Spherical Designs

A Summer Research Project on Finite Tight Frames

Written by
Ken Alexander Rayner

Supervised by
Associate Professor Shayne Waldron

Department of Mathematics
Waipapa Taumata Rau
February 2024

Contents

1 Reflection

My summer research experience at Waipapa Taumata Rau has been a valuable one in both the academic and professional senses.

This was my first opportunity to engage in academic research. The differences between research and the typical delivery of a university course brought both challenges and rewards. A lot more self-motivation was required in the absence of structure that often comes with a taught course, such as timetabled classes, regular assessments and comprehensive course outlines. The greater freedom, however, also allowed me to learn at a pace that adapted to the difficulty of the necessary learning. Such differences have developed my overall skills in self-driven study, which I can now apply throughout my remaining time at University.

Post-graduate studies have always appealed to me as an opportunity once I complete my BCom/BSc conjoint. However, I have always had concerns about the difficulty of adjusting to research-based assessment. Throughout summer, I have enjoyed the independence that comes with research, and the greater sense of reward when you are able to come to some conclusion or produce a result on your own. This experience has definitely broken down some perceived barriers to doing an Honours or Masters degree, and has put me in a better position to make a decision about opportunities to pursue once I graduate.

Lastly, summer research has given me a greater appreciation of the requisite patience for learning, researching and the greater pursuit of knowledge.

As rewarding as the experience has been, I must also acknowledge the people who made it so. Thanks must go to:

- **Dr Shayne Waldron** for his knowledge and insights, as well as his support and encouragement throughout this research project.
- **Ethan Qi** for his advice, especially when it came to picking up the computational package Manopt which was utilised for a big part of my research project.
- **Dr Florian Lehner** for encouraging maths students to apply for the scholarship, recommending me for this project, and for providing all the scholars with morning tea and advice every week throughout summer.

Overall, I have had an enjoyable experience researching in the Mathematics Department and have developed some skills which will benefit me throughout my studies as well as in future career paths.

2 Preliminaries

2.1 Spherical Designs

A spherical design is a set (or sequence) of vectors on a sphere that is considered "optimal" because of some desired properties they exhibit [Wal17]. One such property is providing a quadrature rule.

Spherical designs satisfy the quadrature rule that the average value of a polynomial over the design is equal to the surface integral evaluated over the entire sphere.

More formally, if we let $\mathbb{F} = \mathbb{R}, \mathbb{C}, \mathbb{H}$ denote a field and $\mathbb{S}$ represent the unit sphere over $\mathbb{F}^d$, then a set of n vectors Φ in $\mathbb{F}^d$ is a spherical design if it satisfies

$$\int_{\mathbb{S}} f d\boldsymbol{\sigma} = \frac{1}{|\Phi|} \sum_{\phi \in \Phi} f(\phi)$$

for some family of polynomials over $\mathbb{F}^d$.

A spherical t-design is a set of vectors that satisfies the given quadrature rule for all homogeneous polynomials of degree $1, 2, \ldots, t$ [Wal23].

A spherical half-design of order t is a set of vectors that satisfies the given quadrature rule for all homogeneous polynomials of degree t.

A spherical (t, t)-design, which this project focuses on, is a spherical half-design of order $2t$.

For a set of unit vectors $\Phi = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ the following inequality holds:

$$\sum_{j=1}^{n} \sum_{k=1}^{n} |\langle \mathbf{v}_j, \mathbf{v}_k \rangle|^{2t} \geq c_t\left(\mathbb{F}^d\right)$$

The set of vectors Φ is a spherical design if the lower bound is reached, that is, when the following equality holds:

$$\sum_{j=1}^{n} \sum_{k=1}^{n} |\langle \mathbf{v}_j, \mathbf{v}_k \rangle|^{2t} = c_t\left(\mathbb{F}^d\right)$$

where

$$c_t\left(\mathbb{R}^d\right) = \frac{1 \cdot 3 \cdot 5 \cdots (2t-1)}{d(d+1) \cdots (d+2(t-1))}$$

$$c_t\left(\mathbb{C}^d\right) = \frac{1}{\binom{d+t-1}{t}}$$

$$c_t\left(\mathbb{H}^d\right) = \frac{t+1}{\binom{2d+t-1}{t}}$$

That is, a spherical design minimises the "variational characterisation" [Wal20].

2.2 Finite Tight Frames

[Wal17] formally defines a tight frame as a set of vectors $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ in $\mathbb{F}^d = \mathbb{R}^d, \mathbb{C}^d, \mathbb{H}^d$ for which there exists a frame bound $A > 0$ such that

$$A||\mathbf{u}||^2 = \sum_{\ell=1}^{n} |\langle \mathbf{u}, \mathbf{v}_\ell \rangle|^2, \qquad \forall_{\mathbf{u}\in\mathbb{F}^d}$$

Intuitively, (finite) tight frames are orthonormal bases with some redundancies. That is, you could remove a vector from the set and you would still have enough information to create an orthonormal spanning set [Avi10].

It turns out, that a finite tight frame is actually equivalent to a spherical $(1, 1)$-design. So, we can construct tight frames by constructing spherical (t, t)-designs of order 1 [Moh22].

2.3 Quaternion Numbers

The quaternion numbers are a further extension of the real (and complex) numbers [Wal20].

A quaternion number is defined as

$$q = q_1 + q_2 i + q_3 j + q_4 k \qquad q_1, q_2, q_3, q_3 \in \mathbb{R}$$

where

$$i^2 = -1, \qquad j^2 = -1, \qquad k^2 = -1$$

Similarly, the conjugate is defined as

$$\overline{q} = q_1 - q_2 i - q_3 j - q_4 k \qquad q_1, q_2, q_3, q_3 \in \mathbb{R}$$

The norm of a quaternion number is

$$|q| = q\bar{q} = \sqrt{q_1^2 + q_2^2 + q_3^2 + q_4^2}$$

With quaternion multiplication one has,

$\times$	1	i	j	k
1	1	i	j	k
i	i	-1	k	$-j$
j	j	$-k$	-1	i
k	k	j	$-i$	-1

This table is not a proof of but displays that, in general, multiplication over the quaternions is not commutative. This is an important fact to realise before defining concepts, such as the the dot product, for quaternion vectors as we do below.

Multiplication over the quaternions can be defined by their real components. Let $p, q \in \mathbb{H}$

$$\begin{aligned}
&p \times q \\
=&(a + bi + cj + dk)(e + f + gi + hk) \\
=&(ae - bf - cg - dh) + (af + be + ch - dg)i + (ag - bh + ce + df)j + (ah + bg - cf + de)k
\end{aligned}$$

Defining multiplication as a function in terms of these real components is useful in applications, such as constructing spherical designs.

[Wal20] defines a right vector space over the quaternions as $\mathbb{H}^d$. On a right vector space one can generalise the Euclidean inner product as $\mathbb{H}^d \times \mathbb{H}^d \to \mathbb{H}$ by

$$\langle u, v \rangle = \sum_{\ell=1}^{d} \overline{u_\ell} v_\ell$$

This inner product exhibits conjugate symmetry, linearity in the second variable, and positive definiteness. We do not provide further definitions of these properties as they are not required to understand the contents of this report.

3 Constructing Spherical Designs

3.1 Isomorphism of Vector Spaces

In linear algebra, it is often useful to use the fact that

$$\begin{gathered}\mathbb{C}^1 \simeq \mathbb{R}^2 \\ \mathbb{C}^2 \simeq \mathbb{R}^4 \\ \mathbb{C}^3 \simeq \mathbb{R}^6 \\ \vdots \\ \mathbb{C}^d \simeq \mathbb{R}^{2d}\end{gathered}$$

as vector spaces over the field $\mathbb{R}$.

Similarly, one can also make use of the fact that

$$\begin{gathered}\mathbb{H}^1 \simeq \mathbb{C}^2 \simeq \mathbb{R}^4 \\ \mathbb{H}^2 \simeq \mathbb{C}^4 \simeq \mathbb{R}^8 \\ \mathbb{H}^3 \simeq \mathbb{C}^6 \simeq \mathbb{R}^{12} \\ \vdots \\ \mathbb{H}^d \simeq \mathbb{C}^{2d} \simeq \mathbb{R}^{4d}\end{gathered}$$

as vector spaces over the field $\mathbb{R}$.

MATLAB has native arithmetic for the real numbers and the complex numbers but, despite support for basic quaternion operations, does not support required operations, namely matrix multiplication with quaternion entries.

Isomorphism of vector spaces is helpful in this case since we can express functions of the form

$$f : \mathbb{H}^{d,n} \to \mathbb{R}$$

in the more "digestible" form

$$f : \mathbb{R}^{d,4n} \to \mathbb{R}$$

for computational work.

Such isomorphisms were utilised to write auxiliary functions in MATLAB that would express quaternion operations, such as inner products, as real-valued functions.

While runtime for computations was long, the code was functional and able to reproduce expected results.

Scripts used to construct real, complex and quaternion (t,t)-designs (including supplementary scripts for quaternion operations) are included as appendices at the end of this report.

3.2 Numerical Construction of Spherical (t,t)-designs

Spherical designs are minima of the variational characterisation, so can be computed using software that performs constrained optimisation. For unweighted spherical (t,t)-designs, we seek to minimise the variational characterisation subject to each vector having unit norm.

So, for a given d, t and n we are solving

$$\min_{\mathbf{v}_1,\ldots,\mathbf{v}_n} \quad \sum_{j=1}^{n}\sum_{k=1}^{n} |\langle \mathbf{v}_j, \mathbf{v}_k\rangle|^{2t}$$

$$\text{subject to } ||\mathbf{v}_1|| = \cdots = ||\mathbf{v}_n|| = 1$$

To express the variational characterisation in MATLAB, we represent a set of n unit vectors in $\mathbb{F}^d$ (where $\mathbb{F} = \mathbb{R}, \mathbb{C}, \mathbb{H}$) as a $d \times n$ matrix X of unit vectors in $\mathbb{F}$. We then compute the Gramian matrix (matrix of inner products of column vectors) as $X'X$, perform exponentiation element-wise and then sum all elements.

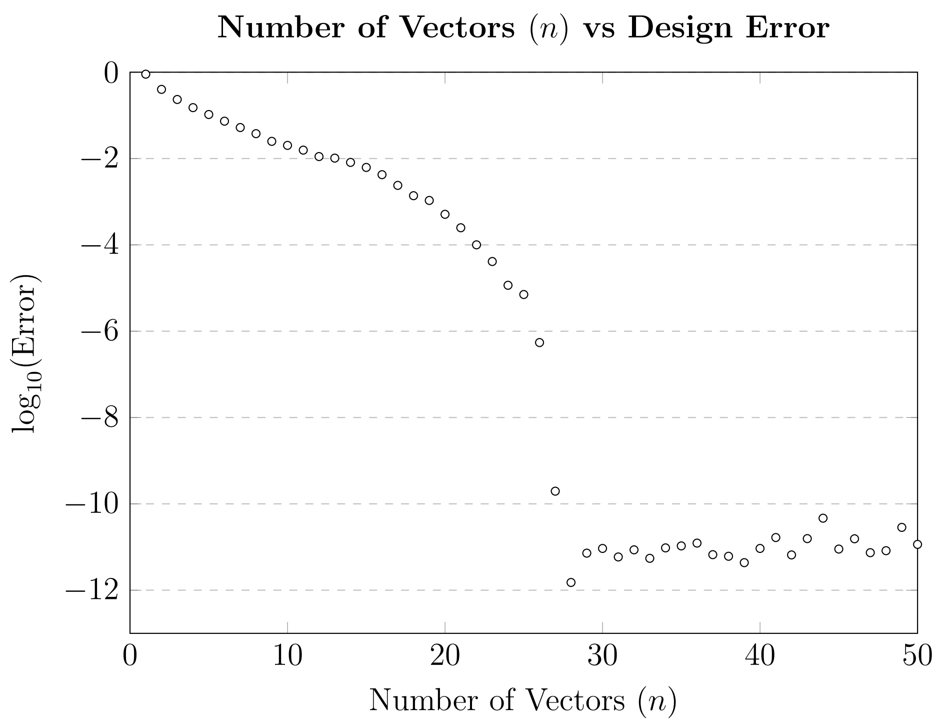
Manopt is a MATLAB package that optimises functions over manifolds. Conveniently, one supported manifold is the "oblique manifold" which consists of matrices where each column has unit norm [Bou+14].

To perform such an optimisation, the user must specify the manifold for optimisation to be performed on, the "cost" function to be minimised, a gradient function (either gradient or hessian) and the solver to be used. The scripts used to construct spherical (t,t)-designs in the real, complex and quaternion cases are attached as appendices.

When constructing designs, many parameters are involved and solvers use numerical methods to optimise (rather than computing local minima using a hessian matrix). This means that results are never "precise". That is, there is a difference between the true minimum of the variational characterisation and the value that Manopt attains. This is numeric error, and we need to know when we have a tolerable level of this error.

We are more interested in the smallest number of vectors for a design of a given order in a particular dimension is, than existence itself. So, one way to verify that we have indeed constructed a design is to iteratively run optimisations for different numbers of vectors and observe when the error "plummets" to zero.

An example is shown below for a $(3,3)$-design in $\mathbb{C}^3$, which we know exists.



Here, we see a drastic change in error from 26 vectors to 27 vectors, which suggests there exists a $(3, 3)$-design with 27 vectors. This reproduces known results from existing research [Bra11].

4 Results

4.1 Caution

Keep in mind when examining these results that in attempts to represent $d \times n$ quaternion matrices as $d \times 4n$ real-valued matrices, I have used real manifolds that do not perfectly represent the desired manifolds.

In my scripts, every four columns is meant to represent a single quaternion vector. This vector would ideally have a unit norm, however, each of these four columns itself has unit norm.

While my scripts normalise each representation of a quaternion vector to have a Frobenius norm of 1 before performing calculations, this still only represents a subset of the unit norm quaternion vectors.

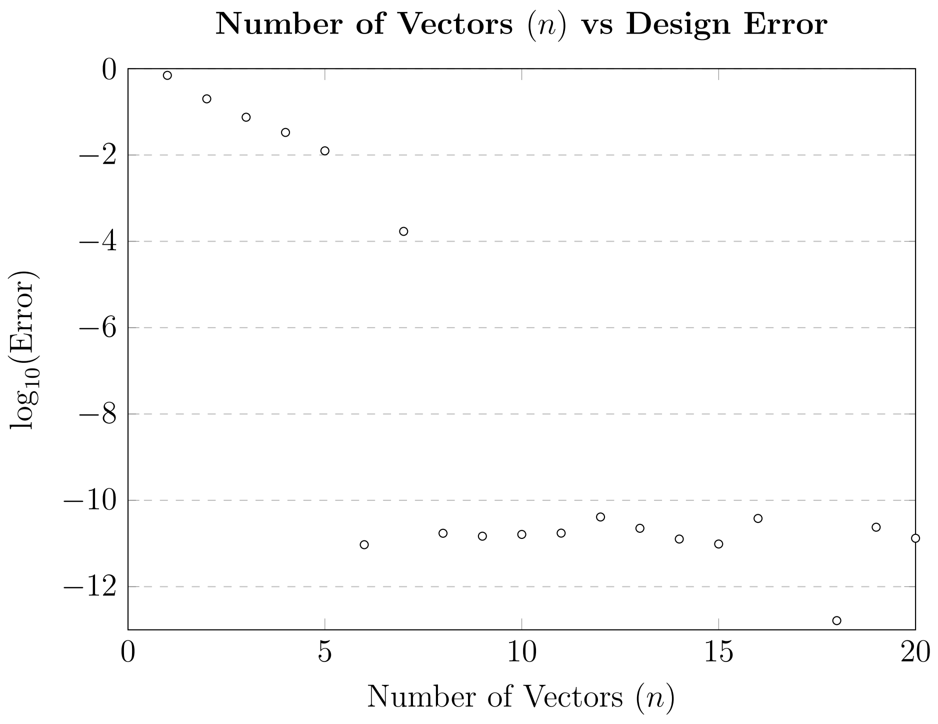
My results still represent existence of spherical (t, t)-designs and in some cases presented have been verified to reflect the minimum number of vectors needed. The implication of this, however, is that it may be the case that designs with fewer vectors do exist.

4.2 Quaternion Designs

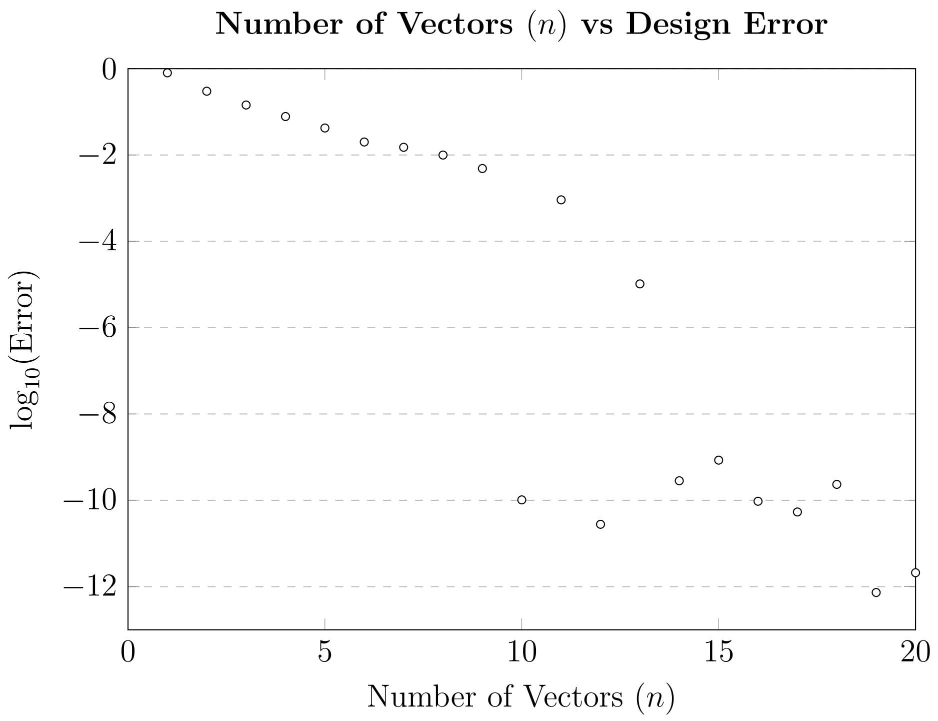
As well as being able to replicate known results for real and complex spherical (t, t)-designs of varying orders over different dimensions, we were able to construct a few designs over the quaternions. The results of constructing real and complex designs are omitted from this report (except for the example given in the previous section).

Below we display some graphs similar to the example in the previous section. These graphs provide numerical evidence of the existence of designs of the given orders (t) over the given dimensions (d).

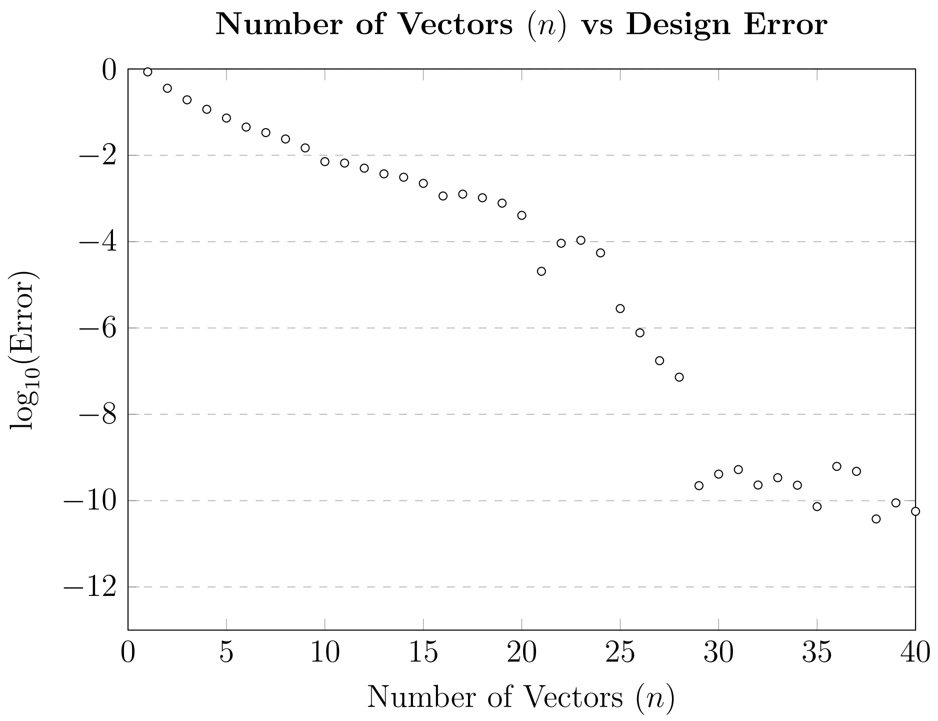
A Spherical $(2, 2)$-design in $\mathbb{H}^2$



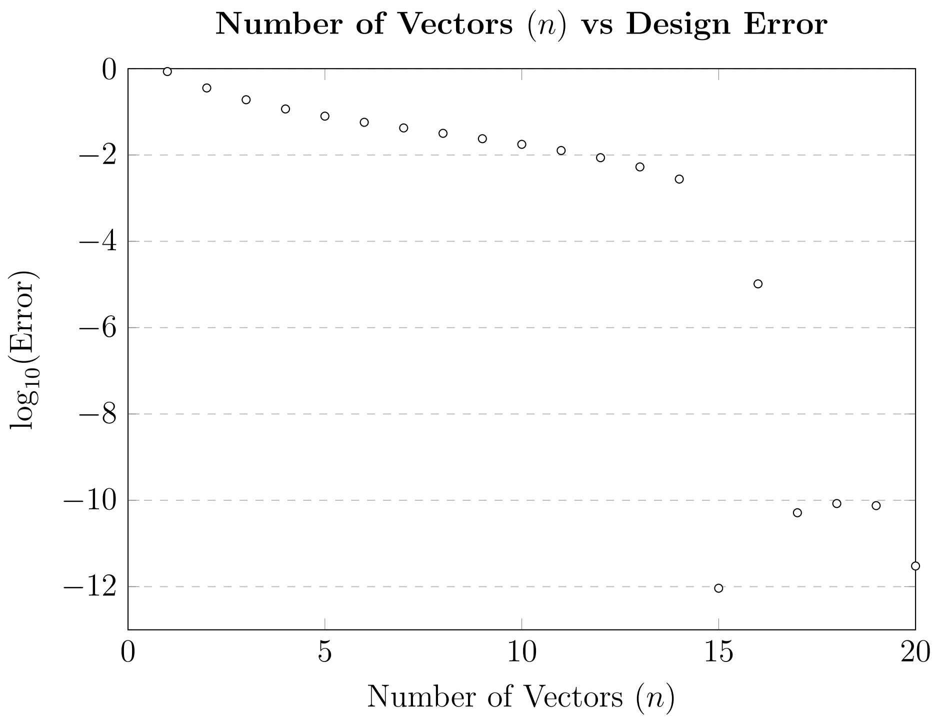
A Spherical $(3,3)$-design in $\mathbb{H}^2$



A Spherical $(4,4)$-design in $\mathbb{H}^2$



A Spherical $(2, 2)$-design in $\mathbb{H}^3$



The following summary table shows the minimum number of vectors (n) required to construct a spherical design of degree t over $\mathbb{H}^d$ for a given value of d.

d	t	n
2	2	6
2	3	10
2	4	29
3	2	15

Again, these first two results agree with results in existing literature [Hog82].

The three examples computed above are relatively low order designs in lower dimensions. Runtime for constructions increases largely even for designs with slightly higher parameters.

4.3 Finite Tight Frames in the Quaternions

It is a well-known fact that in $\mathbb{R}^2$ and $\mathbb{C}^2$, there is a unique set of $n = d + 1 = 3$ equiangular lines, often referred to as the Mercedes-Benz Frame [Wal17].

Based on results from these scripts, it appears that any set of 3 equiangular lines in $\mathbb{H}^2$, is also equivalent to the Mercedes-Benz frame.

Let us denote our computed $(1, 1)$-design in $\mathbb{H}^2$ by X. Using our auxiliary functions in MATLAB, we obtain

$$|X'X| = \begin{bmatrix} 1 & 0.5 & 0.5 \\ 0.5 & 1 & 0.5 \\ 0.5 & 0.5 & 1 \end{bmatrix}$$

Since the inner products of distinct vectors are all the same constant, this is indeed a set of equiangular lines [Wal17]. The value of 0.5 for this constant suggests that this set of equiangular lines is unitarily equivalent to the Mercedes-Benz frame.

It will be of interest to show more rigorously that the Mercedes-Benz frame is the unique set of 3 equiangular lines in $\mathbb{H}^2$. This intuition, however, provides a good starting point for this.

4.4 Furthering Results

It is my goal while continuing this research in the future to be able to create better real representations of quaternion unit vectors, in order to address the problems mentioned at the beginning of this section. I have some ideas I plan on testing of how to implement this, including:

- Instead using a power manifold of spherical manifolds
- Creating a custom manifold that better represents the quaternions

In fact, I did dedicate a decent amount of time of my summer research to adapting some existing manifolds to apply to the quaternions. However, it was not until after this that I found out that MATLAB does not support quaternion matrix multiplication. I hope to find some use for these manifolds in future research of spherical designs and finite tight frames.

Scripts for these manifolds are omitted from the report due to size, but all scripts from this project are accesible at: `https://github.com/ken-rayner/spherical_design/tree/main`.

5 Appendices

5.1 Appendix 1: Construction of Real (t, t)-designs

Constructs a real spherical (t, t)-design design given dimension of vector field, order of design, number of vectors and minimum number of optimisation iterations.

```matlab
%d = dimension of vector space
%t = order of the design
%n = number of vectors in design
%iterations = minimum number of iterations for solver
function error = construct_real_tt_design(d,t,n,iterations)

%define manifold for optimisation problem
%obliquefactory consists of real (d x n) matrices with unit norm columns
manifold = obliquefactory(d,n);

%set up optimisation problem
%manoptAD automatically calculates gradient and hessian of cost function
design.M = manifold;
design.cost = @(X) (1/(n^2))*(sum(sum((abs(X'*X).^(2*t)))));
design = manoptAD(design);

%sense check gradient
checkgradient(design);

%sets minimum number of iterations [GPT GENERATED]
options = struct();
options.miniter = iterations;

%run optimisation
[x,xcost] = trustregions(design,[],options)

%Calculates VC expected
ct_numerator = 1;
for i = 1:2:((2*t)-1)
    ct_numerator = ct_numerator*i
end

ct_denominator = 1;
for j = d:(d+2*(t-1))
    ct_denominator = ct_denominator*j
end

c_t = ct_numerator/ct_denominator

%calculates difference between calculated VC and expected VC
error = design.cost(x) - c_t;

%saves constructed design to global workspace
assignin("base","Design",x);

end
```

Runs iterations of construction script above and stores error from each iteration, outputting graph of error for each number of vectors.

```matlab
%d = dimension of vector space
%t = order of the design
%l = max values of n to iterate for
function real_design_min_vectors(d,t,l)

```

```
%runs construction script for 1 vectors, 2 vectors,..., l vectors
%saves error for each iteration
%saves error vector to global workspace
for i = 1:l
    error(i) = construct_real_tt_design(d,t,i,100);
    assignin("base","Error",error);
end

%plots number of vectors vs log10(error)
plot(1:l,log10(error),'.')

end
```

5.2 Appendix 2: Construction of Complex (t,t)-designs

Constructs a complex spherical (t,t)-design design given dimension of vector field, order of design, number of vectors and minimum number of optimisation iterations.

```
%d = dimension of vector space
%t = order of the design
%n = number of vectors in design
%iterations = minimum number of iterations for solver
function error = construct_complex_tt_design(d,t,n,iterations)

%define manifold for optimisation problem
%obliquecomplexfactory consists of complex (d x n) matrices with unit norm columns
manifold = obliquecomplexfactory(d,n);

%set up optimisation problem
%manoptAD automatically calculates gradient and hessian of cost function
design.M = manifold;
design.cost = @(X) (1/(n^2))*(sum(sum(abs(X'*X).^(2*t))));
design = manoptAD(design);

%sense check gradient
checkgradient(design);

%sets minimum number of iterations [GPT GENERATED]
options = struct();
options.miniter = iterations;

%run optimisation
[x,xcost] = trustregions(design,[],options)

%calculates difference between calculated VC and expected VC
error = design.cost(x) - 1/(nchoosek((d+t-1),t));

%saves constructed design to global workspace
assignin("base","Design",x);

end
```

Runs iterations of construction script above and stores error from each iteration, outputting graph of error for each number of vectors.

```
%d = dimension of vector space
%t = order of the design
%l = max values of n to iterate for
function complex_design_min_vectors(d,t,l)

%runs construction script for 1 vectors, 2 vectors,..., l vectors
```

```
%saves error for each iteration
%saves error vector to global workspace
for i = 1:l
    error(i) = construct_complex_tt_design(d,t,i,100);
    assignin("base","Error",error);
end

%plots number of vectors vs log10(error)
plot(1:l,log10(error),'.')

end
```

5.3 Appendix 3: Construction of Quaternionic (t, t)-designs

Constructs a quaternionic spherical (t, t)-design design given dimension of vector field, order of design, number of vectors and minimum number of optimisation iterations.

```
%d = dimension of vector space
%t = order of the design
%n = number of vectors in design
%iterations = minimum number of iterations for solver
function error = construct_quaternion_tt_design(d,t,n,iterations)

%define manifold for optimisation problem
%obliquefactory consists of complex (d x 4n) matrices with unit norm columns
%we represent a quaternion vector as a (d x 4) matrix
%[NOTE THAT THE CODE NORMALISES THIS, BUT STILL ONLY REPRESENTS A SUBSET OF UNIT NORM
    QUATERNIONS VECTORS]
manifold = obliquefactory(d,(4*n));


%defining the cost function
function quaternion_cost = qcost(X,t,n)

qarray = real_to_quaternion_arrays(X);

[ignore no_cells] = size(qarray);

for l = 1:no_cells
    for o = 1:no_cells
        gram_norm(l,o) = norm(qinnerproduct(qarray{l},qarray{o}));
    end
end

quaternion_cost = (1/n^2)*sum(sum(gram_norm.^(2*t)));

end


%script approximating the gradient [GPT GENERATED]
function gradient = forward_difference_approximation(X, t, n)
    % Compute the quaternion cost function at the original point
    quaternion_cost_original = qcost(X, t, n);

    % Initialize gradient vector
    gradient = zeros(size(X));

    % Iterate through each element of X to calculate the forward difference
    for i = 1:numel(X)
        % Perturb the i-th element positively
        X_perturbed = X;
        X_perturbed(i) = X_perturbed(i) + 0.000001;

        % Recalculate the quaternion cost function with the perturbed X
        quaternion_cost_perturbed = qcost(X_perturbed, t, n);

        % Compute the forward difference for the i-th element
        gradient(i) = (quaternion_cost_perturbed - quaternion_cost_original) / 0.000001;
    end
end


%set up optimisation problem
design.M = manifold;
design.cost = @(X) qcost(X,t,n);
design.egrad = @(X) forward_difference_approximation(X, t, n);

%sense check gradient
checkgradient(design);

%sets minimum number of iterations [GPT GENERATED]
options = struct();
options.miniter = iterations;

%run optimisation
[x,xcost] = trustregions(design,[],options)

%calculates difference between calculated VC and expected VC
error = xcost - (t+1)/nchoosek((2*d + t - 1),t);

%saves constructed design to global workspace
assignin("base","Design",x);

end
```

Runs iterations of construction script above and stores error from each iteration, outputting graph of error for each number of vectors.

```
%d = dimension of vector space
%t = order of the design
%l = max values of n to iterate for
function quaternion_design_min_vectors(d,t,l)

%runs construction script for 1 vectors, 2 vectors,..., l vectors
%saves error for each iteration
%saves error vector to global workspace
for i = 1:l
    error(i) = construct_quaternion_tt_design(d,t,i,100);
    assignin("base","Error",error);
end

%plots number of vectors vs log10(error)
plot(1:l,log10(error),'.')

end
```

Auxiliary function scripts were used to represent quaternion operations as real-valued functions. This script converts a $d \times 4n$ matrix into n $d \times n$ matrices each respectively representing a quaternion vector of dimension d.

```
function q_array = real_to_quaternion_arrays(A)

[real_r real_c] = size(A);

```

```
q_c = real_c/4;

for i = 1:q_c
    quat{i} = A(:,((i-1)*4 + 1):(i*4));
    quat{i} = quat{i} / norm(quat{i},'fro');
end

q_array = quat;

end
```

This script represents a calculation of the inner product (as a real-valued vector) of two quaternion vectors (as real-valued arrays).

```
function in_prod = qinnerproduct(q,p)

[r c] = size(q);

sum = [0 0 0 0];

for i = 2:c
    q(:,i) = -q(:,i);
end

for j = 1:r
    sum = sum + qmult(q(j,:),p(j,:));
end

in_prod = sum;

end
```

This script represents multiplication of two quaternion numbers (as real-valued vectors) and returns their product (as a real-valued vector).

```
function prod = qmult(q1,q2)

a = q1(1);
b = q1(2);
c = q1(3);
d = q1(4);
e = q2(1);
f = q2(2);
g = q2(3);
h = q2(4);

prod = [a*e-b*f-c*g-d*h, a*f+b*e+c*h-d*g, a*g-b*h+c*e+d*f, a*h+b*g-c*f+d*e];

end
```

6 References

[Avi10] S. Aviyente. “Frames and Overcomplete Dictionaries”. In: *Lecture* (2010).

[Baj92] B. Bajnok. “Construction of Spherical t-designs”. In: *Geometriae Dedicata* (1992).

[Bou+14] N. Boumal et al. “Manopt, a Matlab Toolbox for Optimization on Manifolds”. In: *Journal of Machine Learning Research* 15.42 (2014), pp. 1455–1459. URL: `https://www.manopt.org`.

[Bra11] J. Bramwell. “On the existence of spherical (t,t)–designs”. In: *Honours Project, Waipapa Taumata Rau* (2011).

[Hog82] S. G. Hoggar. “Spherical Designs”. In: *European Journal of Combinatorics* (1982).

[HS02] R. H. Hardin and N. J. A. Sloane. “Spherical Designs”. In: (2002). URL: `http://neilsloane.com/sphdesigns/`.

[Moh22] M. Mohammadpour. “The Construction of Complex and Quaternionic Spherical Designs”. PhD thesis. Waipapa Taumata Rau, 2022.

[SC03] N. J. A. Sloane and Phillipe Cara. “Spherical Designs in Four Dimensions”. In: (2003).

[Wal17] S. F. D. Waldron. *An Introduction to Finite Tight Frames*. Honeymoon Valley, Aotearoa: Springer, 2017.

[Wal20] S. F. D. Waldron. “A Variational Characterisation of Projective Spherical Designs over the Quaternions”. In: (2020).

[Wal23] S. F. D. Waldron. “Real and Complex Spherical Designs and their Gramian”. In: (2023).