(Re)constructing code loops

Ben Nagy and David Michael Roberts

Abstract. Write later.

The theory of codes makes for a fascinating study. At their heart, codes are 'merely' subspaces of vector spaces over some small finite field, with certain combinatorial properties. Why do such things exist? Like a lot of exceptional objects in combinatorics, it can come down to: "because". This makes constructing codes sometimes more of an art than something systematic. In this paper, we are going to consider the construction of certain structures closely related to codes, called *code loops*. We will only consider the case where the base field is $\mathbb{F}_2 = \{0, 1\}$, and refer to its elements as *bits*.

The first published example of a code loop appeared as a step in John Conway's construction of the Monster sporadic simple group [**3**]. The code loop Conway used originally appeared in an unpublished manuscript by Richard Parker. A general study of code loops was then made by Robert Griess [**6**]. Griess also proved the existence of code loops by an algorithmic construction, starting from a particular type of code. More recent approaches will be discussed below.

Recall that the elements (or *words*) in a code $\mathcal{C}$, being vectors, can be combined by addition—this is a group operation and hence associative. The elements of a code loop consist of a pair: a code word and one extra bit. The extra bit *twists* the addition so that combination of code loop elements is a *non-associative operation*: $(xy)z \neq x(yz)$ in general.

More specifically, while addition of words in a code is performed by coordinate-wise addition in $\mathbb{F}_2$ (bitwise XOR) the algebraic operation in a code loop is not so easily described. The code loop operation can be reconstructed from a function $\mathcal{C} \times \mathcal{C} \to \mathbb{F}_2$ satisfying certain identities, called a *twisted cocycle*. It is the computation and presentation of these functions that will mainly concern us in this article, using Griess's algorithm [**6**, proof of Theorem 10]. As a result, we will observe some curious features of the Parker loop, obtained via experimentation and, it seems, previously unknown.

1. EXTENSIONS AND COCYCLES. As a warm-up, we will describe a more familiar structure using the techniques that will be used later. Recall that the *quaternion group* Q_8 is the group consisting of the positive and negative basis quaternions:

$$Q_8 = \{1, i, j, k, -1, -i, -j, -k\}$$

The elements of Q_8 satisfy the identities

$$i^2 = j^2 = k^2 = -1, \quad ij = k.$$

There is a surjective group homomorphism $\pi\colon Q_8 \to \mathbb{F}_2 \times \mathbb{F}_2 = V_4$, sending i to $(1, 0)$ and j to $(0, 1)$, and the kernel of π is the subgroup $\{1, -1\} \simeq \mathbb{F}_2$.

Moreover, this kernel is the *center* of Q_8, the set of all elements that commute with every other element of the group. This makes Q_8 an example of a *central extension*: $\mathbb{F}_2 \to Q_8 \to V_4$.

Now Q_8 is a nonabelian group, but both $\mathbb{F}_2$ and V_4 are abelian. One might think that it shouldn't be possible to reconstruct Q_8 from the latter two groups, but it is! That is, if we are given some extra information that uses only the two abelian groups. There is an obvious function $s\colon V_4 \to Q_8$, sending $(0,0)$ to 1, $(1,0)$ to i, $(0,1)$ to j and $(1,1)$ to k. This almost looks like a group homomorphism, but it is not, as $(1,0)+(1,0) = (0,0)$ in V, but $s(1,0)s(0,1) = i^2 \neq 1 = s(0,0)$ in Q_8. We can measure the failure of s to be a group homomorphism by considering the two-variable function

$$d\colon V_4 \times V_4 \to \mathbb{F}_2$$

defined by $(-1)^{d(v,w)} = s(v)s(w)s(v+w)^{-1}$. It is a nice exercise to see that $s(v)s(w)s(v+w)^{-1}$ is always ± 1, so that this definition makes sense. The values of $d(v,w)$ are given as:

$v \setminus w$	00	10	01	11
00	0	0	0	0
10	0	1	1	0
01	0	0	1	1
11	0	1	0	1

where $00 = (0,0)$, $10 = (1,0)$ and so on. If s *were* a homomorphism, d would be constant at 0. One can check that d satisfies the *cocycle identities*:

$$d(v,w) - d(u+v,w) + d(u,v+w) - d(u,v) = 0$$

for all triples $u, v, w \in V_4$. It is also immediate from the definition that $d(0,0) = 0$. An alternative visualisation is given in Figure 1.

The reason for this somewhat mysterious construction is that we can build a bijection of *sets* using s and the isomorphism $\mathbb{F}_2 \simeq \{1,-1\}$, namely

$$\mathbb{F}_2 \times V_4 \simeq (\{1\} \times V_4) \cup (\{-1\} \times V_4) \xrightarrow{\phi} \{1, i, j, k\} \cup \{-1, -i, -j, -k\} = Q_8\,.$$

Now, if we define a new product operation on the underlying set of $\mathbb{F}_2 \times V_4$ by

$$(s,v) *_d (t,w) := (s + t + d(v,w), v + w),$$

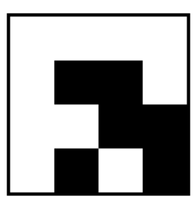

Figure 1. A 4×4 array giving the values of the cocycle $d\colon V_4 \times V_4 \to \mathbb{F}_2$, with white = 0, black = 1.

then the cocycle identities ensure that this is in fact associative and further, a group operation. Finally, ϕ can be checked to be a homomorphism for the group operation on Q_8 and for $*_d$, hence is a group isomorphism.

Thus we can reconstruct, at least up to isomorphism, the nonabelian group Q_8 from the two abelian groups V_4 and $\mathbb{F}_2$, together with the *cocycle* $d\colon V_4 \times V_4 \to \mathbb{F}_2$. If we didn't know about the group structure of Q_8 already, we could construct it from scratch using d. We can construct the Parker loop using a similar approach.

2. TWISTED COCYCLES AND LOOPS. The construction in the previous section is a fairly typical case of reconstructing a central extension from a cocycle (although in general one does not even need the analogue of the group V_4 to be abelian). However, we wish to go one step further, and construct a structure with a *non-associative* product from a pair of abelian groups: the group $\mathbb{F}_2$ and additive group of a vector space V over $\mathbb{F}_2$. Instead of a cocycle, we use a *twisted cocycle*: a function $\alpha\colon V \times V \to \mathbb{F}_2$ like d that instead satisfies

$$\alpha(v,w) - \alpha(u+v,w) + \alpha(u,v+w) - \alpha(u,v) = f(u,v,w),$$

for a special *twisting function* $f\colon V \times V \times V \to \mathbb{F}_2$. We will assume that α satisfies $\alpha(0,v) = \alpha(v,0) = 0$ for all $v \in V$, a property that holds for d in the previous section. From a twisted cocycle the set $\mathbb{F}_2 \times V$ can be given a binary operation $*_\alpha$:

$$(s,v) *_\alpha (t,w) := (s + t + \alpha(v,w), v + w).$$

We denote $\mathbb{F} \times V$ equipped with this binary operation by $\mathbb{F}_2 \times_\alpha V$.

Recall that a *loop* is a set L with a binary operation $\star\colon L \times L \to L$, a unit element $e \in L$ such that $e \star x = x \star e = x$ for all $x \in L$, and such that for each $z \in L$, the functions $r_z(x) = x \star z$ and $\ell_z(x) = z \star x$ are bijections $L \to L$. Informally, this means that every element $z \in L$ has a left inverse and a right inverse for $\star$, and these are unique—but may be different in general. The following is a cute exercise using the twisting function and the assumption that $\alpha(0,v) = \alpha(v,0) = 0$.

Lemma 1. *The operation $*_\alpha$ makes $\mathbb{F}_2 \times_\alpha V$ into a loop, with identity element $(0,\underline{0})$, for $\underline{0}$ the zero vector in V. There is a homomorphism $\pi\colon \mathbb{F}_2 \times_\alpha V \to V$ that projects onto the second factor, and whose kernel is $\mathbb{F}_2 \times \{\underline{0}\}$.*

Groups are examples of loops, but they are in a sense the uninteresting case. Arbitrary loops are quite badly behaved: their product is non-associative in general. But there is a special non-associative case, introduced by Ruth Moufang [**11**], with better algebraic properties.

Definition. A *Moufang loop* is a loop $(L,\star)$ satisfying the identity

$$x \star (y \star (x \star z)) = ((x \star y) \star x) \star z$$

for all choices of elements $x, y, z \in L$.

The most famous example of a Moufang loop is probably the set of non-zero octonions under multiplication. A key property of a Moufang loop L is that any subloop $\langle x,y\rangle < L$ generated by a pair of elements x, y is in fact a group. As a corollary, powers of a *single* element are well-defined, and do not require extra bracketing: $x \star (x \star x) = (x \star x) \star x =: x^3$, for example. Additionally, the left and right inverses *always* agree in a Moufang loop, so that for each $x \in L$, there is a unique x^{-1} such that $x \star x^{-1} = x^{-1} \star x = e$. Importantly for us, code loops, defined below as a special case of the construction of $\mathbb{F} \times_\alpha V$, turn out to be Moufang.

Example 1. Let $V = (\mathbb{F}_2)^3$. The 16-element Moufang loop $M := M_{16}(C_2 \times C_4)$ of [1, Theorem 2] is isomorphic to $\mathbb{F}_2 \times_\mu V$, arising from a twisted cocycle $\mu \colon V \times V \to \mathbb{F}_2$ given by the 8×8 array in Figure 2:

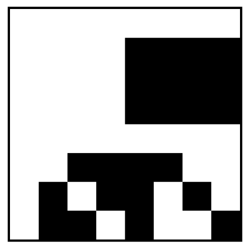

Figure 2. Twisted cocycle for the Moufang loop M, white = 0, black = 1. The order of the row/colum labels is 000, 100, 010, 110, 001, 101, 011, 111.

Notice in particular that the first four columns/rows correspond to the subgroup $U \subset V$ generated by 100 and 010, and that the restriction of μ to $U \times U$ is identically zero (i.e. white). This means that the restriction $M\big|_U < M$ (the subloop of M whose elements are mapped to U by $M \to V$) is isomorphic to the direct product $\mathbb{F}_2 \times U$, and in particular a group.

3. CODES AND CODE LOOPS. To describe the twisting function f for our code loops, we need to know about some extra operations that exist on vector spaces over the field $\mathbb{F}_2$. For W an n-dimensional vector space over $\mathbb{F}_2$ and vectors $v, w \in W$, there is a new vector $v \mathbin{\&} w \in W$ given by

$$v \mathbin{\&} w := (v_1 w_1,\, v_2 w_2,\, \ldots,\, v_n w_n).$$

If we think of such vectors as binary words, then this is bitwise AND. Note that if we take a code $\mathcal{C} \subset (\mathbb{F}_2)^n$, then $\mathcal{C}$ is not guaranteed to be closed under this operation. The other operation takes a vector $v \in W$ and returns its *weight*: the sum, as an integer, of its entries: $|v| := v_1 + \cdots + v_n$. Equivalently, it is the number of nonzero entries in v.

The desired twisting function is a combination of these two, namely $f(u, v, w) := |u \mathbin{\&} v \mathbin{\&} w|$. However, as alluded to above, we are going to ask that further identities hold. For these identities to make sense we need to start with a code with the special property of being *doubly even*.

Definition. A code $\mathcal{C} \subset (\mathbb{F}_2)^n$ is *doubly even* if for every word $v \in \mathcal{C}$, $|v|$ is divisible by 4.

Example 2. The *Hamming (8,4) code* is the subspace of $(\mathbb{F}_2)^8$ spanned by the rows of the matrix

$$\begin{pmatrix} 1&0&0&0&0&1&1&1 \\ 0&1&0&0&1&0&1&1 \\ 0&0&1&0&1&1&0&1 \\ 0&0&0&1&1&1&1&0 \end{pmatrix}$$

and is doubly even.

A more substantial example is given by the Golay code.

Example 3. The (extended binary) Golay code $\mathcal{G} \subset (\mathbb{F}_2)^{24}$ is the span of the following (row) vectors, denoted $b_1, \ldots, b_6$ (left) and $b_7, \ldots, b_{12}$ (right):

000110000000010110100011	101001011100111001111111
101001111101101111110001	100000011100001001001100
000100000000100100111110	000001000000111001001110
010000000010000110101101	100000001000111000111000
000000000010010101010111	100000000100101000010111
100000000000100111110001	011011000001111011111111

This is a different basis from the more usual ones (e.g. [**4**, Figure 3.4]), which can be taken as the rows of a 12×24 matrix whose left half is the 12×12 identity matrix. Our basis, however, allows us to demonstrate some interesting properties below.

The inclusion/exclusion formula applied to counting nonzero entries allows us to show that, for all v and w in any doubly even code $\mathcal{C}$,

$$|v + w| + |v \,\&\, w| = |v| + |w| - |v \,\&\, w| \,.$$

In other words: $|v \,\&\, w| = \frac{1}{2}(|v| + |w| - |v + w|)$, which implies that $|v \,\&\, w|$ is divisible by 2. Thus for words v, w in a doubly even code, both $\frac{1}{4}|v|$ and $\frac{1}{2}|v \,\&\, w|$ are integers.

Definition (Griess [6]). Let $\mathcal{C}$ be a doubly even code. A *code cocycle* $\theta \colon \mathcal{C} \times \mathcal{C} \to \mathbb{F}_2$ is a function satisfying the identities

$$\theta(v, w) - \theta(u + v, w) + \theta(u, v + w) - \theta(u, v) = |u \,\&\, v \,\&\, w| \pmod 2 \tag{1}$$

$$\theta(v, w) + \theta(w, v) = \tfrac{1}{2}|v \,\&\, w| \pmod 2 \tag{2}$$

$$\alpha(v, v) = \tfrac{1}{4}|v| \pmod 2 \tag{3}$$

Remark. What we call a code cocycle, Griess actually calls a 'factor set'. Given that a code cocycle is an example of a twisted cocycle, we prefer a name that indicates this.

There is a notion of what it means for two twisted cocycles to be equivalent, and equivalent twisted cocycles $\alpha \sim \beta$ give isomorphic loops $\mathbb{F}_2 \times_\alpha V \simeq \mathbb{F}_2 \times_\beta V$. As part of [**6**, Theorem 10], Griess proves that all code cocycles for a given doubly even code are equivalent, and hence give isomorphic cocde loops.

It is not obvious, on first consideration, that code cocycles even exist, or how many there are for a given doubly even code. However, Griess gave a proof that inductively constructs code cocycles, and counts how many arbitrary choices can be made along the way, proving that code cocycles do indeed exist. The number of possible code cocycles quickly becomes fearsome: 2^{2^k-k-1}, for $k = \dim \mathcal{C}$ ([**6**, Theorem 10]). For the 4-dimensional Hamming code given above, this is 512, but for the Golay code there are 2^{4083} possible code cocycles, a number with 1230 digits in base 10. However, the set of all possible functions $\mathcal{C} \times \mathcal{C} \to \mathbb{F}_2$ has 2^{k^2} elements; for the Golay code ($k = 12$) this number is astronomical. A brute-force search is completely infeasible.

4. GRIESS'S ALGORITHM AND ITS OUTPUT. The algorithm that Griess describes in the proof of [**6**, Theorem 10] to construct code cocycles for a code $\mathcal{C}$ takes as input an ordered basis $\{b_0, \dots, b_{k-1}\}$ for $\mathcal{C}$. The code cocycle is then built up inductively over larger and larger subspaces $V_i = \mathrm{span}\{b_1, \dots, b_i\}$.

However, the description by Griess is more of an outline, using steps like 'determine the cocycle on such-and-such subset using identity X', where X refers to one of (1), (2), (3), or corollaries of these. We have reconstructed the process in detail in Algorithm 1.

We implemented Algorithm 1 in the language Go [**12**], together with diagnostic tests, for instance to verify the Moufang property. The output of the algorithm is the code cocycle: a matrix of zeroes and ones with rows and columns labelled by words in the given code, and can be displayed as an array of black and white pixels. The pixel colours correspond to ones and zeroes, as in Example 1. For the Golay code the image looks like 16 million pixels (more accurately, 4096×4096) of noise.

As a combinatorial object, the code cocycle $\theta \colon \mathcal{G} \times \mathcal{G} \to \mathbb{F}_2$ constructed from Algorithm 1 using the basis in Example 3 is too large and unwieldy to examine for any interesting structure. Moreover, to calculate with the Parker loop $\mathcal{P} := \mathbb{F}_2 \times_\theta \mathcal{G}$ one needs to know all 16 million or so values of θ. It is thus desirable to have a method that will calculate values of θ by a method shorter than Algorithm 1.

Lemma 2. *Let $\mathcal{C}$ be a doubly even code, θ a code cocycle on it, and $\mathcal{C} = V \oplus W$ a decomposition into complementary subspaces. Then for $v_1, v_2 \in V$ and $w_1, w_2 \in W$,*

$$\begin{aligned} \theta(v_1 + w_1, v_2 + w_2) = {} & \theta(v_1, v_2) + \theta(w_1, w_2) + \theta(v_1, w_1) \\ & + \theta(w_2, v_2) + \theta(v_1 + v_2, w_1 + w_2) \\ & + \tfrac{1}{2}|v_2 \,\&\, (w_1 + w_2)| + |v_1 \,\&\, v_2 \,\&\, (w_1 + w_2)| \\ & + |w_1 \,\&\, w_2 \,\&\, v_2| + |v_1 \,\&\, w_1 \,\&\, (v_2 + w_2)| \pmod 2 . \end{aligned} \tag{4}$$

Proof. sorry. ∎

Observe that in Lemma 2, on the right hand side of (4), the code cocycle θ is only ever evaluated on vectors from the subset $V \cup W \subset \mathcal{C}$. This means that if we throw away all of the array encoding the values of θ except those positions with labels coming from $V \cup W$, then we can still reconstruct arbitrary values of θ using (4). If we assume that $\mathcal{C}$ is $2k$-dimensional, and that V and W are both k-dimensional, then the domain of the restricted θ has $(2^k + 2^k - 1)^2 = 2^{2(k+1)} - 2^{k+2} + 1 = O((2^k)^2)$ elements. Compare this to the full domain of θ, which has $2^{2k} \times 2^{2k} = (2^k)^4$ elements, giving a roughly square-root saving.

And, now it should be clear why the Golay code basis in Example 3 was partitioned into two lists of six vectors: we can reconstruct all 16,777,216 values of the resulting code cycle θ, and hence the Parker loop multiplication, from a mere $2^{14} - 2^{8} + 1 = 16,129$ values. The span of the left column of vectors in Example 3 is the subspace $V \subset \mathcal{G}$, and the span of the right column of vectors is $W \subset \mathcal{G}$.

The top left quadrant of Figure 3 then contains to the restriction of θ to $V \times V$, and the bottom right quadrant the restriction to $W \times W$. The off-diagonal quadrants contain the values of θ restricted to $V \times W$ and $W \times V$.

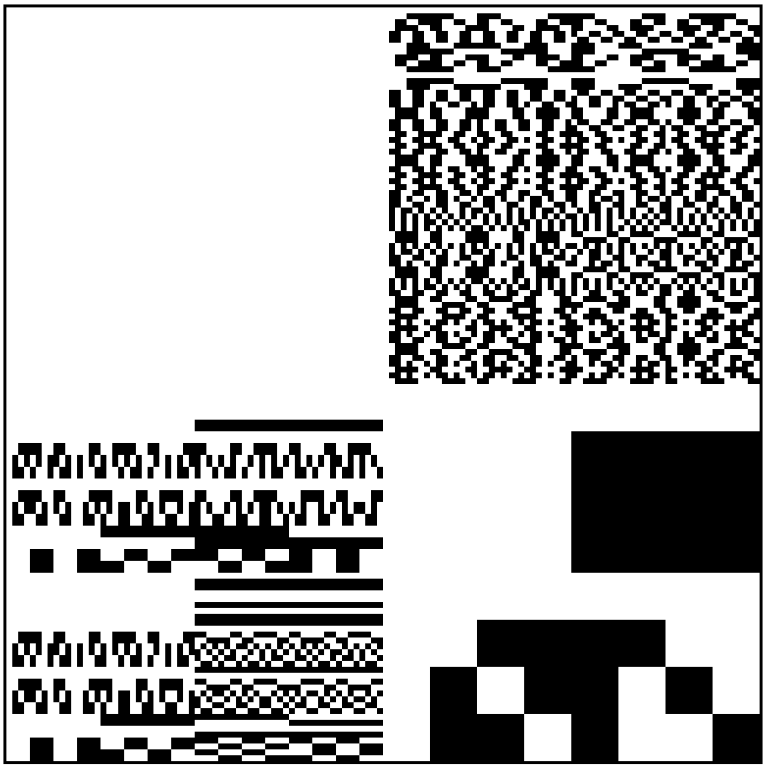

Figure 3. The restriction of the Parker loop code cocycle θ to $(V \cup W)^2$. A machine-readable version is available in [**12**]. The order of the row/column labels is $\underline{0}, b_1, b_2, b_1 + b_2, b_3, \ldots, b_1 + \cdots + b_6$, $b_7, b_8, b_7 + b_8, \ldots, b_7 + \cdots + b_{12}$.

From Figure 3 we can immediately see that the restriction of the Parker loop $\mathcal{P}\big|_V$ reduces to a direct product, as $\theta\big|_{V\times V}$ is identically zero. Moreover, the restriction of $\mathcal{P}\big|_W$ is the direct product $(\mathbb{F}_2)^3 \times M$, where M is the Moufang loop from Example 1. This is because what was a single pixel in Figure 2 is now an 8×8 block of pixels in Figure 3.

5. DISCUSSION AND COMPARISON The Parker loop $\mathcal{P}$ is a sporadic object that has spawned a small industry trying to understand and construct code loops in general, as a class of Moufang loops that are fairly easy to describe.

Aside from the description by Griess using code cocycles, there was an unpublished description by Parker, an unpublished thesis [**8**], characterisations as loops with specified commutators and associators [**2**], an iterative construction using centrally twisted products [**7**], and a construction using groups with triality [**13**]. The LOOPS library [**9**] for software package GAP [**5**] contains all the code loops of order 64 and below, although "the package is intended primarily for quasigroups and loops of small order, say up to 1000". Even the more recent [**14**], which classifies code loops up to order 512 in order to add them to the LOOPS package, falls short of the Parker loop's 8192 elements; the authors say "our work suggests that it will be difficult to extend the classification of code loops beyond order 512". In principle, there is nothing stopping the construction of $\mathcal{P}$ in LOOPS, but it will essentially be stored as a multiplication table, which would comprise $67,108,864$ entries, each of which is a 13-bit element label. The paper [**10**] describes an algebraic formula for a code cocycle that will build the Parker loop, as a combination of the recipe in the proof of Proposition 6.6 and the generating function in Proposition 9.2 of *loc. cit*. This formula is a polynomial with 330 cubic terms and 12 quadratic terms in 24 variables, being coefficients of basis vectors of $\mathcal{G}$. To compare, combined with the small amount of data in Figure 3 together with a labelling of rows/columns by words of the Golay code, Lemma 2 only requires 9 terms, of which four are cubic and the rest come from the $16,129$ stored values of θ. Large numbers of computations in $\mathcal{P}$ require optimising the multiplications, and we have found a space/time trade-off that must surely approach the more effective end of the spectrum.

In addition to computational savings, the ability to visually explore the structure of code loops during experimentation more generally seems a novel advance—the recognition of $(\mathbb{F}_2)^3 \times M$ inside $\mathcal{P}$ was purely by inspection of the picture of the code cocyle then consulting the small list of Moufang loops of small order [**1**]. Discovery of the basis in Example 3 was by walking through the spaces of bases of subcodes and working with the heuristic that more regularity in the appearance of the code cocycle is better. Additionally, our code flagged when subloops thus considered were in fact associative, and hence a group, leading to the discovery of the relatively large elementary subgroups $(\mathbb{F}_2)^7 < \mathcal{P}$ and $(\mathbb{F}_2)^6 < (\mathbb{F}_2)^3 \times M < \mathcal{P}$.

Finally, one can also remark that because of the identity (2), one can reconstruct the top right quadrant of Figure 3 from the bottom left quadrant, and vice versa. Thus one can describe the Parker loop as being generated by the subloops $(\mathbb{F})^7$ and $(\mathbb{F}_2)^3 \times M$, with relations coming from the information contained in the bottom left quadrant of Figure 3, and the formulas (2) and (4). The regularity in that bottom left quadrant is intriguing, and perhaps indicative of further simplifications; this will be left to future work.

ACKNOWLEDGMENT. DMR is supported by the Australian Research Council's Discovery Projects scheme, (project number DP180100383), funded by the Australian Government.

REFERENCES

1. Orin Chein, Moufang Loops of Small Order. I, *Trans. Am. Math. Soc.* **188** (1974) 31–51.
2. Orin Chein and Edgar G. Goodaire, Moufang loops with a unique nonidentity commutator (associator, square), *J. Algebra* **130** (1990) 369–384.
3. John H. Conway, A simple construction for the Fischer–Griess monster group, *Invent. math.* **79** (1985) 513–540.

4. John H. Conway and Neil Sloane, Sphere packings, lattices and groups. Third Edition. Grundlehren der Mathematischen Wissenschaften **290** Springer-Verlag, New York, 1999.
5. The GAP Group, *GAP – Groups, Algorithms, and Programming, Version 4.10.0* (2018) `https://www.gap-system.org`.
6. Robert L. Griess Jr., Code loops, *J. Algebra* **100** (1986) 224–234.
7. Tim Hsu, Explicit constructions of code loops as centrally twisted products, *Math. Proc. Cam. Phil. Soc.* **128** (2000) 223–232.
8. Peter Johnson, Gamma spaces and loops of nilpotence class two, PhD thesis, University of Illinois at Chicago, 1988.
9. Gábor P. Nagy and Petr Vojtěchovský, LOOPS: Package for GAP, Version 3.4.0. `https://www.cs.du.edu/~petr/loops/`, 2017.
10. Sophie Morier-Genoud and Valentin Ovsienko, A series of algebras generalizing the octonions and Hurwitz-Radon identity, *Commun. Math. Phys.* **306** (2011) 83–118, arXiv:1003.0429
11. Ruth Moufang, Zur Struktur von Alternativkörpern *Math. Ann.* **110** (1935) 416–430. doi:10.1007/bf01448037
12. Ben Nagy and David Michael Roberts, `codeloops`, GitHub repository, (2019) `https://github.com/bnagy/codeloops`
13. Gábor P. Nagy, Direct construction of code loops, *Discrete Mathematics* **308** (2008) 5349–5357.
14. Eamonn A. O'Brien and and Petr Vojtěchovský, Code loops in dimension at most 8, *J. Algebra* **473** (2017) 607–626, arXiv:1712.06524.

BEN NAGY is an MPhil. candidate at the University of Adelaide, researching computational stylistics for Latin Poetry.
Department of Classics, Archaeology & Ancient History, The University of Adelaide
benjamin.nagy@adelaide.edu.au

DAVID MICHAEL ROBERTS is a mathematician specialising in category theory, geometry and a hodge-podge of other random topics.
School of Mathematical Sciences, The University of Adelaide
david.roberts@adelaide.edu.au

Data: Basis $B = \{b_0, b_1, \ldots, b_{k-1}\}$ for the code $\mathcal{C}$
Result: Code cocycle $\theta\colon \mathcal{C} \times \mathcal{C} \to \mathbb{F}_2$, encoded as a square array of elements from $\mathbb{F}_2$, with rows and columns indexed by $\mathcal{C}$

`// Initialise`
forall $c_1, c_2 \in C$ **do**
  $\theta(c_1, c_2) \leftarrow 0$
end
$\theta(b_0, b_0) \leftarrow \frac{1}{4}\,|b_0|$

forall $1 \le i \le \text{length}(B)$ **do**
  Define $V_i := \text{span}\{b_0, \ldots, b_{i-1}\}$
  `// (D1) define theta on {bi} x Vi then deduce on Vi x {bi}`
  forall $v \in V_i$ **do**
    if $v \neq 0$ **then**
      $\theta(b_i, v) \leftarrow \text{random}$ `// In practice, random = 0`
      $\theta(v, b_i) \leftarrow \frac{1}{2}\,|v \,\&\, b_i| + \theta(b_i, v)$
    else
      `//` $\theta(b_i, v)$ `is already set to 0`
      $\theta(v, b_i) \leftarrow \frac{1}{2}\,|v \,\&\, b_i|$
    end
  end
  `// (D2) deduce theta on {bi} x Wi and Wi x {bi}`
  forall $v \in V_i$ **do**
    $\theta(b_i, b_i + v) \leftarrow \frac{1}{4}\,|b_i| + \theta(b_i, v)$
    $\theta(b_i + v, b_i) \leftarrow \frac{1}{2}\,|b_i \,\&\, (b_i + v)| + \frac{1}{4}\,|b_i| + \theta(b_i, v)$
  end
  `// (D3) deduce theta on Wi x Wi`
  forall $v_1 \in V_i$ **do**
    forall $v_2 \in V_i$ **do**
      $w \leftarrow b_i + v_2$
      $a \leftarrow \theta(v_1, b_i)$
      $b \leftarrow \theta(v_1, b_i + w)$
      $c \leftarrow \theta(w, b_i)$
      $r \leftarrow \frac{1}{2}\,|v_1 \,\&\, w| + a + b + c$
      $\theta(w, b_i + v_1) \leftarrow r$
    end
  end
  `// (D4) deduce theta on Wi x Vi and Vi x Wi`
  forall $v_1 \in V_i$ **do**
    forall $v_2 \in V_i$ **do**
      $w \leftarrow b_i + v_2$
      $a \leftarrow \theta(w, v_1 + w)$
      $\theta(w, v_1) \leftarrow \frac{1}{4}\,|w| + a$
      $\theta(v_1, w) \leftarrow \frac{1}{2}\,|v_1 \,\&\, w| + \frac{1}{4}\,|w| + a$
    end
  end
end

Algorithm 1: Reverse engineered from proof of [**6**, Theorem 10].