Appendices

A Python Code

The following code has been written using Python. However, the same processes that will be shown can be written with other languages.

The first code function is defined as '`proj(N)`' which constructs a complete game 'of order N'. Each card contains $N+1$ objects. Indeed, as we have already established, some projective planes don't exist for certain orders of N. In section 2.2, for example, we mentioned that $r = 7$ (equivalently $N = 6$) does not produce a working game.

```
def proj(N):
    full_list={i:[] for i in range(1,N**2+N+2)}
    for c in range(0,N+1):
        y=[N*c+b+2 for b in range(N)]
        y.insert(0,1)
        full_list[c+1]=y
    for c in range(N):
        for a in range(2,N+2):
            x=[a]
            dummy=list(range(N))
            for b in range(N):
                x.append(N*(b+1)+dummy[(c+(b)*(a-2))%N]+2)

            full_list[N*(a-1)+2+c]=x
    return full_list

print(proj(3))
```

Recall the set of cards produced in Example 2.7. These were produced using the '`proj(N)`' function and replacing `N` with `3`. We can be sure that if $N = p^n$ for some prime integer p, then the '`proj(N)`' function should return a complete game that satisfies all the rules.

The second code function is defined as '`inci(N)`' which takes a set of cards produced by the '`proj(N)`' function and creates an incidence matrix A. It is clear that A is an $n \times n$ matrix, where $n = N^2 + N + 1$. Suppose we have a collection of cards from '`proj(N)`' and let the i^{th} card have the elements: $\{t_1, t_2, ..., t_r\}$. Then $A_{ij} = 1$ for every $j = t_i$ $(1 \leq i \leq r)$ an $A_{ij} = 0$ otherwise.

```
import numpy as np
def inci(N):
    zero=[0]*(N**2+N+1)
    A=np.array([zero]*(N**2+N+1))
    for i in range(N**2+N+1):
        x=proj(N)[i+1]
        y=[0]*(N**2+N+1)
        for j in x:
            y[j-1]=1
        A[i]=y
    return A

print(inci(3))
```

For example, if we set $N = 2$, we get a 7×7 matrix. The second card produced by '`proj(N)`' is (1,4,5). Therefore, the second row of matrix A gives:

$$(1\ 0\ 0\ 1\ 1\ 0\ 0)$$

Recall that for any incidence matrix of a projective plane, the inner product between two distinct rows must always equal to 1. We can check to make sure that our incidence matrix is valid for a projective plane - or show that for certain orders of N, a valid incidence matrix cannot be constructed. In order to check this, we use the section of code below:

```
from numpy import *
N=6
inc=inci(N)
n=N**2+N+1
for i in range(n):
    for j in range(n):
        if dot(inc[i],inc[j])!=1 and i!=j:
            print('(i,j)=('+str(i+1)+','+str(j+1)+')')
```

This code takes every possible pair of rows from the incidence matrix and checks their inner product. If two rows had an inner product that was **not equal to 1**, then the code will write down the row numbers of which the situation occurs.

If we set $N = 6$, where we know a projective plane of this order doesn't exist, we return a list of pairs (i, j), such as $(i, j) = (8, 20)$ which implies that the 8^{th} and 20^{th} rows do not give an inner product equal to 1.[14] Now, if we set $N = 3$ where a projective plane does exist, hence the existence of a valid incidence matrix, then the code does not yield a single pair, implying that every pair of rows satisfies the condition that inner products must equal to 1.

There is an interesting point to consider. One value of N we are able to try is $N = 12$. By producing a possible incidence matrix, we actually find pairs of rows that **do not** have an inner product equal to 1. So this implies that the projective plane of order 12 does not exist. However, this is not necessarily true because the existence of an order 12 plane has not been disproved yet. In fact, the '`proj(N)`'

[14] This is the same example used in Section (Impossible Games).

code function written above is just one possible algorithm and is by no means solidly proven to be true. Thus, even if it implies that an order 12 plane does not exist, this does not answer the conjecture about the plane of order 12.

Note that it is possible that these codes may not correctly account for every case because it has not been proven that these codes are exhaustive to every case. The only way to discredit these codes is to find a counterexample, such as a value of N that is a prime power but the code yields an incorrect game in the '`proj(N)`' function - or even $N = 12$ if it turns out its projective plane actually exists!