Percy Ludgate’s Logarithmic Indexes

Brian Coghlan

A. *Ludgate’s logarithmic indexes*

Percy Ludgate invented his own logarithmic indexes for multiplication [1][2][3], an entirely new result in 1909. These are now called Irish Logarithms, a discrete analogy to real logarithms where log(j*k) = Iog(j) + log(k). For two operands Z_J and Z_K the index numbers ensure $Z_Y = Z_{J*K} = Z_J + Z_K$, for example $Z_3 = 7$, $Z_5 = 23$, $Z_{15} = 30$. The largest Z_{J*K} is $Z_{7*7} = 66$. C.V.Boys said: “Ludgate … uses for each of the prime numbers below ten in a logarithmic system with a different incommensurable base, which as a fact never appears” [2], i.e. each $Z_X = \log_{N_X}(X)$ has a different (invisible) base N_X. Ludgate almost certainly did not derive his indexes from number theory, either finding them empirically or by a systematic “method”, such as that in Figure 1.

Ludgate’s proposed logarithmic indexes are amenable to construction by a systematic method.

For an example of how these are used in calculations, see Andries de Man’s educational emulator [5], and his citations of similar indexes in the literature [6] cited below:

(1) Jacobi/Zech indexes (1846/1849 [7]), which can be derived from number theory. For example $Z_1 = 0$, $Z_2 = 1$, $Z_3 = 18$, $Z_5 = 44$, $Z_7 = 7$. An alternative is $Z_1 = 0$, $Z_2 = 1$, $Z_3 = 8$, $Z_5 = 44$, $Z_7 = 27$. In both cases the largest Z_{J*K} is $Z_{5*5} = 88$.
(2) Remak indexes (Von K. Hoecken 1913 [8]), $Z_1 = 0$, $Z_2 = 1$, $Z_3 = 13$, $Z_5 = 21$, $Z_7 = 30$, where the largest Z_{J*K} is $Z_{7*7} = 60$.
(3) Korn indexes (Von K. Hoecken 1913 [8]), $Z_1 = 0$, $Z_2 = 8$, $Z_3 = 13$, $Z_5 = 1$, $Z_7 = 30$, where the largest Z_{J*K} is $Z_{7*7} = 60$.

Ludgate said: “The index numbers (which I believe to be the smallest whole numbers that will give the required results)”. His focus on small numbers makes it possible Ludgate knew of Jacobi/Zech indexes, which were in use in astronomy. Remak and Korn indexes give smaller numbers, but were reported after Ludgate had published. Small numbers are important to minimize the length of any mechanism that is used to convert to or from the logarithmic indexes. McQuillan [11][12] has shown that better indexes can be readily be computed using modern machinery.

Ludgate proposed logarithmic indexes with the smallest Z_{J*K} then known.

As happens surprisingly often with novel inventions, in the same year as Ludgate’s 1909 paper a sliderule with Jacobi indexes was designed by Prof.Schumacher of Germany [9] and later manufactured as the Faber Model 366 [10].

Ludgate’s indexes can be derived by an algorithm. He likely did this, but thought of it as a “method”. The method can be expressed as a simple algorithm, see below, which has proven Irish Logarithms exist over the range 0-99, perhaps further.

Ludgate’s proposed logarithmic indexes are amenable to algorithmic construction.
Ludgate’s logarithmic indexes have been shown to exist for 0-99.

B. *Algorithm to derive Irish Logarithms*

Ludgate’s indexes can be derived by algorithm. He likely did this, but thought of it as a “method”. The method of Figure 1 can be expressed as a simple algorithm, for example in Python as in Figure 2. Executing this algorithm yields identical indexes to those given in Tables 1, 2 and 3 of Ludgate’s 1909 paper, see Figure 3.

Irish Logarithms might exist for the range up to any *N*, not just for 0-9 as in Ludgate’s paper. Certainly they have been shown to exist for 0-99 by extending the rather inefficient algorithm of Figure 2 with a basic Sieve of Eratosthenes to generate the seed prime numbers. Beyond that a more efficient algorithm would be necessary, or ideally a number-theoretic proof.

- All these products derive ultimately from primes, so start with first prime J=1 by assigning $Z_1=0$
  Index [0] is now used, so for next prime p=2 the indexes Z_2+[0] must be free
  All indexes above 0 are free, so assign $Z_2=1$
- Then recursively for all products ≤ 9*9=81 for which an index exists, assign a logarithmic index $Z_{J*K} = Z_J + Z_K$
  $Z_4=Z_{2*2}=Z_2+Z_2=1+1=2$ $\quad Z_8=Z_{2*4}=Z_2+Z_4=1+2=3$ $\quad Z_{16}=Z_{4*4}=Z_4+Z_4=2+2=4$
  $Z_{32}=Z_{4*8}=Z_4+Z_8=2+3=5$ $\quad Z_{64}=Z_{8*8}=Z_8+Z_8=3+3=6$
- Indexes [1,2,3] are now used for 1<Y<9
  So for next prime p=3 the indexes Z_3+[0,1,2,3] must be free
  All indexes above 6 are free, so assign $Z_3=7$
- Then recursively for all products ≤ 9*9=81 for which an index exists, assign a logarithmic index $Z_{J*K} = Z_J + Z_K$
  $Z_6=Z_{3*2}=Z_3+Z_2=7+1=8$ $\quad Z_9=Z_{3*3}=Z_3+Z_3=7+7=14$ $\quad Z_{12}=Z_{3*4}=Z_3+Z_4=7+2=9$
  $Z_{18}=Z_{3*6}=Z_3+Z_6=7+8=15$ $\quad Z_{24}=Z_{3*8}=Z_3+Z_8=7+3=10$ $\quad Z_{27}=Z_{3*9}=Z_3+Z_9=7+14=21$
  $Z_{36}=Z_{6*6}=Z_6+Z_6=8+8=16$ $\quad Z_{48}=Z_{6*8}=Z_6+Z_8=8+3=11$ $\quad Z_{54}=Z_{6*9}=Z_6+Z_9=8+14=22$
  $Z_{72}=Z_{8*9}=Z_8+Z_9=3+14=17$ $\quad Z_{81}=Z_{9*9}=Z_9+Z_9=14+14=28$
- Indexes [1,2,3,7,8,14] are now used for 1<Y<9
  So for next prime p=5 the indexes Z_5+[0,1,2,3,7,8,14] must be free
  The next free index for which this is so is Z_5=23+[1,2,3,7,8,14], i.e. 23,24,25,26,30,31,37, the indexes are all free, so assign $Z_5=23$
- Then recursively for all products ≤ 9*9=81 for which an index exists, assign a logarithmic index $Z_{J*K} = Z_J + Z_K$
  $Z_{10}=Z_{5*2}=Z_5+Z_2=23+1=24$ $\quad Z_{15}=Z_{5*3}=Z_5+Z_3=23+7=30$ $\quad Z_{20}=Z_{5*4}=Z_5+Z_4=23+2=25$
  $Z_{25}=Z_{5*5}=Z_5+Z_5=23+23=46$ $\quad Z_{30}=Z_{5*6}=Z_5+Z_6=23+8=31$ $\quad Z_{40}=Z_{5*8}=Z_5+Z_8=23+3=26$
  $Z_{45}=Z_{5*9}=Z_5+Z_9=23+14=37$
- Indexes [1,2,3,7,8,14,23] are now used for 1<Y<9
  So for next prime p=7 the indexes Z_7+[0,1,2,3,7,8,14,23] must be free
  The next free index for which this is so is Z_7=33+[1,2,3,7,8,14,23], i.e. 33,34,35,36,40,41,47,56, the indexes are all free, so assign $Z_7=33$
- Then recursively for all products ≤ 9*9=81 for which an index exists, assign a logarithmic index $Z_{J*K} = Z_J + Z_K$
  $Z_{14}=Z_{7*2}=Z_7+Z_2=33+1=34$ $\quad Z_{21}=Z_{7*3}=Z_7+Z_3=33+7=40$ $\quad Z_{28}=Z_{7*4}=Z_7+Z_4=33+2=35$
  $Z_{35}=Z_{7*5}=Z_7+Z_5=33+23=56$ $\quad Z_{42}=Z_{7*6}=Z_7+Z_6=33+8=41$ $\quad Z_{49}=Z_{7*7}=Z_7+Z_7=33+33=66$
  $Z_{56}=Z_{7*8}=Z_7+Z_8=33+3=36$ $\quad Z_{63}=Z_{7*9}=Z_7+Z_9=33+14=47$
- So for 1<Y<9, Indexes [1,2,3,7,8,14,23,33] are now used
  The only unused integer is Y=0, and although log(0) does not exist, here multiply by 0 must be valid, so Z_0+[1,2,3,7,8,14,23,33] must be free
  The next free index for which this is so is Z_0=50+[1,2,3,7,8,14,23,33], i.e. 50,51,52,53,57,58,64,73,83 the indexes are all free, so assign $Z_0=50$
- Then recursively for all products ≤ 9*9=81 for which an index exists, assign a logarithmic index $Z_{J*K} = Z_J + Z_K$
  $Z_{02}=Z_{0*2}=Z_0+Z_2=50+1=51$ $\quad Z_{03}=Z_{0*3}=Z_0+Z_3=50+7=57$ $\quad Z_{04}=Z_{0*4}=Z_0+Z_4=50+2=52$
  $Z_{05}=Z_{0*5}=Z_0+Z_5=50+23=73$ $\quad Z_{06}=Z_{0*6}=Z_0+Z_6=50+8=58$ $\quad Z_{07}=Z_{0*7}=Z_0+Z_7=50+33=83$
  $Z_{08}=Z_{0*8}=Z_0+Z_8=50+3=53$ $\quad Z_{09}=Z_{0*9}=Z_0+Z_9=50+14=64$ $\quad Z_{00}=Z_{0*0}=Z_0+Z_0=50+50=100$

Figure 1 *Systematic method to derive Ludgate's simple index numbers*
For all products $Y = (1<J<9) * (1<K<9)$ ***assign logarithmic index numbers*** $Z_Y = Z_{J*K} = Z_J + Z_K$
Image reproduced courtesy The John Gabriel Byrne Computer Science Collection [4]

```
#!/usr/bin/env python

import sys

# initialise variables
Z=[-1]*200   # table of complex indexes
PP=[-1]*200  # table of partial products
i = 0;
for p in (1,2,3,5,7,0):
    if Z[p]==-1:  # prime not indexed yet
        free=False
        while free==False and i<=100:
            free=True
            for j in (1,2,3,4,5,6,7,8,9):
                if free==True:
                    if Z[j]<>-1:                # for existing indexes
                        for k in range (1,100):
                            if Z[k]==(i+Z[j]): # check if complex index exists
                                free=False
                                i=i+1
        if free==True: # OK, found a desired free set of indexes
            Z[p]=i     # create new simple index
            PP[i]=p    # create new partial product
            i=i+1
            for j in (1,2,3,4,5,6,7,8,9,0):
                if Z[j]<>-1:  # multiplicand simple index exists
                    for k in (1,2,3,4,5,6,7,8,9,0):
                        if Z[k]<>-1:  # multiplier simple index exists
                            if PP[Z[j]+Z[k]]==-1:    # product not indexed yet
                                if Z[j*k]==-1:
                                    Z[j*k]=Z[j]+Z[k] # create new complex index
                                PP[Z[j]+Z[k]]=j*k    # create new partial product
```

Figure 2 *Python algorithm to derive Ludgate's simple index numbers*
Image reproduced courtesy The John Gabriel Byrne Computer Science Collection [4]

```
final Ludgate Simple Index for each Unit (Table 1):
  0: 50    1:  0    2:  1    3:  7    4:  2    5: 23    6:  8    7: 33    8:  3    9: 14

final Ludgate Complex Index for each Partial Product (Table 2):
  1:  0    2:  1    3:  7    4:  2    5: 23    6:  8    7: 33    8:  3    9: 14   10: 24   12:  9   14: 34
 15: 30   16:  4   18: 15   20: 25   21: 40   24: 10   25: 46   27: 21   28: 35   30: 31   32:  5   35: 56
 36: 16   40: 26   42: 41   45: 37   48: 11   49: 66   54: 22   56: 36   63: 47   64:  6   72: 17   81: 28

final Ludgate Partial Product for each Complex Index (Table 3):
  0:  1    1:  2    2:  4    3:  8    4: 16    5: 32    6: 64    7:  3    8:  6    9: 12
 10: 24   11: 48   12:      13:      14:  9   15: 18   16: 36   17: 72   18:      19:
 20:      21: 27   22: 54   23:  5   24: 10   25: 20   26: 40   27:      28: 81   29:
 30: 15   31: 30   32:      33:  7   34: 14   35: 28   36: 56   37: 45   38:      39:
 40: 21   41: 42   42:      43:      44:      45:      46: 25   47: 63   48:      49:
 50:  0   51:  0   52:  0   53:  0   54:      55:      56: 35   57:  0   58:  0   59:
 60:      61:      62:      63:      64:  0   65:      66: 49   67:      68:      69:
 70:      71:      72:      73:  0   74:      75:      76:      77:      78:      79:
 80:      81:      82:      83:  0   84:      85:      86:      87:      88:      89:
 90:      91:      92:      93:      94:      95:      96:      97:      98:      99:
100:  0
```

Figure 3 *Results from running the algorithm of Figure 2*
Image reproduced courtesy The John Gabriel Byrne Computer Science Collection [4]

REFERENCES,

[1] Percy E.Ludgate, On a Proposed Analytical Machine, Scientific Proceedings of the Royal Dublin Society, Vol.12, No.9, pp.77–91, 28-Apr-1909.
[2] C.V.Boys, A new analytical engine, Nature, Vol.81, pp.14-15, Jul-1909, see elsewhere in the Literature category of this catalog.
[3] Percy E.Ludgate, Automatic Calculating Machines, In "Handbook of the Napier Tercentenary Celebration or modern instruments and methods of calculation", Ed: E.M.Horsburgh, 1914
[4] Trinity College Dublin, The John Gabriel Byrne Computer Science Collection. Available: https://scss.tcd.ie/SCSSTreasuresCatalog/

[5] Andries de Man, Irish Logarithms Animation. Available: http://ajmdeman.awardspace.info/t/irishlog.html
[6] Andries de Man, Irish Logarithms, Available: https://sites.google.com/site/calculatinghistory/home/irish-logarithms-1 and https://sites.google.com/site/calculatinghistory/home/irish-logarithms-1/irish-logarithms-part-2-1
[7] Wikipedia, Zech's Logarithm, Available: https://en.wikipedia.org/wiki/Zech's_logarithm
[8] K. Hoecken, "Die Rechenmaschinen von Pascal bis zur Gegenwart, unter besonderer Berücksichtigung der Multiplikationsmechanismen", Sitzungsberichte Berliner Math. Gesellsch.13, pp.8–29, February 1913.
[9] Dr. Joh. Schumacher, Ein Rechenschieber mit Teilung in gleiche Intervalle auf der Grundlage der zahlentheoretischen Indizes. Für den Unterricht konstruiert, München, 1909.
[10] Dieter von Jezierski, Detlef Zerfowski, Paul Weinmann: A.W. Faber Model 366 - System Schumacher. A Very Unusual Slide Rule, Journal of the Oughtred Society, pp.10-17, Vol.13, No.2, 2004. Available: http://osgalleries.org/journal/displayarticle.cgi?match=13.2/V13.2P10.pdf also at: https://scss.tcd.ie/SCSSTreasuresCatalog/miscellany/TCD-SCSS-X.20121208.002/
[11] David McQuillan, Percy Ludgate's Analytical Machine, Available: http://fano.co.uk/ludgate/
[12] David McQuillan, The Feasibility of Ludgate's Analytical Machine, Available: http://fano.co.uk/ludgate/Ludgate.html