# An Algorithm for Calculating the Simplest Ratios for 2D Temperament Lattices

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. The set of all ratios which can consistently be assigned to a give lattice point are given by

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

If we will typically find an infinite number of solutions to these equations. We would like to assign the simplest fraction which satisfies the above equations to the lattice point. To get the simplest fraction, we need to factor our ratios into primes, . 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. Then a good measure of the simplicity of a fraction is the Tenney distance:

The simplest fraction would have the smallest Tenney distance. Thus, our goal is to minimize the Tenney distance subject to equations (1) and (2). It turns out to be convenient to write this equation in terms of matrices. To that end define

Then we have

And we can write equation (3) as

There are terms in equation (4) which we would like to get as close to zero as possible, and we have variables to fiddle with. We also need to satisfy the two constraints given by equations (1) ands (2). If the were allowed to be real numbers, then we could solve a system of linear equations. We have a total of equations. Thus, if were allowed to be real, we could set all but terms inequation (4) to zero. However, our are constrained to be integers. In fact, there might never even be a solution to both equations (1) and (2). That said, typically this added constraint will make it so that we can solved all but equations. Hence, we can typically expect to set of the terms in equation (4) to zero. Note that we will typically still have an extra integer parameter which we can vary. We can then vary this parameter to try to get the remaining terms of equation (4) as close to zero as possible.

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

Our first task is to find and that satisfy the above equations. Note that and are inputs. It is helpful to rewrite the equations so all the unknowns are in one vector

Here we have a set of equations and unknowns. 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, ***U A V=a,*** where ***a*** is diagonal and ***V*** and ***U*** are invertible. Then we can write

Defining and we can write this as

This is simple to solve since is diagonal. In components we have

where is the rank of . We see there are only solutions if for , divides , and for for , . If these conditions are met, then 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 thes which satisfy the above equation, for some integers and , such that the Tenney distance is minimum. Since, generically, small s equate to small Tenney distances, a good strategy is to search through the various solutions from small s to large s. The equations we need to satisfy are of the form

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 Tenney distance of the solution so far explored (note that when we first start our search, is equal to the Tenney distance of the inputted ratio). To see this, not that if we choose a such that , then the Tenney distance of the solutions we are exploring will all be larger than , no matter what the other s turn 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 Tenney distances, and if we find a solution with a Tenney distance which is smaller than , we can set equal to this new Tenney distance which tightens the bounds of our search, which in turn leads us to our solution faster. Once we pick a , we need to solve the equation

where . Convert it to Smith normal form, ***UAV=B***, 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.