Additive Variable Length Codes for the Integers

Peter Fenwick

Abstract— **This paper introduces a new family of variable length codes for the integers, initially based on the Goldbach conjecture that every even integer is the sum of two primes. For an even integer we decompose the value into its two constituent primes and encode the ordinal numbers of those primes with an Elias gamma code. The method is then elaborated to handle odd integers. The paper then develops a more general method of encoding any integer as the sum of two integers and developing suitable basis sets of integers. Although the codes which are generated by these methods are characterised by widely-varying and unpredictable lengths, they are over some ranges shorter than most other variable length codes.**

I. Introduction

MANY types of information coding and compression involve a transformation which produces a sequence of integers with a highly skewed frequency distribution. For good compression or a compact representation it is then necessary to represent these integers in a "variable-length" form such that each codeword is self-delimiting and small values are represented much more compactly than larger values. Many such codes have been developed, as summarised by Fenwick[1] (though this account has major errors in describing the Golomb code). The present paper will describe a variable length code of novel structure, based initially on the Goldbach conjecture of number theory.

II. Examples of variable length codes

We start by describing some of the better known variable-length codes, the "gamma" and "omega" codes of Elias[2] and then a lesser-known one by Fraenkel and Klein[3].

• The Elias gamma code has several variants, with differing permutations of the component bits of the codeword. In the version used here, to encode an integer N, first write the natural binary representation of N, from its most significant 1 bit (this is the Elias "beta" code). Then precede this by as many zero bits as there are bits following that initial 1. (This string of 0s, with its following 1, is the Elias "alpha" code.) To decode, count the initial zero bits, write down the 1 which terminates the zero sequence and then copy as many following bits as there were leading zeros. Some representative codewords are shown in Table I. A range which includes zero must be offset by 1 because the smallest value representable by a gamma code is 1.

• The Elias omega code also starts with the Elias beta codeword. But now precede that codeword by its "length" (here the number of bits after the initial 1), also as a beta codeword, then the "length of the length", and so on until a "length" of 2 or 3 is encoded directly. Finally the whole codeword is followed by a terminating 0. Thus to encode the value 51, take its binary representation 110011, precede that by its length to give 101 110011, and a further length 10 101 110011. As this length is 2, the codeword is now complete, except for adding a final 0 to give the omega codeword of 10 101 110011 0. Some omega codewords are also shown in Table I. Note that $(N = 1)$ is handled as a special case for the omega code. While the omega code is asymptotically shorter than the gamma code, its length has major discontinuities when a new length component is "phased in". The first such discontinuities occur between 3 and 4, between 15 and 16 ($\omega(15) = 1111110$ and $\omega(16) = 10100100000$) and between 255 and 256; in general for $N = 2^{2^k}$. Because of the sequence of "length", "log(length)", "log(log(length))" and so on, the omega code is sometimes referred to as a "logarithmic-ramp" code.

• Codes based on Fibonacci numbers were first described by Apostolico and Fraenkel[4] and later, in a simplified form, by Fraenkel and Klein[3]. It is the simplest of these latter codes, their C1 code, which is described here. The Fibonacci numbers form a sequence in which each value is the sum of its two predecessors $F_i = F_{i-1} + F_{i-2}$. The first members of the sequence, with $F_1 = F_2 = 1$ are 1, 1, 2, 3, 5, 8, 13, 21, 34,.... Zeckendorf has shown [5] that any integer can be represented as the sum of Fibonacci numbers. The binary vector whose digits indicate the presence or absence of the corresponding F_i, is known as the "Zeckendorf representation" $Z(N)$ of an integer. Omitting the F_1 term and writing with the least-significant bit to the right we have that $30 = 21+8+1$ and $Z(30) = 1010001$. Similarly $42 = 34+8$ and $Z(42) = 10010000$.

The important feature of the Zeckendorf representation is that it cannot have two consecutive ones; any such pair can, by the definition of the Fibonacci numbers, be replaced by a single more-significant 1 bit. Thus if we take the Zeckendorf representation, least-significant bit first, and follow its most-significant 1 by another 1, the illegal combination of two consecutive 1 bits acts as a terminating flag. This is the C1 code of Fraenkel and Klein; they present other codes

Department of Computer Science, The University of Auckland, Private Bag 92019, Auckland, New Zealand. Email : p.fenwick@auckland.ac.nz

TABLE I
Elias gamma and omega, and Fraenkel & Klein C1 codes

N	Elias codes		Fraenkel &
	gamma	omega	Klein C1
1	1	0	11
2	010	10 0	011
3	011	11 0	0011
4	00100	10 100 0	1011
5	00101	10 101 0	00011
13	0001101	11 1101 0	0000011
20	000010100	10 100 10100 0	0101011

which differ in detail but not in principles or general performance. Using the earlier examples $C1(30) = 10001011$ and $C1(42) = 000010011$. The C1 code is simple and efficient, with $C1(N)$ having a length of about $1 + 1.44 \log_2 N$ bits.

III. Components of Variable Length Codes

The examples of the previous section illustrate how variable-length codes combine two essential components.

Value specification Many numeric representations combine a visible *digit vector* **d** and an implicit *weight vector* **w**, so that the represented value N is given by $N = \mathbf{d}.\mathbf{w}$, the scalar product of **d** and **w**. In the most familiar representations, successive digits are related by $w_{i+1} = b.w_i$, giving a polynomial in the base b. Obvious examples are $b = 10$ (decimal) and $b = 2$ (binary).

A less obvious example is that as successive Fibonacci numbers tend to the ratio

$$\lim_{n\to\infty} \frac{F_{n+1}}{F_n} = \frac{\sqrt{5}+1}{2} = 1.618$$

the Zeckendorf representation approximates a polynomial with a base $b = 1.618$.

Again, the Elias gamma code can be rearranged to generate bit-pairs for each underlying binary digit, as $\{0 \rightarrow 00, 1 \rightarrow 01, \text{end} \rightarrow 1x\}$. This resembles a polynomial with $b = \sqrt{2}$ (but not all bit patterns are legal; it terminates on the first occurrence of an odd power of the base).

Length specification Some codes, such as the omega, have a specific digit count but this may require a recursive definition of the length, exactly as in the omega code. (And embedded within the omega code is an alpha code giving the "recursion depth" for the length specification.) Other codes, such as the Fibonacci codes, are terminated by an illegal bit pattern or reserved terminating "comma".

The simplest length specification is the Elias "alpha" code which may be either $0\ldots01$ (stop on the first 1) or $1\ldots10$ (stop on the first 0). (The second form constitutes a polynomial with $b = 1$.) As originally described the gamma code is a combination of an alpha code (for the length) and a beta code (for the value).

IV. Codes based on the Goldbach conjecture

The codes to be described may be regarded as extensions of an alpha code, but with the termination rule "Stop at the *second* 1". In the simpler forms at least they are "constant weight" codes, with exactly two 1-bits in each codeword. The constant weight is combined with the Goldbach conjecture of number theory, that any even integer is the sum of two primes. The first few prime numbers are shown in Table II, but with $P_1 = 1$ rather than the more usual $P_1 = 2$.

A. A simple prime number code (G0)

We introduce the new Goldbach codes by considering a very simple example, the G0 code. As its natural form handles only even integers we encode twice the integer, with an offset. The weight vector is the prime numbers starting from 3, giving the code shown Table III.

TABLE II
The first few prime numbers

Value	1	3	5	7	11	13	17	19
Index	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8
Value	23	29	31	37	41	43	47	53
Index	P_9	P_{10}	P_{11}	P_{12}	P_{13}	P_{14}	P_{15}	P_{16}

TABLE III
Simple prime number code (G0)

value N	encode $2(N+3)$	Sum of primes	codeword
1	8	3+5	11
2	10	3+7	101
3	12	5+7	011
4	14	3+11	1001
5	16	5+11	0101
6	18	5+13	01001
7	20	7+13	00101
8	22	5+17	010001
9	24	11+13	00011
10	26	7+19	0010001
11	28	11+17	000011
12	30	11+19	0001001
13	32	13+19	0000101
14	34	11+23	00010001

weights $\mathbf{w}$ = 3, 5, 7, 11, 13, 17, 19, ...

For values much beyond those shown, the G0 codeword length "runs away" because the primes are relatively dense. But for many of the small values shown the G0 code is one of the best known. Despite its limited range of useful values the range does include most codeword lengths (for example use a beta code with a preceding G0 code for its length).

B. More complex encodings

The G0 code is ultimately limited by its long runs of zeros. For the codes of this section we encode these run-lengths using for example a gamma code. Each codeword now consists of two shorter variable-length components, encoding the ordinal position or index from Table II.

We describe the new "G1" codes by illustrating their construction.

- By the Goldbach conjecture, any even integer N is the sum of two primes, say $P_i + P_j$, where $i \leq j$
- Represent N by the couple $(i, j - i + 1)$.
- Encode the two components i and $j - i + 1$, each with a gamma code. (The second component has an offset of 1 to handle the case $i = j$).

The gamma code is preferred to the Fraenkel & Klein C1 code for the internal codes because it is shorter for very small values, especially 1 and 3. Also note that the run lengths easily exceed the efficient range of G0 codes.

Several values are shown encoded in Table IV, which includes the gamma code length for comparison. The two

TABLE IV

Goldbach G1 encoding of selected even integers

	Components		Goldbach G1 code			Gamma
N	values	indices	pair	encoding	len.	length
2	1 + 1	1, 1	1, 1	1.1	2	3
4	1 + 3	1, 2	1, 2	1.010	4	5
6	3 + 3	2, 2	2, 1	010.1	4	5
8	3 + 5	2, 3	2, 2	010.010	6	7
10	3 + 7	2, 4	2, 3	010.011	6	7
12	5 + 7	3, 4	3, 2	011.010	6	7
14	7 + 7	4, 4	4, 0	00100.1	6	7
16	5 + 11	3, 5	3, 3	011.011	6	9
18	7 + 11	4, 5	4, 2	00100.010	8	9
20	7 + 13	4, 6	4, 3	00100.011	8	9
30	13 + 17	6, 7	6, 2	00110.010	8	9
40	17 + 23	7, 9	7, 3	00111.011	8	11
50	13 + 37	6, 12	6, 7	00110.00111	10	11
60	29 + 31	10, 11	10, 2	0001010.010	10	11
70	29 + 41	10, 13	10, 4	0001010.00100	12	13
80	37 + 43	12, 14	12, 3	0001100.011	10	13
90	43 + 47	14, 15	14, 2	0001110.010	10	13
100	47 + 53	15, 16	15, 2	0001111.010	10	13
40	3 + 37	2, 12	2, 11	010.0001011	10	11
40	11 + 29	5, 10	5, 6	00101.00110	10	11

TABLE V

Goldbach G2 codes, encoding rules

components			
1st	2nd	explanation	comment
110		$N = 1$	special code
111		$N = 2$	special code
0...	0...	even integer	prime + prime
0...	1	prime integer	prime + 0
1 0...	0...	odd integer	1 + prime + prime

components are shown separated by a period ".", which is *not* part of the codeword. The most obvious feature is that the codeword length varies erratically; it is not the stepwise increasing function that we see in the gamma and C1 codes. But despite this variation, most values have shorter codewords than the corresponding gamma codewords.

The last two lines of Table IV show two alternative codings for the value 40. Here using the pair (17, 23) from the main table gives a codeword of 8 bits, while using the more widely-separated values (3, 37) or (11, 29) gives a 10-bit codeword. These illustrate the guideline that the initial component primes should be as close together as possible to give one large encoded value and a small displacement to the next.

But the situation is really more complex than this and a complete search is needed to find the best combination. For example, 50 can be coded as (19+31), with indices [8, 11] and the coded pair {8,4}, for 12 bit cost. 50 can also be encoded as (13+37), indices[6,12] and pair{6,7}, for a cost of 10 bits. The difference is that decreasing the first component from 19 to 13 changes its index from 8 to 6 and the coding cost from 7 to 5 bits. Increasing the second value (difference) from 4 to 7 leaves its cost at 5 bits. The key point is not the values *per se*, but the cost of coding their indices.

V. Developing the general Goldbach Codes

The G1 code is limited to even values. We now extend the coding to include odd integers as well, using two different methods.

• The first method doubles the encoded value N as for the G0 code but then encodes the run lengths as a gamma code. This is an obvious method, but it involves larger values in the run-lengths because each index is increased by about 60% and often requires a longer gamma code representation.

• The second method (the G2 code) handles values in several different ways. While there are three basic situations to consider, we also provide special codes for the smallest values. (Special handling of small values is quite usual in variable-length codes; we saw it earlier in the omega code for 0.) The different cases are summarised in Table V

Special cases. Encode the value 1 as 110 and 2 as 111. (None of the other codes can start with 11 ....)

Even integer. Encode as above, as the sum of two primes. In the earlier codes the difference of the two indices was incremented before coding. Here we increment the first index as well before coding, so that an index of 1 is represented by 010 and allows an initial 1 to signal a non-numeric interpretation.

Prime integer. Emit the index of the number itself (again incremented by 1), followed by a single 1. (The final 1 implies "no component". Alternatively, we can define P_0 as 0; the 1 decodes as a 1 from a gamma code, which decrements down to 0 as the correct index of the "prime number" P_0.)

Odd integer (non-prime). Emit an initial 1 bit, decrement the value by 1 and then encode as an even number as for case 1 above.

Several of the resulting codes are shown in Table VI, including the codeword lengths and the corresponding gamma and omega lengths for comparison. For most values the new codes are no longer than the older ones, and are usually shorter.

VI. Prediction of the Codeword Length

For this analysis it is simplest to consider the G1 code, encoding the even integer $2N$, with two additive components both of $O(N)$. While the prediction is not intended to be precise (this is very difficult because of the unpredictable codeword length), it gives a reasonable picture of the overall codeword behaviour.

The prime number theorem states that $\pi(N)$, the number of primes less than N, is proportional to $\frac{N}{\log N}$. For values up to 1,000, a good approximation is $\pi(N) = 1.15\frac{N}{\log N}$. The gamma codeword of an integer k requires approximately $2\lfloor \log_2 k \rfloor + 1$ bits. The Goldbach codeword represents both of its components as a gamma codeword; if both are approximately $\pi(N)$, each encoded length will be about

$$2\left\lfloor \log_2 \frac{1.15N}{\log N} \right\rfloor + 1 \qquad \text{bits}$$

But the second index is encoded as a difference and if we assume the difference to be about half the magnitude of

TABLE VI
GOLDBACH G2 CODES, WITH GAMMA AND OMEGA LENGTHS

N	Goldbach G2 Code		Gamma	Omega
codeword	Length	Length	Length	
1	110	3	1	1
2	111	3	3	3
3	0101	4	3	3
4	010010	6	5	6
5	0111	4	5	6
6	011010	6	5	6
7	001001	6	5	6
8	011011	6	7	7
9	1011011	7	7	7
10	01100100	8	7	7
11	001011	6	7	7
12	00101010	8	7	7
13	001101	6	7	7
14	00110010	8	7	7
15	100110010	9	7	7
16	0010100100	10	9	11
17	001111	6	9	11
18	00111010	8	9	11
19	00010001	8	9	11
20	00111011	8	9	11
30	0001010010	10	9	11
40	0001100011	10	11	12
50	0001111011	10	11	12
60	000010001010	12	11	12
70	000010011011	12	13	13
80	000010110010	12	13	13
90	000011000010	12	13	13
100	000011001011	12	13	13

the first index, its gamma codeword is 2 bits shorter than the first. The predicted Goldbach codeword length is then (in bits)

$$(2L+1)+(2L-1)=4L, \text{ where } L=\left\lfloor \log_2 \frac{1.15N}{\log N}\right\rfloor$$

The Goldbach length may be compared with the gamma codeword length of $2\lfloor \log_2(2N)\rfloor + 1$ bits and the C1 length of $1.44\log_2(2N)+1$ bits.

The predictions are shown in Figure 1, compared with predicted lengths for the Elias Gamma and Fraenkel & Klein C1 codes. The figure shows that the Goldbach code should be competitive with the other codes over a range of values, being better than the gamma codes for values to 100 and the C1 code for values less than about 40.

VII. Additive codes, beyond the Goldbach codes

While the codes based on the Goldbach conjecture have been shown to work, there should be other codes of a similar nature. In particular, with the Goldbach codes based on a rather special set of numbers we can ask the question "Are there other sets of numbers which, added in pairs, can form all integers within a range?". Given a suitable set of "basis integers" we can then encode values using the very first technique described for primes (the one which handled only even values). The remainder of the paper deals with forming such number sets and codes based upon them, leading to a new class of "Additive Codes".

We restrict ourselves to number *pairs* largely because allowing an arbitrary number of components requires that

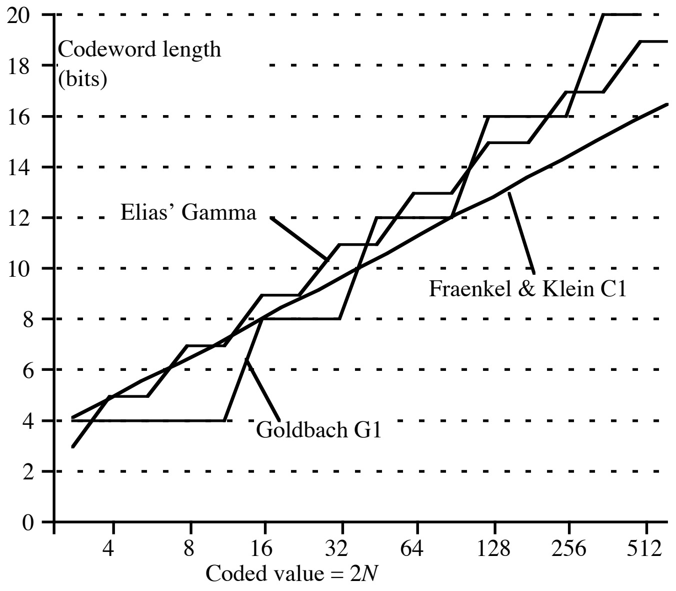

Fig. 1. Theoretical codeword lengths

the number representation be extended to specify the number of components. (There is also the intellectual challenge of what can be done with only two components.) To illustrate, this work grew out of attempts to improve the Fibonacci codes by run-length encoding the zeros which predominate in those codes. The attempts were largely unsuccessful, partly because all variable-length codes *expand* the very short runs, and partly because of the need to encode a count of the runs or of the ones. The realisation that a constant two components sufficed was a considerable simplification.

The numbers are generated from first principles, using a sieve technique, loosely modelled on the Sieve of Eratosthenes. At any stage of the algorithm we have a set of basis integers $\{0, B_1, B_2, B_3, \ldots, B_k\}$ and have noted all possible sums $B_i + B_j$ of pairs of those integers. As a consequence of the algorithm we know that all integers $N < B_k$ can be so generated. Having generated B_k we then insert into the sieve all sums $B_i + B_k$ where $i \le k$. We then search through the successive integers starting from B_k for the next value not in the sieve; this becomes B_{k+1}.

To illustrate, the basis set {0, 1, 2} will add the integers {3 and 4}. The first absent integer is 5, which becomes B_3 and allows us to add {6, 7 and 10} to the sieve. The next generated value, $B_4 = 8$, adds {9, 13 and 16}, with the next gap at $B_5 = 11$.

There is clearly something special about the value 2 in the basis set, because it is generated by (1+1) and is apparently superfluous. A naive application of the algorithm starting with the basis set {0, 1} will just generate all odd integers because the pairwise sums deliver only the even integers and no odd ones. It is therefore necessary to introduce a few suitably chosen even "seed" values such as the 2 above to force some odd values and fill in gaps. The seeds may be regarded as parameters of the code and the choice of values determines the quality and nature of the resultant code.

To illustrate, all values up to 250 can be generated as sums of the pairs of the 37 values shown in Table VII (which includes 0). This code uses 3 seed values. By comparison the Goldbach code for the same range needs 53 primes and rather more complex encoding. Other tables to cover ranges to 100, 500 and 1000 are shown in an Appendix. Examples of the coding are shown in Table VIII, with a format based on Table VI but adapted for the new codes.

These codes were generated by a computer search, finding seed values to minimise the size of the basis set covering a specified range of values. Other optimisations are clearly possible, such as minimising either the average cost of the resultant code, or an appropriately weighted codeword length.

VIII. Evaluation of the Goldbach code

Figure 1 shows the codeword lengths of the Goldbach codes for arguments from 1 to 500. Under the general variation of about ± 2.5 bits about an average value is a fairly consistent behaviour with steps at about $N = 7, 20, 50, 135$ and 320, each adding about 2 bits of length. These steps correspond to the introduction of primes P_{2^k} where the index of the prime suddenly requires a longer gamma codeword. (We note that $P_4 = 7$, $P_8 = 19$, $P_{16} = 53$, $P_{32} = 131$ and $P_{64} = 311$.) The trend is approximated by the codeword length of $2 + \frac{13}{8} log_2 N$, showing that the final codeword length is about 1.625 times that of the original binary value. This may be contrasted with the gamma code whose expansion is 2.0 and the Fraenkel C1 code with an expansion of 1.44.

A better presentation is shown in Figure 3, which shows the lengths averaged over ranges $2^k \leq N < 2^{k+1}$ of coded values. This division is certainly appropriate for the Elias gamma and omega codes because they have natural discontinuities at the octave boundaries. It is less appropriate for the Fraenkel and Klein, Goldbach and additive codes but is still a reasonable way of smoothing out their variations.

We see that over most of the range the new additive code is better than the Elias gamma and omega codes, while for values from about 4 to 30 it is the best of the codes shown. These results are in reasonable agreement with the theoretical predictions, given the uncertainties surounding those predictions. Certainly the additive code (but *not* the Goldbach code itself) is a competitive code for values from about 4 to 40.

TABLE VII
Basis values to generate integers to 250

0	1	2	5	8	11	14	16
20	23	26	29	33	46	50	63
67	80	84	97	101	114	118	131
135	148	152	165	169	182	186	199
203	216	220	233	237			
36 values (53 primes) Seeds = 2, 16, 46							

TABLE VIII
Additive encoding of some integers

	Components		Additive code			Gamma
N	values	indices	pair	encoding	len.	len.
10	3 + 7	3, 5	3, 3	011:011	6	7
11	0 + 11	1, 7	1, 7	1:00111	6	7
12	1 + 11	2, 7	2, 6	010:00110	8	7
13	1 + 12	2, 8	2, 7	010:00111	8	7
14	7 + 7	5, 5	5, 1	00101:1	6	7
15	3 + 12	3, 8	3, 6	011:00110	8	7
16	7 + 9	5, 6	5, 2	00101:010	8	9
17	5 + 12	4, 8	4, 5	00100:00101	10	9
18	7 + 11	5, 7	5, 3	00101:011	8	9
20	9 + 11	6, 7	6, 2	00110:010	8	9
30	5 + 25	4, 9	4, 6	00100:00110	10	9
40	11 + 29	7, 11	7, 5	00111:00101	10	11
50	25 + 25	9, 9	9, 1	0001001:1	8	11
60	29 + 31	11, 12	11, 2	0001011:010	10	11
70	35 + 35	14, 14	14, 1	0001110:1	8	13
80	0 + 80	1, 20	1, 20	1:000010100	10	13
90	29 + 61	11, 17	11, 7	0001011:00111	14	13
100	3 + 97	3, 23	3, 21	011:000010101	14	13

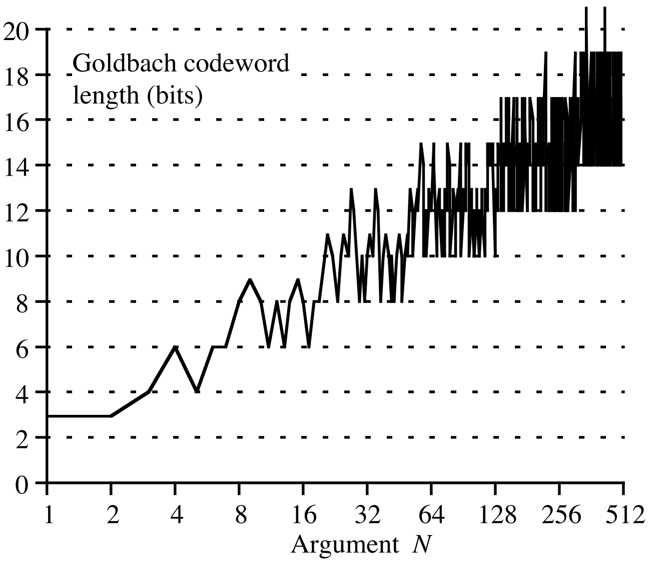

Fig. 2. Goldbach codeword lengths

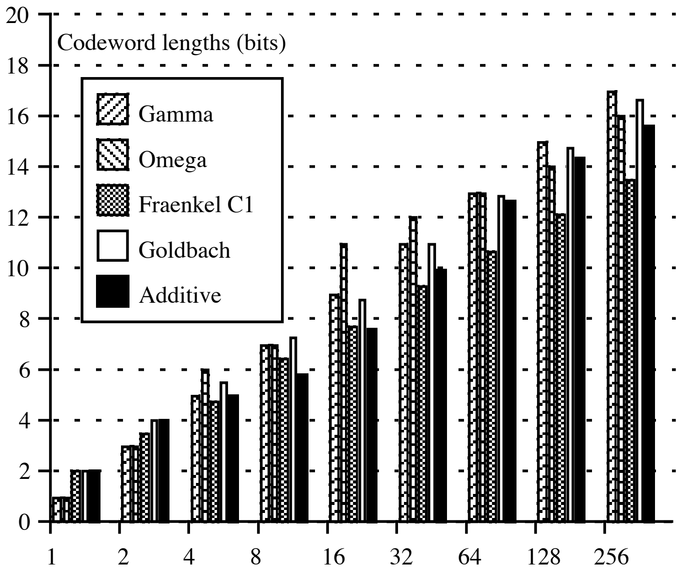

Fig. 3. Comparison of codeword lengths

TABLE IX
Average codeword lengths for limits of 500 and 128

	1	2–3	4–7	8–15	16–31	32–63	64–127
C1 code	2	3.5	4.75	6.38	7.69	9.22	10.61
Add. 500	2	4	5	5.75	7.63	9.83	12.63
Add. 128	2	4	5	6.25	7.88	9.06	11.34

Further experiments (results not shown) indicate that there is considerable scope for tuning to a particular situation by adjusting the seed values.

For example, changing the range to 128 (more in line with binary values) produced 24 basis values and seeds of 8, 20 and 24, compared with 21 values and seeds of 8, 10 and 16 for a range to 100. The codeword lengths over the octaves to 127 are shown in Table IX, comparing the two additive codes with the lengths of the Fraenkel C1 code over the same range. It is clear that the additive code can be tuned by adjusting the seeds and the detailed rules for generating the basis numbers.

IX. Conclusions

This paper has introduced a new family of variable length codes, based on representing a value as the sum of two components of a set of integers and encoding the indices of the components in that set. The codes have been shown as competitive with established variable-length codes. For values up to 500 they are generally better than the Elias gamma and omega codes and competitive with the Fraenkel and Klein's C1 Fibonacci code. The "basis tables" for the new additive codes contain some special "seed" values which can be used as parameters when tuning the code to a particular application.

X. Acknowledgements

The author thanks Dr Mark Titchener, who originally offered the vague suggestion that the Goldbach conjecture might be useful for constructing variable length codes, but with neither of us having any idea as to how it might be done.

The author also thanks the referees for many perceptive and useful comments.

References

[1] P.M. Fenwick, "Punctured Elias Codes for variable-length coding of the integers", *Technical Report 137*, Department of Computer Science, The University of Auckland, 1996. Available : www.cs.auckland.ac.nz/~peter-f/ftplink/TechRep137.ps

[2] P. Elias, "Universal Codeword Sets and Representations of the Integers", *IEEE Trans. Inform. Theory*, Vol IT 21, No 2, pp 194–203, Mar 1975

[3] A.S. Fraenkel and S.T. Klein, "Robust universal complete codes for transmission and compression", *Discrete Applied Mathematics* Vol 64 (1996) pp 31–55.

[4] A. Apostolico, A.S. Fraenkel, "Robust transmission of unbounded strings using Fibonacci representations", *IEEE Trans. Inform. Theory*, Vol IT–33 (1987), pp 238–245.

[5] E. Zeckendorf, "Représentation des nombres naturels par une somme de nombres de Fibonacci ou de nombres de Lucas", *Bull. Soc. Roy. Sci. Liège*, Vol 41 (1972), pp 179–182

TABLE X
Basis integers for numbers to 100, 500 and 1000

0	1	3	5	7	8	10	16
22	28	34	40	46	52	58	64
70	76	82	88	94			
Maximum Value = 100; Seeds = 8, 10, 16							
21 values (25 primes)							

0	1	2	5	8	11	14	17
20	23	26	29	32	34	38	41
44	47	50	53	56	59	62	65
69	82	104	117	139	152	174	187
209	222	244	257	279	292	314	327
349	362	384	397	419	432	454	467
489							
Maximum Value = 500; Seeds = 2, 34, 82							
49 values (95 primes)							

0	1	2	5	8	11	14	17
20	23	26	29	32	35	38	41
44	47	50	52	56	59	62	65
68	71	74	77	80	83	86	89
92	95	98	101	105	154	158	207
211	260	264	313	317	366	370	419
423	472	476	525	529	578	582	631
635	684	688	737	741	790	794	843
847	896	900	949	953			
Maximum Value = 1000; Seeds = 2, 52, 154							
69 values (168 primes)							

XI. Appendix

Table X shows tables of basis integers for generating numbers to 100, 500 and 1000. We emphasise that they are not unique. Note that larger ranges are covered in fewer numbers than there are primes in the same range.