\text{LB,CoG}}_{i,j} - \mathbf{p}^{\mathcal{G},\text{LB}}_{i,j}) &= d^{\text{LB,CoG}}\boldsymbol{L}^{\mathcal{G}}_{i,j}, \\ \boldsymbol{L}^{\text{len}}_{i,j}(\mathbf{p}^{\mathcal{G},\text{LT}}_{i,j} - \mathbf{p}^{\mathcal{G},\text{LT,CoG}}_{i,j}) &= d^{\text{LT,CoG}}\boldsymbol{L}^{\mathcal{G}}_{i,j}. \end{aligned} \tag{M9}$$

Leg Length Bounds: **(KVC)** We define the leg length bounding constraints via (8), as

$$L^{\min} \leq \boldsymbol{L}^{\text{len}}_{i,j} \leq L^{\max}. \tag{M10}$$

Note that the upper-bounding leg length constraints are second order cone constraints in terms of $\mathbf{p}^{\mathcal{BP},\text{TP}}_i$ and $\mathbf{R}^{\mathcal{BP},\text{TP}}_i$ (as the interiors of spheres), while the lower-bounding constraints are nonconvex as sphere exteriors.

Leg Angle Deviation: **(KVC)** The bottom-plate leg angle deviation constraints are defined via (9), after multplying through by denominators, as The bottom-plate leg angle deviation constraints are defined via (9), as

$$\boldsymbol{L}^{\text{len}}_{i,j}\|\boldsymbol{L}^{\mathcal{BP},\text{rest}}_{i,j}\|\cos(\theta^{\max}) \leq \boldsymbol{L}^{\mathcal{BP}}_{i,j} \cdot \boldsymbol{L}^{\mathcal{BP},\text{rest}}_{i,j} \quad \forall i \in I^P, j \in J. \tag{M11}$$

while the top-plate constraints are defined via (10), after multiplying through by denominators, as

$$\boldsymbol{L}^{\text{len}}_{i,j}\|\boldsymbol{L}^{\mathcal{BP},\text{rest}}_{i,j}\|\cos(\theta^{\max}) \leq \boldsymbol{L}^{\mathcal{BP}}_{i,j} \cdot (\mathbf{R}^{\text{BP,TP}}_i \boldsymbol{L}^{\mathcal{BP},\text{rest}}_{i,j}) \quad \forall i \in I^P, j \in J. \tag{M12}$$

Note that all leg angle deviation constraints are second-order cone constraints given fixed rotation matrices.

Continuous Translation Enforcement: **(KVC)** We enforce that legs do not break the surface of the bottom plate of any SP via (11), as

$$\mathbf{p}^{\mathcal{BP},\text{LT}}_{i,j,[3]} \geq \mathbf{p}^{\mathcal{BP},\text{LB}}_{i,j,[3]} \tag{23}$$

Extreme Pose Prevention: **(KVC)** To help prevent extreme and difficult-to-reach poses for each SP, we enforce (12), as

$$\mathbf{R}^{\text{BP,TP}}_{i,[k,k]} \geq \cos(\theta^{\mathbf{R},\max}), \quad k = 1, 2, 3 \tag{24}$$

End Effector: **(KVC)** We enforce that the end effector pose is exactly as desired via

$$\begin{aligned} \mathbf{p}^{\mathcal{G},\text{P}}_{N_P-1} &= \mathbf{p}^{\mathcal{G},\text{Pgoal}} \\ \mathbf{R}^{\mathcal{G},\text{P}}_{N_P-1,[:,1]} &= \mathbf{R}^{\mathcal{G},\text{P,goal}}_{[:,1]} \\ \mathbf{R}^{\mathcal{G},\text{P}}_{N_P-1,[:,2]} &= \mathbf{R}^{\mathcal{G},\text{P,goal}}_{[:,2]} \end{aligned} \tag{M13}$$

where $\mathbf{p}^{\mathcal{G},\mathrm{P},\mathrm{goal}}$ and $\mathbf{R}^{\mathcal{G},\mathrm{P},\mathrm{goal}}$ are computed from $\boldsymbol{\Upsilon}^{\mathcal{G},\mathrm{goal}}$.
Objective: The objective is to minimize maximal leg force

$$\min \tau^{\max}. \tag{M14}$$

where the constraints defining the maximum force $\tau^{\max}$ are introduced in Section 3.4.3.
Initialization: As the model is solved only to local optimality via IPOPT due to the intractability of a global solve even with $N_P = 2$, an initial solution for the pose is required. We choose to initialize via the same-SP initialization scheme in Section 3.3.2, as it performed much better in our numerical testing when compared to the spline-based scheme in Section 3.3.1, despite yielding valid poses less often before optimization.

3.4.3 Forces

Referring to (15), the *transposed adjoint* of a transformation matrix $\mathbf{T}$ is

$$\mathrm{Adj}\,(\mathbf{T})^\top = \begin{bmatrix} \mathbf{R}^\top & \mathbf{R}^\top[\mathbf{p}]^\top \\ \mathbb{0} & \mathbf{R}^\top \end{bmatrix}. \tag{25}$$

For shorthand, we let $\mathbf{A}_i := \mathrm{Adj}\left(\mathbf{T}_i^{\mathcal{G}}\right)^\top$ for each $i \in I$.

Each column of the transposed inverse Jacobian J^{t} as in (13) via (M15), can be computed as (see [21] for discussion on single SPs)

$$\boldsymbol{L}_{i,j}^{\mathrm{len}} J^{\mathrm{t}}{}_{i,[:,j]} = \begin{bmatrix} [\mathbf{p}_{i,j}^{\mathcal{G},\mathrm{LB}}]\boldsymbol{L}_{i,j}^{\mathcal{G}} \\ \boldsymbol{L}_{i,j}^{\mathcal{G}} \end{bmatrix} \quad \forall i \in I^P, j \in J. \tag{M15}$$

Note that $\boldsymbol{L}_{i,j}^{\mathrm{len}} = \|\mathbf{p}_{i,j}^{\mathcal{G},\mathrm{LT}} - \mathbf{p}_{i,j}^{\mathcal{G},\mathrm{LB}}\| = \|\mathbf{p}_{i,j}^{\mathcal{BP},\mathrm{LT}} - \mathbf{p}_{i,j}^{\mathcal{BP},\mathrm{LB}}\|$ is the length of the corresponding leg, as modelled in (M1).

To compute the global-frame wrenches for each SP, we use (17) for $i = 0, \ldots, N_P - 2$ and (18) for $i = N_P - 1$.

Finally, to compute the leg forces, and defining $\boldsymbol{\tau}_i = [\tau_{i,1}, \ldots, \tau_{i,6}]^\top$ for each i, we use the vector constraints

$$\begin{aligned} J^{\mathrm{t}}{}_i \boldsymbol{\tau}_i &= \mathbf{A}_i W_i && \forall i \in I^P \\ \tau^{\max} &\geq \tau_{i,j} && \forall i \in I^P, j \in J \\ \tau^{\max} &\geq -\tau_{i,j} && \forall i \in I^P, j \in J \end{aligned} \tag{M16}$$

where the jth component of the resulting solution $\boldsymbol{\tau}_i$ is the force on the jth leg of the ith SP. We then minimize over τ.

Notice that $\mathbf{A}_i W_i$ can be written as:

$$\mathbf{A}_i W_i = \begin{bmatrix} (\mathbf{R}_i^{\mathcal{G},\mathrm{P}})^\top & (\mathbf{R}_i^{\mathcal{G},\mathrm{P}})^\top [\mathbf{p}_i^{\mathcal{G},\mathrm{P}}]^\top \\ \mathbb{0} & (\mathbf{R}_i^{\mathcal{G},\mathrm{P}})^\top \end{bmatrix} \begin{bmatrix} W_i^R \\ W_i^P \end{bmatrix} = \begin{bmatrix} (\mathbf{R}_i^{\mathcal{G},\mathrm{P}})^\top (W_i^R + [\mathbf{p}_i^{\mathcal{G},\mathrm{P}}]^\top W_i^P) \\ (\mathbf{R}_i^{\mathcal{G},\mathrm{P}})^\top W_i^P \end{bmatrix}.$$

Thus, the first constraint of (M16) is equivalent to, and implemented as,

$$J^{\mathrm{t}}{}_i \boldsymbol{\tau}_i \quad = \begin{bmatrix} (\mathbf{R}_i^{\mathcal{G},\mathrm{P}})^\top (W_i^R + [\mathbf{p}_i^{\mathcal{G},\mathrm{P}}]^\top W_i^P) \\ (\mathbf{R}_i^{\mathcal{G},\mathrm{P}})^\top W_i^P \end{bmatrix} \quad i \in I^P. \tag{M17}$$

Note that this constraint is bilinear in nature, since W_i^P is a constant.

If $w_2 \neq 0$, we also enforce that the force on each leg does not exceed the maximum allowable force, via

$$\tau^{\max} \leq f^{\max} \tag{M18}$$

3.4.4 Improving Robustness

Due to the nature of iterative local nonlinear optimization via e.g. IPOPT, the equality constraints defining various intermediate variables such as leg lengths, rotation matrices, the inverse Jacobian etc, and even primary constraints such as end-effector position, can become violated as the solver attempts to resolve violations of the physical constraints defining a valid pose. We have observed that this can sometimes lead to instability within the optimizer, particularly if the end effector is moved from the goal position during optimization.

To address this instability while controlling for computational time, we implement an iterative-refinement scheme around the basic nonlinear optimizer. Here, we leverage the fact that the variables $\boldsymbol{\Upsilon}_i^{\mathcal{G},\mathrm{P}}$ uniquely define the Assembler. Thus, if an optimization seems to be converging slowly or returns with a 'locally infeasible' status, we re-initialize the model every so often, and reset the end effector to the correct location.

To this end, we set the maximum total internal iteration count as 2500. Every k iterations, if the solver has not yet converged, we stop the solve, re-initialize using the plate locations and corrected the end effector location, then continue. We start with $k = 500$, but for each internal solver error we divide k in half and try again, with up to five such retries.

Occasionally, locally infeasible solutions can occur at valid poses, due to numerical difficulties in resolving force-related equality constraints. This results in sub-optimal, but kinematically valid, poses. When using IPOPT as the nonlinear solver, we have observed that the solver can often recover from such poses after re-initialization. Thus, to handle this contingency, on the first consecutive 'locally infeasible' result, we deduct k iterations from the remaining total (as if the solver had run k iterations) and re-initialize as usual. On the consecutive second locally

infeasible result for the same pose, we report an optimization failure due to local infeasibility within the solver.

3.4.5 Extensions

Here, we describe some possible extensions to the model that could be useful to improve the stability of the optimized pose solutions.

Objective: In the objective, we could take into consideration both maximal leg force and the angular deviation λ_i of the consecutive plates from the resting poses. In this case, the objective becomes

$$\min w_1 \tau^{\max} + w_2 \|\boldsymbol{\lambda}\|^2, \tag{26}$$

where λ_i for $i = 1, \ldots, N_p - 1$ can be formally defined as

$$\|\mathbf{p}_i^{\mathcal{BP},\mathrm{TP}}\| \|\mathbf{p}_i^{\mathcal{BP},\mathrm{TP,rest}}\| \cos(\lambda_i) = \mathbf{p}_i^{\mathcal{BP},\mathrm{TP}} \cdot \mathbf{p}_i^{\mathcal{BP},\mathrm{TP,rest}}. \tag{27}$$

Now, as the model for λ_i^2 is highly nonlinear, we instead model 1-cos(λ_i), as it is nonnegative, zero at $\lambda_i = 0$, and approximately quadratic for λ_i near zero. Indeed, for small λ, we have $\|\lambda\|^2 \approx 2\|1 - \cos(\lambda)\|_1$, which is supported element-wise by the fact that $\lim_{x\to 0} \frac{1-\cos(x)}{x^2} = \frac{1}{2}$. Thus, for the objective, we use

$$\min w_1 \tau + w_2 \sum_{i \in I^P} x_i, \tag{28}$$

where x_i is defined from cos(λ_i) via the second-order cone constraint.

$$x_i \geq (2(1 - \cos(\lambda_i)))^2 \quad \forall i \in I^P. \tag{29}$$

Note that one only needs to model the lower bound on x_i in (27), as the optimization will seek to minimize x_i in order to maximize cos(λ_i). Furthermore, we replace cos(λ_i) with an intermediate variable $c\lambda_i$, defined via (27) as

$$\|\mathbf{p}_i^{\mathcal{BP},\mathrm{TP}}\| \|\mathbf{p}_i^{\mathcal{BP},\mathrm{TP,rest}}\| c\lambda_i = \mathbf{p}_i^{\mathcal{BP},\mathrm{TP}} \cdot \mathbf{p}_i^{\mathcal{BP},\mathrm{TP,rest}}. \tag{30}$$

Note that we require an intermediate variable and constraint to model $\|\mathbf{p}_i^{\mathcal{BP},\mathrm{TP}}\|$.

In practice, setting $w_2 > 0$ can help improve the stability and manipulability of the Assembler in the resulting poses.

Sensitivity to End-Effector Forces: To improve the robustness of the pose to internal forces, we can add phantom forces to each leg corresponding with the sensitivity of the leg to the force applied to the end effector.

To compute the sensitivity of the force on leg i to the force $\boldsymbol{F}^{\mathrm{EE}}$ applied on the end effector, we first note

$$\begin{aligned} W^{R,\mathrm{EE}} &= [\mathbf{R}_{N_P}^{\mathcal{G},\mathrm{P}} \boldsymbol{v}^{\mathcal{TP},m,\mathrm{EE}} + \mathbf{p}_{N_P}^{\mathcal{G}}] \boldsymbol{F}^{\mathrm{EE}} \\ W^{P,\mathrm{EE}} &= \boldsymbol{F}^{\mathrm{EE}}, \end{aligned} \tag{31}$$

where $\boldsymbol{v}^{\mathcal{TP},m,\mathrm{EE}}$ is the position of the end effector extension from the top plate of the Assembler, in the reference frame of the top plate. Thus, since the only component of the wrench depending on $\boldsymbol{F}^{\mathrm{EE}}$ is W^{EE}, we thus obtain

$$\begin{aligned} J^{\mathrm{t}}{}_i (\nabla_{\boldsymbol{F}^{\mathrm{EE}}} \boldsymbol{\tau}_i)^\top &= \begin{bmatrix} (\mathbf{R}_i^{\mathcal{G},\mathrm{P}})^\top ([\mathbf{R}_{N_P}^{\mathcal{G},\mathrm{P}} \boldsymbol{v}^{\mathcal{TP},m,\mathrm{EE}} + \mathbf{p}_{N_P}^{\mathcal{G}}] + [\mathbf{p}_i^{\mathcal{G},\mathrm{P}}]^\top) \\ (\mathbf{R}_i^{\mathcal{G},\mathrm{P}})^\top \end{bmatrix} \\ &= \begin{bmatrix} (\mathbf{R}_i^{\mathcal{G},\mathrm{P}})^\top ([\mathbf{R}_{N_P}^{\mathcal{G},\mathrm{P}} \boldsymbol{v}^{\mathcal{TP},m,\mathrm{EE}}] + [\mathbf{p}_{N_P}^{\mathcal{G}} - \mathbf{p}_i^{\mathcal{G},\mathrm{P}}]) \\ (\mathbf{R}_i^{\mathcal{G},\mathrm{P}})^\top \end{bmatrix} \quad i \in I^P, \end{aligned} \tag{32}$$

where the jth row of $(\nabla_{\boldsymbol{F}^{\mathrm{EE}}} \boldsymbol{\tau}_i)^\top$ gives the gradient of the force on leg j of SP i w.r.t. the force on the end effector.

4 Trajectory Planning

The Assembler is designed to be a movable platform that can help with complicated operations. As such, it is important that it can move from one position to another. We will adapt the tools in the prior section to develop trajectory optimization techniques. We present three approaches: a naïve direct transformation, a force optimization using trust regions, and a RRT* version that hinges on sub-paths computed from the force optimization approach.

Equations in this section are either labeled as $(N\#)$, $(T\#)$, or $(\#)$ to denote equations for naïve method, trust method, or calculations, respectively.

4.1 Naïve Direct Transformation

We establish the simplest approach to move from one pose to another that we call the naïve method. This approach is ignorant to force calculations, and thus is very prone to generating infeasible trajectories that violate the maximum force bounds. The approach simply uses a linear interpolation of the vector of leg lengths throughout the motion. For each target leg-length vector $\boldsymbol{L}_{i,j}^{\mathrm{goal}}$, $i \in I^P$, $j \in J$, the FK problem is solved to obtain the full trajectory planning solution. To solve this problem, we first attempt the widely-used Newton-Raphson approach, as in e.g. [23, 26]. If this does not succeed, we then leverage a force-free version of the nonlinear optimizer to solve the FK problem, in which the end effector is allowed to move,

and we optimize the squared leg-length-distance from the goal. More formally, to construct a QCQP model for the FK problem, we begin with the model in Section 3.4, then remove all force-related constraints and the end effector constraint (M13). We then optimize

$$\min \sum_{i \in I^P} \sum_{j \in J} (\boldsymbol{L}_{i,j} - \boldsymbol{L}_{i,j}^{\text{goal}})^2. \tag{N1}$$

Note that the FK problem for the Assembler can decompose into separate FK problems for each SP.

4.2 Trust Region Optimization

We introduce a trust region method for trajectory planning of the SP. This algorithm first optimizes the starting and goal poses, then iteratively tries to make small steps towards the goal pose until it is sufficiently close, while trying to maintain forces that do not exceed those in the starting or ending poses.

4.2.1 Additional Definitions

To define the optimization problem for a single step, we first introduce the following additional parameters for the full optimization.

Parameter	Size	Description
$\boldsymbol{\Upsilon}^{\mathcal{G},EE,\text{start}}$		: Starting end effector position
$\boldsymbol{\Upsilon}^{\mathcal{G},EE,\text{goal}}$		: Goal end effector position
$\lambda^{\text{force}} \in [0,1]$		: Coefficient for force-related objective terms
$\lambda^{\text{pose}} \in [0,1]$		: Coefficient for objective terms related to distance from the final goal pose
$\lambda^{\text{avg}} \in [0,1]$		: Additional multiplier for average-force-related objective term
ε^{pos}		: Positional plate motion limit for each Stewart platform per iteration
ε^{rot}		: Rotational plate motion limit for each Stewart platform per iteration
$\boldsymbol{F}^{\text{EE}}$		: The force vector applied to the end effector
$\boldsymbol{v}^{\mathcal{TP},m,\text{EE}}$		: The application point for $\boldsymbol{F}^{\text{EE}}$ w.r.t. the top plate of the assembler

Note that we normalize the non-negative objective weights with $\lambda^{\text{force}} + \lambda^{\text{pose}} = 1$. We then compute the full optimized starting and ending poses by solving the QCQP model in Section 3.4 with IPOPT, then compute the maximum force observed in either pose.

Parameter	Description
$\boldsymbol{\Upsilon}_i^{\mathcal{G},\text{P},\text{start}}$	: Global-frame starting pose for plate $i \in I$
$\boldsymbol{\Upsilon}_i^{\mathcal{BP},\text{TP},\text{start}}$	: SP-frame starting pose for plate $i \in I$

$\boldsymbol{\Upsilon}_i^{\mathcal{G},\mathrm{P},\mathrm{goal}}$	: Global-frame goal pose for plate $i \in I$
$\boldsymbol{\Upsilon}_i^{\mathcal{BP},\mathrm{TP},\mathrm{goal}}$	: SP-frame goal pose for SP $i \in I^P$
$f^{\max,\mathrm{ends}}$	: Maximum force in either path endpoint, the starting and goal poses

Note that, for practical use, the starting pose would be specified directly as the current pose of the Assembler.

For each iteration, we define the pose from the previous iteration as

Parameter	Description
$\boldsymbol{\Upsilon}_i^{\mathcal{G},P,\mathrm{init}}$	: Initial global-frame pose for plate $i \in I$ for the current iteration
$\boldsymbol{\Upsilon}_i^{\mathcal{BP},\mathrm{TP},\mathrm{init}}$	: Initial SP-frame pose for plate $i \in I^P$ for the current iteration

Finally, we introduce the following additional variables for each iteration.

Variable	Size	Description
τ^{viol}		: Maximum force above the maximum allowable force in any leg
τ^{ends}		: Maximum force above $f^{\max,\mathrm{ends}}$ in any leg
$\tau_{i,j}^{\mathrm{abs}}$		: Absolute value of $\tau_{i,j}$ for leg $i \in I^P$, $j \in J$
W^{EE}		: End-effector wrench at the current pose

4.2.2 Constraints

We begin with the model in Section 3.4, omitting the end effector constraints (M13) and the explicit force constraint (M18). We also redefine the end effector wrench definition W^{EE} to account for the fact that the end effector is no longer at a fixed position.

We then add constraints to ensure that no plate moves a distance further than $\varepsilon^{\mathrm{pos}}$ from $\mathbf{p}_i^{\mathcal{G},P,\mathrm{init}}$, measured in two-norm, and rotates no further than $\varepsilon^{\mathrm{rot}}$ from $\mathbf{R}_i^{\mathcal{G},\mathrm{P},\mathrm{init}}$, measured in matrix Frobenius norm. The Frobenius norm was chosen for its superior performance and robustness compared to more direct angle-based measures in preliminary testing, combined with its simplicity and convexity. Finally, we add constraints to define τ^{viol} and τ^{ends}, then define the objective function.

Motion Limit: We ensure that positional and rotational motions are sufficiently controlled via

$$\begin{aligned} \|\mathbf{R}_i^{\mathcal{G},\mathrm{P}} - \mathbf{R}_i^{\mathcal{G},\mathrm{P},\mathrm{init}}\|_F^2 &\leq \left(\varepsilon^{\mathrm{rot}}\right)^2 \quad i \in I, i \geq 1 \\ \|\mathbf{p}_i^{\mathcal{G},\mathrm{P}} - \mathbf{p}^{\mathcal{G},\mathrm{P},\mathrm{init}}\|^2 &\leq \left(\varepsilon^{\mathrm{pos}}\right)^2 \quad i \in I, i \geq 1. \end{aligned} \tag{T1}$$

Force Definitions: To define wrenches, we first define the now-variable rotational component

$W^{R,\mathrm{EE}}$ of the end effector wrench W^{EE} via

$$\begin{aligned} \mathbf{p}^{\mathcal{G},m,\mathrm{EE}} &= \mathbf{R}_{N_P}^{\mathcal{G},\mathrm{P}} \boldsymbol{v}^{\mathcal{TP},m,\mathrm{EE}} + \mathbf{p}_{N_P}^{\mathcal{G},\mathrm{P}} \\ W^{R,\mathrm{EE}} &= [\mathbf{p}^{\mathcal{G},m,\mathrm{EE}}] \boldsymbol{F}^{\mathrm{EE}} \end{aligned} \tag{T2}$$

where the translational component $W^{P,\mathrm{EE}} = \boldsymbol{F}^{\mathrm{EE}}$ of the wrench is constant. We then use (18) and (17) to define plate wrenches as before. Next, we define the additional required force-related variables with

$$\begin{aligned} \tau^{\mathrm{viol}} &\geq \tau^{\max} - f^{\max} && \forall i \in I^P, j \in J \\ \tau^{\mathrm{ends}} &\geq \tau^{\max} - f^{\max,\mathrm{ends}} && \forall i \in I^P, j \in J \\ \tau_{i,j}^{\mathrm{abs}} &\geq \tau_{i,j} && \forall i \in I^P, j \in J \\ \tau_{i,j}^{\mathrm{abs}} &\geq -\tau_{i,j} && \forall i \in I^P, j \in J \\ \tau^{\mathrm{viol}} &\geq 0 \\ \tau^{\mathrm{ends}} &\geq 0 \end{aligned} \tag{T3}$$

Note that, through τ^{viol}, the max-force constraints on the legs are moved to the objective with a high coefficient. For the purposes of balancing objective terms related to forces, distances, and rotation angles, we assume forces and distances are measured in SI units (i.e. Newtons and meters). Note that the rotation terms are unit-independent, as they are measured via Frobenius norms of unitary rotation matrices.

4.2.3 Objective

To assist in defining the objective functions, we define the following expressions for the sake of readability. These definitions are the 'distance-to-goal' metrics for the objective function

$$\begin{aligned} \tau^{\text{1-norm}} &= \sum_{i \in I^P} \sum_{j \in J} \tau_{i,j}^{\mathrm{abs}}, \\ \mathcal{E}^{\mathrm{pos}} &= \sum_{i \in I^P} \|\mathbf{p}_i^{\mathcal{BP},\mathrm{TP}} - \mathbf{p}_i^{\mathcal{BP},\mathrm{TP},\mathrm{goal}}\|^2, \\ \mathcal{E}^{\mathrm{rot}} &= \sum_{i \in I^P} \|\mathbf{R}_i^{\mathcal{BP}} - \mathbf{R}_i^{\mathcal{BP},\mathrm{goal}}\|_F^2. \end{aligned} \tag{T4}$$

With these definitions, we define the objective function as

$$\min \lambda^{\mathrm{force}} (\tau^{\mathrm{ends}} + \tfrac{\lambda^{\mathrm{avg}}}{6N_P} \tau^{\text{1-norm}} + 1000 \tau^{\mathrm{viol}}) + \lambda^{\mathrm{pose}} (\mathcal{E}^{\mathrm{pos}} + \tfrac{1}{4} \mathcal{E}^{\mathrm{rot}}). \tag{T5}$$

Note that the decision to divide the $\mathcal{E}^{\mathrm{rot}}$ by 4 serves to balance the position and rotation distance metrics, and performed well in preliminary testing compared to other coefficients. For

Assemblers consisting of SP's of different sizes, this divisor should be re-scaled according to the size of the SP, while the choice of λ^{force} should be re-scaled according to the weight of the SP.

4.2.4 Iteration

The trust region method proceeds by solving the problem described above until convergence. However, the process often stagnates, reaching positions where the objective to reduce forces overrides the objective to move towards the goal pose, or even moves in the wrong direction to improve average forces. When stagnation occurs due with high maximum forces, we divide λ^{force} by 2, set $\lambda^{\text{pose}} = 1 - \lambda^{\text{force}}$, and try again with the same initial positions. Similarly, if either stagnation or motion in the wrong direction occurs due to average forces, we divide λ^{avg} by 4 and try again with the same initial positions. The method converges when the distance between the current pose and the goal pose is small enough that the goal pose is a valid solution for the next iteration.

More formally, for the current iteration, define

$$\begin{aligned} \mathcal{E}^{\text{max,pos}} &= \max_{i \in I, i \geq 1} \|\mathbf{p}_i^{\mathcal{G},\text{P,init}} - \mathbf{p}^{\mathcal{G},\text{P,goal}}\|, \\ \mathcal{E}^{\text{max,rot}} &= \max_{i \in I, i \geq 1} \|\mathbf{R}_i^{\mathcal{G},\text{P,init}} - \mathbf{R}_i^{\mathcal{G},\text{P,goal}}\|_F. \end{aligned} \tag{33}$$

Then *convergence* occurs when both $\mathcal{E}^{\text{max,pos}} \leq \varepsilon^{\text{pos}}$ and $\mathcal{E}^{\text{max,rot}} \leq \varepsilon^{\text{rot}}$. Early termination occurs if the process has stagnated or moved in the wrong direction too many times, or if too many iterations have been reached.

Due to the multi-objective nature of the trust region method, it's possible that insufficient progress towards the goal can occur during the solve. We call this a *stagnation* and formally define this to occur when one of the following criteria are met after optimizing:

$$\begin{aligned} \max_{i \in I, i \geq 1} \|\mathbf{p}_i^{\mathcal{G},\text{P}} - \mathbf{p}^{\mathcal{G},\text{P,init}}\| &\leq \tfrac{\varepsilon^{\text{pos}}}{100}, \\ \max_{i \in I, i \geq 1} \|\mathbf{R}_i^{\mathcal{G},\text{P}} - \mathbf{R}_i^{\mathcal{G},\text{P,init}}\|_F &\leq \tfrac{\varepsilon^{\text{rot}}}{100}. \end{aligned} \tag{34}$$

Even if $\lambda^{\text{avg}} = 0$, stagnation can occur in the optimizer if $\tau^{\max} = f^{\text{max,ends}}$ and further progress requires forces above $f^{\text{max,ends}}$. Thus, if stagnation occurs, we conclude that the average-force term is the culprit only if $\tau^{\max} \leq f^{\text{max,ends}} - 0.001$, where the 1mN subtraction is added to be conservative, helping to prevent the algorithm from stagnating with repeated, futile reductions of λ^{avg}.

We define a step to be in the *wrong direction* if both the positional and rotational components of the motion have moved somewhat away from the goal pose, i.e. if for an iteration k we have $\mathcal{E}_k^{\text{max,pos}} \geq \mathcal{E}_{k-1}^{\text{max,pos}} + 10^{-4}$ and $\mathcal{E}_k^{\text{max,rot}} \geq \mathcal{E}_{k-1}^{\text{max,rot}} + 10^{-4}$. We allow

motion in the wrong direction in order to reduce maximum leg forces that seemed excessive, i.e. if $\tau^{\max} > f^{\max,\text{ends}}$. If we reject the motion step, then the position-related terms in the objective function have worsened while the max-force-related terms have not improved, and so the average force term must be the culprit.

Define the maximum allowed number of stagnated iterations as $n^{\max,\text{stag}}$, and define the maximum number of iterations as $k^{\max}$. The iteration then proceeds as in Algorithm 1. For shorthand, we define the solution path as a list of poses from the starting pose to the goal pose. As defined in Section 2, a *pose* is the corresponding list of $\mathbf{p}_i^{\mathcal{G},\text{P}}$'s and $\mathbf{R}_i^{\mathcal{G},\text{P}}$'s.

Algorithm 1: General procedure for trust-region trajectory planning method

Input : A starting position $pose^{\text{start}}$ and ending position $pose^{\text{ends}}$.
Output : An integer K and a sequence of poses $pose_0, \ldots, \text{pose}_k$

$n^{\text{stag}} \leftarrow 0, \quad k \leftarrow 1, \quad pose_0 \leftarrow pose^{\text{start}}$
while *not converged and* $n^{stag} < n^{\max,stag}$ *and* $k \leq k^{\max}$ **do**
  $pose^{\text{init}} \leftarrow pose_{k-1}$
  Solve the trust region problem from position $pose^{\text{init}}$ to compute solution *pose*
  if *stagnation detected or (wrong direction detected and* $\tau_{k-1}^{ends} = 0$*)* **then**
    if *wrong direction* **then**
      $\lambda^{\text{avg}} \leftarrow \frac{1}{4}\lambda^{\text{avg}}, \quad n^{\text{stag}} \leftarrow n^{\text{stag}} + 1$
    else
      $\lambda^{\text{force}} \leftarrow \frac{1}{2}\lambda^{\text{force}}, \quad \lambda^{\text{pose}} \leftarrow 1 - \lambda^{\text{force}}, \quad n^{\text{stag}} \leftarrow n^{\text{stag}} + 1$
    end if
  else
    $pose_k \leftarrow pose, \quad k \leftarrow k + 1$
  end if
end while
if *converged* **then**
  $pose_k \leftarrow pose^{\text{goal}}$
end if

To improve consistency of this algorithm, we run it within a backtracking scheme: if convergence fails, then we assume the iteration got sidetracked by forces, and restart after dividing λ^{force} by 4, for up to 5 total restarts. Then, if convergence succeeds, but maximal mid-path forces exceed maximum path-endpoint forces, we run again with $pose^{\text{start}}$ and $pose^{\text{ends}}$ switched to try to find a better path. We only keep this reversed solution if it converges and yields better worst-case mid-path forces.

5 Experimental Results

The results in Sections 5.2 and 5.3 were coded in Python 3.7, while the optimization steps were performed in IPOPT 3.11.1 [37] via Pyomo 5.7 [4, 12]. They were run on a laptop running Windows 10 with 32GB of RAM, using an Intel Core i7-9750H CPU processor (2.6GHz, 6 Cores, 12 threads). Additional computations were run on a desktop computer running Windows 10 with 64GB of RAM, using an AMD Ryzen 9 3900X 12-Core processsor running at 3.79Ghz.

For these computational studies, we use the following parameters. All parameters are given in SI units, i.e. kilograms, meters, seconds, and Newtons for masses, distances, time, and forces, respectively.

Parameter	Value
N_P	$= 4$
$\boldsymbol{\Upsilon}_i^{\mathcal{BP},\mathrm{TP,rest}}$	$= [0, 0, 0.5069351, 0, 0, 0]^\top$
$\mathbf{p}_{0,j}^{\mathcal{BP},\mathrm{LB}}$	$= \begin{bmatrix} 0.150037 & 0.150037 & -0.040202 & -0.109834 & -0.109834 & -0.040202 \\ -0.040202 & 0.040202 & 0.150037 & 0.109834 & -0.109834 & -0.150037 \\ 0.016637 & 0.016637 & 0.016637 & 0.016637 & 0.016637 & 0.016637 \end{bmatrix}^\top$
$\mathbf{p}_{0,j}^{\mathcal{TP},\mathrm{LT,rest}}$	$= \begin{bmatrix} 0.109834 & 0.109834 & 0.040202 & -0.150037 & -0.150037 & 0.040202 \\ -0.109834 & 0.109834 & 0.150037 & 0.040202 & -0.040202 & -0.150037 \\ -0.016637 & -0.016637 & -0.016637 & -0.016637 & -0.016637 & -0.016637 \end{bmatrix}^\top$
$\theta^{\max}$	$= 55°$
$\theta^{\mathbf{R},\max}$	$= 60°$
$(L^{\min}, L^{\max})$	$= (0.38044, 0.580434)$
$f^{\max}$	$= 889.644$
$m^{\mathrm{P}}_{(i=0:N_P)}$	$= [7.235, 14.47, 14.47, 14.47, 7.235]^\top$
m^{LT}	$= 0.15$
m^{LB}	$= 0.2$
$d^{\mathrm{LT,CoG}}$	$= 0.05$
$d^{\mathrm{LB,CoG}}$	$= 0.089$
g	$= 9.81$

For even i, The values of $\mathbf{p}_{i,j}^{\mathcal{BP},\mathrm{LB}}$ and $\mathbf{p}_{i,j}^{\mathcal{TP},\mathrm{LT,rest}}$ are equal to $\mathbf{p}_{0,j}^{\mathcal{BP},\mathrm{LB}}$ and $\mathbf{p}_{0,j}^{\mathcal{TP},\mathrm{LT,rest}}$, respectively. For odd i, we have

$$\begin{aligned} \mathbf{p}_{i,j}^{\mathcal{BP},\mathrm{LB}} &= R^{\mathrm{odd}} \mathbf{p}_{0,j}^{\mathcal{BP},\mathrm{LB}} \\ \mathbf{p}_{i,j}^{\mathcal{TP},\mathrm{LT,rest}} &= R^{\mathrm{odd}} \mathbf{p}_{0,j}^{\mathcal{TP},\mathrm{LT,rest}} \end{aligned} \tag{35}$$

Where R^{odd} is a 30-degree rotation about the z-axis,

$$R^{\text{odd}} = \begin{bmatrix} \frac{\sqrt{3}}{2} & -\frac{1}{2} & 0 \\ \frac{1}{2} & \frac{\sqrt{3}}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{36}$$

5.1 Pose Generation

We describe the procedure for generating the dataset of poses that we use to generate end-effector goal positions for the computational studies in this work. This generation begins with the generation of individual random poses for the intended SP configuration. Given that the target 4-SP Assembler consists of four separate robots, poses can be generated by applying kinematic operations on each constituent SP separately, then stacking them, resulting in a full pose for the Assembler.

Pose generation for the individual SPs came in two varieties: Uniform and Extreme. Uniform poses were generated by applying the FK algorithm to a SP with a set of leg lengths uniformly generated from their minimum and maximum extensions. Configurations that resulted in an error or were at the "home" position (via error correction) were discounted, and iteration continued until a set number of poses (for the purpose of this paper, 10,000) were generated. Extreme poses followed the same procedure, with the added caveat that poses which did not meet a minimum measure of rotation magnitude of 30-degrees were also rejected, leaving only poses with the requisite tilt factor. Tilt was selected as being more important than translation because a tilt in one platform has a much greater change potential in a stack end effector than does translation.

Once individual SP poses were completed, the 4-SP stacked poses could be constructed. There were three categories of poses which we trialed: Uniform, Extreme, and Repeated. Uniform poses drew from the aforementioned uniform individual poses, while Extreme drew from the extreme poses. For both, four poses are chosen at random from their constituent files and applied to create a stacked pose. The end effector position (topmost plate of the topmost SP) is recorded, along with the plate positions and leg lengths of all constituent platforms. The Repeated poses differ slightly in generation. They too draw from the extreme pose file, but only one individual SP pose is chosen, and is applied to each platform in the stack, such that the ultimate pose is severely biased towards tilt in the direction of the constituent SP pose. For the purposes of this analysis, each type of pose generator produced three files consisting of 100, 1,000, and 10,000 poses respectively. The 100 pose file is intended purely for testing, whereas the two larger datasets for each type were earmarked for analysis.

5.2 Pose Optimization

In this section, we compare the SIK and OPT methods for solving the IK problem on the instances generated in Section 5.1. We also include the force-related information for the initially generated poses (GEN) as a reference.

Table 1 summarizes the performance of the SIK and OPT methods on various instances, while Table 2 summarizes some additional performance characteristics determined during meta-analysis. A pose is said to be *valid* the end effector and base plates are in the correct pose, and all constraints related to kinematic validity are satisfied. A valid pose is said to be *force-valid* if the maximum-force constraints are also satisfied. The Avg % and Avg Max % Reduced metrics measure the percent reduction of forces out of the instances for which both methods yield kinematically valid poses. The % Improved metric measures the percentage of poses for which OPT yielded a better pose then SIK, out of the poses for which at least one of the methods yielded a valid pose. Finally, the Avg OPT Time metric measures the average time required for the OPT method to terminate, regardless of termination status.

Dataset	Solver	Valid poses	Force-valid poses
Uniform	GEN	10,000	627
	SIK	9,048	4,416
	OPT	10,000	9,895
Extreme	GEN	10,000	323
	SIK	8,905	3,459
	OPT	10,000	9,903
Repeated	GEN	10,000	1479
	SIK	2,191	187
	OPT	10,000	8,317

Table 1: Performance metrics for SIK and OPT, out of 10,000 random poses.

Dataset	Avg % Reduced	Avg Max % Reduced	% Improved	Avg OPT Time
Uniform	34.03%	57.4%	100%	1.15s
Extreme	33.85%	58.51%	100%	1.18s
Repeated	37.81%	64.67%	100%	1.19s

Table 2: Improvement of OPT over SIK, along with average computational time for 10,000 random poses.

Note that the optimizer is always able to find kinematically valid poses when initialized

via the same-SP approach, even when the initial guesses are not kinematically valid. This effectiveness is especially noticeable for poses from the difficult Repeated datasets, for the success rate from SIK was only about 22%.

Figure 5 showcases the force-performance for the SIK and OPT approaches. From the figure, note that, particularly for the repeated dataset, the problems yielding kinematically valid poses for SIK are significantly less likely to yield force-invalid poses in OPT.

Figure 6 showcases the computational performance of the optimizer over the Extreme and Repeated instances. The Uniform performance plot is omitted due to its strong similarity to the Extreme plot, but with one outlier requiring more than 16s. Note that the time-performance for the Repeated poses is more polarized than for other datasets: ∼55% (vs. ∼40%) of poses solve in less than 1s, but ∼10% (vs. ∼3%) require more than 2s to solve.

From Table 1 and Figure 5, we see a very strong degree of improvement in the resulting forces compared to the SIK heuristic. The forces are at least halved between 70-80% of the time, with improvement factors over 8 in some cases. This improvement, combined with the fast computation times observed in Figure 6, renders the OPT approach a viable method for the generation of force-optimized poses for SPs. Note that, in terms of forces, even the SIK approach was typically able to find much better poses than were initially generated, when it succeeded in generating a kinematically valid pose.

5.3 Trajectory Planning

In this section, we demonstrate the effectiveness of the trust region method, then showcase how this method can be used in conjunction with RRT* to obtain good obstacle-avoiding paths very quickly after pre-processing.

To demonstrate the effectiveness of the trust region method, we showcase a difficult motion: a transition between opposite bent-over poses. We showcase using the thin-plated SP with $\varepsilon^{\text{pos}} = 0.2$, $\varepsilon^{\text{rot}} = \frac{\pi}{3}$, $\lambda^{\text{force}} = 0.04$, $\lambda^{\text{pose}} = 0.96$, and $\lambda^{\text{avg}} = 0.05$. The sample motion, along with a plot showing the progression of the motion in terms of the forces and the maximum plate global-frame distances from any plate to the ending pose, is shown in Figure 7.

To demonstrate the consistency and performance of the trust region method, we compare the trust region method with the direct transformation and the RRT* approach.

A comparison trajectory planning results for all datasets are shown in Table 3. The settings used for the trust-region method were $\varepsilon^{\text{pos}} = 0.1$, $\varepsilon^{\text{rot}} = \frac{\pi}{6}$, $\lambda^{\text{force}} = 0.04$, $\lambda^{\text{pose}} = 0.96$, and $\lambda^{\text{avg}} = 0.05$. Some primary metrics of solution are given as

1. Behaved: A path is considered to be behaved if the maximum leg forces mid-motion do not exceed the maximum forces at the starting or ending pose.

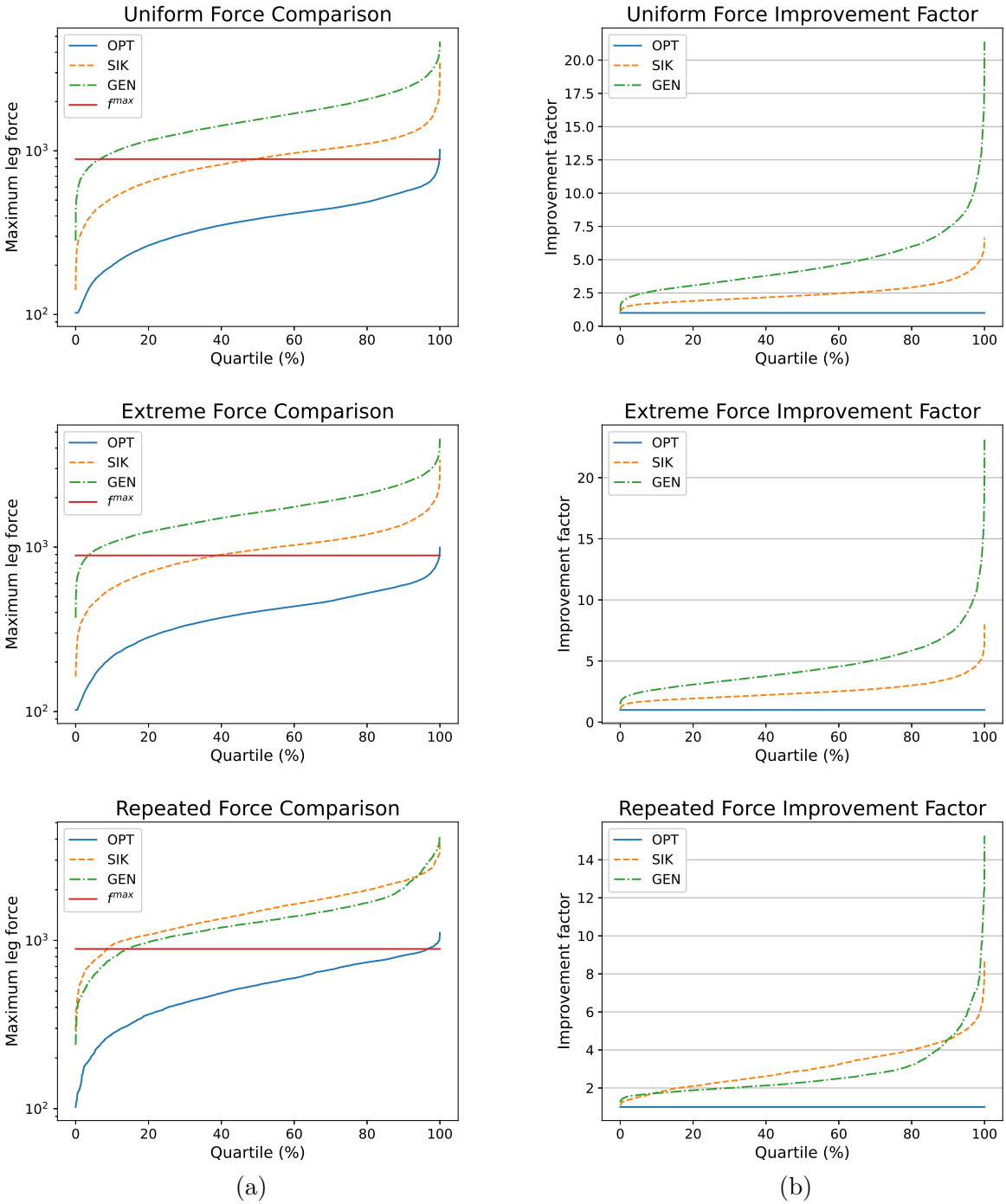


Figure 5: Pose comparison of OPT, GENerated, and SIK forces, considering the poses for which both approaches yielded kinematically valid poses. (a) Comparison of forces, (b) Factor of improvement of OPT over each method. Forces are in Newtons.

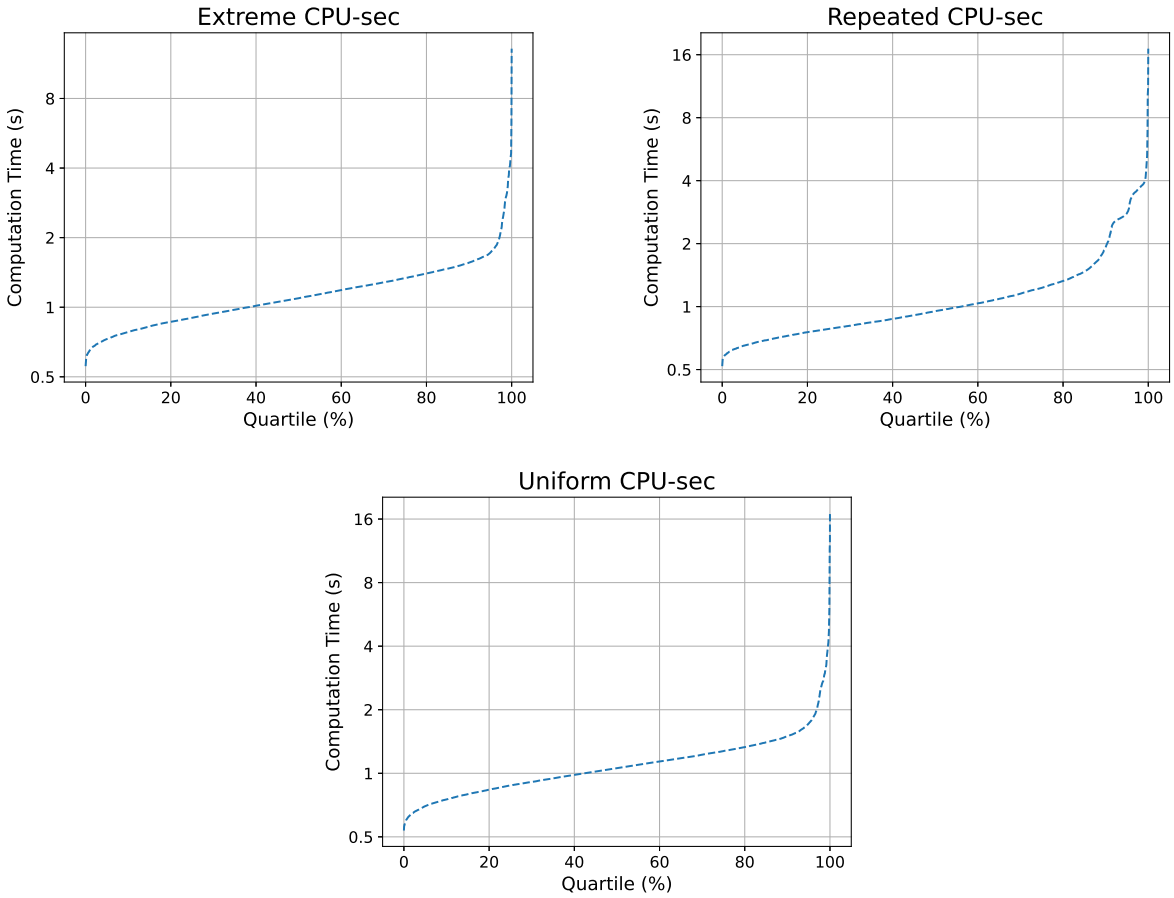


Figure 6: Computation times for OPT over 10,000 poses for the Extreme and Repeated instance families.

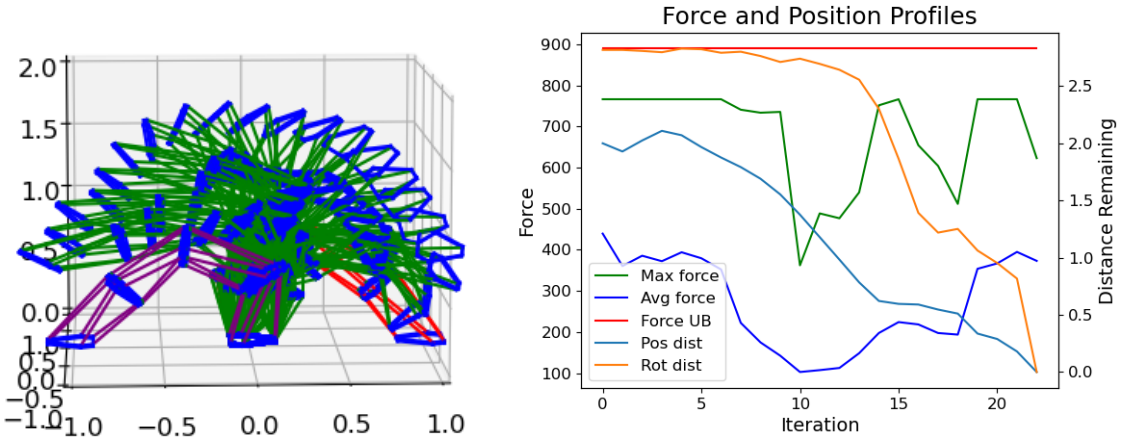


Figure 7: Force-controlled transition between bent-over poses using trust region method. (Left) Axes units are in meters. (Right) Force in Newtons.

2. Force-Valid: A path is considered to be force-valid if either all leg forces are valid throughout the motion, or if the path is behaved with excessive leg forces at either the starting or ending pose.

3. Avg OPT Time: The average optimization time across all tests.

Dataset	Method	% Behaved	% Force-Valid	Avg OPT Time
Uniform	Trust	100%	100%	15.06s
	Naïve	1.0%	78.5%	7.79s
Extreme	Trust	99.9%	100%	15.35s
	Naïve	1.3%	75.1%	8.34s
Repeated	Trust	99.8%	100%	22.20s
	Naïve	0.3%	19.7%	9.08s

Table 3: Comparison of trajectory planning results for trust-region vs naïve FK, out of 1000 random poses.

We show a more complete picture of the maximum leg forces resulting from the two approaches in Figure 8, and compare the motion energies in Figure 9. To estimate the motion energies, we assume that extending a leg under tensile forces and contracting a leg under compression forces are free operations, and count only the portions of the motions that require energy input.

The results reveal that the trust region method is far more consistent, with maximum forces only rarely exceeding those at the end effectors (in 0.1% of cases), and with maximum-force improvements as high as a factor of 5. However, the gains in maximum forces and consistency come at a price: the motion energies tend to be far higher, exceeding the more direct motions by as much as a factor of 7. Furthermore, the naïve approach solves roughly twice as quickly as the trust region approach. Thus, in practice, we recommend first computing the naïve motion, then applying the trust region method if the motion is either near-infeasible or very poorly behaved, with mid-motion max-forces far higher then endpoint max-forces.

Note that, in practice, any leg motion requires some baseline motor energy usage even under zero net forces, which would further strengthen the energy advantage gained from the shorter naïve motions.

5.4 Range of Motion Analysis

Using a stochastic method, we analyzed the range of motion for the SIK method for both the thin and thick-plated SPs. The results are shown in Figure 10.

For a single SP, the simplified range of motion (ROM) methodology is to create a number of concentric scaled unit spheres centered at the platform's rest pose (a sphere with a certain number of points evenly distributed about its exterior surface) and calculate IK for each point, along with perturbing the rotations at the extremes. Points that succeed are admitted into the ROM cloud, and those that do not are discarded. The final ROM graphic is created by utilizing the technique Alpha-Shapes in order to create a concave hull illustrating the reachability of the platform.

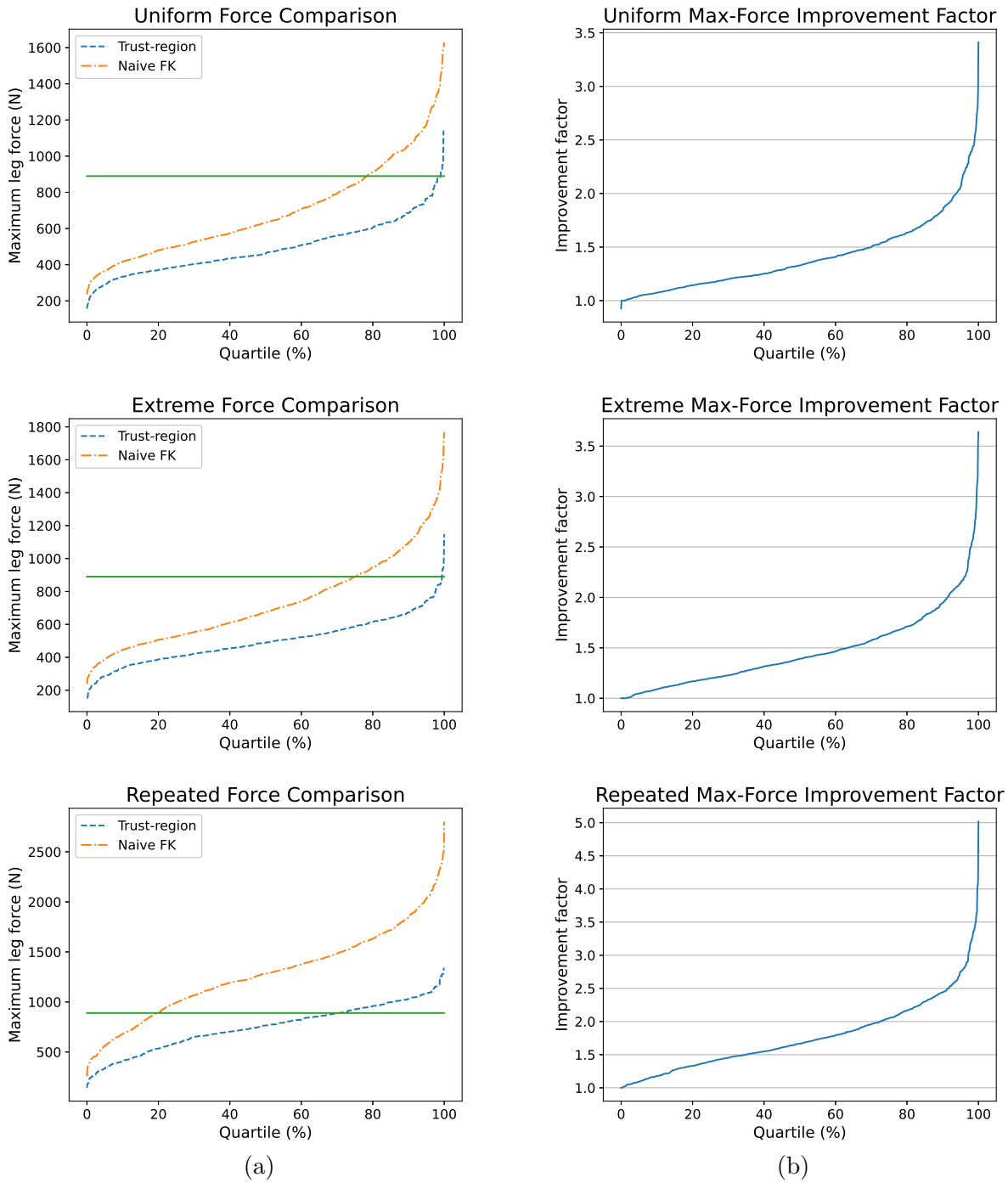


Figure 8: Comparison of maximum forces between trajectory planning methods. (a) Comparison of maximum forces, (b) Factor of improvement of trust-region method over naïve interpolated FK.

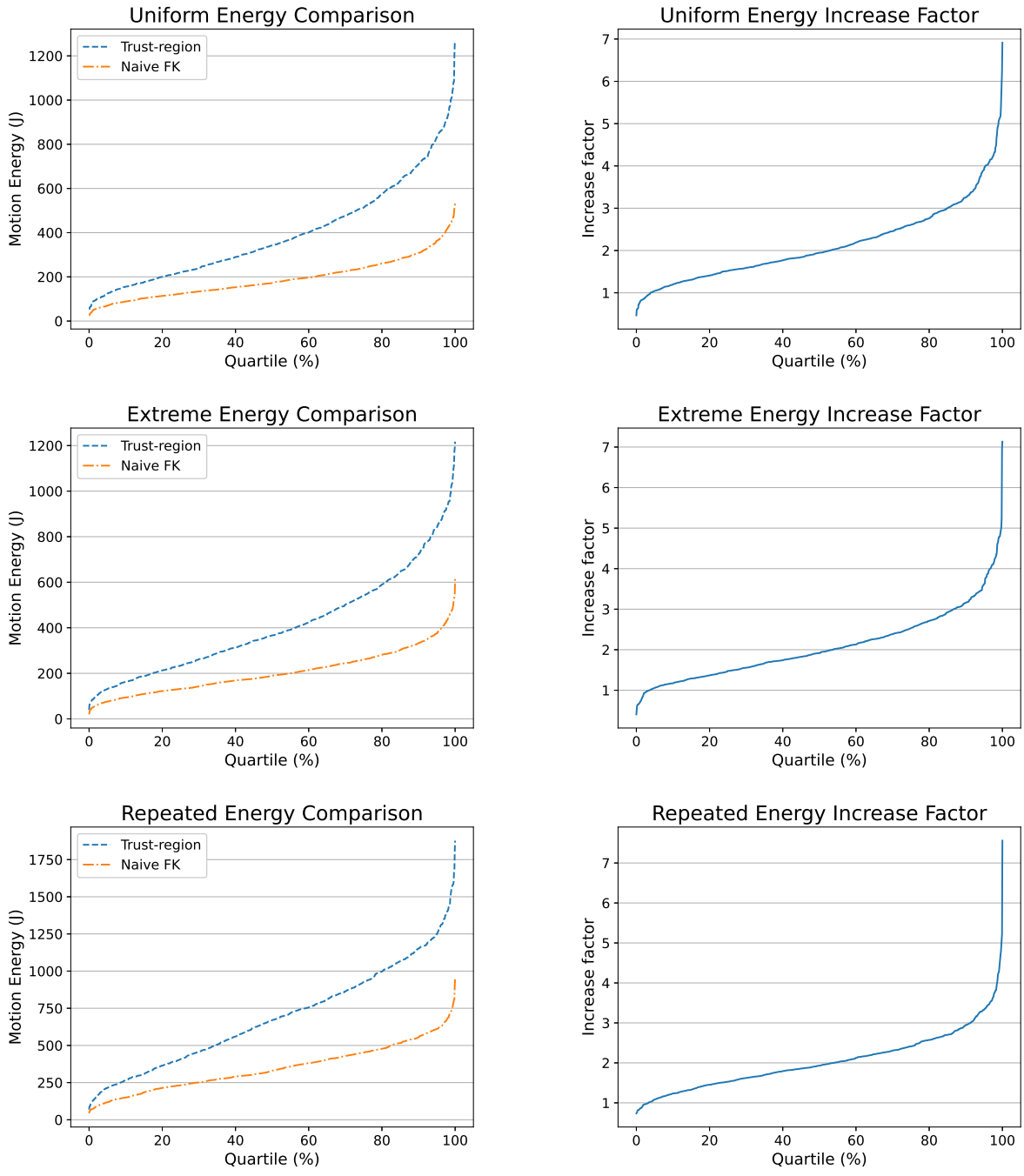


Figure 9: Comparison of motion energies between trajectory planning methods. (a) Comparison of motion energies, (b) Extra-energy factor of trust-region method over naïve interpolated FK.

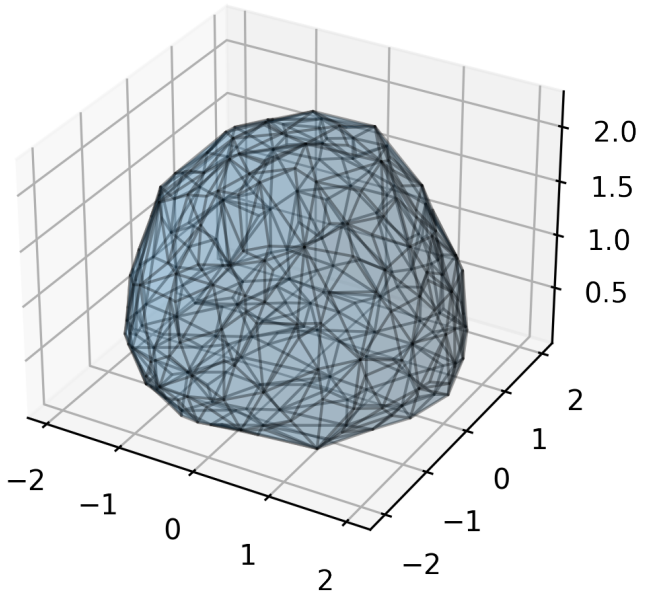


Figure 10: Range of motion approximation for the test Assembler used in our computational experiments. Axes units are in meters. Plot shows points where the end-effector can be located. Note that there is a hole in the middle of this image since certain shorter end-effector points are not reachable.

6 Conclusion

We have presented a fast approach for the force-optimization of stacked SPs that can significantly improve their range of motion under heavy loads. In particular, the optimization step can result in reductions of the worst-case leg forces by as much as an order of magnitude in some cases. The model of forces used in this work has been verified via hardware testing, showing that the modeled and real forces are very similar under rotation. Further, we have presented a fast, consistent trust-region trajectory planning approaches based on this pose optimization scheme, and demonstrated the effectiveness of the approach in comparison to a simple length-based optimization approach.

In the future, the ultimate goal is to use these stacked SP structures to perform automated assembly operations. To improve on the pose optimization approach, we suggest exploring the use of more robust heuristics in conjunction with the local optimizer proposed in this work. For example, to improve pose optimization given fixed base and end-effector positions, one could quickly compute rough bounds on the possible poses for the middle plates, then apply heuristics such as a genetic algorithm or a bidirectional search, with frequent or intermittent local optimization steps, to quickly hone in on a solution. Alternatively, one could construct some fast heuristic for generating randomized feasible or near-feasible poses for a fixed end-effector position, then locally optimize a moderate number of random poses in parallel to seek a stronger solution with limited additional computational cost.

Lastly, it may be possible to make our objective function for the trajectory optimization more realistic. Our objective tracks the amount of energy used over time. However, since the amperage needed for expansion and contraction of the leg joints is a nonlinear function of the movement, we may be able to study this function using our hardware to model a better objective function that reflects the amperage used for each movement. This will require further hardware experimentation.

Acknowledgements

Some of this material is based on work supported by NASA via contract 80LARC20P0020 for the “Assemblers: A modular and reconfigurable manipulation system for autonomous in-space assembly” project.

Additionally, this material was also partially based on work sponsored by the Northrop Grumman Undergraduate Research Experience in Industrial & Systems Engineering at Virginia Tech project. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Northrop

Grumman Corporation.

R. Hildebrand and B. Beach are funded by AFOSR grant FA9550-21-0107. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the the Air Force Office of Scientific Research.

References

[1] Balaban, D., Cooper, J., and Komendera, E. (2019). Inverse kinematics and sensitivity minimization of an n-stack Stewart platform. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6794–6799. IEEE.

[2] Bangjun, L., Likun, P., and Tingtao, M. (2012). Improving dynamic performance of Stewart platforms through optimal design based on evolutionary multi-objective optimization algorithms. In *Proceedings of the 1st International Conference on Mechanical Engineering and Material Science*, pages 294–298. Atlantis Press.

[3] Bingul, Z. and Karah, O. (2012). Dynamic modeling and simulation of Stewart platform. In *Serial and Parallel Robot Manipulators - Kinematics, Dynamics, Control and Optimization*, pages 19–42. InTech.

[4] Bynum, M. L., Hackebeil, G. A., Hart, W. E., Laird, C. D., Nicholson, B. L., Siirola, J. D., Watson, J.-P., and Woodruff, D. L. (2021). *Pyomo–optimization modeling in python*, volume 67. Springer Science & Business Media, third edition.

[5] Charters, T., Enguica, R., and Freitas, P. (2009). Detecting singularities of Stewart platforms. *Mathematics-in-Industry Case Studies Journal*, 1:66–80.

[6] Chen, H., Chen, W., and Liu, J. (2007). Optimal design of Stewart platform safety mechanism. *Chinese Journal of Aeronautics*, 20(4):370–377.

[7] Cortes, J. and Simeon, T. (2003). Probabilistic motion planning for parallel mechanisms. In *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, pages 4354–4359. IEEE.

[8] Dorsey, J., Sutter, T., and Wu, K. (1992). Structurally adaptive space crane concept for assembling space systems on orbit. Technical report, NASA Langley Research Center.

[9] Dragan, A. and Srinivasa, S. (2014). Integrating human observer inferences into robot motion planning. 37(4):351–368.

[10] Ernandis, R. (2021). *Sampling Based Motion Planning for Minimizing Position Uncertainty with Stewart Platforms*. PhD thesis, University of Maryland, College Park.

[11] Grosch, P., Gregorio, R. D., Lopez, J., and Thomas, F. (2010). Motion planning for a novel reconfigurable parallel manipulator with lockable revolute joints. In *2010 IEEE International Conference on Robotics and Automation*, pages 4697–4702. IEEE.

[12] Hart, W. E., Watson, J.-P., and Woodruff, D. L. (2011). Pyomo: modeling and solving mathematical programs in python. *Mathematical Programming Computation*, 3(3):219–260.

[13] Ichnowski, J., Avigal, Y., Satish, V., and Goldberg, K. (2020). Deep learning can accelerate grasp-optimized motion planning. 5(48).

[14] Islam, F., Nasir, J., Malik, U., Ayaz, Y., and Hasan, O. (2012). RRT*-Smart: Rapid convergence implementation of RRT* towards optimal solution. IEEE.

[15] Karaman, S. and Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. 30(7):846–894.

[16] Kuffner, J. and LaValle, S. (2000). RRT-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 2, pages 995–1001 vol.2. IEEE.

[17] LaValle, S. M. (1998). Rapidly-exploring random trees : a new tool for path planning. *The annual research report*.

[18] Lazard, D. and Merlet, J.-P. (1994). The (true) Stewart platform has 12 configurations. volume 3, pages 2160 – 2165 vol.3.

[19] Lei, Z. and Xiaolin, D. (2013). Optimize the redundant 6-DOF Stewart platform based on ant colony optimization. In *Proceedings of 2013 3rd International Conference on Computer Science and Network Technology*, pages 1238–1241. IEEE.

[20] Li, Y.-W., Wang, J.-S., and Wang, L.-P. (2002). Stiffness analysis of a Stewart platform-based parallel kinematic machine. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, volume 4, pages 3672–3677.

[21] Lynch, K. M. and Park, F. C. (2017). *Modern robotics: mechanics, planning, and control*. Cambridge University Press, Cambridge, UK. OCLC: ocn983881868.

[22] Majid, M. Z. A., Huang, Z., and Yao, Y. L. (2000). Workspace analysis of a six-degrees of freedom, three-prismatic- prismatic-spheric-revolute parallel manipulator. *The International Journal of Advanced Manufacturing Technology*, 16(6):441–449.

[23] Merlet, J. P. (2006). *Parallel Robots*. Springer.

[24] Miura, K. and Furuya, H. (1988). Adaptive structure concept for future space applications. *AIAA Journal*, 26(8):995–1002.

[25] Nguyen, C., Antrazi, S., Zhou, Z.-L., and Campbell, C. (1991). Experimental study of motion control and trajectory planning for a Stewart Platform robot manipulator. In *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, volume 2, pages 1873–1878 vol.2. IEEE Comput. Soc. Press.

[26] Nguyen, C. C. and Antrazi, S. (1990). Trajectory planning and control of a 6 dof manipulator with Stewart platform-based mechanism. Technical report, NASA Goddard Space Flight Center.

[27] Osa, T., Esfahani, A. M. G., Stolkin, R., Lioutikov, R., Peters, J., and Neumann, G. (2017). Guiding trajectory optimization by demonstrated distributions. 2(2):819–826.

[28] Quintero-Pena, C., Kyrillidis, A., and Kavraki, L. E. (2021). Robust optimization-based motion planning for high-DOF robots under sensing uncertainty. IEEE.

[29] Ríos, A., Hernández, E. E., and Valdez, S. I. (2021). A two-stage mono- and multi-objective method for the optimization of general UPS parallel manipulators. *Mathematics*, 9(5):543.

[30] Santos, J. C. and da Silva, M. M. (2017). Investigation of motion planning methods with a kinematically redundant manipulator.

[31] Schulman, J., Duan, Y., Ho, J., Lee, A., Awwal, I., Bradlow, H., Pan, J., Patil, S., Goldberg, K., and Abbeel, P. (2014). Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, 33(9):1251–1270.

[32] Sun, T. and Lian, B. (2018). Stiffness and mass optimization of parallel kinematic machine. *Mechanism and Machine Theory*, 120:73–88.

[33] Szynkiewicz, W. and Błaszczyk, J. (2011). Optimization-based approach to path planning for closed chain robot systems. 21(4):659–670.

[34] Toz, M. and Kucuk, S. (2013). Dexterous workspace optimization of an asymmetric six-degree of freedom Stewart–Gough platform type manipulator. *Robotics and Autonomous Systems*, 61(12):1516–1528.

[35] Virtanen, P. et al. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*.

[36] Volz, A. and Graichen, K. (2018). An optimization-based approach to dual-arm motion planning with closed kinematics. IEEE.

[37] Wächter, A. and Biegler, L. T. (2005). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. 106(1):25–57.

[38] Wang, P., Yang, H., and Xue, K. (2015). Jerk-optimal trajectory planning for Stewart platform in joint space. In *2015 IEEE International Conference on Mechatronics and Automation (ICMA)*, pages 1932–1937. IEEE.

[39] Williams, R. L. (1995). Survey of active truss modules. In *Volume 1: 21st Design Automation Conference*. American Society of Mechanical Engineers.

[40] Xie, Z., Li, G., Liu, G., and Zhao, J. (2017). Optimal design of a Stewart platform using the global transmission index under determinate constraint of workspace. *Advances in Mechanical Engineering*, 9(10):1687814017720880.

[41] Yokoi, K., Komoriya, K., and Tanie, K. (1992). A method for solving inverse kinematics of variable structure truss arm with high redundancy. *Journal of Intelligent Material Systems and Structures*, 3(4):631–645.

[42] Zhang, B. (2005). *DESIGN AND IMPLEMENTATION OF A 6 DOF PARALLEL MANIPULATOR WITH PASSIVE FORCE CONTROL*. PhD thesis, University of Florida.

[43] Zucker, M., Ratliff, N., Dragan, A., Pivtoraiko, M., Klingensmith, M., Dellin, C., Bagnell, J. A. D., and Srinivasa, S. (2013). Chomp: Covariant hamiltonian optimization for motion planning. *International Journal of Robotics Research*, 32(9):1164 – 1193.