Example R

Study: The Divisors of Seventy-Two

MUTABLE NUMBER ANALYSIS

In the short Archive article *Music By Mutable Numbers* the argument is advanced that tonal music, stripped down to a protean structure of nested harmonic series, forms essentially an implied sequence of values, *mutable base numbers*, written in an acoustic physical notation which our ears intuitively understand as commensurable harmony. In support of this contention – that chords in tonal music are in essence positional numbers written in sound, (i.e. parts of harmonic series h1 through hn) – a demonstration 'composition' or 'tonal procedure' is presented below, which performs the overtly numerical task of finding the divisors of a given whole number. In principle any given number. Judged on its merits, in purely musical terms, the piece is trivial, consisting as it does of repetitive arpeggios and scale passages. However, this procedure does result in a 'composition' of sorts which is both recognisably tonal music and recognisably practical mathematics. (Essentially the same sequence of chords is 'repackaged' in a rather more palatable form in Ex_R2/pdf and Ex_R2/MID so as to hopefully demonstrate that this really is tonal music.) The example given below seeks out the divisors of seventy-two, though equally the procedure could be applied to any number that lies within the range of musical instruments, and in theory could be applied to any positive whole number.

The procedure is also given below in the parallel form of a computer program. This program, as in Chapter 3 of *Journey to the Heart of Music* is written in the almost readable prose of the BASIC programming language (parallel versions in AWK and Perl can be found in the SCRPT.ZIP directory). All these versions of the procedure/program require a digital electronic computer to function. That is, a physical device capable of handling positional binary numbers by means of representing the digits zero and one as the absence or presence of a defined level of electrical potential within the computer's circuitry. Going back in time to the nineteenth century, the mathematician Charles Babbage designed similar devices: the mechanical difference and analytical engines. Though never finished in their own day, these were likewise physical devices, but machines that used cog wheels and cylinders to represent the digits of positional decimal numbers – with which we are all familiar. Theoretically, there is little to distinguish between the modern computer and Babbage's engines beside the technicalities of operation and of course a huge speed differential. Interestingly, Ada Lovelace (the daughter of Lord Byron, who collaborated with Babbage on the project, and is the author of the fullest account of the analytical engine's true potential) suggested that among other things the device might: "compose elaborate and scientific pieces of music".

Similarly, in seeking to use positional mutable numbers as the basis of operation for computation, an appropriate physical device is required, that is, a physical device specifically designed to match the particular characteristics of its operational number system. For mutable numbers (–i.e. chords in tonal music) an appropriate physical device is a musical instrument, though one might imagine far more powerful oscillatory processors, with frequency ranges and sensitivities greatly in excess of that required for the pursuit of music. However, the instruments we have and use to make music are entirely adequate for the demonstration of tonal computation in sound.

First the mathematics. An underlying mechanism for finding the divisors of a given number is described in Chapter 7, in the section on ‘*bow waves’* in the Table of Harmonic Series – or Sieve of Eratosthenes. (In regular mathematics I believe it is called the difference of squares.) There the formula $N = n^2 - s^2$ was derived and described, for any odd number N. A slightly more complicated formula was found for where N is an even number. However, a further development of the simpler odd-number relationship, allows the inclusion of even numbers within a single algorithm. The best way of seeing how this algorithm works is visually, by picturing the numbers as areas – as squares and rectangles. Figures R.1 and R.2 illustrate this extension of the $N = n^2 - s^2$ relationship from odd to even numbers with examples. The odd number N = 133 is examined and then the even number N = 82.

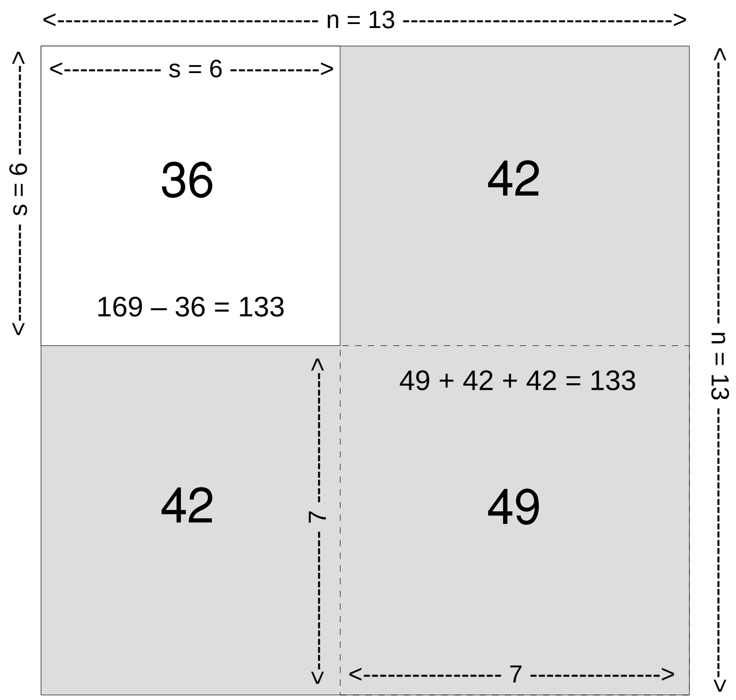


Figure R.1 (13.17) Number = 133. By subtracting s^2 from n^2 the number N is deduced (i.e. 13 x 13 minus 6 x 6 = 133). Looking at the squares of n and s, superimposed, reveals that N 133 (the grey area) is composed of another square, plus two identical rectangles: 7 x 7 + 42 + 42 = 133 which may be combined into one rectangle 7 × 19. Therefore divisors of N are 7 and 19 (i.e. 7 + (42 + 42)/7).

Because the same ‘remainder’ relationship holds true for both odd and even numbers, with the two gray leftover rectangles and one square (Figures R.1–2) necessarily taking integer values when whole numbers divide N (despite n and s themselves not always being whole numbers), this characteristic allows a simple algorithm to be devised: whenever a perfect whole numbered square (i.e. gray 2 × 2 square below), equal to or less than the given number ‘N’, is subtracted from area N, leaving over two rectangular areas; then, two whole numbered divisors of N will be: 1) the root of that perfect (gray) square, and 2) the sum of that root, plus the area of the two rectangles divided by that root (i.e. the sum of the non-root sides of the rectangles).

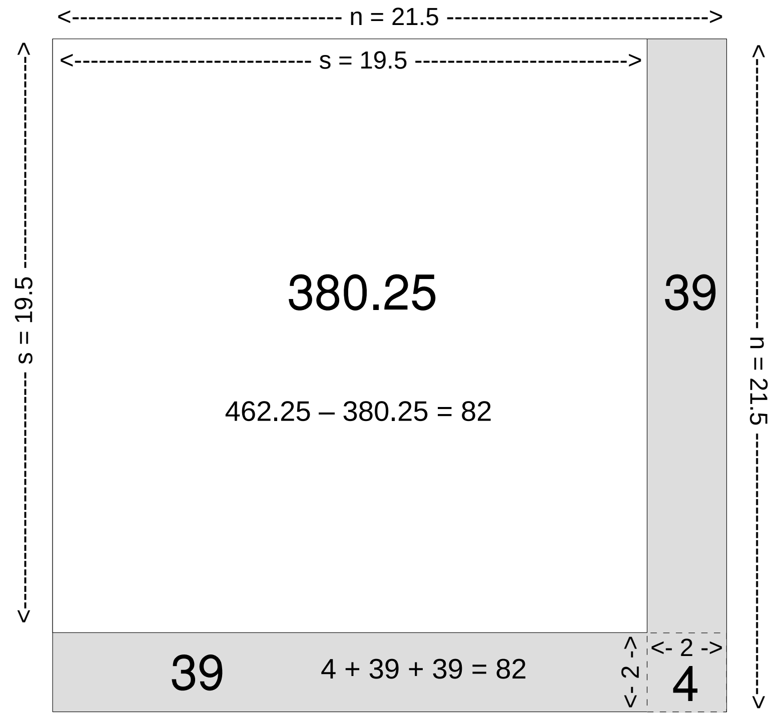


Figure R.2 (13.18) Number = 82. The same relationship as illustrated in Figure 13.17, holds true for even numbers too, in spite of n and s being fractional: 21.5 x 21.5 minus 19.5 x 19.5 equals 82. The imposition of s^2 upon n^2 again reveals another square and two identical rectangles: 2 x 2 + 39 + 39 = 82 which translates into the rectangle shown in Figure R.3 (13.19). Therefore divisors are 2 and 41 = 2 + (39 + 39)/2.

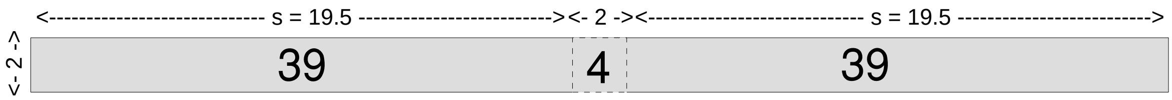


Figure R.3 (13.19) Rotating one leftover rectangle perhaps makes things clearer: area N = 82 = 2 x (19.5 + 2 + 19.5)

Thus when presented with any number ‘N’ for which one wishes to find the divisors, first calculate the largest perfect square equal to or less than N and then proceed in whole numbered steps downward from this square, testing each descending square in turn against the algorithm. Whenever the procedure produces an integer result for the ‘leftover’ rectangles, two divisors of N have been found. Essentially, the algorithm anchors the largest (gray) perfect square that will fit within ‘area N’, in N’s bottom right corner, and sequentially compresses this (gray) square, in whole number steps, to one.

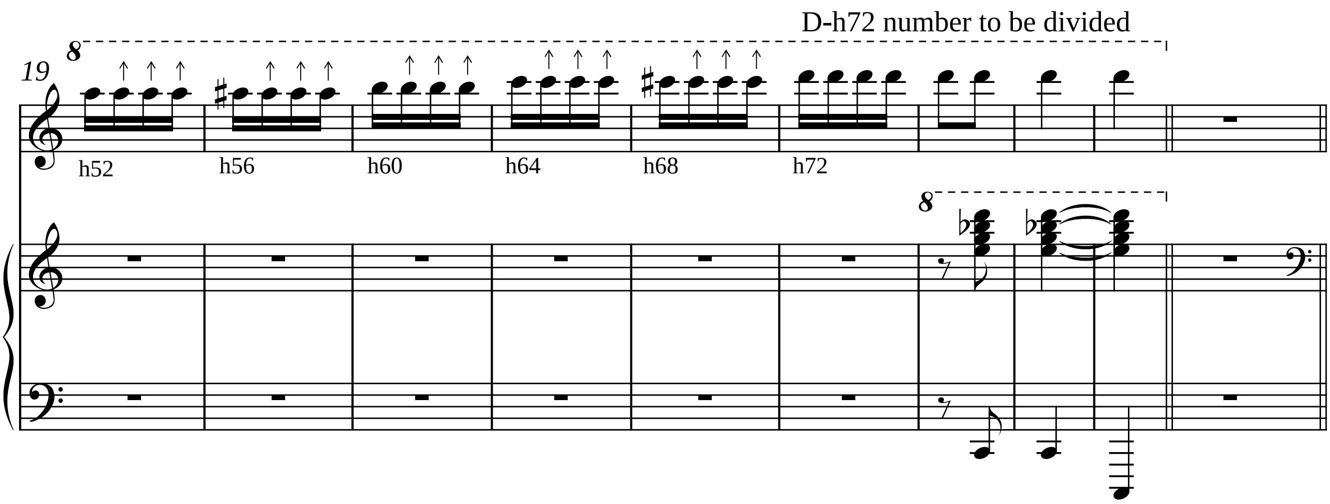
In the BASIC programming language (BBC Basic V) this procedure could be written out as three steps: First, acquire the number to be divided; second, find the largest perfect square that is less than or equal to it; and third, check each perfect square from this largest square down to the unit square in integer steps for leftover rectangles with whole number areas. Whenever the result meets this criterion, print out the whole numbered divisors found.

```
REMARK delineate number, for example 72.
PRINT "Please specify whole number to be divided"'
INPUT note_number
REMARK Loop 1. find largest perfect square equal to or less than note_number.
sqrt = 0
REPEAT
  sqrt = sqrt + 1
  square = sqrt * sqrt
UNTIL square >= note_number
IF square > note_number THEN sqrt = sqrt - 1
REMARK Loop 2. work down from value of sqrt to 1 in whole steps.
WHILE sqrt >= 1
  square = sqrt * sqrt
  difference = note_number - square
  result = difference / sqrt
  REM test if result is a whole number.
  IF result = INT(result) THEN
    divisor_1 = sqrt
    divisor_2 = sqrt + result
    PRINT "Divisors: "; divisor_1; " x "; divisor_2
  ENDIF
  sqrt = sqrt - 1
ENDWHILE
END
```

Applying the selfsame procedure as given in the above BASIC program, but using mutable base numbers operating upon the ‘physical devices’ that we call musical instruments (and writing out the progress through each loop exhaustively), produces the following score for the input number seventy-two. The score requires microtonal notes to be played in the upper part (violin), indicated by small arrows above the notes where one staff note covers a range of two or four frequencies. For example, the written top C may stand for four frequency inflections – h64, h65, h66 and h67 – of a notional root or fundamental frequency: h1.

An Explicit Demonstration of Tonal Computation in Mutable Base Numbers

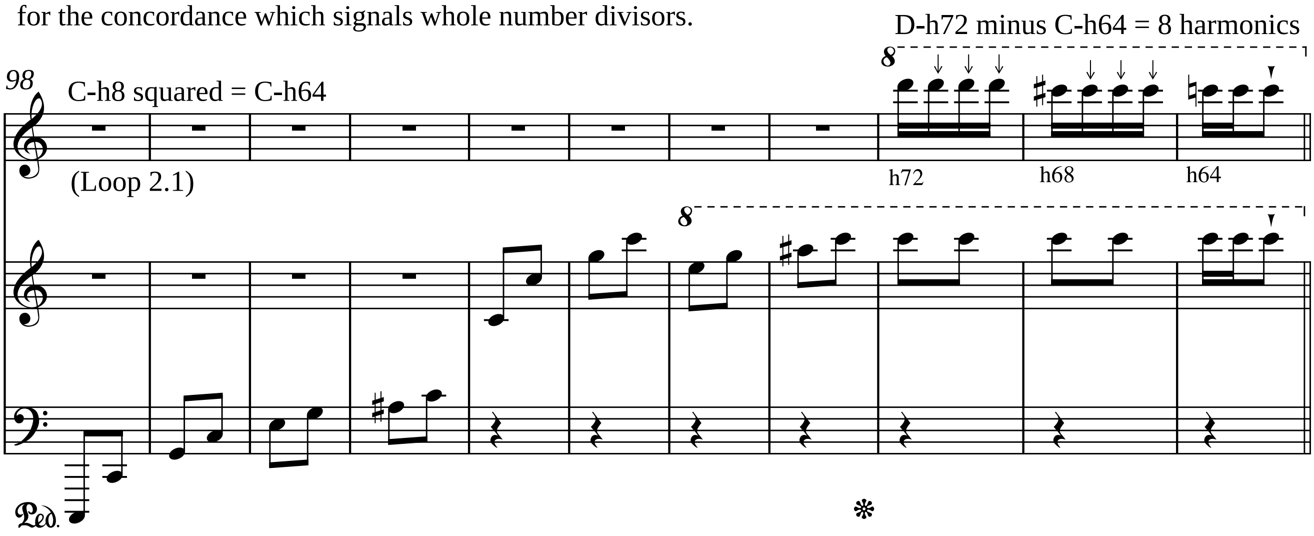
[Where more than one harmonic of the fundamental tone C-h1 is represented by a single note,
e.g. F♯h22 and F♯h23 above, arrows (↑↓) are used to distinguish between them.
The arrow symbol indicates roughly an eighth-tone, quarter-tone or three eighth-tones as appropriate.]

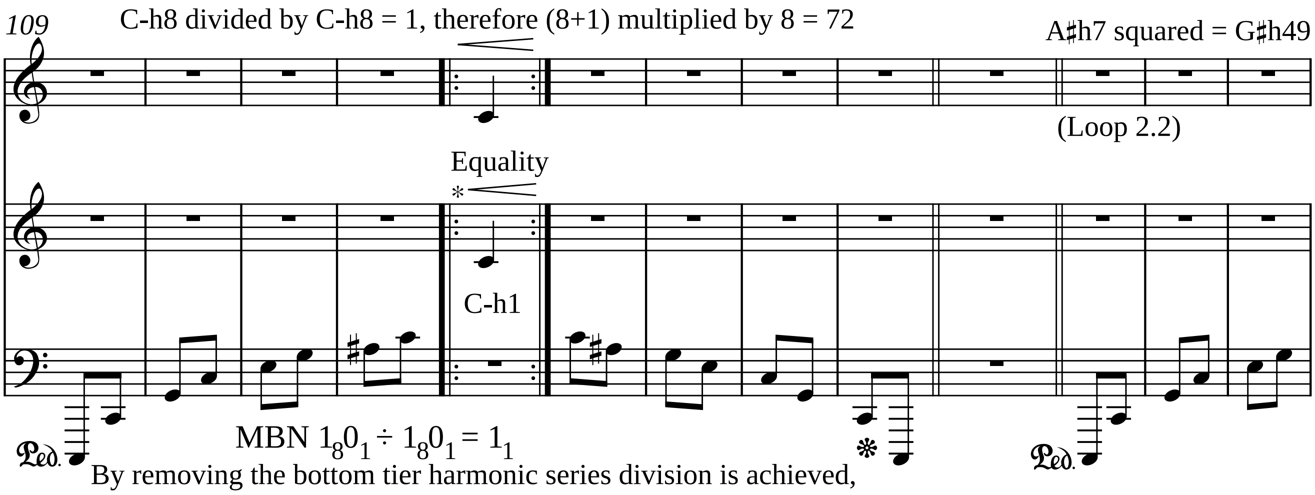


REMARK Loop 1. Find largest square number equal to or less than D-h72

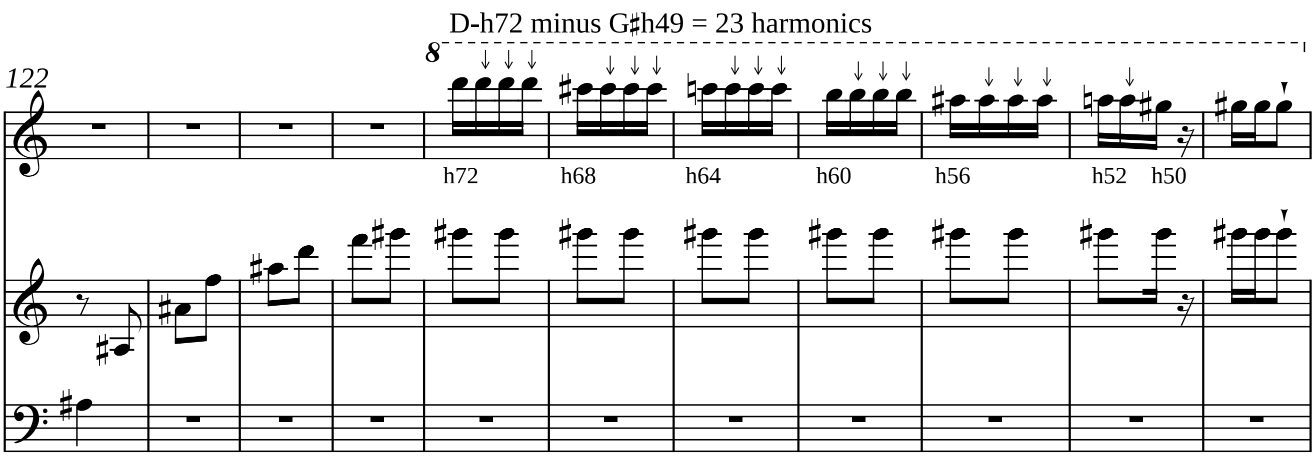

66
9 squared
(Loop 1.9)
64 < 72
Ped.
79
therefore 8 squared is largest square number equal to or less than D-h72
9 squared, E-h81 is greater than D-h72
81 > 72
81 > 72
Ped.
88
64 < 72
64 < 72

REMARK Loop 2. Check each square number in descending order from C-h64 for the concordance which signals whole number divisors.





By removing the bottom tier harmonic series division is achieved, here division by eight leaves C-h1 of the second tier nested series.
–i.e. Decimal 8 ÷ 8 = 1

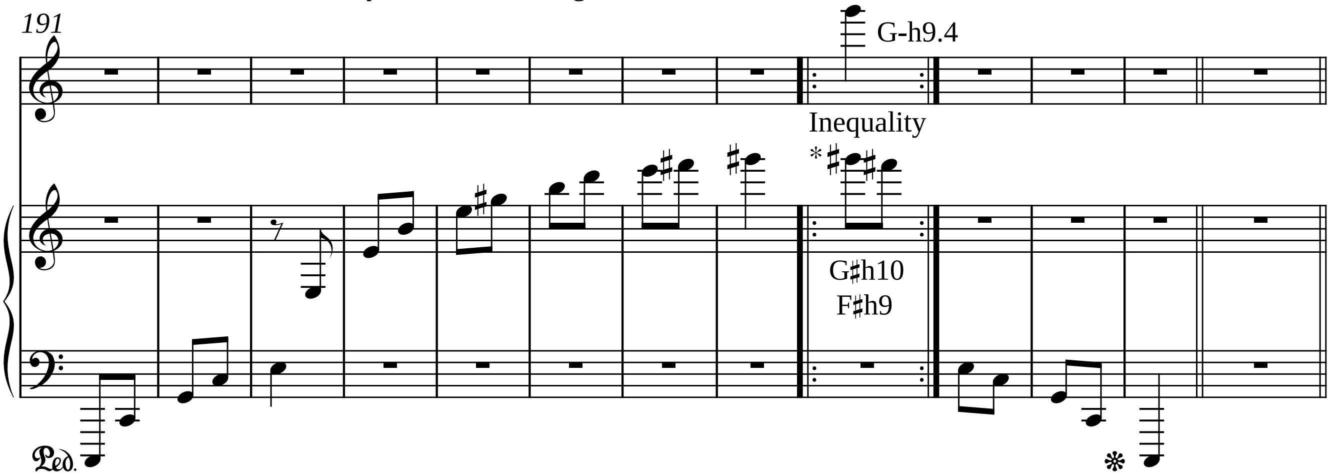


* Repeat ad lib. with crescendo and allargando.

* Repeat ad lib. with crescendo and allargando.

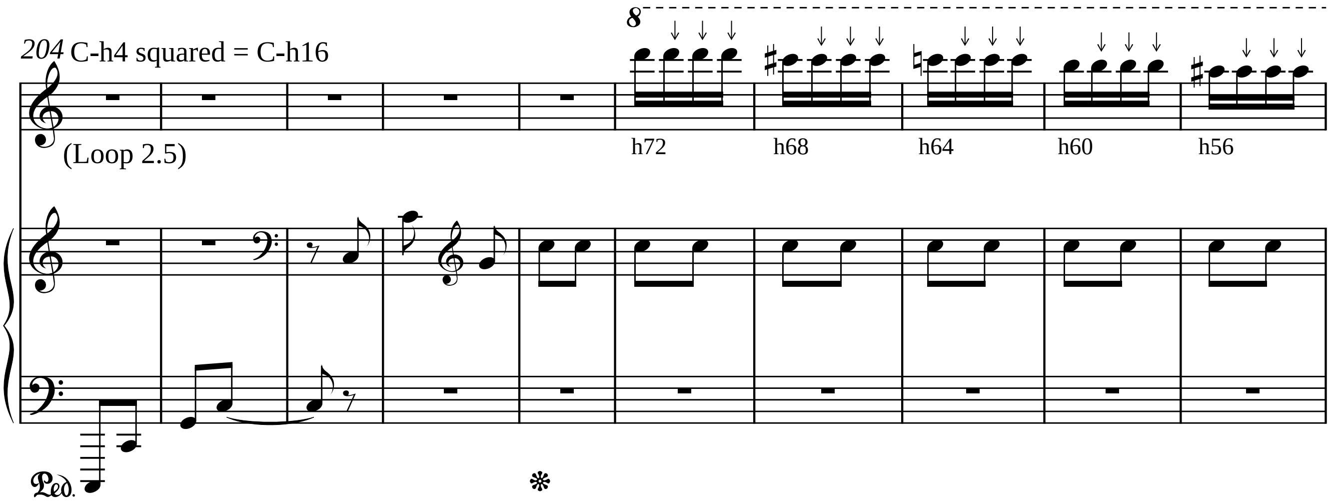
* Repeat ad lib. with crescendo and allargando.

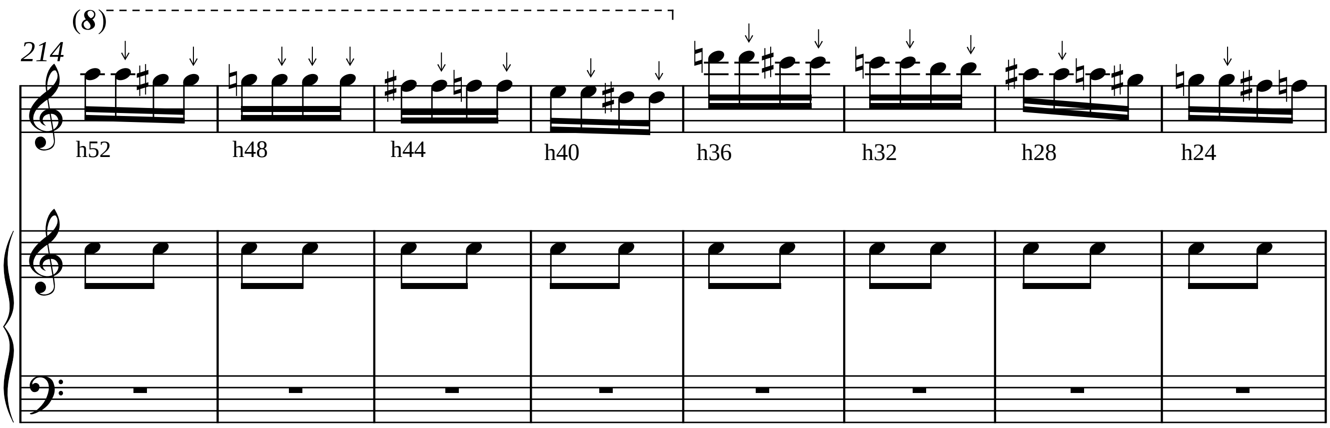
G-h47 divided by E-h5 = non-integer value between 9 and 10, therefore no whole number divisors



MBN $(9_5 0_1 + 2_1) \div 1_5 0_1 = 9_1 2_5$ (i.e. $2_1 \times 0_1 1_5 = 0_1 2_5$)
Decimal $(45 + 2) \div 5 = 9.4$

D-h72 minus C-h16 = 56 harmonics





* Repeat ad lib. with crescendo and allargando.

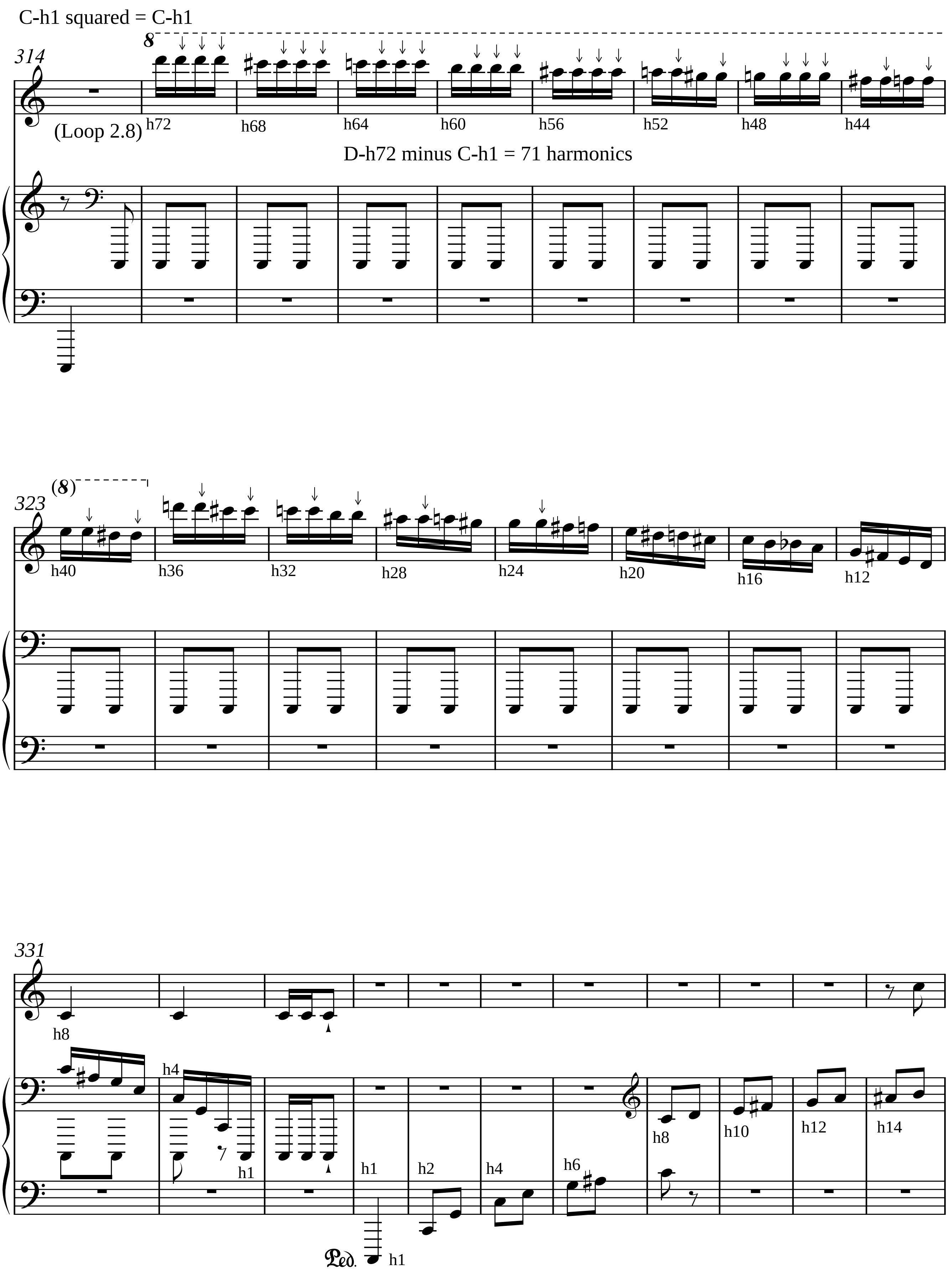
* Repeat ad lib. with crescendo and allargando.

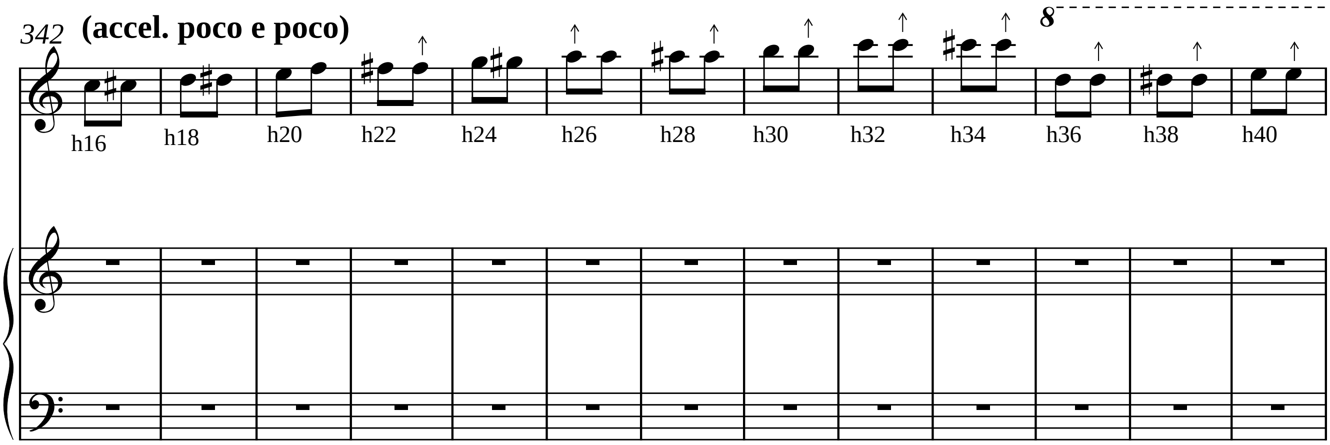
* Repeat ad lib. with crescendo and allargando.

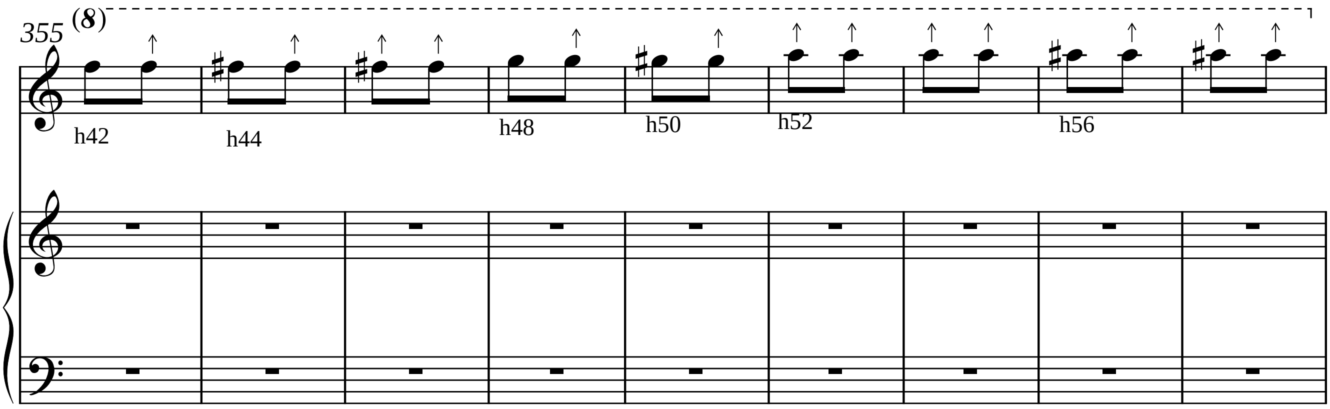
MBN $34_2 0_1 \div 1_2 0_1 = 34_1$

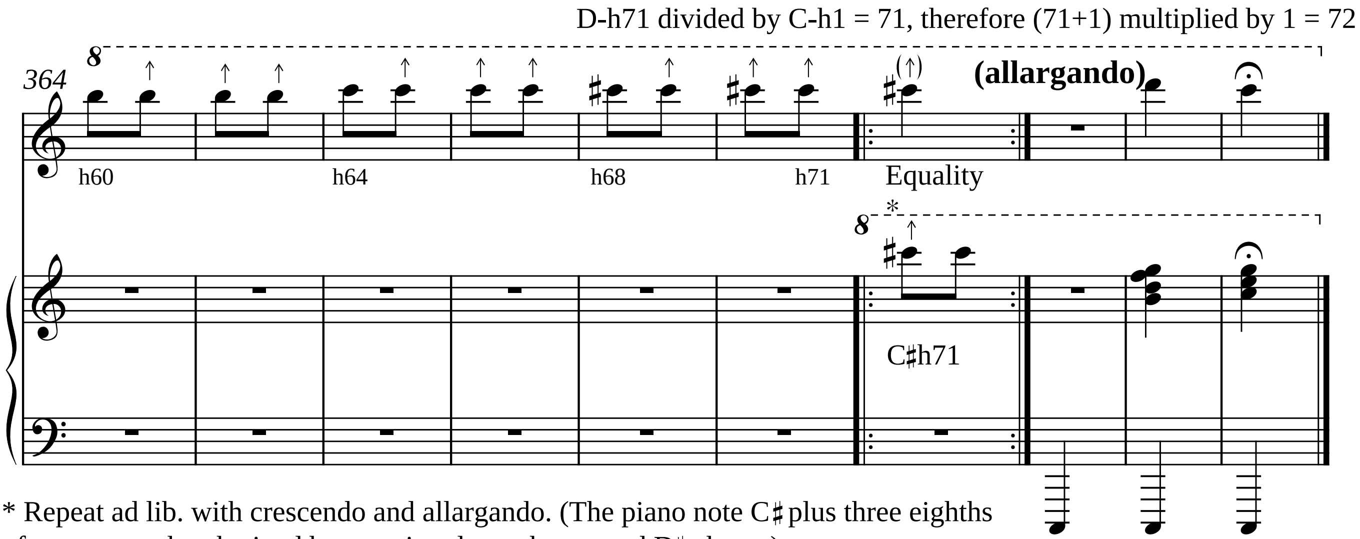
Decimal 68 ÷ 2 = 34

* Repeat ad lib. with crescendo and allargando.


C-h1 squared = C-h1
314
(Loop 2.8)
h72
h68
h64
h60
h56
h52
h48
h44
D-h72 minus C-h1 = 71 harmonics
323
h40
h36
h32
h28
h24
h20
h16
h12
331
h8
h4
h1
h1
h2
h4
h6
h8
h10
h12
h14
h1







* Repeat ad lib. with crescendo and allargando. (The piano note C♯ plus three eighths of a tone may be obtained by retuning down the unused D♯ above.)