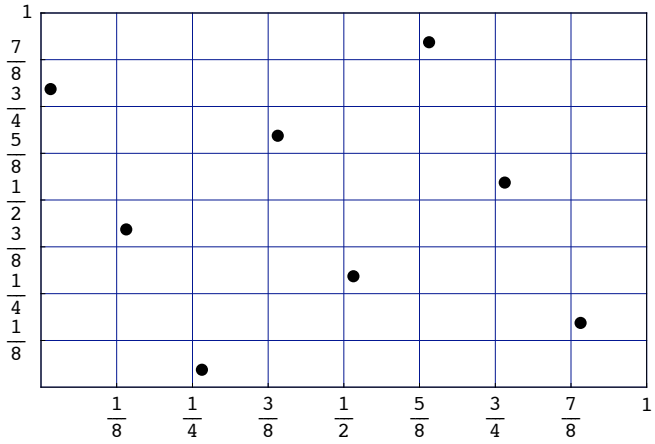
Nets and Sequences, Faure sequence, and scrambled Faure sequence

Introduction

Here I will describe the generation of the Faure sequence, and a way of randomizing it. For the definitions of the terms (t,s)-sequence and (t,m,s)-nets see "Random Number Generation and Quasi-Monte Carlo Methods" by Harald Niederreiter (published by SIAM). The basic distinction is simple: the sequence is an infinite sequence that has good uniformity, and the net is only a finite set of points with good uniformity. Since it is much easier to distribute a finite set as evenly as possible (as opposed to making sure an infinite sequence provides good uniformity for all of its finite segments), the discrepancy bound for a net is $O\left(\log N^{s-1} / N\right)$, better than that of a sequence given by $O(\log N^s / N)$. Here s is the dimension of the problem, and N is the number of points.

The Faure sequence is a (0,s)-sequence. If we take a finite point set from the Faure sequence as follows $\{q_k$: $\text{jb}^m \le k \le <(j+1)\, b^m\}$, then we obtain a (0,m,s)-net. Here b is the base of the sequence, which depends on the dimension s. Here is a picture that will help clarify the definitions:



What you see above is a plot of a (0,m,s)-net, where s=dimension=2, and m=3. It is obtained from the Faure sequence with s=2 and b=2, by extracting from the Faure sequence its elements with indices $4 * 2^3$ `through` $5 * 2^3 - 1$. (j=4 in the above definition)

The coordinates of the plotted vectors are:

$$\left\{\left\{\frac{1}{64}, \frac{51}{64}\right\}, \left\{\frac{33}{64}, \frac{19}{64}\right\}, \left\{\frac{17}{64}, \frac{3}{64}\right\}, \left\{\frac{49}{64}, \frac{35}{64}\right\}, \left\{\frac{9}{64}, \frac{27}{64}\right\}, \left\{\frac{41}{64}, \frac{59}{64}\right\}, \left\{\frac{25}{64}, \frac{43}{64}\right\}, \left\{\frac{57}{64}, \frac{11}{64}\right\}\right\}$$

and the first 35 elements of the Faure sequence are:

$$\left\{\left\{\frac{1}{2}, \frac{1}{2}\right\}, \left\{\frac{1}{4}, \frac{3}{4}\right\}, \left\{\frac{3}{4}, \frac{1}{4}\right\}, \left\{\frac{1}{8}, \frac{5}{8}\right\}, \left\{\frac{5}{8}, \frac{1}{8}\right\}, \left\{\frac{3}{8}, \frac{3}{8}\right\}, \left\{\frac{7}{8}, \frac{7}{8}\right\},\right.$$
$$\left\{\frac{1}{16}, \frac{15}{16}\right\}, \left\{\frac{9}{16}, \frac{7}{16}\right\}, \left\{\frac{5}{16}, \frac{3}{16}\right\}, \left\{\frac{13}{16}, \frac{11}{16}\right\}, \left\{\frac{3}{16}, \frac{5}{16}\right\}, \left\{\frac{11}{16}, \frac{13}{16}\right\}, \left\{\frac{7}{16}, \frac{9}{16}\right\},$$
$$\left\{\frac{15}{16}, \frac{1}{16}\right\}, \left\{\frac{1}{32}, \frac{17}{32}\right\}, \left\{\frac{17}{32}, \frac{1}{32}\right\}, \left\{\frac{9}{32}, \frac{9}{32}\right\}, \left\{\frac{25}{32}, \frac{25}{32}\right\}, \left\{\frac{5}{32}, \frac{5}{32}\right\}, \left\{\frac{21}{32}, \frac{21}{32}\right\},$$
$$\left\{\frac{13}{32}, \frac{29}{32}\right\}, \left\{\frac{29}{32}, \frac{13}{32}\right\}, \left\{\frac{3}{32}, \frac{15}{32}\right\}, \left\{\frac{19}{32}, \frac{31}{32}\right\}, \left\{\frac{11}{32}, \frac{23}{32}\right\}, \left\{\frac{27}{32}, \frac{7}{32}\right\}, \left\{\frac{7}{32}, \frac{27}{32}\right\},$$
$$\left.\left\{\frac{23}{32}, \frac{11}{32}\right\}, \left\{\frac{15}{32}, \frac{3}{32}\right\}, \left\{\frac{31}{32}, \frac{19}{32}\right\}, \left\{\frac{1}{64}, \frac{51}{64}\right\}, \left\{\frac{33}{64}, \frac{19}{64}\right\}, \left\{\frac{17}{64}, \frac{3}{64}\right\}, \left\{\frac{49}{64}, \frac{35}{64}\right\}\right\}$$

Note that the last 4 elements in the above list are elements of the net.

Generating the Faure sequence

Let's assume we want to generate 5 dimensional Faure vectors, and at most 1,000,000 of them.

In[1]:=
```
dim = 5;
max = 1000000;
```

Compute the smallest prime number **p** greater than or equal to **dim.** This is the base of the sequence, which was denoted by b in the Introduction.

In[3]:=
```
p = NextPrime[dim - 0.1]
```

Out[3]= 5

In this example, the dimension **dim** happens to be the same as the base, **p**. Now compute the number of digits in base p = 5 expansion of integer **max**

In[4]:=
```
uppersize = Floor[Log[p, max] + 1]
```

Out[4]= 9

Next compute the so-called Pascal matrices that are used in generating the Faure sequence. The ij-th entry of the matrix is:

In[5]:=
```
pascal[1] := IdentityMatrix[
  uppersize];
pascal[h_] := Table[
  Mod[Binomial[j - 1, i - 1] *
   (h - 1)^(j - i), p],
  {i, uppersize}, {j, uppersize}];
```

This is what they look like:

In[7]:=
```
pascal[1] // MatrixForm
```

Out[7]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

In[8]:=
```
pascal[2] // MatrixForm
```

Out[8]//MatrixForm=

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 & 0 & 1 & 2 & 3 \\ 0 & 0 & 1 & 3 & 1 & 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 & 4 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

We need the function below which extracts a square submatrix of a given size d, starting from the upper left corner of the input matrix

In[9]:=
```
submatrix[mat_, d_] :=
 mat[[1 ;; d, 1 ;; d]]
```

For example, consider the matrix below:

In[10]:=
```
emat = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}
```

Out[10]=
```
{{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}
```

In[11]:=
```
emat // MatrixForm
```

Out[11]//MatrixForm=

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

If we apply the **submatrix** function with **d** = 2 to **emat** we get:

In[12]:=
```
MatrixForm[submatrix[emat, 2]]
```

Out[12]//MatrixForm=

$$\begin{pmatrix} 1 & 2 \\ 4 & 5 \end{pmatrix}$$

The following generates the nth Faure vector in dimension dim = 5.

In[13]:=
```
faure[n_] := Module[{digits,size},
   digits = Reverse[IntegerDigits[n, p]];
   size = Length[digits];
   Table[Mod[submatrix[pascal[h], size].digits, p] .
       p^(-Range[size]), {h, dim}]];
```

For example, the first and 7th vectors are:

```
In[14]:= faure[1]
```

$$\text{Out[14]}= \left\{\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}\right\}$$

```
In[15]:= faure[7]
```

$$\text{Out[15]}= \left\{\frac{11}{25}, \frac{16}{25}, \frac{21}{25}, \frac{1}{25}, \frac{6}{25}\right\}$$

Let's dissect the code. Consider the case n = 7. We first write 7 in its base **p** = 5 expansion, and then take the mirror image of it:

```
In[16]:= digits = Reverse[IntegerDigits[7, p]]

Out[16]= {2, 1}
```

To find the 4th component of **faure[7]**, we compute the pascal matrix, **pascal[4]**:

```
In[17]:= pascal[4]

Out[17]= {{1, 3, 4, 2, 1, 3, 4, 2, 1}, {0, 1, 1, 2, 3, 0, 3, 3, 1}, {0, 0, 1, 4, 4, 0, 0, 3, 2},
          {0, 0, 0, 1, 2, 0, 0, 0, 3}, {0, 0, 0, 0, 1, 0, 0, 0, 0}, {0, 0, 0, 0, 0, 1, 3, 4, 2},
          {0, 0, 0, 0, 0, 0, 1, 1, 2}, {0, 0, 0, 0, 0, 0, 0, 1, 4}, {0, 0, 0, 0, 0, 0, 0, 0, 1}}
```

We multiply the matrix **pascal[4]** with the reverse digits **digits**, however, the dimensions are not the same. You need to take the 2 by 2 submatrix of pascal[1] as we discussed before: (2 is the **size** of the set called **digits**)

```
In[18]:= submatrix[pascal[4], 2] // MatrixForm
```

Out[18]//MatrixForm=

$$\begin{pmatrix} 1 & 3 \\ 0 & 1 \end{pmatrix}$$

```
In[19]:= submatrix[pascal[4], 2] . digits

Out[19]= {5, 1}
```

This vector needs to be reduced mod p = 5. The following does the reduction.

```
In[20]:= Mod[submatrix[pascal[4], 2] . digits, 5]

Out[20]= {0, 1}
```

Now you use these digits to compute a decimal number in base p = 5, like we did in van der Corput sequence:

```
In[21]:= 0 * 1/5 + 1 * 1/5^2
```

$$\text{Out[21]}= \frac{1}{25}$$

In the code the above computation is done as follows: First note the following.

```
In[22]:= 5^(-Range[2])

Out[22]= {1/5, 1/25}
```

Next line takes the dot product of the above with **digits** {0,1}:

In[23]:=
```
Mod[submatrix[pascal[4], 2] . digits, 5] . 5^(-Range[2])
```

Out[23]= $\frac{1}{25}$

This is the fourth component of **faure[7]**. You do a similar calculation as **h** = 1,..., **dim**.

Generating the scrambled Faure sequence

If I need to use a single Faure sequence to get a single estimate for the problem I am solving, I would use one particular realization of a scrambled Faure sequence. If I need multiple estimates, and if I want the estimates to be independent so that I can apply statistics to estimate mean and variance, then I would use different realizations of a scrambled Faure sequence, and use each realization to get an estimate.

Here I will describe one scrambling approach; there are many others. It is by Matousek and called linear scrambling of the Faure sequence (see "On the L2-Discrepancy for Anchored Boxes", Journal of Complexity 14, 527-556, 1998). This scrambling approach was also used by Tezuka. There is one difference in my implementation; I do not have the random additive vector Matousek has in his paper.

The construction is similar to the regular Faure sequence. The only difference is in the "generator" matrices. In Faure, the generator matrices were the Pascal matrices. Here, the generator matrices will be obtained by multiplying the Pascal matrix, from the left, by randomly generated lower triangular matrices. One random matrix is needed for each dimension. Since we have dim = 5, we need 5 of these random matrices.

The following function **matrix[k]** computes a lower triangular square matrix of size **uppersize** whose non-diagonal entries are random integers between 0 and p-1, and diagonal entries are random integers between 1 and p-1.

In[24]:=
```
u[i_, j_] := Which[i < j, 0, i == j, RandomInteger[{1, p - 1}], i > j, RandomInteger[{0, p - 1}]];

matrix[k_] := matrix[k] = SparseArray[{i_, j_} /; i ≥ j → u[i, j], {uppersize, uppersize}];
```

The way I define these matrices using the syntax **matrix[i_]:=matrix[i]** makes *Mathematica* save them in memory, and read the values from memory whenever they are called. So when you evaluate matrix[1], the code generates a random matrix and sets it to matrix[1]. Next time you call matrix[1], you will get the same random matrix, not a new one. If you want to re-generate these matrices use the Clear function to erase them from memory.

In[26]:=
```
matrix[1] // MatrixForm
```

Out[26]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 3 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 1 & 3 & 4 & 0 & 0 & 0 & 0 & 0 \\ 3 & 4 & 2 & 1 & 3 & 0 & 0 & 0 & 0 \\ 4 & 0 & 3 & 0 & 1 & 2 & 0 & 0 & 0 \\ 1 & 2 & 4 & 4 & 4 & 2 & 2 & 0 & 0 \\ 2 & 2 & 1 & 0 & 0 & 3 & 2 & 1 & 0 \\ 0 & 4 & 1 & 2 & 1 & 3 & 0 & 4 & 3 \end{pmatrix}$$

In[27]:=
```
matrix[5] // MatrixForm
```

Out[27]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 2 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 3 & 3 & 3 & 1 & 3 & 0 & 0 & 0 & 0 \\ 4 & 4 & 4 & 2 & 3 & 1 & 0 & 0 & 0 \\ 4 & 1 & 3 & 3 & 2 & 0 & 2 & 0 & 0 \\ 1 & 3 & 0 & 2 & 2 & 1 & 1 & 3 & 0 \\ 3 & 0 & 2 & 1 & 0 & 2 & 0 & 2 & 3 \end{pmatrix}$$

Now we define the crucial generator matrices. They are obtained by multiplying the matrix **matrix[h]** and **pascal[h],** and reducing the product mod p, as h = 1,..., dim.

In[28]:=
```
genmatrix[h_] := genmatrix[h] =
  Mod[matrix[h] . pascal[h], p];
```

Some generator matrices:

In[29]:=
```
genmatrix[3] // MatrixForm
```

Out[29]//MatrixForm=

$$\begin{pmatrix} 3 & 1 & 2 & 4 & 3 & 1 & 2 & 4 & 3 \\ 4 & 2 & 2 & 0 & 2 & 3 & 4 & 4 & 0 \\ 1 & 3 & 4 & 1 & 2 & 2 & 1 & 3 & 2 \\ 2 & 4 & 4 & 4 & 2 & 4 & 3 & 3 & 3 \\ 0 & 2 & 1 & 2 & 0 & 0 & 4 & 2 & 4 \\ 0 & 4 & 0 & 1 & 3 & 4 & 1 & 1 & 4 \\ 4 & 1 & 4 & 2 & 1 & 3 & 1 & 4 & 2 \\ 1 & 1 & 1 & 2 & 3 & 4 & 1 & 1 & 1 \\ 1 & 4 & 4 & 4 & 4 & 4 & 2 & 1 & 2 \end{pmatrix}$$

Here is our basic scrambled Faure sequence. Compare this code with **faure[n]**.

In[30]:=
```
sfaure[n_] := Module[{digits, size},
  digits = Reverse[IntegerDigits[
    n, p]]; size = Length[
   digits]; Table[
   Mod[submatrix[genmatrix[h],
     size] . digits , p] .
   p^(-Range[size]), {h, dim}]];
```

Recall that this is a vector sequence of dimension dim=5. For example, the 7th element of the sequence is (compare this with faure[7]):

In[31]:=
```
sfaure[7]
```

Out[31]= $\left\{\frac{11}{25}, \frac{16}{25}, \frac{2}{5}, \frac{3}{25}, \frac{7}{25}\right\}$

■ Example

Let's compute some 2-dimensional scrambled Faure vectors and plot them. First I clear the old constants and functions defined before:

```
ClearAll[dim, p, max, uppersize, pascal, u, matrix, genmatrix, sfaure]
```

The dimension is 2, so the prime base is 2. I need at most 1000 vectors, so I define:

In[33]:=
```
dim = 2; p = 2; max = 1000;
```

In[34]:=
```
uppersize = Floor[Log[p, max] + 1]
```

Out[34]= 10

In[35]:=
```
pascal[1] := IdentityMatrix[uppersize];
pascal[h_] := Table[Mod[Binomial[j - 1, i - 1] * (h - 1)^(j - i), p],
  {i, uppersize}, {j, uppersize}];
```

In[37]:=
```
u[i_, j_] := Which[i < j, 0, i == j, RandomInteger[{1, p - 1}], i > j, RandomInteger[{0, p - 1}]];

matrix[k_] := matrix[k] = SparseArray[{i_, j_} /; i ≥ j → u[i, j], {uppersize, uppersize}];
```

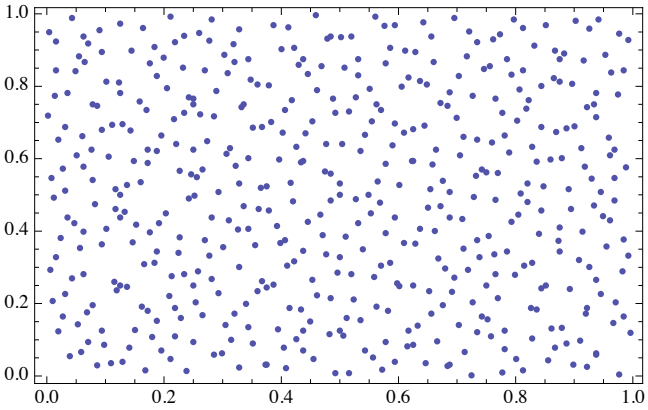
In[39]:=
```
genmatrix[h_] := genmatrix[h] = Mod[matrix[h] . pascal[h], p];
```

In[40]:=
```
sfaure[n_] := Module[{digits, size}, digits = Reverse[IntegerDigits[n, p]];
   size = Length[digits];
   Table[Mod[submatrix[genmatrix[h], size] . digits , p] . p^(-Range[size]), {h, dim}]];
```

Here is the plot of the first 500 two-dimensional scrambled Faure vectors:

In[41]:=
```
ListPlot[Table[sfaure[n], {n, 500}], Frame → True]
```

Out[41]=



Here is a scrambled Faure net, extracted in a similar way to the one in the Introduction.

In[42]:=
```
net = Table[sfaure[i], {i, 4 * 2^3, 5 * 2^3 - 1}]
```

Out[42]= $\left\{\left\{\frac{1}{64}, \frac{27}{32}\right\}, \left\{\frac{47}{64}, \frac{3}{8}\right\}, \left\{\frac{3}{8}, \frac{1}{32}\right\}, \left\{\frac{27}{32}, \frac{11}{16}\right\}, \left\{\frac{3}{16}, \frac{11}{32}\right\}, \left\{\frac{17}{32}, \frac{7}{8}\right\}, \left\{\frac{21}{64}, \frac{17}{32}\right\}, \left\{\frac{59}{64}, \frac{3}{16}\right\}\right\}$

I next put these vectors in a grid. The meaning of a net: every elementary interval of area of 1/8 contains exactly one point of the sequence.

In[43]:=
```
grids = Table[i / 8, {i, 0, 8}]
```

Out[43]= $\left\{0, \frac{1}{8}, \frac{1}{4}, \frac{3}{8}, \frac{1}{2}, \frac{5}{8}, \frac{3}{4}, \frac{7}{8}, 1\right\}$

In[44]:=
```
ListPlot[net, PlotStyle → PointSize[0.02`],
 Frame → True, AxesOrigin → {0, 0}, GridLines → {grids, grids},
 PlotRange → {{0, 1}, {0, 1}}, FrameTicks → {grids, grids, None, None}]
```

Out[44]=