Example 1

Creating and plotting a pattern and its generating set

This example demonstrates, how to compute all points belonging to a pattern $\mathcal{P}(\mathbf{M})$ and how to plot them in different ways.


```
Author:          Ronny Bergmann
Created:         2013-08-15
Last Changed:    2014-12-13
```


License

Loading the Library

The MPAWL is located in the parent directory (see `MPAWL.m`) in order to load the library, we add its path to **$Path**.

In[1]:=
```
$Path = Join[$Path, {ParentDirectory[NotebookDirectory[]]}];
SetDirectory[NotebookDirectory[]];(*Set to actual directory*)
Needs["MPAWL`"];
```

z::shdw : Symbol z appears in multiple contexts {MPAWL`Visualization`, Global`}; definitions in context MPAWL`Visualization` may shadow or be shadowed by other definitions. ≫

Looking at a pattern

For a given matrix, e.g.

In[4]:=
```
mM = {{32, 4}, {-1, 8}}; MatrixForm[mM]
```

Out[4]//MatrixForm=

$$\begin{pmatrix} 32 & 4 \\ -1 & 8 \end{pmatrix}$$

we obtain the pattern by using the function

In[5]:=
```
? pattern
```

pattern[mM]

generates the set of points inside the unit cube, whose multiplication with mM results in an integral vector. The matrix mM must be in pattern normal form for this fast algorithm to work, see **getPatternNromalform**.

Options

Target → *"Unit"* | "Symmetric"

target domain of the modulus, either the unit cube or the unit cube shifted by −1/2.

validateMatrix → *True* | False

whether to perform a check (via **isMatrixValid**) on the matrix mM.

In[6]:=
```
? getPatternNormalform
```

getPatternNormalform[mM]

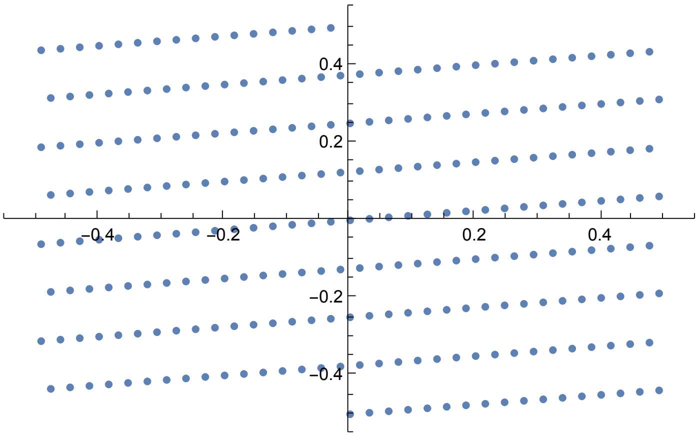
Return the normal form of the matrix mM, that generates the same pattern, i.e. the corresponding matrix is in upper triangular form and the dominant value is on the diagonal.

Options

validateMatrix → *True* | False
whether to perform a check (via **isMatrixValid**) on the matrix mM.

In[7]:=
```
patPts = pattern[getPatternNormalform[mM], Target → "Symmetric"];
```

In[8]:=
```
ListPlot[patPts,
 PlotRange → {{-1 / 2, 1 / 2}, {-1 / 2, 1 / 2}}, PlotRangePadding → 0.05]
```

Out[8]=


The rank of this lattice is given by

In[9]:=
```
patternDimension[mM]
```

Out[9]= 1

, i.e. we only need one vector, that spans these points by its integral scales, this vector is e.g.

In[10]:=
```
{v} = patternBasis[mM]
```

Out[10]= $\left\{\left\{\frac{2}{65}, \frac{1}{260}\right\}\right\}$

The Smith form also provides the set of scalars, a*v, a=0,...,ϵ, such that these are the pattern (when using modulo 1 or the shifted modulo operation for the symmetric case)

In[11]:=
```
IntegerSmithForm[mM, ExtendedForm → False] // MatrixForm
```

Out[11]//MatrixForm=
$$\begin{pmatrix} 1 & 0 \\ 0 & 260 \end{pmatrix}$$

In[12]:=
```
ϵ = IntegerSmithForm[mM, ExtendedForm → False][[2, 2]]
```

Out[12]= 260

So the pattern can also be given by using **modM[]** and this one vector, i.e.

In[13]:=
```
? modM
```

modM[k, mM]

calculates the modulus of k with respect to the matrix mM, i.e. the vector h in the unit cube such that k = h + mM*z.

Options

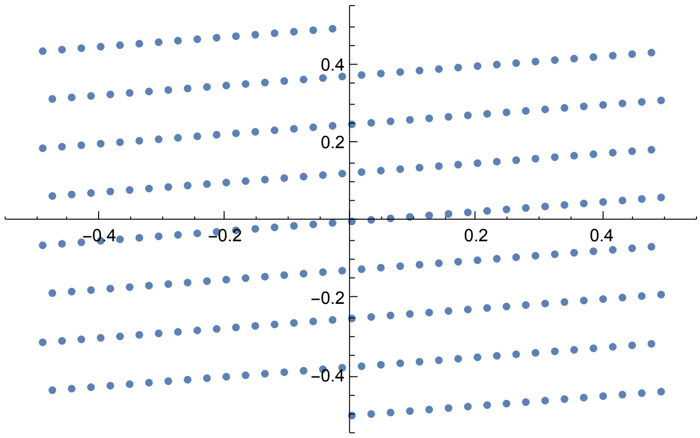
Target → *"Unit"* | "Symmetric"

target domain of the modulus, either the unit cube or the unit cube shifted by $-\frac{1}{2}$.

validateMatrix → *True* | False

whether to perform a check (via **isMatrixValid**) on the matrix mM.

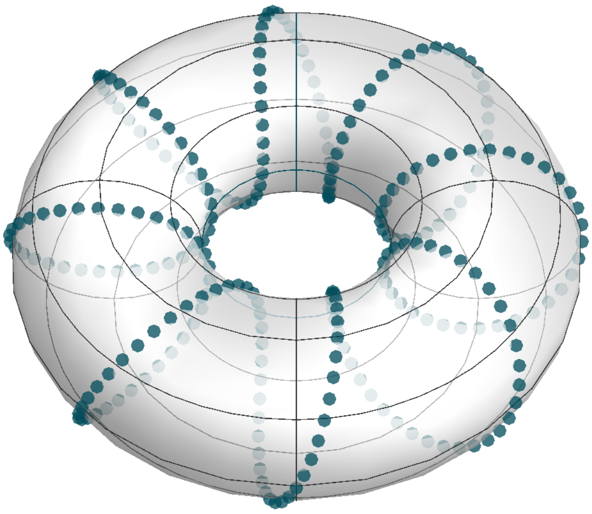
In[14]:=
```
ListPlot[
 Table[modM[k * v, IdentityMatrix[2], Target → "Symmetric"], {k, 0, ϵ - 1} ],
 PlotRange → {{-1 / 2, 1 / 2}, {-1 / 2, 1 / 2}}, PlotRangePadding → 0.05]
```

Out[14]=


Finally, another possibility to look at a pattern is, to plot it on a torus. This emphasizes the fact, that all additions in the application are seen modulo 1, which is the same as "running around" on the torus.

In[15]:= `plotOnTorus[mM]`

Out[15]=



The corresponding generating set

The generating set the MPAW Library work with, is the one of **Transpose[mM]**. We can also plot this, but now we really need the usual matrix (not its pattern normal form) to obtain the generating set

In[16]:= `? generatingSet`

generatingSet[mM]

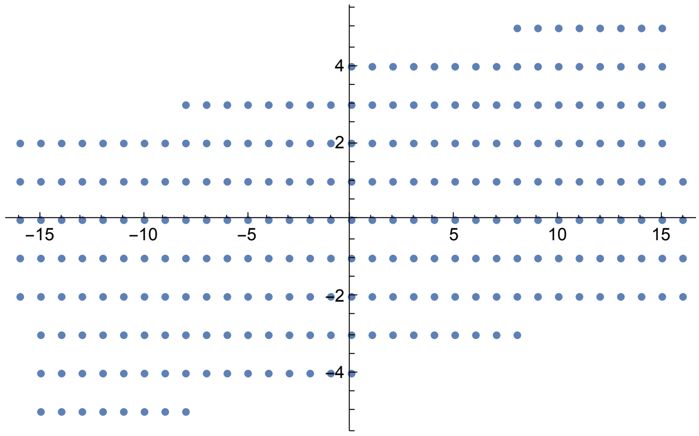
Returns the set of integer vectors originating from the corresponding pattern[mM] by multiplying each element with mM.

Options

Target → *"Unit"* | "Symmetric"
target domain of the modulus, either the unit cube or the unit cube shifted by −1/2.

validateMatrix → *True* | False
whether to perform a check (via **isMatrixValid**) on the matrix mM.

In[17]:=
```
ListPlot[generatingSet[Transpose[mM], Target → "Symmetric"]]
```

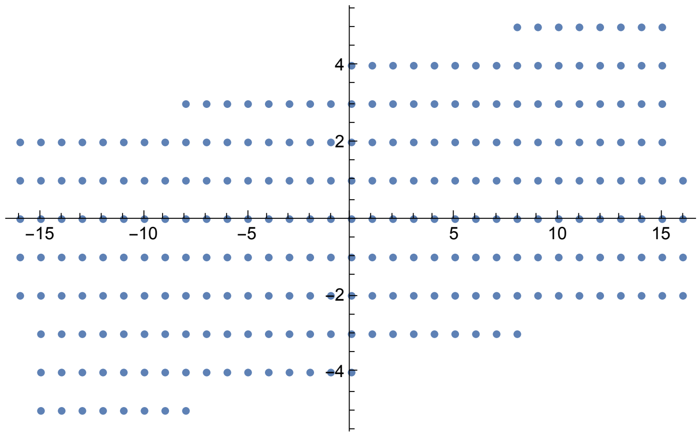
Out[17]=


Here, this set is also given as the scalar multiplicities of the same number of vectors as for the pattern above, i.e. the one vector

In[18]:=
```
{g} = generatingSetBasis[Transpose[mM]]
```

Out[18]= {{0, 1}}

In[19]:=
```
ListPlot[Table[modM[k * g, Transpose[mM], Target → "Symmetric"], {k, 0, ϵ - 1}]]
```

Out[19]=


And we further can decompose an element of the generating set into its basis vector fractions, which is here obtaining the corresponding **k**:

In[20]:=
```
? generatingSetBasisDecomp
```

generatingSetBasisDecomp[k,mM]

For the standard Basis of the Generating Set provided by generatingSetBasis[mM] the (integer) Coefficients, that reconstruct x from the basis (up to equivalence with respect to mod[k,mM].

Options

Target → *"Unit"* | "Symmetric"
target domain of the modulus, either the unit cube or the unit cube shifted by −1/2.

validateMatrix → *True* | False
whether to perform a check (via **isMatrixValid**) on the matrix mM.

```
In[21]:= generatingSetBasisDecomp[{3, 3}, Transpose[mM], Target → "Symmetric"]
Out[21]= {27}

In[22]:= modM[27 * g, Transpose[mM], Target → "Symmetric"]
Out[22]= {3, 3}
```