A spreadsheet Demo of Brocot's method

Henry Casson 2011

The computer science/mathematics column in American Scientist has been written for a number of years by Brian Hayes, very much in the excellent tradition of Martin Gardner.
I enjoyed all of these, but spent most time on a column entitled "On the Teeth of Wheels", in 2000. Hayes relates that in a previous column a mathematical problem had been solved with the use of a mathematical technique known as the Stern-Brocot tree. At that time he did not need to follow the references to these two gentlemen, but he subsequently did so. Stern turns out to be Moriz Stern, Gauss's successor in a chair at Gottingen, notable as the first Jew to attain such a rank in Germany, and Brocot turned out to be our old friend Achille Brocot, whose name and escapement are always visible to horologists. The two worked completely independently, Stern producing a work of number theory, and Brocot a practical application.
The tree is a way of setting out a series of what are also known as Farey sequences, Farey upholding the honor of England.
The reason that a horologist becomes involved is that the tree ends up by laying out every possible fraction in its lowest terms, in ascending order. Any clock which displays complicated indications, particularly of course astronomical ratios, and does not use an analog device such as a cam, must do so with a gear train. The problem solved is to find the best gear train to produce a non-integer ratio using pairs of gears with integer numbers of teeth. There are obviously other constraints such as a reasonable maximum and minimum number of teeth in a gear or pinion.
Hayes describes Brocots method lucidly. The paper can be found at

http://www.americanscientist.org/issues/pub/on-the-teeth-of-wheels

Brocot is harder to track down. It is on the web, courtesy of Google books.
The Revue Chronometrique, Vol 3 can be found at:

TinyURL.com/3fbrzmu note: click on "front cover image" upper left

The article is between pages 186 and 194. It is in French, of course. There is a translation by John Kirk in HSN 2008-2. A copy will be posted on the HSN web site.

Brocot's table is also hard to find, but in case anyone really wants it, there is a copy at

TinyURL.com/3qr7u3u

I wanted to explore the algorithm, and did so by putting it in a spreadsheet. I think that this is more a demonstration than a working tool, but it did find me a useful gear set when engaged in a discussion of the Prague astronomical clock. If you seriously need a gear ratio, the trick is to consult the superior wizard, the aforementioned John Kirk at geartrains.com.

Starting out we see at the top left corner

	A	B	C	D	E	F
1	Numerator	Denominator	Error	Ratio	accuracy	Process
2	0	1		0.00000		

The user starts out by filling in a numerator in the yellow square A2, and optionally a denominator in A3.

	A	B	C	D	E	F
1	Numerator	Denominator	Error	Ratio	accuracy	Process
2	3.14159	1		3.14159		

D2 contains the ratio of these.

Now we start Brocots method.

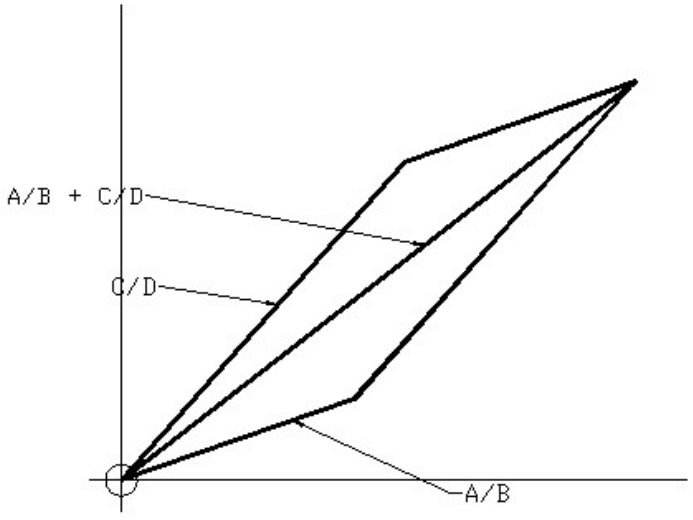
The integer ratio we want obviously lies between the next integer below and the next integer above the non-integer ratio.

3	=FLOOR(Numerator/Denominator,1)
4	=CEILING(Numerator/Denominator,1)

In the next row we put the mean of these.

In subsequent rows we put the mediant, *either of the previous row and the floor, or of the previous row and the ceiling, depending on which gives us the smaller error*. This is the basis of the Brocot algorithm. The mediant of two fractions $\frac{A}{B}$ and $\frac{C}{D}$ is the fraction $\frac{A+B}{C+D}$ It is guaranteed to lie between them.

This can be seen intuitively by just graphing the addition.



The conditional formulae are
For column A
=IF(ABS(C5+$C$4)<ABS(C5+$C$3),A5+$A$4,A5+$A$3)
And similarly for B and C
=IF(ABS(C5+$C$4)<ABS(C5+$C$3),B5+$B$4,B5+$B$3)
=IF(ABS(C5+$C$4)<ABS(C5+$C$3),C5+$C$4,C5+$C$3)

Having entered the original ratio the spreadsheet evaluates all the formulae, producing this:

	A	B	C	D	E	F
1	Numerator	Denominator	Error	Ratio	accuracy	Process
2	3.14159	1		3.14159		
3	3	1	-0.14159	3.00000	0.14159	
4	4	1	0.85841	4.00000	0.85841	
5	7	2	0.71682	3.50000	0.35841	
6	10	3	0.57523	3.33333	0.191743	
7	13	4	0.43364	3.25000	0.10841	
8	16	5	0.29205	3.20000	0.05841	

Now we are interested in the best ratio, but also in one that can be made into a gear train, so we need both to sort the list, and to factorise the numerators and denominators.
For this we have to leave the spreadsheet environment and go to the visual basic macro programming language. This is a complete programming environment available to all Microsoft office components. It is really Pascal, but Bill did not want to scare away his customers.
The “Process “ button has attached to it a Macro, which is an executable program that runs when you click on it. For security reasons, since these can be nasty, you have to enable their use. So now hit the button
The macro is called Showfactors, and it starts by factorizing the numerator and denominator, by the simple minded method of dividing them by the first 55 prime numbers and seeing if there is a remainder. 55 primes because the 55th prime is 257, a reasonable stopping point.
I could have typed in a list of these divisors, but as with anyone who misspent his middle age in programming, it was less effort to put in the sieve of Eratosthenes and let the computer do it.

```
Const NumPrimes = 55
Const MaxPrime = 257
Const Numerator = 1, Denominator = 2
Dim prime(1 To NumPrimes) As Integer
Dim theRow, theCol As Integer

Sub ShowFactors()
 Sieve
 theRow = 3
 theCol = 6
 ScreenUpdating = False   ' speedup
 While Cells(theRow, 1).Value <> 0
  Factorise theRow, Numerator
  Factorise theRow, Denominator
  theCol = 6
  theRow = theRow + 1
 Wend
 Sort
 Censor
 Cells(1, 1).Select   'back to the top
End Sub
```

The sieve

```
Sub Sieve()
Dim I, J As Integer
Dim PrimeArray(1 To MaxPrime) As Boolean
    For I = 2 To 257
       PrimeArray(I) = True
    Next I
    For I = 2 To Int(Sqr(MaxPrime))
       If PrimeArray(I) = True Then
           For J = 2 To Int(MaxPrime / I)
               PrimeArray(I * J) = False
           Next J
       End If
    Next I
    J = 1
    For I = 1 To MaxPrime
        If PrimeArray(I) = True Then
           prime(J) = I
           J = J + 1
        End If
   Next I
End Sub
```

Factorisation.

```
Sub Factorise(R, C As Integer)
Dim Count, PR, N As Integer
Dim theNum As Long
Count = 0
theNum = Cells(R, C).Value
For PR = 1 To NumPrimes
 N = prime(PR)
 While theNum / N = Int(theNum / N)
    theNum = Int(theNum / N)
    Count = Count + 1
 Wend
 If Count = 1 Then Cells(R, theCol).Value = N Else
  If Count > 1 Then Cells(R, theCol).Value = N & "^" & Count
 If Count > 0 Then theCol = theCol + 1
 Count = 0
 Next PR
 If theNum > 1 Then
 Cells(R, theCol).Value = theNum
 theCol = theCol + 1
 End If
 If C = Numerator Then
 Cells(R, theCol).Value = "/"
 theCol = theCol + 1
 If theNum = 1 Then Cells(R, theCol).Value = 1
 End If
End Sub
```

Then we sort with the least errors on top, and remove the unwanted rows (A factor over 257, or a multiple of a previous value)
To do this sort in place the formulae in the spreadsheet are replaced by their values, so the spreadsheet cannot be reused. That is why it is provided as a template, and you have to open a new document each time you use it. Do not save the result as a template, and keep a spare copy!

A more experienced programmer could certainly improve this enormously, but I have spent enough time on it.

The Sort

```
Sub Sort()
    ScreenUpdating = False ' speedup
    Range("A3:BH262").Select
    Selection.Copy
    Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
        :=False, Transpose:=False
    Application.CutCopyMode = False
    Selection.Sort Key1:=Range("E4"), Order1:=xlAscending, Header:=xlGuess, _
        OrderCustom:=1, MatchCase:=False, Orientation:=xlTopToBottom, _
        DataOption1:=xlSortNormal
End Sub
```

Removing unwanted rows

```
Sub Censor()
Dim R As Integer
Dim M As Long
 R = 3
 While Cells(R, 1).Value > 0
 M = Maxval(R)
  If (M > MaxPrime) Then
   Rows(R).Select
   Selection.Delete Shift:=xlUp
  Else: R = R + 1
  End If
 Wend
 R = 3
 While Cells(R, 1).Value > 0
  If Cells(R, 5).Value = Cells(R - 1, 5).Value Then
   Rows(R).Select
   Selection.Delete Shift:=xlUp
  Else: R = R + 1
  End If
 Wend
End Sub



Function Maxval(R)
Maxval = 1
C = 6
V = Cells(R, C).Value
While V <> 0
 If IsNumeric(V) Then Maxval = V
 If Maxval > MaxPrime Then Exit Function
 C = C + 1
 V = Cells(R, C).Value
Wend
End Function
```

Bob will host a copy of the spreadsheet, and a copy of John Kirk's translation on the HSN161 site at www.hsn161.com/HSN/gearing.php

Henry Casson 2011