# An Algorithm For calculating the Period and Generator for a Ratio in 2D Temperaments

In general, for a temperament we are given ratios and how many periods and generators it takes to reach those ratios in the temperament:

A given point on our temperament lattice can be identified by how many period vectors, , and how many generator vectors, , it takes to reach the point. Given an input ratio , for it to be represented by the temperament, we must have:

Where is Ratio 1 etc., and the s satisfy the equations:

Let be the total number of distinct primes, , which are required to factor all of our ratios . Also let be the multiplicity of the th prime in the factorization of the th ratio. A first test of whether our inputted ratio, , can be represented by the temperament, is whether can be factored into the primes, , which are required to factor all our ratios . Assuming this is the case, let be the multiplicity of the th prime in the factorization of our inputted ratio. Then, from equation (1) we must have

Let’s combine our equations for , and into one big matrix equation

Here the unknowns are our s and and . It is helpful to rewrite the equations so all the unknowns are in one vector

Note that above, we have equation and unknowns. Typically the number of primes equals the number of , so . In this case, we expect, to have at most one solution. If, however there are more ratios than primes, then . We then have more unknowns than equations. In this case there could be multiple solutions. This is typically the case when the temperament is 1 dimensional.  
We can solve the above system of equations by reducing the left most matrix to Smith normal form. Defining

we can write our matrix equation as

We now decompose into Smith normal form, ,where ***a*** is diagonal and and are invertible matrices with integer matrix inverses. Then we can write

Defining and we can write this as

Note that for efficiency, we can write

Where is just with the last two columns lopped off. The equation is simple to solve since is diagonal. In components we have

where is the rank of . Note that since both and are integer matrices, we know is also an integer matrix. Thus, we see there are solutions if and only if for , divides , and for for , . If these conditions are met, then the set of all our solutions are given by

where all the s range over the set of integers. Once we have , we can calculate our and via

Of course, we don’t really care about the . So, we really only need

Where

Or writing out components,

Which can be multiplied out to give

And then rearranged to yield

Our job now is to find theand which satisfy the above equation, given some integers and . Note that if then the above equation becomes

and there is only one solution. This is expected to typically be the case for a 2D temperament. Also note that this is always a solution, since it can be obtained by setting all the to zero. However, there are times when there may be more than one solution, such as in the case of a 1D temperament. In this case, it would be desirable to find the solution such that is minimum.

In perms of solving the problem more generally, first consider

Where

Or written as matrices,

First consider . By Bézout's identity we see that we must have

Where is an integer which we are free to choose. However, we are bounded in our choice of by the inequality , where is the smallest distance of the solutions so far explored (note that when we first start our search, we can set equal to the distance of the solution with all s set to zero). To see this, not that if we choose a such that , then the distance of the solutions we are exploring will all be larger than , no matter what the other turns out to be. In terms of efficiency, it makes sense to choose start our search with a such that is small since smaller s tend to correspond to smaller distances, and if we find a solution with a distance which is smaller than , we can set equal to this new distance which tightens the bounds of our search, which in turn leads us to our solution faster. So to choose a , it makes sense to set to zero in the above equation, solve for , and then round it to the nearest integer. Once we pick a , we need to solve the equation

where . Convert it to Smith normal form, , we obtain

Which can easily be solved

and we can write as

Plugging this into equation (5) we get

Which we can rearrange to

Now the first row of this equation is solved for any set of (that is what we just solved) so lets drop the first row from the above equation:

This can be written

Where

We can now continue recursively. Again, by Bézout's identity, we see that we must have

Where is an integer which we get to choose. We are now bounded in our choice of by . We then continue this process recursively.