MANCHESTER 1824
The University of Manchester

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.

List of abbreviations

AUJ Actuated Universal Joint

2D Two Dimensional

3D Three Dimensional

COM Centre of Mass

DLS Damped Least Squares

DOF Degree(s) of Freedom

PID Proportional Integral Derivative

SHTP Sensor Hub Transport Protocol

MSB Most Significant Bit

LSB Least Significant Bit

I^2C Inter-Integrated Circuit

SPI Serial Peripheral Interface

UART Universal Asynchronous Receiver-Transmitter

DIO Digital Input/Output

IMU Inertial Measurement Unit

GOOP Graphical Object Oriented Programming

CS Chip Select

DFT Discrete Fourier Transform

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

A Literature Review of Terrestrial Robots with Robotic Tails and Their Functions

1.1 Introduction

The field of terrestrial robots with robotic tails is incredibly diverse, reflecting the many functions of tails in the animal kingdom. Even discounting tails used for fluid dynamics (swimming and flying robots), and focusing only on robots that use their tail during "terrestrial" locomotion (broadly defined as when a robot is moving along a contiguous surface, or jumping from one surface to another surface), there are many applications of robotic tails. In order to make sense of the state of the field, an abstract categorisation system is considered based on the environment the robot is in when the tail is active, the specific action the robot is taking to move itself in space, and what the specific function of the tail is with respect to the robot dynamics. Using this categorisation system, various examples are explored from a set of research articles selected using specific keywords. From this, conclusions can be derived about the general design and operation of robotic tails, which can be used to influence and guide the research covered in chapters **??**.

1.2 Research Methodology

Using three online publication repositories, *IEEE Xplore*, *Scopus* and *Web of Science*, a search query was conducted to find relevant publications. The query was tailored to include all publications with **tail*** or **appendage** in the title along with **robot*** (* indicates a wildcard suffix), but to exclude publications that concerned swimming, water walking, or flying robots, as using a tail as a rudder to influence fluid dynamics was outside of the scope of the research. Further exclusions were added upon experimentation with the query in order to remove false positives in areas such as chemistry (as molecules are often described as having "tails") or medicine (as it did not pertain to mobile robots and usually concerned biological structures such as proteins and cells). The date range was set from January 1980 to December 2018 to exclude outdated publications. The exact queries for each repository can

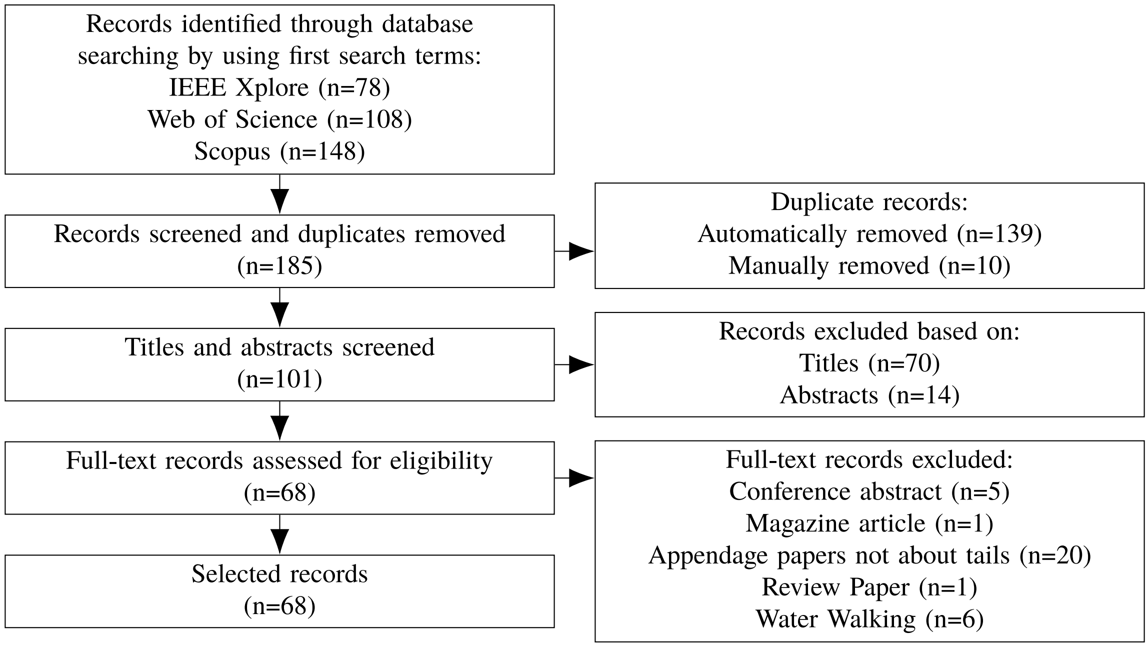


Figure 1.1: Flowchart of the publication selection process.

be found in table **??**.

As a result XX publications were discovered. After duplicates were removed and after screening abstracts and full texts, a total of XX unique publications were selected for inclusion. A flowchart of this process can be found in figure **??**.

1.3 Tail Functions of Terrestrial Robots

1.3.1 Categorisation System

The resulting publications represented a wide array of different robot designs and multiple forms of locomotion. Therefore, a simple categorisation system was required in order to better understand the majority of the functions of these tails. After careful analysis of the literature, two questions could be asked that provided common answers:

1. Is the robot on the ground, in the air or transitioning between those states when the tail is active?
2. What precisely is the robot doing when the tail is active?
3. What does the tail do to the dynamics of the robot (i.e. what would happen if there *wasn't* a tail present)?

Each question can then assign a tiered category to a publication based on the answer, the three tiers named *Environment*, *Action* and *Function*.

1. **Environment:** The general domain the robotic system is operating in when the tail is active. From the reviewed literature, three categories have been created:

- *Terrestrial:* A robot with an active tail when the robot is touching a surface, such as a robot driving along the ground.
- *Aerial:* A robot with an active tail when the robot is in free space, such as a robot which *has jumped* into the air, or is falling off a ledge.
- *Transition:* A robot with an active tail when the robot is just about to transition from between the two previous environments, such as a robot *just about* to jump into the air.

2. **Action:** The specific action the robot is performing when the tail is active. Most actions are unique to each environment, except for *hopping*.

   - *Straight:* A robot travelling across a surface maintaining its direction of travel.
   - *Accelerating:* A robot changing its velocity in the direction of travel across a surface. In the literature, this was a robot coming to complete stop, and starting from stationary.
   - *Turning:* A robot changing its direction of travel across a surface, such as a robot turning a corner.
   - *Balancing:* A robot undergoing external disturbances while travelling across a surface, typically due to adverse terrain, such as a robot navigating a rough and uneven surface without falling over.
   - *Hopping:* A robot executing a sequence of periodic jumps in order to travel across a surface, similar to the method of locomotion of a Kangaroo.
   - *Jumping:* A robot executing isolated non-periodic jumps, typically to transition from one surface to another at a different altitude and/or orientation.
   - *Falling:* A robot transitioning from one surface to another at a different altitude via the influence of gravity.

3. **Function:** The purpose of the tail when the robot is performing the action. These categories usually apply to multiple actions.

   - *Stability:* The tail is used to *maintain* some aspect of the robot's position and/or orientation from the start of the action.
   - *Initiation:* The tail is used to *change* some aspect of the robot's position and/or orientation from the start of the action.
   - *Amplification:* The tail is used to *amplify* the effects of an action by other parts of the robot (such as the legs) which changes the position and/or orientation.

For example, a robot with a tail that helps it increase it's apex when hopping, is in the *Transition* environment as it is about to transition from being on the ground to in the air when

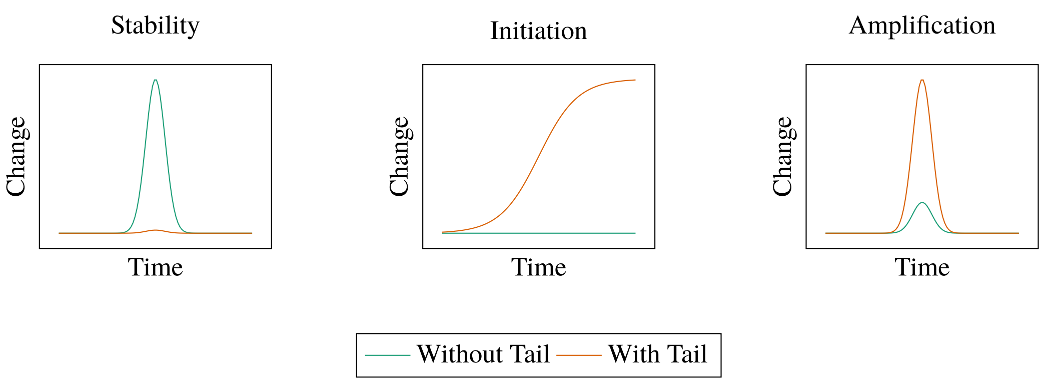


Figure 1.2: Abstract graphs of the different functions of the tail, with the magnitude representing some kind of change in the robot's position and/or orientation.

the tail is active. The robot itself locomotes by hopping, so it is performing a *Hopping* action, and since without the tail the robot would still be capable of hopping, but would not have such a tall apex, the tail can be considered to be performing *Amplification* of the robot's existing capabilities. Overall, this results in the categorisation *Transition* → *Hopping* → *Amplification*.

Another example is a robot with a tail that prevents it from falling over on rough terrain. The robot is in the *Terrestrial* environment as it is on the ground when the tail is active. The robot itself is performing a *Balancing* action as it is trying to remain upright during locomotion, and since without the tail the robot would fall over, the tail can be considered to be maintaining the *Stability* of the robot. Overall, this results in the categorisation *Terrestrial* → *Balancing* → *Stability*.

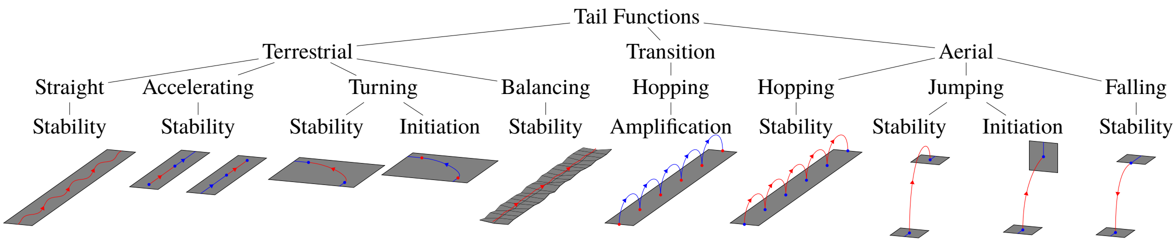


Figure 1.3: Tree diagram of all the categorisations found in the publications with accompanying visual diagrams.

1.3.2 Examples from each Category

1.3.3 *Terrestrial* → *Straight* → *Stability*

zhang2016effects uses a quadrupedal robot (Dcat), with a gait controlled by a **CPG!** (**CPG!**), an open loop control concept based on neural networks found in the spinal chords of vertebrates and invertebrate thoracic ganglia **ijspeert2008central**. A **CPG!** in robotics is typically constructed from a network of oscillators represented by a system of **ODE!** (**ODE!**), which accepts the input of another oscillator as a parameter in the system. By chaining these together, complex synchronised trajectories can be generated for multiple actuators in robot locomotion.

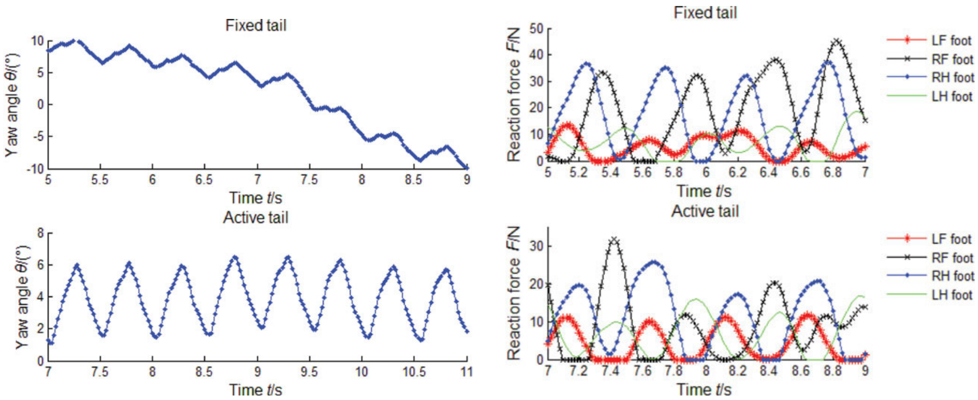


Figure 1.4: Data from **zhang2016effects** showing the effects of a static and dynamic tail on maintaining robot heading in a trotting gait.

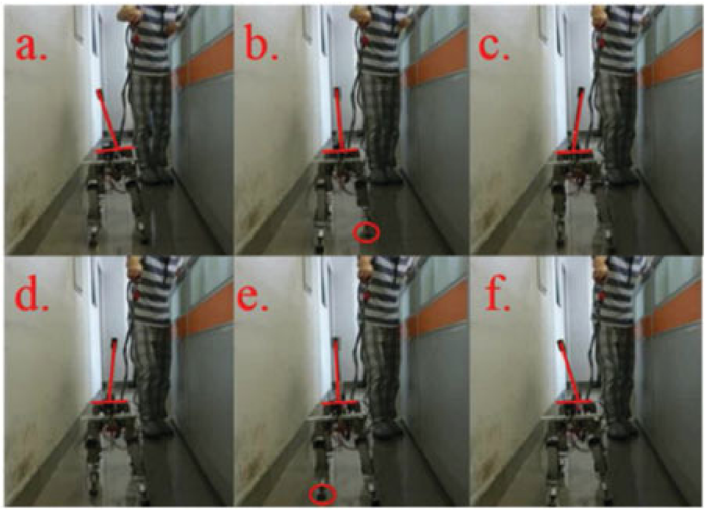


Figure 1.5: Image from **zhang2016effects** showing how the tail moves during the gait in order to correct for heading drift.

Upon initial experiments with a trotting locomotion with no active tail, the robot wouldn't maintain a set heading, it would slowly begin to drift in a circle. Visual observations noted that the robot would topple onto its front left leg that was in "swing" phase (lifted off the ground), and it would drag on the ground until in "stance" phase. This resulted in a difference between the **GRF!** (**GRF!**) of the left and right feet which caused the drift in locomotion path.

By implementing a swinging 1 **DOF!** (**DOF!**) tail that imparted an opposing torque to the direction of the topple, the differences between the left and right **GRF!** were reduced, and the robot could maintain its heading, as can be seen in figure **??**.

1.3.4 *Terrestrial → Accelerating → Stability*

patel2014rapid used inspiration from the Cheetah (*Acinonyx jubatus*) to improve the acceleration and braking capabilities of a wheeled robot (Dima). The research is based on findings from **williams2009pitch**, which shows that quadruped acceleration and deceleration in the animal kingdom is limited by their ability to constrain body pitch to prevent toppling over. It can be considered analogous to a motorcycle: accelerate too fast and the vehi-

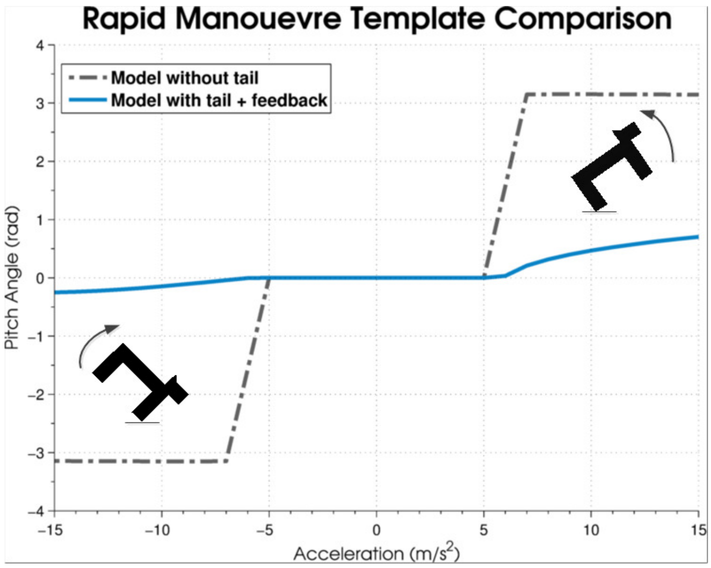


Figure 1.6: Simulation Data from **patel2014rapid** showing the how the body pitch would be reduced when accelerating of braking with an active tail.

Figure 1.7: Images from **patel2014rapid** showing the robot performing a rapid acceleration test with the tail.

cle will "pop a wheelie" and potentially flip backwards, decelerate too fast and the opposite may occur.

Using a combined state feedback and **PI!** (**PI!**) controller based on the angular position of the tail, and the angular velocity of the tail and body, the researchers were able to increase the acceleration and braking capabilities of the robot by using the tail to generate an opposing torque to the direction of body pitch, as can be seen in figure **??**. This was verified by running a series of experiments, increasing the acceleration/braking magnitude until the robot failed to complete the test by toppling over.

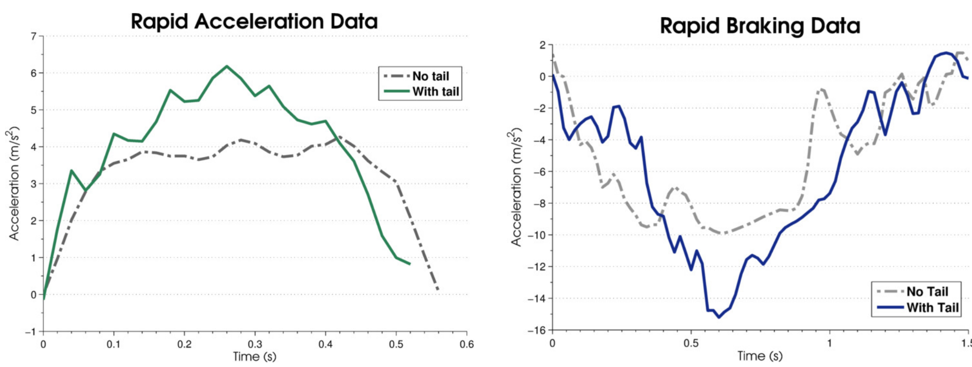


Figure 1.8: Experimental Data from **patel2014rapid** showing the maximum acceleration achieved with and without an active tail.

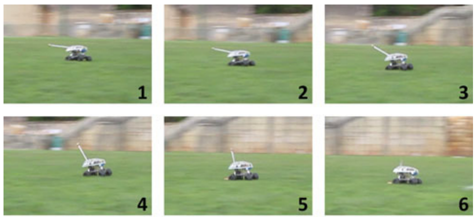


Figure 1.9: Images from **patel2015conical** showing the robot performing a turn with the tail.

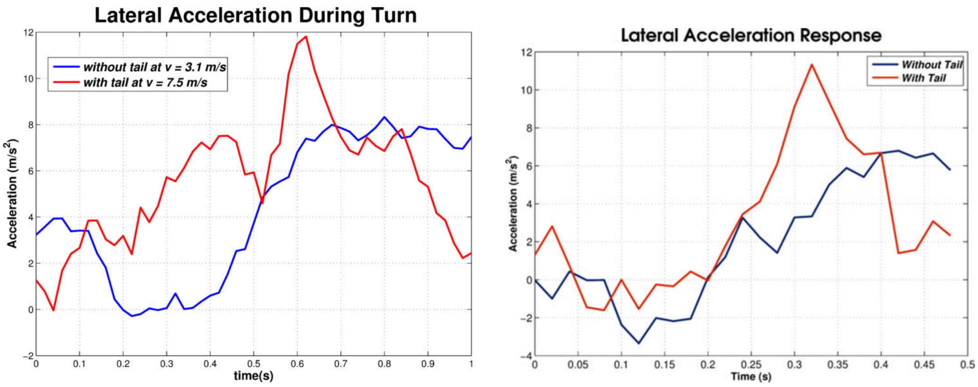


Figure 1.10: Experimental Data from **patel2013rapid** (left) and **patel2015conical** (right) showing the maximum lateral acceleration achieved with and without and active tail. Note the right graph manages similar results to the left graph.

1.3.5 *Terrestrial → Turning → Stability*

patel2013rapid and **patel2015conical** took similar inspiration from the Cheetah to allow for tighter turns by allowing greater lateral acceleration. **patel2013rapid** swings the tail out in a single motion in the direction of the turn, producing an opposing torque to the centrifugal force that would otherwise topple the robot during the turn. In contrast, **patel2015conical** moves the tail constantly in a conical motion, the direction of rotation in the direction of the turn. This also produces an opposing torque in the same fashion, but was not limited in duration, as in the first strategy the tail would eventually contact the ground. This allowed for turns of longer duration to be stabilised.

The control system and experimental procedures were similar to those in **patel2014rapid**.

1.3.6 *Terrestrial → Turning → Initiation*

casarez2013using

1.3.7 *Terrestrial → Balancing → Stability*

borisova2020analysis

1.3.8 *Transition → Hopping → Amplification*

1.3.9 *Aerial → Hopping → Stability*

1.3.10 *Aerial → Jumping → Initiation*

1.3.11 *Aerial → Jumping → Stability*

1.3.12 *Aerial → Falling → Stability*

1.4 Bio-Inspiration

1.5 Conclusion and Discussion

Chapter 2

Creating a Configurable Payload for Instability Experiments

2.1 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!** (**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 can then be run with each of these test points to generate the test data.

2.2 Configurable Payload

2.2.1 Design

The payload consists of a matrix of cubes of various materials packed tightly into a **3D!** (**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.

2.2.2 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 $\mathrm{kg\,m^{-3}}$) of a material to be used:

$$\mathcal{P} = \{\rho_1, \rho_2 \ldots \rho_n \mid \rho_i > 0\} \tag{2.1}$$

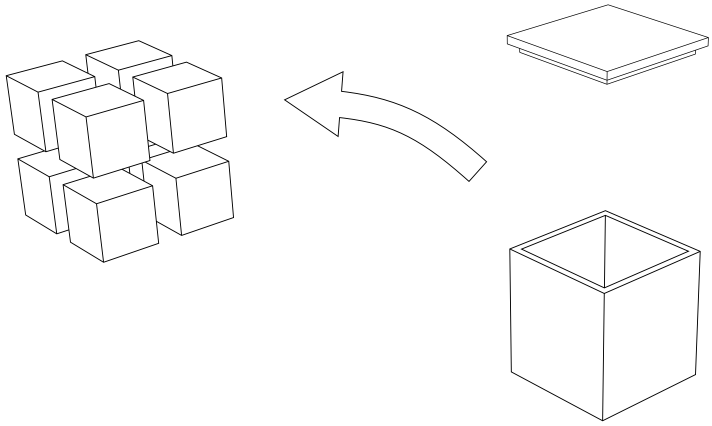


Figure 2.1: Concept drawing of the configurable payload.

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

Table 2.1: The materials chosen for the cubes.

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:

$$\boldsymbol{C} = (c_{ijk}) \in \mathbb{R}^{n^n} \mid (c_{ijk}) \in \mathcal{P} \tag{2.2}$$

2.2.2.1 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:

$$M(\boldsymbol{C}) = \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} c_{ijk} a^3 \right) + m_c \tag{2.3}$$

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:

$$R(\boldsymbol{C}) = \frac{\left(\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} c_{ijk} a^4 \left(\begin{bmatrix} i & j & k \end{bmatrix} - n + 1 \right) \right) + \boldsymbol{r}_c}{M(\boldsymbol{C})} \tag{2.4}$$

2.2.2.2 Results

2.3 Test Points

2.3.1 Test Point Sets

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 $\begin{bmatrix} m_i & \boldsymbol{r}_i \end{bmatrix}$.

2.3.1.1 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 $x = -y$.

$$\mathcal{E} = \left\{ \boldsymbol{x} \subset \mathcal{Z} \mid \begin{cases} M(\boldsymbol{x}) = \text{max}\left\{M(\mathcal{Z})\right\} \\ M(\boldsymbol{x}) = \text{min}\left\{M(\mathcal{Z})\right\} \\ R(\boldsymbol{x})_x = \text{max}\left\{R(\mathcal{Z})_x\right\} \\ R(\boldsymbol{x})_x = \text{min}\left\{R(\mathcal{Z})_x\right\} \\ R(\boldsymbol{x})_y = \text{max}\left\{R(\mathcal{Z})_y\right\} \\ R(\boldsymbol{x})_y = \text{min}\left\{R(\mathcal{Z})_y\right\} \\ R(\boldsymbol{x})_x = \text{max}\left\{R(\mathcal{Z})_x\right\} \wedge R(\boldsymbol{x})_x = R(\boldsymbol{x})_y \\ R(\boldsymbol{x})_x = \text{min}\left\{R(\mathcal{Z})_x\right\} \wedge R(\boldsymbol{x})_x = R(\boldsymbol{x})_y \\ R(\boldsymbol{x})_x = \text{max}\left\{R(\mathcal{Z})_x\right\} \wedge R(\boldsymbol{x})_x = -R(\boldsymbol{x})_y \\ R(\boldsymbol{x})_x = \text{min}\left\{R(\mathcal{Z})_x\right\} \wedge R(\boldsymbol{x})_x = -R(\boldsymbol{x})_y \end{cases} \right\} \tag{2.5}$$

2.3.1.1.1 Mass Limited $\mathcal{E}$ When this set was generated, it was found that $M\left(\mathcal{E}_1\right)$ was greater than the chosen robot arm could safely lift. Therefore, $M(\boldsymbol{x}) = \text{max}\left\{M(\mathcal{Z})\right\}$ 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 **??** can then be used to find the nearest point in configuration space.

Cube Set ($\mathcal{C}$)

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

$$\mathcal{C} = \left\{\boldsymbol{x} \subset \mathcal{C} \mid \begin{bmatrix}\pm\frac{b}{2} & \pm\frac{b}{2} & \pm\frac{b}{2}\end{bmatrix} = \boldsymbol{x}\right\} \tag{2.6}$$

Balanced Set ($\mathcal{B}$)

The balanced set is defined by q points in $\mathcal{C}$ subject to the constraint $R(\boldsymbol{x})_x = 0 \wedge R(\boldsymbol{x})_y = 0$. 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 **??**.

$$\begin{aligned} m_r &= \frac{\max\{M(\mathcal{Z})\} - \min\{M(\mathcal{Z})\}}{q+1} \\ \boldsymbol{z} &= \begin{bmatrix}m_r & 2m_r & \cdots & qm_r\end{bmatrix} \\ \mathcal{B} &= \{\boldsymbol{x} \subset \mathcal{C} \mid z_i = \boldsymbol{x} \wedge R(\boldsymbol{x})_x = 0 \wedge R(\boldsymbol{x})_y = 0\} \end{aligned} \tag{2.7}$$

2.3.2 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 $n = 3$.

2.3.2.1 Simulated Annealing ($n = 3$)

Simulated annealing **kirkpatrick1983optimization** 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 $c_{ijk} \neq c_{ijk}$:

$$\mathcal{N}(\boldsymbol{C}) = \{x \subset \mathcal{Z} \mid \exists!\,(x_{ijk} \neq c_{ijk})\} \tag{2.8}$$

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**
  $t = T\left(\frac{k_{max}}{k}\right)$
  $\boldsymbol{C}_{new} = \mathcal{N}(\boldsymbol{C}) \xleftarrow{R} x$
  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 **serafini1994simulated**, so it is possible to generate test points that approximate a desired mass and **COM!** simultaneously.

2.3.2.1.1 Cooling Function The function which controls the probability of exiting local minima (known as the *temperature*) is known as the *cooling function*. This function can be any function which monotonically decreases (except in adaptive simulated annealing where it is

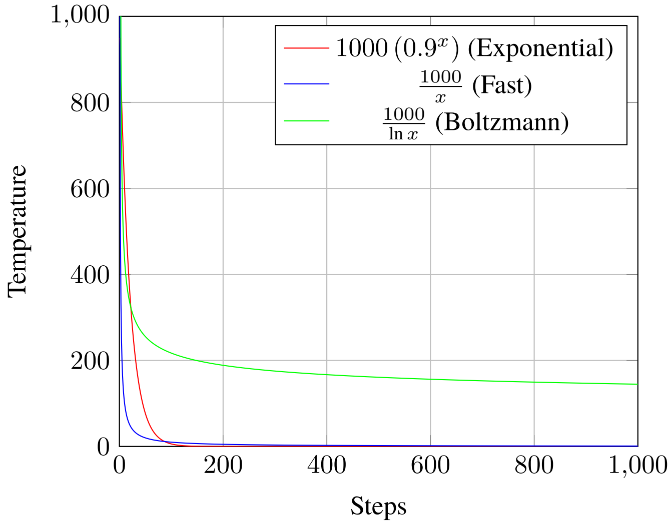


Figure 2.2: Various temperature cooling profiles for simulated annealing, assuming 1000 steps.

dependant on the accuracy of the current approximation). Different functions will result in a different cooling profile, generally decreasing quickly in the first few steps, and then slowing down after that.

2.3.2.2 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.

$$NN\left(t, \mathcal{Z}\right) = \min\left\{\|t - x\|_2 \mid \forall x \in \mathcal{Z}\right\} \tag{2.9}$$

2.3.3 Selected Method

Initially an $n = 3$ configuration was used with the acceptance probability function *Rule M* from **serafini1994simulated**. 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 $\boldsymbol{w} \in \mathbb{R}^4 \mid w_i \in (0, 1)$.

$$P\left(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w}, t\right) = \alpha \underbrace{\prod_{i=1}^{m} \min\left\{1, e^{\frac{w_i(x_i - y_i)}{t}}\right\}}_{\text{Rule P}} + (1 - \alpha) \underbrace{\min\left\{1, \max_{i=1,\ldots,m}\left\{1, e^{\frac{w_i(x_i - y_i)}{t}}\right\}\right\}}_{\text{Rule W}} \tag{2.10}$$

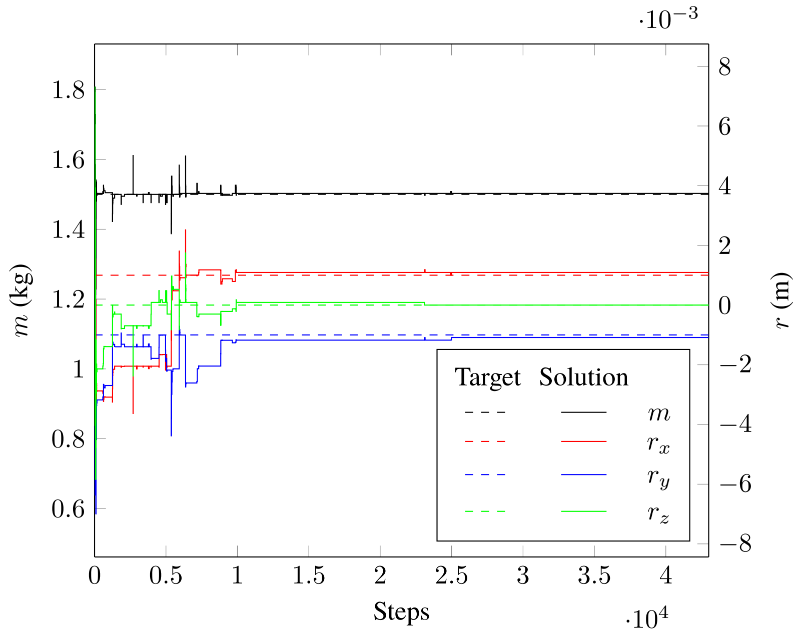


Figure 2.3: Simulated annealing output for the target $[1.5 \quad 0.001 \quad 0.001 \quad 0.001]$ with $\alpha = 0.997$ and even weighting $w_m, \boldsymbol{w_r} = 0.25$ for 4.3×10^4 steps.

m	$\boldsymbol{r}$
Extrema Set ($\mathcal{E}$)	
1.291	$[0.011 \quad -0.000 \quad 0.000]$
1.291	$[0.000 \quad 0.011 \quad 0.000]$
1.291	$[-0.011 \quad -0.000 \quad -0.000]$
1.291	$[-0.000 \quad -0.011 \quad -0.000]$
0.516	$[0.000 \quad 0.000 \quad 0.000]$
1.133	$[0.010 \quad 0.010 \quad 0.000]$
1.133	$[-0.010 \quad -0.010 \quad 0.000]$
1.133	$[0.010 \quad -0.010 \quad 0.000]$
1.133	$[-0.010 \quad 0.010 \quad 0.000]$

Table 2.2: Table of the vectors of $\mathcal{E}$, excluding the mass-limited element.

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 $\mathcal{C}$.

2.3.4 Results

2.4 Conclusion and Discussion

Target		Nearest		L2 Norm Error
m	$\boldsymbol{r}$	m	$\boldsymbol{r}$	
Extrema Set ($\mathcal{E}$)				
1.770	$\begin{bmatrix}0.000 & 0.000 & 0.000\end{bmatrix}$	1.291	$\begin{bmatrix}-0.000 & 0.000 & -0.008\end{bmatrix}$	4.789×10^{-1}
Cube Set ($\mathcal{C}$)				
$*$	$\begin{bmatrix}-0.007 & -0.007 & -0.007\end{bmatrix}$	1.061	$\begin{bmatrix}-0.006 & -0.006 & -0.006\end{bmatrix}$	1.055×10^{-3}
$*$	$\begin{bmatrix}-0.007 & -0.007 & 0.007\end{bmatrix}$	1.061	$\begin{bmatrix}-0.006 & 0.006 & -0.006\end{bmatrix}$	1.895×10^{-2}
$*$	$\begin{bmatrix}-0.007 & 0.007 & -0.007\end{bmatrix}$	1.061	$\begin{bmatrix}0.006 & -0.006 & -0.006\end{bmatrix}$	1.895×10^{-2}
$*$	$\begin{bmatrix}-0.007 & 0.007 & 0.007\end{bmatrix}$	1.061	$\begin{bmatrix}0.006 & 0.006 & -0.006\end{bmatrix}$	1.895×10^{-2}
$*$	$\begin{bmatrix}0.007 & -0.007 & -0.007\end{bmatrix}$	1.061	$\begin{bmatrix}-0.006 & -0.006 & 0.006\end{bmatrix}$	1.895×10^{-2}
$*$	$\begin{bmatrix}0.007 & -0.007 & 0.007\end{bmatrix}$	1.061	$\begin{bmatrix}-0.006 & 0.006 & 0.006\end{bmatrix}$	1.895×10^{-2}
$*$	$\begin{bmatrix}0.007 & 0.007 & -0.007\end{bmatrix}$	1.061	$\begin{bmatrix}0.006 & -0.006 & 0.006\end{bmatrix}$	1.895×10^{-2}
$*$	$\begin{bmatrix}0.007 & 0.007 & 0.007\end{bmatrix}$	1.061	$\begin{bmatrix}0.006 & 0.006 & 0.006\end{bmatrix}$	1.055×10^{-3}
Balanced Set ($\mathcal{B}$)				
0.516	$\begin{bmatrix}0.000 & 0.000 & *\end{bmatrix}$	0.832	$\begin{bmatrix}0.000 & 0.000 & -0.003\end{bmatrix}$	3.156×10^{-1}
1.258	$\begin{bmatrix}0.000 & 0.000 & *\end{bmatrix}$	1.192	$\begin{bmatrix}-0.000 & -0.000 & 0.000\end{bmatrix}$	6.630×10^{-2}
2.000	$\begin{bmatrix}0.000 & 0.000 & *\end{bmatrix}$	1.291	$\begin{bmatrix}-0.000 & 0.000 & -0.008\end{bmatrix}$	7.088×10^{-1}

Table 2.3: 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.

??

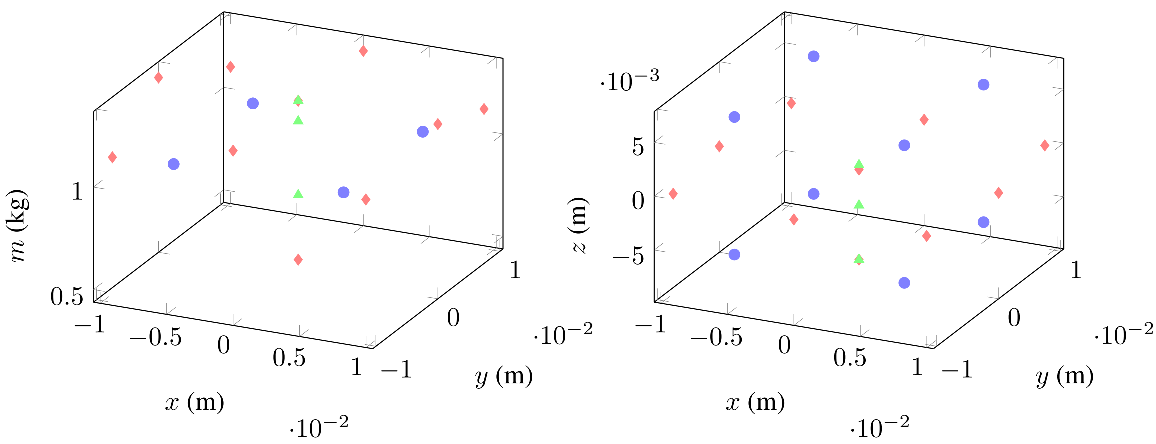


Figure 2.4: Mass and **COM!** coordinates for each test point set.

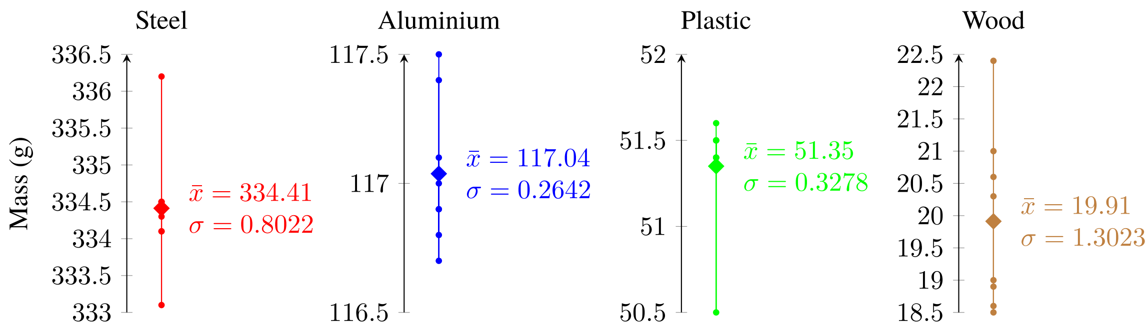


Figure 2.5: Masses for each set of eight 35 mm cubes of the configurable payload for each of the four materials, with the mean mass $\bar{x}$ and standard deviation σ of each set.

m	$\boldsymbol{r}$	Material Matrix	3D Preview
	Extrema Set ($\mathcal{E}$)		
1.291	$[-0.000 \quad 0.000 \quad -0.008]$	$\begin{bmatrix} \text{S} & \text{W} & \text{A} & \text{S} \\ \text{W} & \text{A} & \text{W} & \text{W} \end{bmatrix}$	
1.291	$[0.011 \quad -0.000 \quad 0.000]$	$\begin{bmatrix} \text{W} & \text{S} & \text{W} & \text{A} \\ \text{W} & \text{A} & \text{W} & \text{S} \end{bmatrix}$	
1.291	$[0.000 \quad 0.011 \quad 0.000]$	$\begin{bmatrix} \text{W} & \text{W} & \text{S} & \text{A} \\ \text{W} & \text{W} & \text{A} & \text{S} \end{bmatrix}$	
1.291	$[-0.011 \quad -0.000 \quad -0.000]$	$\begin{bmatrix} \text{A} & \text{W} & \text{S} & \text{W} \\ \text{S} & \text{W} & \text{A} & \text{W} \end{bmatrix}$	
1.291	$[-0.000 \quad -0.011 \quad -0.000]$	$\begin{bmatrix} \text{A} & \text{S} & \text{W} & \text{W} \\ \text{S} & \text{A} & \text{W} & \text{W} \end{bmatrix}$	
0.516	$[0.000 \quad 0.000 \quad 0.000]$	$\begin{bmatrix} \text{W} & \text{W} & \text{W} & \text{W} \\ \text{W} & \text{W} & \text{W} & \text{W} \end{bmatrix}$	
1.133	$[0.010 \quad 0.010 \quad 0.000]$	$\begin{bmatrix} \text{W} & \text{W} & \text{W} & \text{S} \\ \text{W} & \text{W} & \text{W} & \text{S} \end{bmatrix}$	
1.133	$[-0.010 \quad -0.010 \quad 0.000]$	$\begin{bmatrix} \text{S} & \text{W} & \text{W} & \text{W} \\ \text{S} & \text{W} & \text{W} & \text{W} \end{bmatrix}$	
1.133	$[0.010 \quad -0.010 \quad 0.000]$	$\begin{bmatrix} \text{W} & \text{S} & \text{W} & \text{W} \\ \text{W} & \text{S} & \text{W} & \text{W} \end{bmatrix}$	
1.133	$[-0.010 \quad 0.010 \quad 0.000]$	$\begin{bmatrix} \text{W} & \text{W} & \text{S} & \text{W} \\ \text{W} & \text{W} & \text{S} & \text{W} \end{bmatrix}$	

Table 2.4

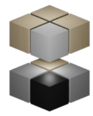
m	$\boldsymbol{r}$	Material Matrix	3D Preview
		Cube Set ($\mathcal{C}$)	
1.061	$\begin{bmatrix}-0.006 & -0.006 & -0.006\end{bmatrix}$	$\begin{bmatrix} \mathrm{S} & \mathrm{A} & \mathrm{A} & \mathrm{W} \\ \mathrm{A} & \mathrm{W} & \mathrm{W} & \mathrm{W} \end{bmatrix}$	
1.061	$\begin{bmatrix}-0.006 & 0.006 & -0.006\end{bmatrix}$	$\begin{bmatrix} \mathrm{A} & \mathrm{W} & \mathrm{S} & \mathrm{A} \\ \mathrm{W} & \mathrm{W} & \mathrm{A} & \mathrm{W} \end{bmatrix}$	
1.061	$\begin{bmatrix}0.006 & -0.006 & -0.006\end{bmatrix}$	$\begin{bmatrix} \mathrm{A} & \mathrm{S} & \mathrm{W} & \mathrm{A} \\ \mathrm{W} & \mathrm{A} & \mathrm{W} & \mathrm{W} \end{bmatrix}$	
1.061	$\begin{bmatrix}0.006 & 0.006 & -0.006\end{bmatrix}$	$\begin{bmatrix} \mathrm{W} & \mathrm{A} & \mathrm{A} & \mathrm{S} \\ \mathrm{W} & \mathrm{W} & \mathrm{W} & \mathrm{A} \end{bmatrix}$	
1.061	$\begin{bmatrix}-0.006 & -0.006 & 0.006\end{bmatrix}$	$\begin{bmatrix} \mathrm{A} & \mathrm{W} & \mathrm{W} & \mathrm{W} \\ \mathrm{S} & \mathrm{A} & \mathrm{A} & \mathrm{W} \end{bmatrix}$	
1.061	$\begin{bmatrix}-0.006 & 0.006 & 0.006\end{bmatrix}$	$\begin{bmatrix} \mathrm{W} & \mathrm{W} & \mathrm{A} & \mathrm{W} \\ \mathrm{A} & \mathrm{W} & \mathrm{S} & \mathrm{A} \end{bmatrix}$	
1.061	$\begin{bmatrix}0.006 & -0.006 & 0.006\end{bmatrix}$	$\begin{bmatrix} \mathrm{W} & \mathrm{A} & \mathrm{W} & \mathrm{W} \\ \mathrm{A} & \mathrm{S} & \mathrm{W} & \mathrm{A} \end{bmatrix}$	
1.061	$\begin{bmatrix}0.006 & 0.006 & 0.006\end{bmatrix}$	$\begin{bmatrix} \mathrm{W} & \mathrm{W} & \mathrm{W} & \mathrm{A} \\ \mathrm{W} & \mathrm{A} & \mathrm{A} & \mathrm{S} \end{bmatrix}$	

Table 2.5

m	$\boldsymbol{r}$	Material Matrix	3D Preview
		Balanced Set ($\mathcal{B}$)	
0.832	$\begin{bmatrix}0.000 & 0.000 & -0.003\end{bmatrix}$	$\begin{bmatrix} \mathrm{A} & \mathrm{A} & \mathrm{W} & \mathrm{A} \\ \mathrm{W} & \mathrm{W} & \mathrm{A} & \mathrm{W} \end{bmatrix}$	
1.192	$\begin{bmatrix}-0.000 & -0.000 & 0.000\end{bmatrix}$	$\begin{bmatrix} \mathrm{P} & \mathrm{W} & \mathrm{W} & \mathrm{S} \\ \mathrm{S} & \mathrm{P} & \mathrm{P} & \mathrm{P} \end{bmatrix}$	
1.291	$\begin{bmatrix}-0.000 & 0.000 & -0.008\end{bmatrix}$	$\begin{bmatrix} \mathrm{S} & \mathrm{W} & \mathrm{A} & \mathrm{S} \\ \mathrm{W} & \mathrm{A} & \mathrm{W} & \mathrm{W} \end{bmatrix}$	

Table 2.6

Chapter 3

Investigating the use of a 2DOF Pendulum Tail for Compensating for Instability when Carrying a Payload

3.1 Introduction

3.2 Control Experiments

3.2.1 Results

3.3 Conclusion and Discussion

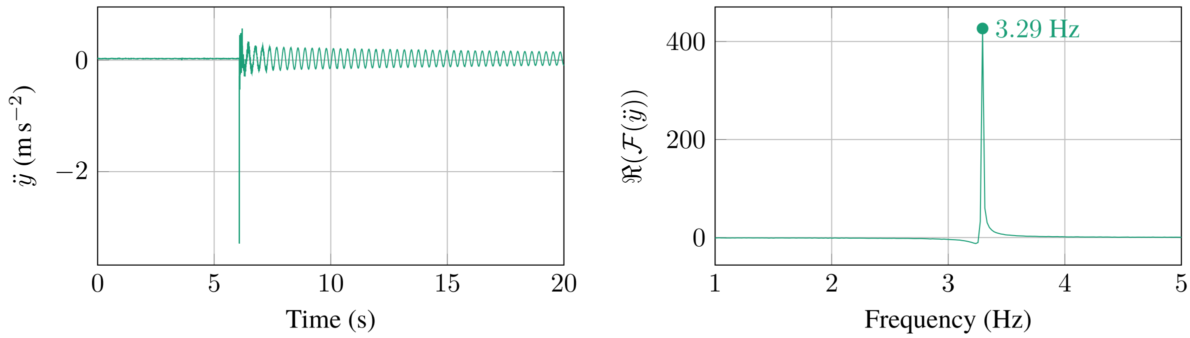
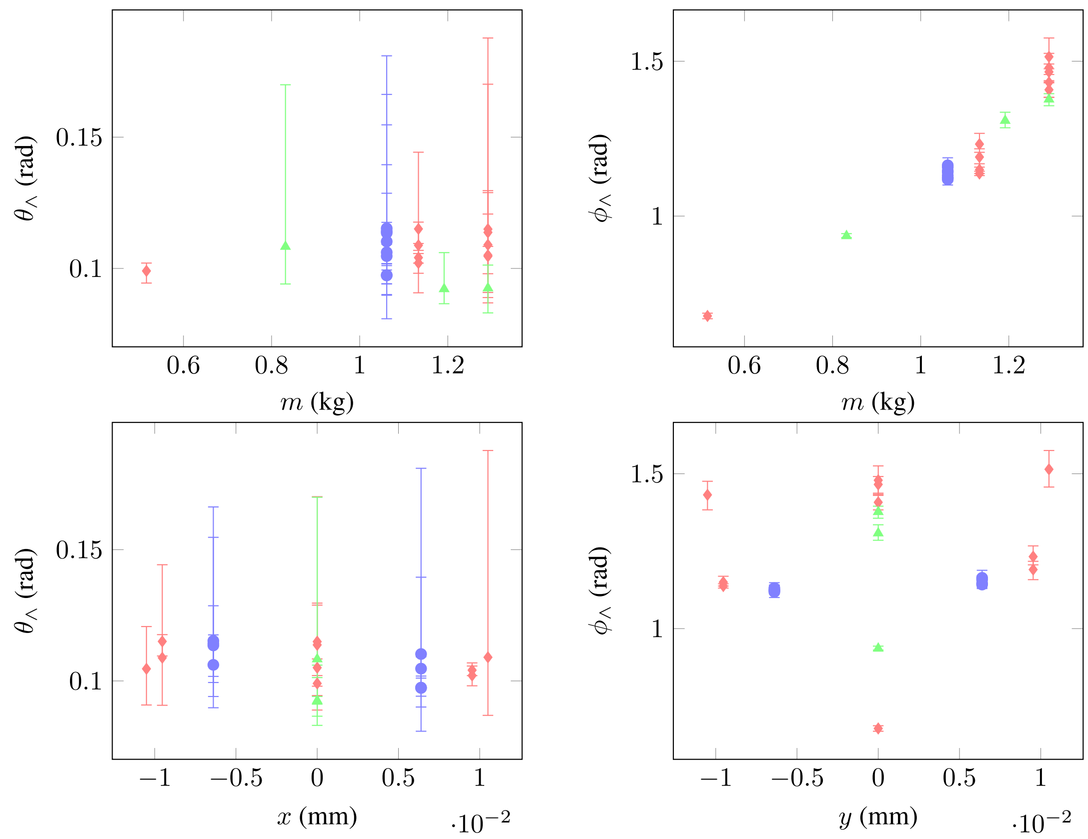


Figure 3.1: Resonant frequency of the tail calculated from accelerometer data captured during an impact with a steel hammer. The left graph is the accelerometer data, and the right graph is the real part from a real **DFT!** (**DFT!**) on the data.

??



0.15
0.1
$\theta_\wedge$ (rad)
0.6
0.8
1
1.2
m (kg)
1.5
1
$\phi_\wedge$ (rad)
0.6
0.8
1
1.2
m (kg)
0.15
0.1
$\theta_\wedge$ (rad)
−1
−0.5
0
0.5
1
x (mm)
$\cdot 10^{-2}$
1.5
1
$\phi_\wedge$ (rad)
−1
−0.5
0
0.5
1
y (mm)
$\cdot 10^{-2}$

Figure 3.2

Appendices