Thesis Title

A thesis submitted to the University of Manchester for the degree of
Doctor of Philosophy
in the Faculty of Science and Engineering

2022

Damian Crosby
Department of Mechanical, Aerospace and Civil Engineering

Contents

Word Count: X

List of figures

List of tables

List of publications

Publications go here.

2D	Two Dimensional
3D	Three Dimensional
COM	Centre of Mass
DLS	Damped Least Squares

Abstract

This is abstract text.

Declaration of originality

I hereby confirm that no portion of the work referred to in the thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright statement

Acknowledgements

Acknowledgements go here.

Chapter 1

Creating a Configurable Payload for Instability Experiments

1.1 Introduction

In order to generate a diverse set of test data for the experiments in chapter **??**, a configurable payload was conceived, an object that could be configured to have a wide range of masses and COM. A series of test points can then be generated which have a specific mass and COM, and a matching algorithm can be used to find the configuration that mostly closely matches these parameters. The experiments in chapter **??** can then be run with each of these test points to generate the test data.

1.2 Payload Design

The payload consists of a matrix of cubes of various materials packed tightly into a Three Dimensional (3D) printed container. The cubes are designed to be changed after each experimental run to alter the mass and COM of the payload. A lid on the container prevents the cubes from falling out during the experiment, and the exterior design of the box may accommodate additional features to improving the handling of the payload by the robot arm.

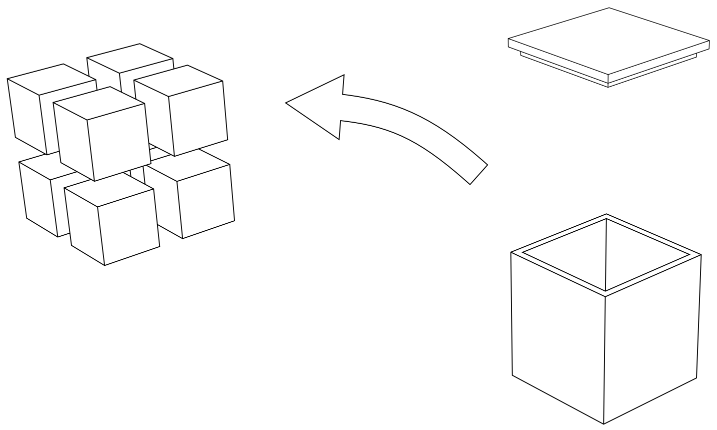

Figure 1.1: Concept drawing of the configurable payload.

Material	Variant	Density (kg m^{-3})
Wood	Pine	860
Plastic	Acrylic	1200
Aluminium	6082	2700
Steel	EN3B	8060

Table 1.1: The materials chosen for the cubes.

1.3 Payload Configuration Space

In order to find a configuration that closely matches a desired test point, first the payload has to be abstracted mathematically, so the mass and COM can be calculated for a given configuration. Firstly we can consider a positive real set of material densities $\mathcal{P} \in \mathbb{R}^+$, each element the density (in kg m^{-3}) of a material to be used:

$$\mathcal{P} = \{\rho_1, \rho_2 \ldots \rho_n \mid \rho_i > 0\} \tag{1.1}$$

Each configuration can then be defined as an $n \times n \times n$ matrix $\boldsymbol{C}$, such that each element is an element of $\mathcal{P}$, where n^3 is the number of cubes in the matrix:

$$\boldsymbol{C} = (c_{ijk}) \in \mathbb{R}^{n^n} \mid (c_{ijk}) \in \mathcal{P} \tag{1.2}$$

1.3.1 Mass and COM functions

To calculate the mass of the configuration, we can take the sum of all the cube densities multiplied by their volume a^3, where a is the cube edge length, plus the container mass m_c:

$$M(\boldsymbol{C}) = \left(\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} c_{ijk}a^3\right) + m_c \tag{1.3}$$

To calculate the COM, we can take the sum of each cube mass multiplied by its position relative to the centroid of the center cube (c_{222}), which can be calculated from the cube indexes ijk, plus the container COM $\boldsymbol{r}_c$ if non-zero:

$$R(\boldsymbol{C}) = \frac{\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} c_{ijk}a^4\left(\begin{bmatrix} i & j & k\end{bmatrix} - n + 1\right)\right) + \boldsymbol{r}_c}{M(\boldsymbol{C})} \tag{1.4}$$

1.4 Test Points

With $\mathcal{Z}$ as the set of all permutations of $\boldsymbol{C}$, test points can be derived from subsets of $\mathcal{Z}$ defined either by logical expressions on $\mathcal{Z}$, or the nearest neighbours of $\mathcal{Z}$ from a set of coordinates. A test point is defined as a $\mathbb{R}^4$ vector containing the target mass and COM concatenated as $\begin{bmatrix} m_i & \boldsymbol{r}_i \end{bmatrix}$.

1.4.1 Extrema Set ($\mathcal{E}$)

The extrema set is designed to test the extremas of the space of $\mathcal{Z}$ for both $M(\mathcal{Z})$ and $R(\mathcal{Z})$. The extrema set if defined from a set of logical constraints. The first two constraints of the set find the maximum and minimum values of the payload mass using $M(\mathcal{Z})$, and the next four constraints use the payload COM using $M(\mathcal{C})$ to get the maximum and minimum values of the x and y component of the COM. Finally, the last four constraints define the diagonal maximum and minimum values where the COM components match $x = y$ or $x = -y$.

$$\mathcal{E} = \left\{ \boldsymbol{x} \subset \mathcal{Z} \mid \begin{cases} M(\boldsymbol{x}) = \max\{M(\mathcal{Z})\} \\ M(\boldsymbol{x}) = \min\{M(\mathcal{Z})\} \\ R(\boldsymbol{x})_x = \max\{R(\mathcal{Z})_x\} \\ R(\boldsymbol{x})_x = \min\{R(\mathcal{Z})_x\} \\ R(\boldsymbol{x})_y = \max\{R(\mathcal{Z})_y\} \\ R(\boldsymbol{x})_y = \min\{R(\mathcal{Z})_y\} \\ R(\boldsymbol{x})_x = \max\{R(\mathcal{Z})_x\} \wedge R(\boldsymbol{x})_x = R(\boldsymbol{x})_y \\ R(\boldsymbol{x})_x = \min\{R(\mathcal{Z})_x\} \wedge R(\boldsymbol{x})_x = R(\boldsymbol{x})_y \\ R(\boldsymbol{x})_x = \max\{R(\mathcal{Z})_x\} \wedge R(\boldsymbol{x})_x = -R(\boldsymbol{x})_y \\ R(\boldsymbol{x})_x = \min\{R(\mathcal{Z})_x\} \wedge R(\boldsymbol{x})_x = -R(\boldsymbol{x})_y \end{cases} \right\} \tag{1.5}$$

Mass Limited $\mathcal{E}$

When this set was generated, it was found that $M(\mathcal{E}_1)$ was greater than the chosen robot arm could safely lift. Therefore, $M(\boldsymbol{x}) = \max\{M(\mathcal{Z})\}$ in $\mathcal{E}$ was changed to $\begin{bmatrix} m_{max} & 0 & 0 & 0 \end{bmatrix}$ where m_{max} is the safe mass limit that the robot arm can lift. One of the search methods described in section 1.5 can then be used to find the nearest point in configuration space.

1.4.2 Cube Set ($\mathcal{C}$)

The cube set is defined by the vertices of a cube of size b centred around the COM origin $\begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$.

$$\mathcal{C} = \left\{ \boldsymbol{x} \subset \mathcal{C} \mid \begin{bmatrix} \pm\frac{b}{2} & \pm\frac{b}{2} & \pm\frac{b}{2} \end{bmatrix} = \boldsymbol{x} \right\} \tag{1.6}$$

1.4.3 Balanced Set ($\mathcal{B}$)

The balanced set is defined by q points in $\mathcal{C}$ subject to the constraint $R(\boldsymbol{x})_x = 0 \land R(\boldsymbol{x})_y = 0$. This can be defined as a “balanced” set as the COM x and y components are both zero. The points are evenly spaced between the maximum and minimum mass as defined in section 1.4.1.

$$\begin{aligned} m_r &= \frac{\max\{M(\mathcal{Z})\} - \min\{M(\mathcal{Z})\}}{q+1} \\ \boldsymbol{z} &= \begin{bmatrix} m_r & 2m_r & \cdots & qm_r \end{bmatrix} \\ \mathcal{B} &= \{\boldsymbol{x} \subset \mathcal{C} \mid z_i = \boldsymbol{x} \land R(\boldsymbol{x})_x = 0 \land R(\boldsymbol{x})_y = 0\} \end{aligned} \tag{1.7}$$

1.5 Test Point Matching Search Methods

While we are guaranteed an exact result for elements of $\mathcal{E}$ as every element is defined by constraints on known configurations, for $\mathcal{C}$ and $\mathcal{B}$ as the elements are defined numerically, there is no guarantee that any element will have an exact match in solution space. The cardinality of $\mathcal{Z}$ is also important. It is defined as $|\mathcal{Z}| = |\mathcal{P}|^{n^n}$ which increases super exponentially with n. For example, when $|\mathcal{P}| = 4$, $n = 2$ results in 256 permutations and $n = 3$ results in approximately 1.8×10^{16} permutations. It’s very clear that when $n > 2$ for non-trivial cardinalities of $\mathcal{P}$, any kind of brute-force method is not computationally tractable. The only exception is for $\mathcal{E}_1$ and $\mathcal{E}_2$, where the solution is trivial as $\mathcal{E}_1 = \boldsymbol{C} \mid \min(\mathcal{P}) \forall c_{ijk}$ and $\mathcal{E}_2 = \boldsymbol{C} \mid \max(\mathcal{P}) \forall c_{ijk}$. Therefore, we investigated both a brute-force nearest neighbour method for when $n = 2$, and a simulated annealing method for when $n = 3$.

1.5.1 Simulated Annealing ($n = 3$)

Simulated annealing [1] is a modification to a gradient descent optimisation that allows the algorithm the chance to “jump out” of local minima early on (even though the approximation becomes temporarily worse). However, as the number of remaining steps decreases, that probability becomes smaller, becoming more and more like gradient descent. First, like any gradient descent algorithm, two things need to be generated, the initial configuration $\boldsymbol{C_0}$, which can be random or manually selected, and the function $\mathcal{N}(\boldsymbol{C})$ which creates a set of all the “neighbours” of $\boldsymbol{C}$. In this case, this can be defined as the subset of $\mathcal{Z}$ where the difference between $\boldsymbol{C}$ and an element of $\mathcal{N}(\boldsymbol{C})$ is one and only one $c_{ijk} \neq c_{ijk}$:

$$\mathcal{N}(\boldsymbol{C}) = \{x \subset \mathcal{Z} \mid \exists!\,(x_{ijk} \neq c_{ijk})\} \tag{1.8}$$

Then the simulated annealing function can be described as follows:

1. Set $\boldsymbol{C}$ to the initial permutation $\boldsymbol{C_0}$.

2. For each of the optimisation steps:

   (a) Set the temperature value t with function $T\left(\frac{k_{max}}{k}\right)$ which takes into account the number of remaining steps.

   (b) Set $\boldsymbol{C}_{new}$ as a random element from the set of all neighbours of $\boldsymbol{C}$ as defined by $\mathcal{N}(\boldsymbol{C})$.

   (c) Use acceptance probability function $P(E(\boldsymbol{C}), E(\boldsymbol{C}_{new}), t)$ where $E(\boldsymbol{C})$ is the energy function (in this case the absolute difference $E(\boldsymbol{C}, g) = |M(\boldsymbol{C}) - g|$ or $E(\boldsymbol{C}, g) = |R(\boldsymbol{C}) - g|$ where g is the target could be used to look for a lower energy state in either mass or COM) to generate a value. Note this function is dependent on the temperature t.

   (d) Compare that value with a random uniformly distributed real number between 0 and 1. If greater than or equal to, then replace $\boldsymbol{C}$ with $\boldsymbol{C}_{new}$. Otherwise, keep it the same.

   (e) Repeat with $\boldsymbol{C}$ until there are no remaining steps.

3. Return the approximated permutation $\boldsymbol{C}$.

```
$\boldsymbol{C} = \boldsymbol{C}_0$
for $k \leftarrow 1, k_{max}$ do
    $t = T\left(\frac{k_{max}}{k}\right)$
    $\boldsymbol{C}_{new} = \mathcal{N}(\boldsymbol{C}) \overset{R}{\leftarrow} x$
    if $P(E(\boldsymbol{C}), E(\boldsymbol{C}_{new}), t) \geq x \sim U([0,1])$ then
        $\boldsymbol{C} = \boldsymbol{C}_{new}$
    end if
end for
return $\boldsymbol{C}$
```

Using this algorithm an array of approximate configurations can be easily generated from a desired array of test points. Simulated annealing can also be adapted for multi-objective optimisation [2], so it is possible to generate test points that approximate a desired mass and COM simultaneously.

Cooling Function

1.5.2 Brute-Force Nearest Neighbour ($n = 2$)

If $\mathcal{Z}$ is suitably small, as it is when $n = 2$ then a brute-force method can be used which is guaranteed to find the nearest element to the target within a finite time. This can be done by calculating the L2 norms between the target vector t and all the elements of $\mathcal{Z}$ and finding the minimum. If several elements of $\mathcal{Z}$ have the minimum norm, then one is chosen at random from this set.

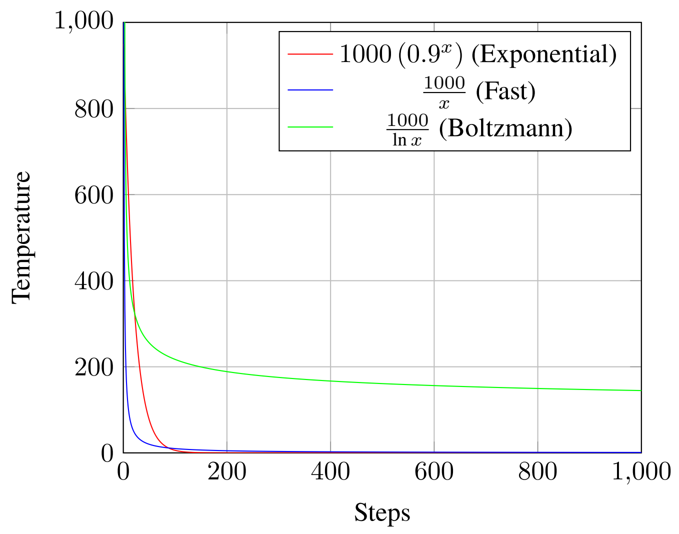

Figure 1.2: Various temperature cooling profiles for simulated annealing, assuming 1000 steps.

$$NN\left(t, \mathcal{Z}\right) = \min\left\{\left\|t - x\right\|_2 \mid \forall x \in \mathcal{Z}\right\} \tag{1.9}$$

1.5.3 Selected Method

Initially an $n = 3$ configuration was used with the acceptance probability function *Rule M* from [2]. This is a weighted blend of two other algorithms defined in the paper, *Rule P* and *Rule W* with a weighting coefficient $\alpha \in (0, 1) \subset \mathbb{R}$. There is also a weighting vector for each element of the test point $\boldsymbol{w} \in \mathbb{R}^4 \mid w_i \in (0, 1)$.

$$P\left(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w}, t\right) = \alpha \underbrace{\prod_{i=1}^{m} \min\left\{1, e^{\frac{w_i\left(x_i - y_i\right)}{t}}\right\}}_{\text{Rule P}} + \left(1 - \alpha\right) \underbrace{\min\left\{1, \max_{i=1,\ldots,m}\left\{1, e^{\frac{w_i\left(x_i - y_i\right)}{t}}\right\}\right\}}_{\text{Rule W}} \tag{1.10}$$

Unfortunately it was difficult to find a stable and consistent result even after a long time running the algorithm.

Therefore as an alternative, the $n = 2$ configuration was used, with larger cubes to compensate. This successfully produced all three test point sets, after some small adjustments of b in order to get closer to a single configuration for each point of $\mathcal{C}$.

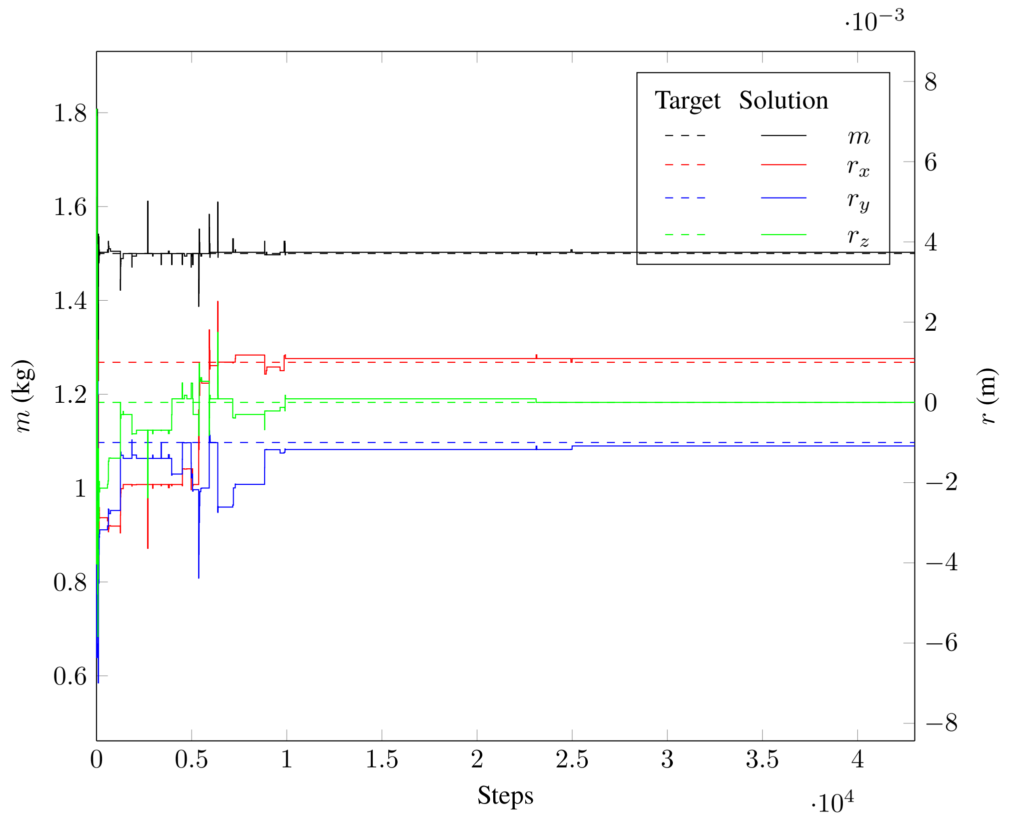

Figure 1.3: Simulated annealing output for the target $[1.5 \quad 0.001 \quad 0.001 \quad 0.001]$ with $\alpha = 0.997$ and even weighting $w_m, \boldsymbol{w_r} = 0.25$ for 4.3×10^4 steps.

m	$\boldsymbol{r}$
Extrema Set ($\mathcal{E}$)	
1.751	$\begin{bmatrix}0.012 & 0.000 & 0.000\end{bmatrix}$
1.751	$\begin{bmatrix}0.000 & 0.012 & 0.000\end{bmatrix}$
1.751	$\begin{bmatrix}-0.012 & 0.000 & 0.000\end{bmatrix}$
1.751	$\begin{bmatrix}-0.000 & -0.012 & 0.000\end{bmatrix}$
0.516	$\begin{bmatrix}0.000 & 0.000 & 0.000\end{bmatrix}$
1.133	$\begin{bmatrix}0.010 & 0.010 & 0.000\end{bmatrix}$
1.133	$\begin{bmatrix}-0.010 & -0.010 & 0.000\end{bmatrix}$
1.133	$\begin{bmatrix}0.010 & -0.010 & 0.000\end{bmatrix}$
1.133	$\begin{bmatrix}-0.010 & 0.010 & 0.000\end{bmatrix}$

Table 1.2: Table of the vectors of $\mathcal{E}$, exclusing the mass-limited element.

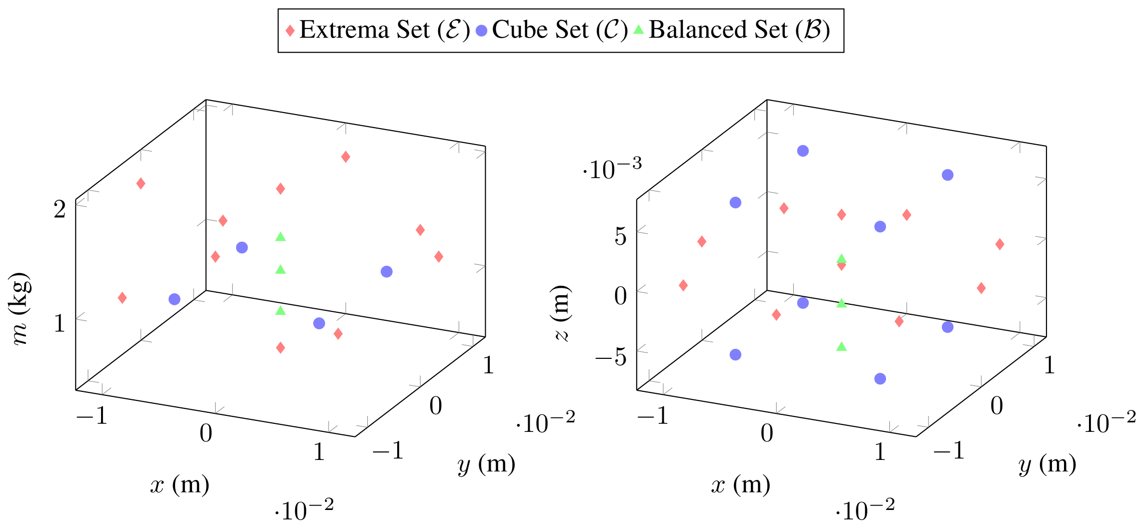

Figure 1.4: Mass and COM coordinates for each test point set.

Target		Nearest		L2 Norm Error
m	$\boldsymbol{r}$	m	$\boldsymbol{r}$	
Extrema Set ($\mathcal{E}$)				
2.000	$[0.000 \quad 0.000 \quad 0.000]$	1.909	$[0.000 \quad 0.000 \quad 0.004]$	9.154×10^{-2}
Cube Set ($\mathcal{C}$)				
$*$	$[-0.007 \quad -0.007 \quad -0.007]$	1.061	$[-0.006 \quad -0.006 \quad -0.006]$	1.055×10^{-3}
$*$	$[-0.007 \quad -0.007 \quad 0.007]$	1.061	$[-0.006 \quad 0.006 \quad -0.006]$	1.895×10^{-2}
$*$	$[-0.007 \quad 0.007 \quad -0.007]$	1.061	$[0.006 \quad -0.006 \quad -0.006]$	1.895×10^{-2}
$*$	$[-0.007 \quad 0.007 \quad 0.007]$	1.061	$[0.006 \quad 0.006 \quad -0.006]$	1.895×10^{-2}
$*$	$[0.007 \quad -0.007 \quad -0.007]$	1.061	$[-0.006 \quad -0.006 \quad 0.006]$	1.895×10^{-2}
$*$	$[0.007 \quad -0.007 \quad 0.007]$	1.061	$[-0.006 \quad 0.006 \quad 0.006]$	1.895×10^{-2}
$*$	$[0.007 \quad 0.007 \quad -0.007]$	1.061	$[0.006 \quad -0.006 \quad 0.006]$	1.895×10^{-2}
$*$	$[0.007 \quad 0.007 \quad 0.007]$	1.061	$[0.006 \quad 0.006 \quad 0.006]$	1.055×10^{-3}
Balanced Set ($\mathcal{B}$)				
0.516	$[0.000 \quad 0.000 \quad *]$	0.832	$[0.000 \quad 0.000 \quad -0.003]$	3.156×10^{-1}
1.258	$[0.000 \quad 0.000 \quad *]$	1.192	$[-0.000 \quad -0.000 \quad 0.000]$	6.630×10^{-2}
2.000	$[0.000 \quad 0.000 \quad *]$	1.478	$[-0.000 \quad -0.000 \quad -0.007]$	5.219×10^{-1}

Table 1.3: Table of the target and actual vectors for $\mathcal{C}$, $\mathcal{B}$ and the mass limited element of $\mathcal{E}$ with the L2 norm error. $*$ notation indicates "don't care" and is excluded from the search algorithm.

1.6 Conclusion and Discussion

References

[1] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[2] P. Serafini, "Simulated annealing for multi objective optimization problems," in *Multiple criteria decision making*, Springer, 1994, pp. 283–292.

Chapter 2

Creating a Configurable Payload for Instability Experiments

2.1 Introduction

2.2 Control Experiments

2.2.1 Results

2.3 Conclusion and Discussion

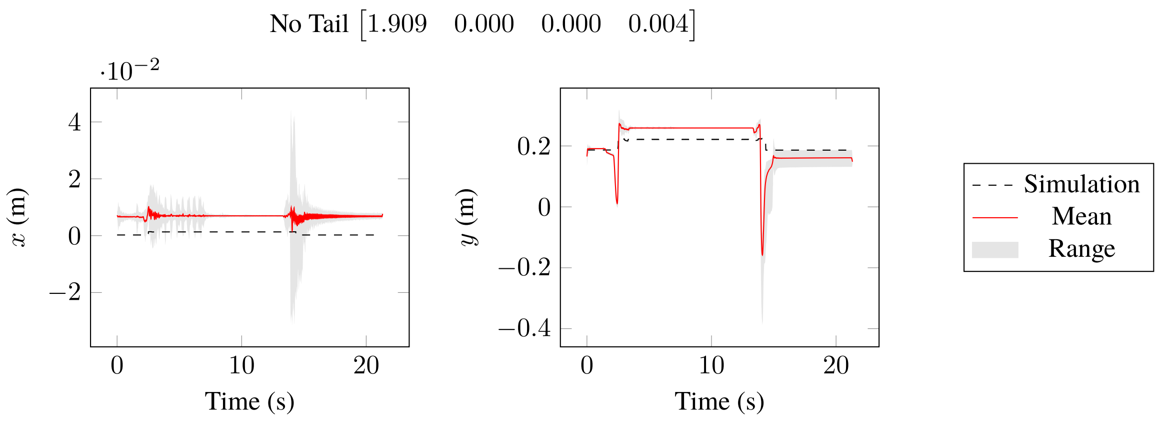

Figure 2.1: COM x and y position of the test rig along the test trajectory for the mass maximum element of the extrema set with no tail.

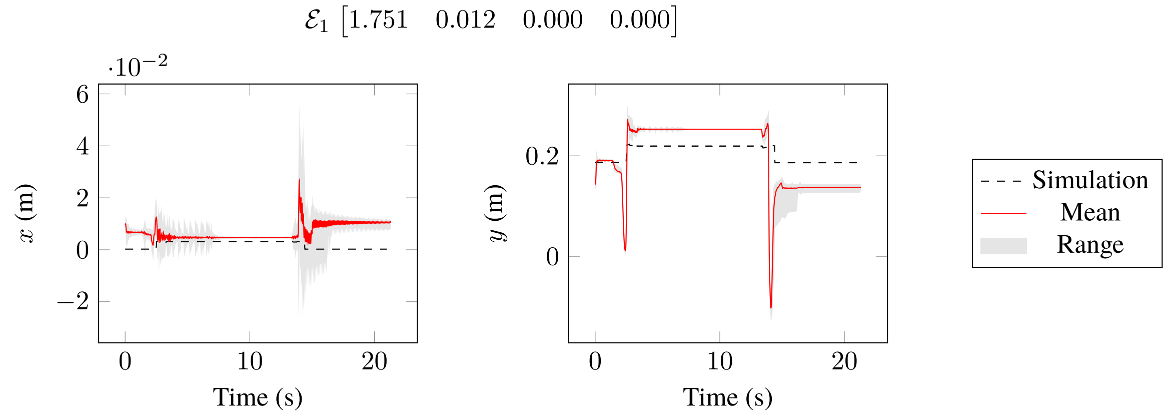

Figure 2.2: COM x and y position of the test rig along the test trajectory for the mass minimum element of the extrema set with no tail.

Chapter 3

Optimisation Study for Multi-Segment Tails for Centre of Mass Control

3.1 Introduction

One of the most noticeable differences between robotic and animal tails, as discovered in chapter **??**, is the far greater number of segments in most animal tails when compared to robot tails.

3.2 Model Definition

We can consider a tail with an arbitrary number of segments as a chain of bodies connected by revolute joints, where $\boldsymbol{l}$ are the lengths of the bodies, $\boldsymbol{m}$ are the masses of the bodies, $\boldsymbol{l_c}$ are the offsets of the COM from the origin of each body along the axis of the chain (assuming the COM remains on that axis for the sake of simplicity), and $\boldsymbol{\theta}$ are the angles of each revolute joint.

The transformation matrices for a given body's origin and COM can then be computed.

$$\begin{aligned} T_i &= \begin{bmatrix} \cos\theta_i & -\sin\theta_i & l_i \\ \sin\theta_i & \cos\theta_i & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ T_{c_i} &= \begin{bmatrix} \cos\theta_i & -\sin\theta_i & l_{c_i} \\ \sin\theta_i & \cos\theta_i & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{3.1}$$

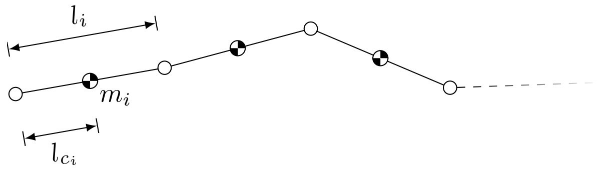

Figure 3.1: Diagram of a 2D tail, with all parameters annotated.

The forward kinematics for the tail is then computed. This will give us the position of the endpoint of the tail for a given set of joint angles $\boldsymbol{\theta}$.

$$T(\boldsymbol{\theta}) = \prod_{i=1}^{n} T_i \tag{3.2}$$

A similar equation can be used for the forward *kinetics*, with the addition of the body masses and COM offsets as parameters. This will give us the position of the COM for the entire tail.

$$R(\boldsymbol{\theta}) = \frac{\sum_{i=1}^{n} m_i \prod_{j=1}^{i-1} (T_j)\, T_{c_i}}{\sum_{i=1}^{n} m_i} \tag{3.3}$$

3.3 Inverse Kinetics

As we can compute the forward kinetics in a very similar fashion to the inverse kinematics, it follows that the inverse kinetics can be computed in a similar fashion. Firstly we define the Jacobian.

$$\boldsymbol{J}_R = \begin{bmatrix} \frac{\partial r_{13}}{\partial \theta_1} & \frac{\partial r_{13}}{\partial \theta_1} & \dots & \frac{\partial r_{13}}{\partial \theta_n} \\ \frac{\partial r_{23}}{\partial \theta_1} & \frac{\partial r_{23}}{\partial \theta_1} & \dots & \frac{\partial r_{23}}{\partial \theta_n} \end{bmatrix} \tag{3.4}$$

Then, assuming a COM position defined by $\begin{bmatrix} q_x & q_y \end{bmatrix}^\intercal$ the COM velocity can be calculated using the Jacobian.

$$\begin{bmatrix} \dot{q}_x \\ \dot{q}_y \end{bmatrix} = \boldsymbol{J}_R \cdot \dot{\theta} \tag{3.5}$$

To get the joint velocities instead for a given COM velocity, we can use a minimisation algorithm to calculate the optimal trajectory of $\boldsymbol{\theta}$ in order to minimise one or more undesirable aspects.

3.4 Minimisation Algorithms

3.4.1 Damped Least Squares

Damped Least Squares (DLS), also known as Levenberg-Marquardt [1], is perhaps one of the simplest algorithms to implement.

$$\dot{\boldsymbol{\theta}} = \boldsymbol{J}^\intercal \left(\boldsymbol{J}\boldsymbol{J}^\intercal \lambda^2 \boldsymbol{I} \right) \begin{bmatrix} \dot{q}_x \\ \dot{q}_y \end{bmatrix} \tag{3.6}$$

Where $\boldsymbol{I}$ is the identity matrix and λ is a suitable damping constant.

While it is easy to implement, DLS can only minimise the joint velocities, expressed as the euclidean norm $\|\dot{\boldsymbol{\theta}}\,(t)\,\|_2$.

3.4.2 Quadratic Programming

Quadratic programming

$$\begin{aligned}
\boldsymbol{\tau} &= D\left(\boldsymbol{\theta}\right)\ddot{\boldsymbol{\theta}} + C\left(\boldsymbol{\theta},\dot{\boldsymbol{\theta}}\right) + G\left(\boldsymbol{\theta}\right) \\
\boldsymbol{Q} &= \alpha\boldsymbol{I} + (1-\alpha)D\left(\boldsymbol{\theta}\right)^2 \\
\boldsymbol{p} &= \alpha\lambda\dot{\boldsymbol{\theta}} + (1-\alpha)D\left(\boldsymbol{\theta}\right)^\intercal\left(C\left(\boldsymbol{\theta},\dot{\boldsymbol{\theta}}\right) + G\left(\boldsymbol{\theta}\right)\right) \\
\ddot{\boldsymbol{\theta}} &= \min_{\ddot{\boldsymbol{\theta}}}\frac{1}{2}\ddot{\boldsymbol{\theta}}^\intercal\boldsymbol{Q}\ddot{\boldsymbol{\theta}} + \boldsymbol{p}^\intercal\ddot{\boldsymbol{\theta}}\text{ s.t.}\left\{\left[\boldsymbol{J}\ddot{\boldsymbol{\theta}}\right]_1 = \ddot{x}_1 - \left[\dot{\boldsymbol{J}}\dot{\boldsymbol{\theta}}\right]_1\right.
\end{aligned} \tag{3.7}$$

3.5 Results

3.6 Conclusion and Discussion

References

[1] S. R. Buss, "Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods," *IEEE Journal of Robotics and Automation*, vol. 17, no. 1-19, p. 16, 2004.

Appendices