

## Example 2

# The Fourier transform

This example demonstrates, that the Fourier transform can be used for two different kinds of data sets.

---

|               |                |
|---------------|----------------|
| 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`"];
```

## The data set as a matrix

Let's look at a matrix having more than one cycle (in contrast to the matrix from Example 1). For

```
In[4]:= mM = {{16, 4}, {0, 16}}; MatrixForm[mM]  
Out[4]/MatrixForm=  

$$\begin{pmatrix} 16 & 4 \\ 0 & 16 \end{pmatrix}$$

```

We have

```
In[5]:= patternDimension[mM]
```

```
Out[5]= 2
```

and

```
In[6]:= {v1, v2} = patternBasis[mM]
```

```
Out[6]= {{0,  $\frac{1}{4}$ }, { $\frac{1}{64}$ ,  $-\frac{1}{16}$ }}
```

where the elementary divisors are

```
In[7]:= {e1, e2} = Diagonal[IntegerSmithForm[mM, ExtendedForm → False]]
```

```
Out[7]= {4, 64}
```

and hence



```
In[11]:= ? FourierTransformTorus
```

FourierTransformTorus[mM, b]

Perform the Fourier transform on the pattern with respect to mM. b is either a vector of length  $m = \text{Abs}[\text{Det}[\mathbf{mM}]]$  or addressing the points with respect to the basis of the pattern, i.e. the cycles having the length of the elementary divisors.

## Options

Validate → *True* | *False*

whether to perform a check (via **isMatrixValid**) on the matrix mM and the check, whether the Origin is in Range.

Compute  $\rightarrow$  “*Numeric*” | “Exact”

Providing numerical data, the Fourier method is used to perform the transform using FFT techniques. If all entries of  $\mathbf{mM}$  and  $\mathbf{b}$  are given exact, the “Exact” computation can be used to obtain the exact transform

```
In[12]:= hatb = FourierTransformTorus[mM, b]
```

[illegible]

which can also be switched to exact computations



```
In[16]:= FourierTransformTorus[mM, hatb, Compute  $\rightarrow$  "Exact"]
```

```
Out[16]= { {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
            0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
            0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},  
          {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
            0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
            0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},  
          {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
            0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},  
          {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
            0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},  
          {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
            0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
```

But we could also reshape the data to be a vector by ordering the indices  $\{0, 0\}, \dots, \{3, 63\}$  lexicographically, i.e.  $\{0, 0\}, \{0, 1\}, \dots, \{1, 0\}, \{1, 1\}, \dots, \{3, 62\}, \{3, 63\}$ . For details see [1].

```
In[17]:= ? reshapeData
```

```
reshapeData[M,data,direction]
```

Perform a reshape of data, where direction denotes

True: From vector to matrix

False: The other way around

## Options

Validate → *True* | *False*

whether to perform a check (via **isMatrixValid**) on the matrix mM and the check, whether the Origin is in Range.

```
In[18]:= b2 = reshapeData[mM, b, False]
```

[illegible]

```
In[19]:= hatb2 = FourierTransformTorus[mM, b2]
```

[illegible]

again using the above ordering of the indices but now with respect to the generating set of **Trans'**: **pose[mM]**. Of course the inverse transform yields again

```
In[20]:= FourierTransformTorus[mM, hatb2]
```

[illegible]

but of course this works if and only if the ordering of the elements in the vector is as just explained. Notice that while the input shape is analyzed automatically, i.e. whether its a vector or a matrix of data, the usual computation is always done numerically and the exact computation must be activated by the option `Compute->` . Also the exact computation yields `$Failed` if any numeric number is given as does the numeric one if any element is not a number.

## Literature

- [1] R.Bergmann, *The fast Fourier transform and fast wavelet transform for patterns on the torus*,

Appl. Comp. Harmon. Anal. 35 (2013) 39–51, doi: 10.1016/j.acha.2012.07.007.