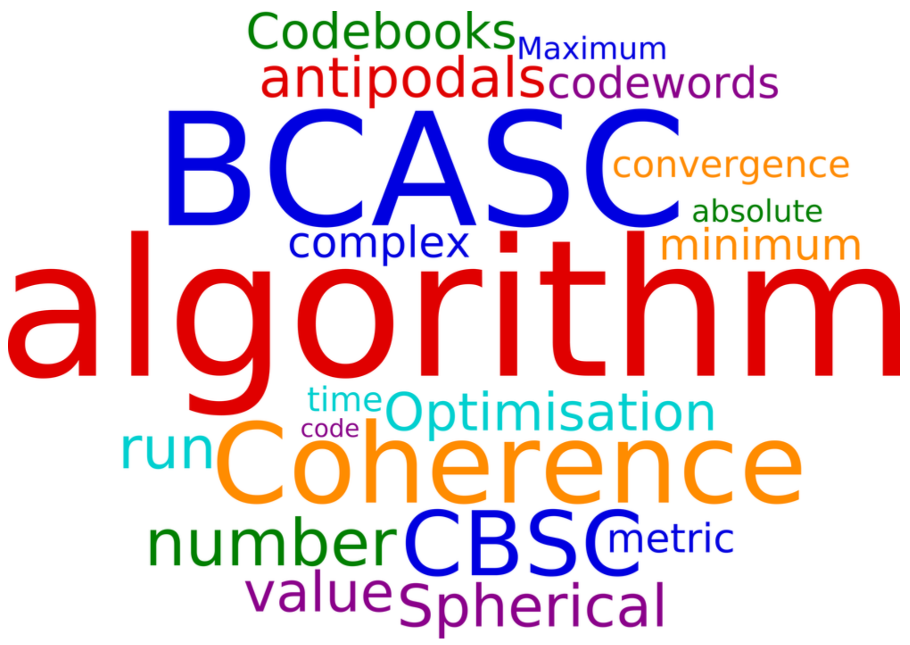

Abbreviations

BCASC	best complex antipodal spherical codes
CBGC	coherence-based Grassmannian codebook
CS	compressive sensing
NRF	National Research Foundation of South Africa
RHS	right-hand side

A coherence-based algorithm for optimising rank-1 Grassmannian codebooks

Heinrich E. A. Laue, *Student Member, IEEE,* and Warren P. du Plessis, *Senior Member, IEEE*

***Abstract*—An iterative algorithm for the numerical optimisation of rank-1 Grassmannian codebooks is presented. This algorithm achieves the same results as the current state-of-the-art algorithm, but requires a median of 9.52 times less computation time. This improvement is achieved by reformulating the optimisation problem to directly minimise coherence instead of maximising the Euclidean distance between codewords, thereby removing the need to consider large numbers of complex antipodals. The run-time improvement achieved allows the optimisation of the larger codebooks required in many applications.**

***Index Terms*—Rank-1 Grassmannian codebooks, coherence, optimisation, compressive sensing, sensing matrix.**

I. Introduction

The complex Grassmannian manifold $\mathbb{G}(R, M)$ is the set of all R-dimensional subspaces in $\mathbb{C}^M$. Finding the best packing of subspaces involves maximising the minimum distance between a set of N subspaces [1]. A rank-R Grassmannian codebook consists of N codewords, with each codeword being an $M \times R$ matrix with orthonormal columns. Each subspace is represented by a codeword since the span of R orthogonal vectors is an R-dimensional subspace. Optimised Grassmannian codebooks are of particular interest in multi-antenna communication systems [2]–[4], and recently, compressive sensing (CS) [5], [6].

Various approaches to designing Grassmannian codebooks exist [5]. In [3], optimal codebooks are constructed analytically from difference sets for certain dimensions. For other dimensions, a numerical method based on the Lloyd algorithm is proposed [3]. An alternating projection method which searches for optimal codebooks by alternately enforcing structural and spectral conditions is proposed in [7]. A sequential optimisation framework proposed in [4] formulates a minimax codebook problem, which is solved by sequentially approximating the $\max$ operator by the p-norm as $p \to \infty$.

The best complex antipodal spherical codes (BCASC) algorithm [5] follows a similar sequential procedure to [4] to obtain rank-1 codebooks (or antipodal spherical codes), which are shown to be better than those obtained by existing approaches. The BCASC algorithm seeks to maximise the minimum Euclidean distance between codewords. Rank-1 codewords represent lines, and all phase shifts of a codeword lie on the same line and are termed complex antipodals. It is therefore necessary to consider codewords and all their antipodals when computing Euclidean distances. This drastically increases the computational complexity of the problem since complex codewords have an infinite number of antipodals.

An algorithm for numerically optimising rank-1 codebooks with similar results to the BCASC algorithm, but with almost an order-of-magnitude reduction in run time is proposed. This is achieved by using a coherence-based metric on the Grassmannian manifold without explicit consideration of the codeword antipodals, enabling the optimisation of rank-1 codebooks of larger sizes within the same time.

Manuscript received 15 February 2017; accepted 27 March 2017.

This work is based on the research supported in part by the National Research Foundation of South Africa (NRF) (Grant specific unique reference number (UID) 85845). The NRF Grantholder acknowledges that opinions, findings and conclusions or recommendations expressed in any publication generated by the NRF supported research are that of the author(s), and that the NRF accepts no liability whatsoever in this regard.

The authors are with the Department of Electrical, Electronic and Computer Engineering, University of Pretoria, Pretoria 0002, South Africa (e-mail: laueheinrich@gmail.com; wduplessis@ieee.org)

II. Sequential Codebook Optimisation

Consider an $M \times N$ matrix $\mathbf{B}$ representing a rank-1 codebook with N column vectors $\left\{\mathbf{b}_n = [b_{1,n} \cdots b_{M,n}]^{\mathrm{T}}\right\}_{n=1}^{N}$ as codewords. The coherence of $\mathbf{B}$ is defined as [5]

$$\mu(\mathbf{B}) = \max_{n \neq l} \frac{|\langle \mathbf{b}_n, \mathbf{b}_l \rangle|}{\|\mathbf{b}_n\| \, \|\mathbf{b}_l\|} \tag{1}$$

where $\langle \mathbf{b}_n, \mathbf{b}_l \rangle = \mathbf{b}_n^{\mathrm{H}} \mathbf{b}_l$ and $n, l \in \{1, \ldots, N\}$. For unit-length vectors, the denominator in (1) is 1 and may be suppressed. The lower bound on the achievable coherence is given by [5]

$$\mu_c(M, N) = \begin{cases} \sqrt{\frac{N-M}{M(N-1)}} & \text{if } N \leq M^2, \\ \max\left(\sqrt{\frac{1}{M}}, \sqrt{\frac{2N-M^2-M}{(M+1)(N-M)}}, 1 - 2N^{\frac{-1}{M-1}}\right) & \text{if } M^2 < N \leq 2(M^2-1), \\ \max\left(\sqrt{\frac{2N-M^2-M}{(M+1)(N-M)}}, 1 - 2N^{\frac{-1}{M-1}}\right) & \text{if } 2(M^2-1) < N. \end{cases} \tag{2}$$

The coherence-optimisation problem is given by [3], [5]

$$\min \max_{n \neq l} |\langle \mathbf{b}_n, \mathbf{b}_l \rangle| \quad \text{subject to} \quad \|\mathbf{b}_n\|^2 = 1 \quad \forall n. \tag{3}$$

In order to obtain a smooth approximation to the $\max$ operator a series of sub-problems given by [4], [5]

$$\min \left(\sum_{n \neq l} |\langle \mathbf{b}_n, \mathbf{b}_l \rangle|^p \right)^{1/p} \quad \text{subject to} \quad \|\mathbf{b}_n\|^2 = 1 \;\; \forall n \tag{4}$$

with $p \geq 1$ may be solved with increasing values of p, thereby approximating the ∞-norm by the p-norm with $p \to \infty$ [4].

Coherence is well-defined on the Grassmannian manifold and is inversely equivalent to the distance metric $\left(1 - |\langle \mathbf{b}_n, \mathbf{b}_l \rangle|^2\right)^{1/2}$ common in line-packing problems [2], [4]. Minimising the maximum coherence of a codebook is equivalent to maximising the minimum distance between the lines represented by the codebook [5].

The approach in [4] uses optimisation algorithms which are able to enforce the orthonormality constraint in higher-rank codebooks. However, it is not necessary to resort to complicated algorithms when rank-1 codebooks which have only unit-length constraints are considered.

By considering rank-1 codebooks only, the BCASC algorithm is able to use a simpler optimisation technique which maximises the minimum Euclidean distance between codewords [5]. The Euclidean distance between the codewords alone does not lie on the Grassmannian manifold since it measures distances between codewords and not the lines spanned by the codewords. To ensure that optimisation is performed on the manifold, this approach requires maximising the distance between codewords and all their antipodals leading to

$$\max \min_{\substack{n\neq l \\ \phi\in[0,2\pi)}} \left\|\mathbf{b}_n - \mathbf{b}_l e^{j\phi}\right\| \quad \text{subject to} \quad \|\mathbf{b}_n\|^2 = 1 \quad \forall n. \tag{5}$$

The value of this approach is shown by the fact that the BCASC algorithm achieves lower coherence values for rank-1 codebooks than previous approaches [5].

III. Best Complex Antipodal Spherical Codes

As the current state-of-the-art approach to finding rank-1 codebooks, the BCASC algorithm is outlined below [5].

Ignoring codeword antipodals, (5) can be reformulated as

$$\min \sum_{n\neq l} \|\mathbf{b}_n - \mathbf{b}_l\|^{-p} \quad \text{subject to} \quad \|\mathbf{b}_n\|^2 = 1 \quad \forall n. \tag{6}$$

where $p \to \infty$. The method of Lagrange multipliers can be used to show that the conditions for a local minimum are

$$\mathbf{b}_n = \underline{\sum_{n\neq l} \frac{\mathbf{b}_n - \mathbf{b}_l}{\|\mathbf{b}_n - \mathbf{b}_l\|^{p+2}}} \quad \forall n \tag{7}$$

where underlining denotes the normalisation $\underline{\mathbf{u}} = \mathbf{u}/\|\mathbf{u}\|$.

The optimised codebook $\mathbf{B}$ can be found by interpreting (7) as a set of forces at optimisation step k ($\mathbf{f}_n^{(k)}$ is given by the right-hand side (RHS) of (7)) and adding a gain factor α multiplied by the matrix of forces ($\mathbf{F}^{(k)} = \left[\mathbf{f}_1^{(k)} \cdots \mathbf{f}_N^{(k)}\right]$) to the current matrix $\mathbf{B}^{(k-1)}$ giving

$$\mathbf{B}^{(k)} = \underline{\mathbf{B}^{(k-1)} + \alpha\mathbf{F}^{(k)}}, \quad k = 0, 1, \ldots \tag{8}$$

where underlining denotes normalisation $\underline{\mathbf{U}} = \{\mathbf{u}_n/\|\mathbf{u}_n\|\}_{n=1}^N$ to ensure that all $\|\mathbf{b}_n\|^2 = 1$. The result converges to a fixed point as $k \to \infty$ when α is small enough [5].

Extending (7) to consider the antipodals results in

$$\mathbf{f}_n^{(k)} = \underline{\int_0^{2\pi} \sum_{n\neq l} \frac{\mathbf{b}_n^{(k)} - \mathbf{b}_l^{(k)} e^{j\phi}}{\|\mathbf{b}_n^{(k)} - \mathbf{b}_l^{(k)} e^{j\phi}\|^{p+2}}\, d\phi} \tag{9}$$

which can be solved using numerical integration. Alternatively, the integral can be approximated by a summation to give

$$\mathbf{f}_n^{(k)} \approx \underline{\sum_{q=1}^{Q} \sum_{n\neq l} \frac{\mathbf{b}_n^{(k)} - \mathbf{b}_l^{(k)} e^{j2\pi q/Q}}{\|\mathbf{b}_n^{(k)} - \mathbf{b}_l^{(k)} e^{j2\pi q/Q}\|^{p+2}}} \tag{10}$$

where the number of antipodals considered, Q, determines the accuracy of the approximation. The main drawback of the BCASC algorithm is the computationally complexity resulting from the need to consider a large number of antipodals.

A random codebook may be used as the starting point for the first sub-problem. The result of each sub-problem is then used as starting point for the next sub-problem which uses a larger value of p. This process is repeated for $p \to \infty$.

IV. Coherence-Based Codebook Optimisation

An algorithm with a coherence metric which avoids the need to explicitly consider the antipodals is proposed to address the computational complexity of the BCASC algorithm.

Instead of using the metric $\sum_{n\neq l} \|\mathbf{b}_n - \mathbf{b}_l\|^{-p}$ as in (6), consider minimisation of the metric

$$g(\mathbf{B}) = \sum_{n\neq l} \left(|\langle\mathbf{b}_n, \mathbf{b}_l\rangle|^2 - \mu_t^2\right)^p, \quad p = 2, 4, \ldots \tag{11}$$

which minimises the absolute difference between the squares of the pair-wise absolute dot products and some target value $\mu_t = \beta\,\mu_c(M, N)$. Squaring the absolute value provides a smooth metric, and restricting p to even values ensures that the summed terms increase monotonically from their minima.

A non-zero target value is used in (11) to improve the rate of convergence [4]. However, the algorithm described in Section V uses a fixed iteration gain for each sub-problem making it prone to instability when $\beta = 1$, so it is necessary to reduce μ_t by choosing $\beta < 1$ to improve stability.

In order to enforce the unit-length constraint on the codewords, the Lagrange function is defined as

$$g(\mathbf{B}, \boldsymbol{\lambda}) = g(\mathbf{B}) + \sum_{n=1}^{N} \lambda_n \left(\|\mathbf{b}_n\|^2 - 1\right) \tag{12}$$

where $\boldsymbol{\lambda} = \{\lambda_n\}_{n=1}^N$ are the Lagrange multipliers for the N unit-length constraints. The conditions for finding the constrained minimum of (11) are [8]

$$\frac{\partial g(\mathbf{B}, \boldsymbol{\lambda})}{\partial \Re\{b_{m,n}\}} = 0, \quad \frac{\partial g(\mathbf{B}, \boldsymbol{\lambda})}{\partial \Im\{b_{m,n}\}} = 0 \quad \text{and} \quad \frac{\partial g(\mathbf{B}, \boldsymbol{\lambda})}{\partial \lambda_n} = 0 \tag{13}$$

for all $b_{m,n}$ and λ_n. These conditions are then used to derive the force vector at iteration k in Appendix A, leading to

$$\mathbf{f}_n^{(k)} = \underline{-2 \sum_{n\neq l} \left(|\langle\mathbf{b}_n^{(k)}, \mathbf{b}_l^{(k)}\rangle|^2 - \mu_t^2\right)^{p-1} \langle\mathbf{b}_l^{(k)}, \mathbf{b}_n^{(k)}\rangle \cdot \mathbf{b}_l^{(k)}} \tag{14}$$

The proposed algorithm iteratively updates the codebook using (8) leading to the coherence-based Grassmannian codebook (CBGC) algorithm.

In (14), the factor $\langle\mathbf{b}_l^{(k)}, \mathbf{b}_n^{(k)}\rangle \cdot \mathbf{b}_l^{(k)}$ is the vector projection of $\mathbf{b}_n^{(k)}$ onto $\mathbf{b}_l^{(k)}$. Due to the negative sign in (14), the weighted sum of projections are subtracted from $\mathbf{b}_n^{(k)}$ in (8), with those larger or smaller than the target μ_t given preference via the factor $\left(|\langle\mathbf{b}_n^{(k)}, \mathbf{b}_l^{(k)}\rangle|^2 - \mu_t^2\right)^{p-1}$, with larger p increasing the effect. Thus $\mathbf{b}_n^{(k)}$ is altered to reduce the components it has in common with other codewords.

Algorithm 1. Generalised coherence optimisation algorithm adapted from [5].

```
1: procedure OPTIMISE COHERENCE($M, N, \alpha_{\text{init}}$)
2:     $\mathbf{B}^{(0)}_{M\times N} \leftarrow$ random complex codebook
3:     $\varepsilon \leftarrow 10^{-10}$                          ▷ Convergence parameter
4:     $\alpha \leftarrow \alpha_{\text{init}}$
5:     $p_{\max} \leftarrow 2^9, p \leftarrow 2$
6:     $k_{\max} \leftarrow 10^5$                   ▷ Maximum number of iterations
7:     while $p \le p_{\max}$ do
8:         $k \leftarrow 1$
9:         while $k \le k_{\max}$ and $\left\|\mathbf{b}_n^{(k)} - \mathbf{b}_n^{(k-1)}\right\| < \varepsilon \;\; \forall n$ do
10:            Calculate $\mathbf{F}^{(k)} = \left[\mathbf{f}_1^{(k)}\, \mathbf{f}_2^{(k)} \cdots \mathbf{f}_N^{(k)}\right]$
11:            $\mathbf{B}^{(k)} = \underline{\mathbf{B}^{(k-1)} + \alpha\mathbf{F}^{(k)}}$        ▷ Update $\mathbf{B}$
12:            $k \leftarrow k+1$                   ▷ Increase iteration number
13:        end while
14:        $p \leftarrow 2p$                          ▷ $p$ for next sub-problem
15:        $\alpha \leftarrow \alpha_{\text{init}}/(p-1)$            ▷ $\alpha$ for next sub-problem
16:    end while
17: end procedure
```

V. Algorithm Implementation

Both the BCASC and CBGC algorithms were implemented using the generalisation of the BCASC algorithm shown in Algorithm 1. This was done to compare only the effect of the updates in (10) and (14) by using the same algorithm.

The main differences between Algorithm 1 and the form in [5] are the initial value of p and the convergence criteria. Algorithm 1 utilises codeword convergence $\left\|\mathbf{b}_n^{(k)} - \mathbf{b}_n^{(k-1)}\right\|$ rather than the convergence of force vectors to codewords $\left\|\mathbf{f}_n^{(k)} - \mathbf{b}_n^{(k)}\right\|$. This change is necessary as force vectors do not converge to codewords in the CBGC algorithm, while the codewords themselves always converge successfully.

The BCASC and CBGC algorithms were implemented in MATLAB R2016a and run on machines with two 6-core Intel Xeon E5-2630 processors and 32 GB of memory.

The BCASC algorithm may use numerical integration with (9) or a finite number of antipodals with (10) [5]. In order to vectorise the operations and to exploit MATLAB's parallel-processing capabilities, the discrete approximation was used, allowing the antipodals to be precomputed at each iteration.

The number of antipodals considered by the generalised BCASC algorithm (Q) was selected by comparing the results of ten tests for Q = 50, 100 and 200 shown in Table I. All coherence values were identical to three significant figures. In most cases the run times were longer for Q = 50 than for Q = 100 as a result of incomplete consideration of the antipodals, while Q = 200 only served to increase the run times. Therefore, Q = 100 was used in subsequent tests.

The results obtained using the original BCASC algorithm described in [5, Fig. 1] for Q = 100 without the changes mentioned above are also shown in Table I. The changes resulted in coherences which are equivalent to within 0.15%, and run times which are faster or only 3.6% slower in one case. These observations demonstrate that the BCASC algorithm is not compromised by the changes made.

The starting codebook $\mathbf{B}^{(0)}$ was generated by normalising the columns of a matrix whose elements had their real and imaginary parts drawn from a zero-mean Gaussian distribution. The variance of this distribution has no effect as the codes are normalised. The initial gain value, α_{init}, was set to 0.9 as in [5]. However, the CBGC algorithm required lower gain for

TABLE I
BCASC Coherence and Run Times of Ten Tests for Various Values of Q using Generalised and Original Forms

		Min. coherence, Q =				Mean run time (s), Q =			
M	N	50	100	200	100 [5]	50	100	200	100 [5]
4	7	0.354	0.354	0.354	0.354	292.0	2.22	3.35	8.21
4	16	0.447	0.447	0.447	0.447	1 010	10.70	13.78	39.67
4	64	0.688	0.688	0.688	0.687	23 816	25 471	26 891	30 531
5	10	0.333	0.333	0.333	0.333	1 915	2 312	3 590	2 231
5	16	0.390	0.390	0.390	0.390	1 855	1 751	2 218	2 837

stability and thus used $\alpha_{\text{init}} = 0.04$. A value of $\beta = 0.5$, midway between the BCASC approach ($\beta = 0$) and the optimum coherence ($\beta = 1$), resulted in convergence in all test cases.

Due to the large exponents p, a force vector in the CBGC algorithm would occasionally round to a zero vector before normalisation. Testing whether this occurred was thus necessary to avoid attempting to normalise a zero vector.

VI. Run-Time and Coherence Results

The minimum and maximum coherences and run times for ten tests are compared in Table II for codebooks of various dimensions obtained using the BCASC and CBGC algorithms. The lower bound from (2) and the BCASC coherences published in [5] are also shown.

The CBGC algorithm resulted in coherences equal to or better than those of the generalised implementation of the BCASC algorithm except $M = 4$, $N = 20$, where only the maximum coherence was worse. The CBGC algorithm also had minimum coherences equal to or better than the BCASC results published in [5] except for $M = 4$, $N = 64$.

The run times of the CBGC algorithm were significantly better than those of the BCASC algorithm except for $M = 4$, $N = 20$, where the minimum run time was marginally worse. When the mean run times of the BCASC and CBGC algorithms were compared for the codebook sizes in Table II, the CBGC algorithm was found to be a median of 9.52 times faster than the BCASC algorithm.

The number of iterations in Table II shows no clear benefit for either algorithm, with the CBGC algorithm requiring a median of only 14% fewer iterations, and each algorithm requiring fewer iterations for some of the problems.

A greater run-time improvement may be anticipated as the CBGC force vector in (14) has Q = 100 times fewer terms than the BCASC force vector in (10) and the median of the iterations are similar. However, the optimised implementation of (10) is better able to exploit the vectorisation [9] capabilities of the MATLAB development environment used during testing, thereby diminishing the run-time improvement achieved.

VII. Conclusion

An iterative algorithm which makes use of a coherence-based metric for optimising rank-1 Grassmannian codebooks has been proposed. Evaluating this algorithm against the Euclidean-distance-based BCASC algorithm within the same algorithmic framework has shown that the proposed algorithm results in almost an order-of-magnitude improvement in run time while achieving comparable or better coherence results.

TABLE II
COHERENCES, RUN TIMES AND ITERATIONS OF TEN TESTS FOR THE BCASC ($Q = 100$) AND CBGC ALGORITHMS

		Coherence						Run time (s)				Total number of iterations			
		μ_c	BCASC	Minimum		Maximum		Minimum		Maximum		Minimum		Maximum	
M	N	(2)	from [5]	BCASC	CBGC	BCASC	CBGC	BCASC	CBGC	BCASC	CBGC	BCASC	CBGC	BCASC	CBGC
2	8	0.7500	0.7950	0.7950	0.7942	0.7950	0.7942	1 827.09	4.09	1 861.76	5.40	800 203	9 419	800 302	12 516
3	16	0.6202	0.6491	0.6499	0.6491	0.6499	0.6494	3 354.15	184.49	3 522.21	294.97	457 639	211 024	466 508	339 380
4	5	0.2500	0.2500	0.2500	0.2500	0.2500	0.2500	0.24	0.03	0.32	0.09	162	61	195	83
4	6	0.3162	0.3277	0.3282	0.3276	0.3282	0.3276	21.61	5.54	23.28	5.76	11 066	16 592	11 883	17 119
4	7	0.3536	0.3536	0.3536	0.3536	0.3536	0.3536	2.04	0.22	2.39	0.28	977	518	1 143	647
4	8	0.3780	0.3780	0.3780	0.3780	0.3780	0.3780	2 041.57	255.73	2 069.23	263.36	800 022	600 033	800 023	600 034
4	9	0.3953	0.4022	0.4025	0.4022	0.4025	0.4022	63.34	23.05	66.32	24.09	20 905	48 475	21 939	50 194
4	10	0.4082	0.4118	0.4124	0.4118	0.4124	0.4118	39.80	11.20	47.50	11.61	11 320	21 301	12 313	21 813
4	16	0.4472	0.4472	0.4472	0.4472	0.4472	0.4472	9.44	0.83	12.29	1.13	1 170	947	1 501	1 298
4	20	0.5000	0.5000	0.5000	0.5000	0.5268	0.5316	57.15	57.86	2 074.32	447.25	5 019	55 682	184 045	409 252
4	64	0.6000	0.6869	0.6877	0.6874	0.6888	0.6884	20 944.52	2 828.72	29 267.22	3 176.66	339 264	626 272	480 875	701 432
5	6	0.2000	0.2000	0.2000	0.2000	0.2000	0.2000	0.63	0.06	0.77	0.08	289	109	357	134
5	7	0.2582	0.2670	0.2676	0.2667	0.2676	0.2667	270.53	4.10	275.53	4.57	121 747	10 498	123 379	11 567
5	8	0.2928	0.2955	0.2957	0.2954	0.2957	0.2954	623.44	12.35	633.72	12.69	229 401	28 091	232 326	28 684
5	9	0.3162	0.3207	0.3212	0.3205	0.3212	0.3205	247.97	57.24	276.22	59.78	76 273	118 701	84 569	121 465
5	10	0.3333	0.3333	0.3333	0.3333	0.3334	0.3333	106.86	0.95	3 784.14	377.52	27 297	1 699	900 000	700 002
5	16	0.3830	0.3889	0.3897	0.3889	0.3914	0.3904	951.62	116.40	2 978.83	321.67	110 763	132 058	347 026	366 100
5	43	0.4956	–	0.5007	0.5005	0.5380	0.5376	5 142.81	94.65	14 943.86	1 443.43	139 375	35 525	407 761	545 384
6	37	0.4082	–	0.4148	0.4082	0.4164	0.4151	6 559.04	242.48	12 449.61	1 160.20	208 980	107 876	397 956	507 834
7	42	0.3492	–	0.3720	0.3541	0.3773	0.3757	6 116.10	199.19	27 019.78	2 139.75	154 812	56 563	685 308	570 066

APPENDIX
CBGC DERIVATION

Define the functions

$$g_{n,l}(\mathbf{B}) = \left(|\langle \mathbf{b}_n, \mathbf{b}_l \rangle|^2 - \mu_t^2 \right)^p \quad \text{and} \tag{15}$$

$$g(\lambda_n) = \lambda_n \left(\|\mathbf{b}_n\|^2 - 1 \right). \tag{16}$$

The equilibrium conditions in (13) for a given value of m and n now become (all terms not dependent on n fall away)

$$\sum_{n \neq l} \frac{\partial g_{n,l}(\mathbf{B})}{\partial \Re\{b_{m,n}\}} + \frac{\partial g(\lambda_n)}{\partial \Re\{b_{m,n}\}} = 0, \tag{17}$$

$$\sum_{n \neq l} \frac{\partial g_{n,l}(\mathbf{B})}{\partial \Im\{b_{m,n}\}} + \frac{\partial g(\lambda_n)}{\partial \Im\{b_{m,n}\}} = 0 \quad \text{and} \tag{18}$$

$$\frac{\partial g(\lambda_n)}{\partial \lambda_n} = 0\,. \tag{19}$$

Now consider

$$\frac{\partial g_{n,l}(\mathbf{B})}{\partial \Re\{b_{m,n}\}} = p \left(|\langle \mathbf{b}_n, \mathbf{b}_l \rangle|^2 - \mu_t^2 \right)^{p-1} \times \frac{\partial}{\partial \Re\{b_{m,n}\}} |\langle \mathbf{b}_n, \mathbf{b}_l \rangle|^2 \tag{20}$$

$$= p \cdot h_{n,l} \times \frac{\partial}{\partial \Re\{b_{m,n}\}} |\langle \mathbf{b}_n, \mathbf{b}_l \rangle|^2\,.. \tag{21}$$

Then calculate

$$\frac{\partial}{\partial \Re\{b_{m,n}\}} |\langle \mathbf{b}_n, \mathbf{b}_l \rangle|^2 = \frac{\partial}{\partial \Re\{b_{m,n}\}} \left([\Re\{\langle \mathbf{b}_n, \mathbf{b}_l \rangle\}]^2 + [\Im\{\langle \mathbf{b}_n, \mathbf{b}_l \rangle\}]^2 \right) \tag{22}$$

$$= 2\Re\{\langle \mathbf{b}_n, \mathbf{b}_l \rangle\} \left(\frac{\partial}{\partial \Re\{b_{m,n}\}} \Re\left\{ \textstyle\sum_{m'=1}^{M} b_{m',n}^* b_{m',l} \right\} \right) + 2\Im\{\langle \mathbf{b}_n, \mathbf{b}_l \rangle\} \left(\frac{\partial}{\partial \Re\{b_{m,n}\}} \Im\left\{ \textstyle\sum_{m'=1}^{M} b_{m',n}^* b_{m',l} \right\} \right) \tag{23}$$

$$= 2\Re\{\langle \mathbf{b}_n, \mathbf{b}_l \rangle\} \left(\frac{\partial}{\partial \Re\{b_{m,n}\}} \Re\{ b_{m,n}^* b_{m,l} \} \right) + 2\Im\{\langle \mathbf{b}_n, \mathbf{b}_l \rangle\} \left(\frac{\partial}{\partial \Re\{b_{m,n}\}} \Im\{ b_{m,n}^* b_{m,l} \} \right) \tag{24}$$

$$= 2\big(\Re\{b_{m,l}\} \Re\{\langle \mathbf{b}_n, \mathbf{b}_l \rangle\} + \Im\{b_{m,l}\} \Im\{\langle \mathbf{b}_n, \mathbf{b}_l \rangle\} \big) \tag{25}$$

$$= r_{m,n,l}\,. \tag{26}$$

Next consider

$$\frac{\partial g(\lambda_n)}{\partial \Re\{b_{m,n}\}} = \frac{\partial}{\partial \Re\{b_{m,n}\}} \lambda_n \Re^2\{b_{m,n}\} = 2\lambda_n \Re\{b_{m,n}\}\,. \tag{27}$$

Combining (17), (21), (26) and (27) and evaluating gives

$$\Re\{b_{m,n}\} = \frac{p}{2\lambda_n} \left[-\textstyle\sum_{n \neq l} h_{n,l} \cdot r_{m,n,l} \right] = \frac{p}{2\lambda_n} u_{m,n}\,. \tag{28}$$

Following similar logic for the imaginary case gives

$$\Im\{b_{m,n}\} = \frac{p}{2\lambda_n} \left[-\textstyle\sum_{n \neq l} h_{n,l} \cdot s_{m,n,l} \right] = \frac{p}{2\lambda_n} v_{m,n} \tag{29}$$

where

$$s_{m,n,l} = 2\left(\Im\{b_{m,l}\} \Re\{\langle \mathbf{b}_n, \mathbf{b}_l \rangle\} - \Re\{b_{m,l}\} \Im\{\langle \mathbf{b}_n, \mathbf{b}_l \rangle\} \right) \tag{30}$$

analogous to $r_{m,n,l}$ in (26). Combining (19), (28) and (29) and evaluating gives

$$2\lambda_n = p \sqrt{\textstyle\sum_{m'=1}^{M} u_{m',n}^2 + v_{m',n}^2}\,. \tag{31}$$

The codeword $\mathbf{b}_n$, taken as the force vector, is found by combining (28), (29) and (31) for $m \in \{1, \ldots, M\}$ to give

$$\mathbf{f}_n = (\mathbf{u}_n + j\mathbf{v}_n) / \|\mathbf{u}_n + j\mathbf{v}_n\|\,. \tag{32}$$

Combining the real and imaginary parts of $\mathbf{f}_n$ leads to (14).

References

[1] J. H. Conway, R. H. Hardin, and N. J. Sloane, "Packing lines, planes, etc.: Packings in Grassmannian spaces," *Experiment. Math.*, vol. 5, no. 2, pp. 139–159, 1996.

[2] D. J. Love, R. W. Heath, and T. Strohmer, "Grassmannian beamforming for multiple-input multiple-output wireless systems," *IEEE Trans. Inf. Theory*, vol. 49, no. 10, pp. 2735–2747, Oct. 2003.

[3] P. Xia, S. Zhou, and G. B. Giannakis, "Achieving the Welch bound with difference sets," *IEEE Trans. Inf. Theory*, vol. 51, no. 5, pp. 1900–1907, May 2005.

[4] A. Medra and T. N. Davidson, "Flexible codebook design for limited feedback systems via sequential smooth optimization on the Grassmannian manifold," *IEEE Trans. Signal Process.*, vol. 62, no. 5, pp. 1305–1318, Mar. 2014.

[5] H. Zörlein and M. Bossert, "Coherence optimization and best complex antipodal spherical codes," *IEEE Trans. Signal Process.*, vol. 63, no. 24, pp. 6606–6615, Dec. 2015.

[6] H. E. A. Laue and W. P. du Plessis, "Compressive direction-finding antenna array," in *Proc. IEEE-APS Topical Conf. on Antennas and Propag. in Wireless Commun. (APWC)*, Sep. 2016, pp. 158–161.

[7] I. S. Dhillon, J. R. Heath, T. Strohmer, and J. A. Tropp, "Constructing packings in Grassmannian manifolds via alternating projection," *Experimental mathematics*, vol. 17, no. 1, pp. 9–35, 2008.

[8] J. Nocedal and S. J. Wright, *Numerical optimization*. New York, USA: Springer-Verlag, 1999.

[9] W. P. du Plessis, "Efficient computation of array factor and sidelobe level of linear arrays [EM programmer's notebook]," *IEEE Antennas and Propagation Magazine*, vol. 58, no. 6, pp. 102–114, Dec. 2016.