Example 6

Wavelet Decomposition

For a set of dilation matrices this demo illustrates nested spaces build by scaling functions of de la Vallée Poussin type.


```
Author:          Ronny Bergmann
Created:         16.08.2013
Last Changed:    16.08.2013
```


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`"];
```

We first use the already known Dirichlet kernels to decompose a sampled image of a certain Box spline. Then we also decompose the Box spline with the de la Vallée Poussin means in order to derive a more localised picture of the edged we want to detect. The examples are taken from chapter 4 in [1] and more details to the wavelet transform and its complexity can be found in [2].

Sample the Box spline

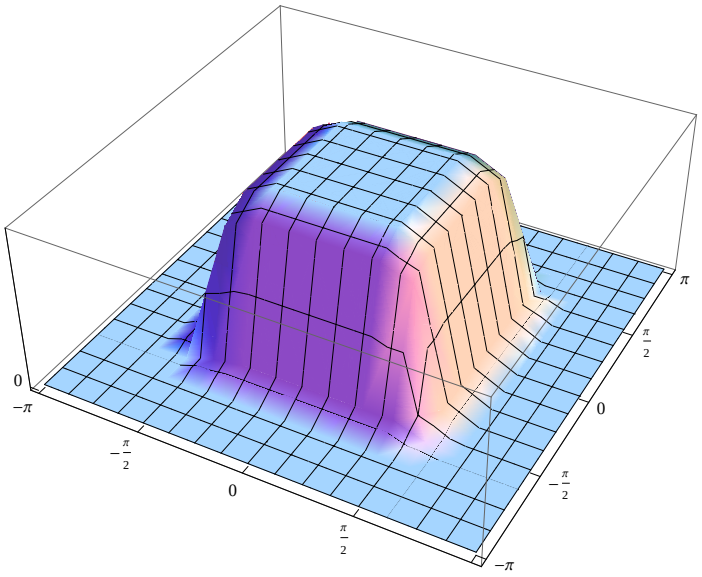
We take the Box spline from Example 4 but introduce a new vizualization of the Fourier series

In[4]:=
```
mΞ = π {{1, 0}, {0, 1}, {1 / 8, 0}, {0, 1 / 8}, {1 / 8, -1 / 8}};
ct = Sum[x / 2,  {x, mΞ}];
```

In[6]:=
```
f[X_] := evaluateBoxSpline[N[mΞ], N[X + ct]]
```

In[7]:=
```
Plot3D[f[{x1, x2}], {x1, -π, π}, {x2, -π, π},
 Ticks → {{-π, -π / 2, 0, π / 2, {π, ""}}, {-π, -π / 2, 0, π / 2, π}, {0, 1 / 2, 1}}]
```

Out[7]=



In[8]:=
```
mM = {{128, 0}, {0, 128}};
```

By this size (16384 points) the sampling takes a while, hence loading the samples is recommended.

In[9]:=
```
? sampleFunction
```

sampleFunction[mM, f]

Sample the function f on the pattern of mM on the 2π–periodic torus.

Options

Debug → “None” | “Text” | “Time” | “Text&Time”
activate text output, either just text, timings or both

File → None
to specify a File where the resulting coefficients are stored and may be loaded the same way again.

Method → “Point” | “Point set”
How to perform the sampling, where usually the sampling is performed by calling f for each point individually. The other method creates a set X of all points and performs f[X], assuming, f can handle a set of points and returns a set of function values.

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

In[10]:=
```
data =
  sampleFunction[mM, f, Debug → "Text&Time", File → "example6/sampling-f.dat"];
```

```
Loading data from file "example6/sampling-f.dat"...

failed. Proceed with sampling...

Sampling the function...

Sampling took 1324.651684 seconds.

Saving data to file "example6/sampling-f.dat".
```

We will also use the save option (File ->) for the coefficients of the Dirichlet kernel and its bracket sums

The Dirichlet kernel

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

DirichletKernel[mM]

provides a dirichlet kernel, which is a special case of the de la Vallé e Poussin mean, where g=0, hence pyramidalFunction[d,0] is used. Here, the same options as for the de la Vallee Poussin mean apply.

In[12]:=
```
{ckD, DBS} = DirichletKernel[mM,
   File → {"example6/DirichletKernelM.dat", "example6/DirichletKernelBS.dat"},
   BracketSums → True, Debug → "Text", Orthonormalize → True];
```

```
Loading de la Vallée Poussin mean from file "example6/DirichletKernelM.dat"...

failed.

Computing de la Vallée Poussin coefficients...

Orthonormalization...

Loading Bracket sums from file "example6/DirichletKernelBS.dat"...

failed. Computing Bracket sums to obtain the basis transform coefficients...
```

In[13]:=
```
originD = (Dimensions[ckD] + 1) / 2; maxD = originD - 1;
```

In[14]:=
```
? changeBasis
```

changeBasis[mM,Coeffs, bracketSums]

Perform a change of Basis on the coeffs, that are the DFT of some samples and bracketSums represenst the Bracket sums of a certain function, i.e. all are nonzero such that the lagrange function exists.

Options

Input → “Frequency” | “Time”
Domain of the discrete Input set. If “Time” is given, a Fourier transform is performed before the change of basis

Output → “Frequency” | “Time”
Domain of the discrete Output set. If “Time” is given, a Fourier transform is performed after the change of basis

Validate → True | False
whether to perform a check (via isMatrixValid[mM]) on the matrix mM and the check, whether the Origin is in Range.

Debug → “None” | “Text” | “Time” | “Text&Time”
activate text output, either just text, timings or both.

In[15]:=
```
cdata = changeBasis[mM, data, DBS, Debug → "Text&Time", Input → "Time"];
```

```
The Fourier transform of the input took 0.044535 seconds.
The change of basis took 0.014605 seconds.
```

In[16]:=
```
? getFourierFromSpace
```

getFourierFromSpace[coefficients, ckSpace, originIndex, mM]

The coefficients represent the Fourier transform of the weights which applied to the translates −−− with respect to mM − of a function φ result in a function f and φ with its Fourier coefficients ckSpace (where originIndex is the index representing the origin), this function reconstructs the Fourier coefficients of f.

Options

Validate → True | False
whether to perform a check (via isMatrixValid[mM]) on the matrix mM and the check, whether the Origin is in Range of the Fourier coefficients.

In[17]:=
```
ckdata = getFourierFromSpace[cdata, ckD, originD, mM];
```

Which can (despite using functions and Plot3D also be visualized as an image (where red indicates positive, blue the negative values)

In[18]:=
```
? discretePlotFourierSeries
```

discretePlotFourierSeries[resolution,coefficients,origin]

For an array of Fourier coefficients, where origin represents the index of the origin, an Image of size resolution is produced, showing the function either in color (where red indicated positive and blue negative values), grayscale or absolute value.

Options

ReturnVal → "AbsoluteImage" | "Image" | "ColorImage"
Specify, which kind of Image should be produced
Debug → "None" | "Text" | "Time" | "Image"
or any combination of these Words in one String (i.e. concatenated via "&") to produce intermediate results, indicate progress and display computation times.
ColorLegend → False | True
display a range of values and their corresponding colors to the right of the plot.

The Image is included in a plot, and hence accepts all its options, providing some non−standard Options as new standards. The same holds for the createBarLegend and hence BarLegend the ColorLegend option consists of.

In order to get the scaling - we have to compensate the (smaller) FOurier transform with respect to mM by multiplying with its Determinant

In[19]:=
```
discretePlotFourierSeries[{512, 512}, Sqrt[Abs[Det[mM]]] * ckdata, originD,
 Frame → False, Axes → False, Debug → "Text&Time", ReturnVal → "ColorImage"]
```

```
Padding Data...
Fourier transforming...
Shifting back...
Creating Color Image...
The range of Values is from -2.80441×10^-5 to 1.01377×10^-1.
```

Out[19]=

The Wavelet tranfsorm

If we now introduce two (dyadic) subspaces

In[20]:= `?delaValleePoussinSubspaces`

"delaValleePoussinSubspaces[g,mM,mJ]\n\nFor any dyadic dilation matrix mJ and a function g fulfilling the properties:\n\n1) nonnegativity\n2) strictly positive on the shifted (symmetric) unit cube\n3) For every x the sum over all integer shifts g(x+z) equals 1.\n\nthis method divides the mM–invariant space V (having dimension m=|Det[mM]) into\ntwo orthogonal subspaces by returning two coefficient arrays, whose Fourier\ntransforms are weights to the sum of translates of any function, that spans V\nwith its translates. Both functions provide linear independence with respect to\nmN = Inverse[mJ].mM\n\nFor simplicity g might be given as a nonnegative number $t \le \frac{1}{2}$ which corresponds to\nthe pyramidal (tensor product of 1D de la Vallée Poussin means) function, where the\nsupport is $-\frac{1}{2}-t$ to $\frac{1}{2}+t$ in each dimension.\n\n!\(*StyleBox[\"Options\",FontWeight→\"Bold\"]\)\n\nOrthonormalize → \!\(*StyleBox[\"True\",\nFontSlant→\"Italic\"]\) | False\n Perform an orthonormalization of the translates of both functions with\n respect to mN.\nValidate → \!\(*StyleBox[\"True\",\nFontSlant→\"Italic\"]\) | False\n whether to perform a check (via isMatrixValid[mM]) on the matrix mM, mJ\n and mN.\nFile → \!\(*StyleBox[\"None\",\nFontSlant→\"Italic\"]\) | {String,String}\n save the coefficients of both functions to a file each.\nDebug → \!\(*StyleBox[\"“None”\",\nFontSlant→\"Italic\"]\) | “Text” | “Time”\n or any combination of these Words in one String (i.e. concatenated via “&”)\n to produce intermediate results, indicate progress and display computation\n times."

In[21]:=
```
{coeffsDXS, coeffsDXW} = DirichletKernelSubspaces[mM,
   dilationMatrix2D["X"], Debug → "Text&Time", Orthonormalize → True];
```

```
Computing the scaling function coefficients...

Scaling function coefficients computed in 28.195969 seconds.

Computing the wavelet function coefficients...

Wavelet function coefficients computed in 2.160853 seconds.

Orthonormalizing coefficients...

Orthonormalization took 7.020041 seconds.

Orthonormalizing coefficients...

Orthonormalization took 7.356178 seconds.
```

We can now decompose with a Wavelet transform

In[22]:=
```
? WaveletTransformTorus
```

WaveletTransformTorus[mM, mJ, data, φNCoeffs, ψNCoeffs]
WaveletTransformTorus[mM, mJ, ckData, ckφM, ckφN, ckψN, originIndex]

For a decomposition of mM = mJ∗mN, i.e. mN is integral and all three are regular,
and |Det[mJ]| = 2] this method performs a wavelet decomposition of the data.

On the one hand, data and both subspaces are given as coefficients of the space
that is decomposed. Then the result consists of two arrays {dataS, dataW}
representing the two parts of data as coefficients of their corresponding spaces.

On the other hand, if data and all three spaces are given as Fourier
coefficients, the result are the two parts again as Fourier coefficients. This
second method is based on the first and hence slower.

Options

Debug → "None" | "Text" | "Time" | "Text&Time"
activate text output, either just text, timings or both

Validate → True | False
whether to perform a check (via isMatrixValid[mM]) on the matrix mM
and the check, whether the Origin is in Range.

In[23]:=
```
{dataXS, dataXW} = WaveletTransformTorus[mM,
   dilationMatrix2D["X"], cdata, coeffsDXS, coeffsDXW, Debug → "Text&Time"];
```

```
Performing the Wavelet transform...

Performing the Wavelet transform took 7.415260 seconds.
```

But we only look at the Wavelet part now, again by reconstructing and creating the image, so we first need their Fourier coefficients

In[24]:=
```
ckDXW = getFourierFromSpace[coeffsDXW, ckD, originD, mM];
```

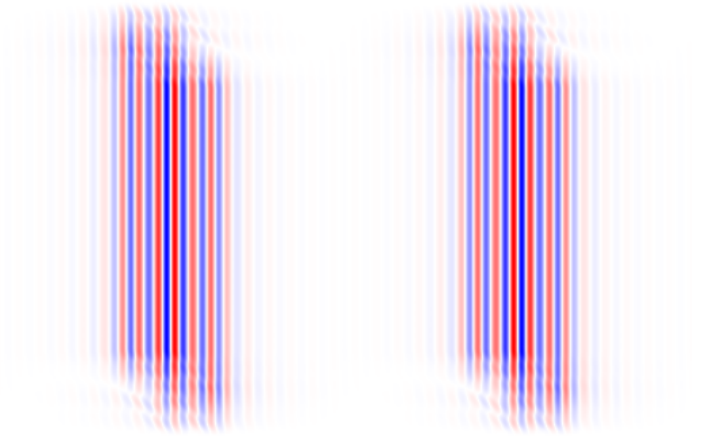
and can then recreate the Fourier coefficients of the data, which is in the space of this function. The origin is the same, *but* the matrix of course, is now N

In[25]:=
```
ckDataXW = getFourierFromSpace[dataXW,
   ckDXW, originD, Inverse[dilationMatrix2D["X"]].mM];
```

In[26]:=
```
discretePlotFourierSeries[{512, 512}, Sqrt[128 * 64] * ckDataXW, originD,
 Frame → False, Axes → False, Debug → "Text&Time", ReturnVal → "ColorImage"]
```

```
Padding Data...

Fourier transforming...

Shifting back...

Creating Color Image...

The range of Values is from -1.16764 × 10^-4 to 1.16764 × 10^-4.
```

Out[26]=



And we can also perform that e.g. for the Quincux matrix

In[27]:=
```
{coeffsDDS, coeffsDDW} =
  DirichletKernelSubspaces[mM, dilationMatrix2D["D"], Orthonormalize → True];
```

In[28]:=
```
{dataDS, dataDW} = WaveletTransformTorus[mM,
   dilationMatrix2D["D"], cdata, coeffsDDS, coeffsDDW, Debug → "Text&Time"];
```

```
Performing the Wavelet transform...

Performing the Wavelet transform took 7.616307 seconds.
```

In[29]:=
```
ckDDW = getFourierFromSpace[coeffsDDW, ckD, originD, mM];
```

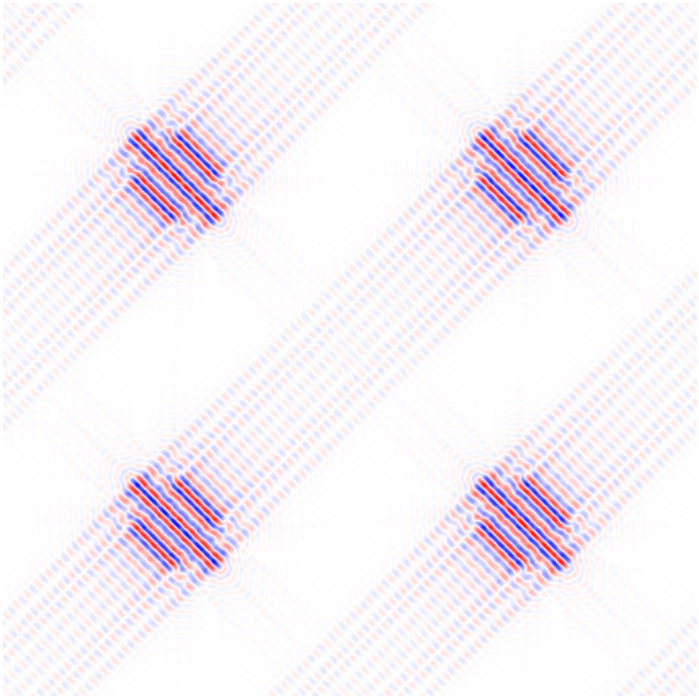
In[30]:=
```
ckDataDW = getFourierFromSpace[dataDW,
   ckDDW, originD, Inverse[dilationMatrix2D["D"]].mM];
```

In[31]:=
```
discretePlotFourierSeries[{512, 512}, Sqrt[128 * 64] * ckDataDW, originD,
 Frame → False, Axes → False, Debug → "Text&Time", ReturnVal → "ColorImage"]
```

```
Padding Data...
Fourier transforming...
Shifting back...
Creating Color Image...
```

The range of Values is from -5.65499×10^{-6} to 5.65499×10^{-6}.

Out[31]=



The de la Vallée Poussin case

In[106]:=
```
?delaValleePoussinMean
```

delaValleePoussinMean[g,mM]

Generate the de la Vallée Poussin Kernel φ_M^{ϕ} in Fourier coefficients
based on the function g and the matrix mM. While mM is an integral regular
matrix of dimension d∗d, the function g has to fullfill the three properties:

1) nonnegativity
2) strictly positive on the shifted (symmetric) unit cube
3) For every x the sum over all integer shifts g(x+z) equals 1.

For simplicity g might be given as a nonnegative number $t \le \frac{1}{2}$ which corresponds to
the pyramidal (tensor product of 1D de la Vallée Poussin means) function, where the
support is $-\frac{1}{2}-t$ to $\frac{1}{2}+t$ in each dimension.
Or an array of such numbers, performing the same idea with different width in each direction,
hence the array must be the same dimension as each dimension of mM.

Options

BacketSums → False | True
: Compute the Bracket sums and return an array {ckV, BSV} consisting of the
Fourier coefficients ckV and the Bracket Sums BSV of the kernel.

Orthonormalize → True | False
: Perform an orthonormalization of the translates of the kernel with respect
to mM.

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

File → None | String | {String,String}
: save the Fourier coefficients of the kernel to a file. If the Bracket sums
are computed too, there have to be two string, the first representing the
kernel, the second the Bracket sums to be saved.

Support → $\frac{1}{2}$ | p
: specifies the support of g, if it is given as a function, to extend the shifted unit cube
by p in each dimension. This can also be given as a d−dimensional vector, where each entry
is nonnegative and smaller than $\frac{1}{2}$.

Debug → “None” | “Text” | “Time”
: or any combination of these Words in one String (i.e. concatenated via “&”)
to produce intermediate results, indicate progress and display computation
times.

In[33]:=
```
{ckdlVP, dlVPBS} = delaValleePoussinMean[1 / 14, mM,
   File → {"example6/dlVPKernel.dat", "example6/dlVPKernelBS.dat"},
   BracketSums → True, Debug → "Text", Orthonormalize → True];
```

```
Loading de la Vallée Poussin mean from file "example6/dlVPKernel.dat"...
failed.
Computing de la Vallée Poussin coefficients...
Orthonormalization...
Loading Bracket sums from file "example6/dlVPKernelBS.dat"...
failed. Computing Bracket sums to obtain the basis transform coefficients...
```

In[34]:=
```
cdata2 = changeBasis[mM, data, dlVPBS, Debug → "Text&Time", Input → "Time"];
The Fourier transform of the input took 0.029104 seconds.
The change of basis took 0.020563 seconds.
```

Then, the previous commands can also ber performed on multiple levels by using

In[35]:= **? decomposeData2D**

decomposeData2D[g(Vec), JSet(Vec), mM, data]
decomposeData2D[l,g,JSet, mM, data, ckφM]

Decompose the data given as coefficients with respect to ckφφ this
method decomposes the data on different dialtionPaths given by JSet(Vec),
where the corresponding de la Vallee Poussin means (and wavelets) are given by
g(Vec) (see delaValleePoussinMean for restrictions on g). The dilation matrices
are given in terms of the characters used by dilationMatrix2D.

Both g and JSet might be given as vectors. If both are vectors, gVec has to be
one element longer than JSetVec. If one is given as a vector, the other is
expanded to a vector of corresponding length, having constant entries. This
length determines the number of levels of the wavelet decomposition, while g
also specifies additionally the starting space of mM.

If none of them is a vector, the second definition is needed to specify a level
depth of decomposition.

Options

ImagePrefix → "Img" | String
prefix that is used to save the images; the path in terms of the matrices
is appended. This String may also contain Folders, that are relative to the
Notebook.
ImageSuffix → ".png" | String
suffix to determine the file format of the images.
DataPrefix → "Data" | String
prefix that is used to save the data files; here an S or a W for the two
subspaces is added and the decomposition path in terms of the matrices is
appended. This String may also contain Folders, that are relative to the
Notebook.

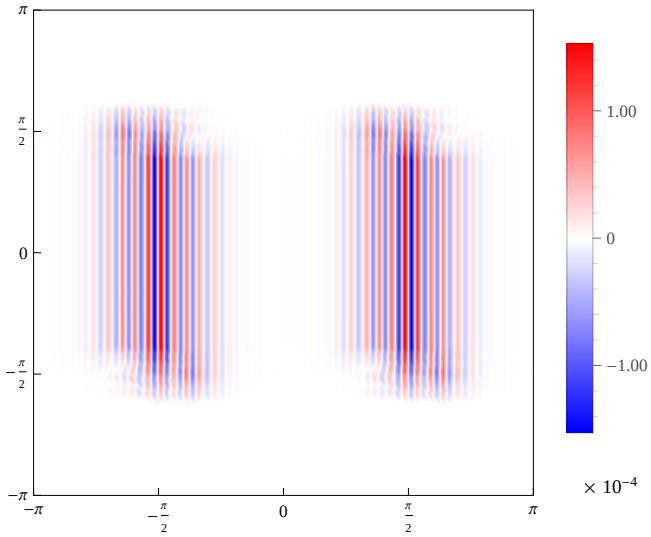
Setting any of those String to an empty String disables the Image or Coefficient
savings.

PlotResolution → 64 | nonnegative Integer
resolution of the resulting Image generated by discreteFourierSeries
computeWavelet –> → True | False
whether to compute the wavelet part of the specific level and produce it's image
computeScale → False | True
whether to compute the scale part of the specific level and produce it's image
Validate → True | False
whether to perform a check (via isMatrixValid[mM]) on the matrix mM and mJ(s)
Validity of data. If some matrices mN in the decomposition are not valid,
the algorithm continues on the other leaves.
Debug → "None" | "Text" | "Time" | "Image"
or any combination of these Words in one String (i.e. concatenated via "&")
to produce intermediate results, indicate progress and display computation
times.

and any Option of discretePlotFourierSeries (including Plot and BarLegend), that
are passed on and any Option of Export (especially Resolution but not ImageSize)
Show (especially ImageSize). By the last two, Fonts are scaled, but a pixel size
of the image is not that easy to be computed.

In[36]:=
```
decomposeData2D[1 / 8, {{"X"}}, mM, cdata2, ImagePrefix → "",
 ImageSuffix → "", DataPrefix → "", Debug → "Text&Time&Image"]
```

```
Computing de la Vallée Poussin coefficients...

Creating the de La Vallée Poussin kernel took 2.547396 seconds.

Orthonormalization...

Orthonormalization took 20.890320 seconds.

Decompose ( 128  0 )  with X
          (  0  128 )

Computing the scaling function coefficients...

Scaling function coefficients computed in 29.734126 seconds.

Computing the wavelet function coefficients...

Wavelet function coefficients computed in 2.140414 seconds.

Orthonormalizing coefficients...

Orthonormalization took 7.098800 seconds.

Orthonormalizing coefficients...

Orthonormalization took 7.441582 seconds.

Performing the Wavelet transform...

Performing the Wavelet transform took 7.484827 seconds.

Padding Data...

Fourier transforming...

Shifting back...

Creating Color Image...

The range of Values is from -1.53182 × 10^-4 to 1.53182 × 10^-4.
```



For the small lines we decompose multiple level, where the first one shows, that the isolation in the Dirichlet case is a small technical artefact.

In[37]:=
```
decomposeData2D[1 / 14, {{"D"}, {"Y"}, {"Y"}, {"Y"}, {"X"}}, mM, cdata2,
 ImagePrefix → "", ImageSuffix → "", DataPrefix → "example6/dlVP",
 Debug → "Text&Time&Image", ColorLegend → False]
```

```
Loading de la Vallée Poussin mean from file "example6/dlVPckS.dat"...
```

failed.

Computing de la Vallée Poussin coefficients...

Creating the de La Vallée Poussin kernel took 2.078787 seconds.

Orthonormalization...

Orthonormalization took 18.812911 seconds.

Decompose $\begin{pmatrix} 128 & 0 \\ 0 & 128 \end{pmatrix}$ with D

Loading subspace coefficients from "
example6/dlVPD-S.dat" and "example6/dlVPD-W.dat"...

failed. Compute from scratch.

Computing the scaling function coefficients...

Scaling function coefficients computed in 29.769737 seconds.

Computing the wavelet function coefficients...

Wavelet function coefficients computed in 2.137466 seconds.

Orthonormalizing coefficients...

Orthonormalization took 7.243729 seconds.

Orthonormalizing coefficients...

Orthonormalization took 7.672707 seconds.

Performing the Wavelet transform...

Performing the Wavelet transform took 7.884330 seconds.

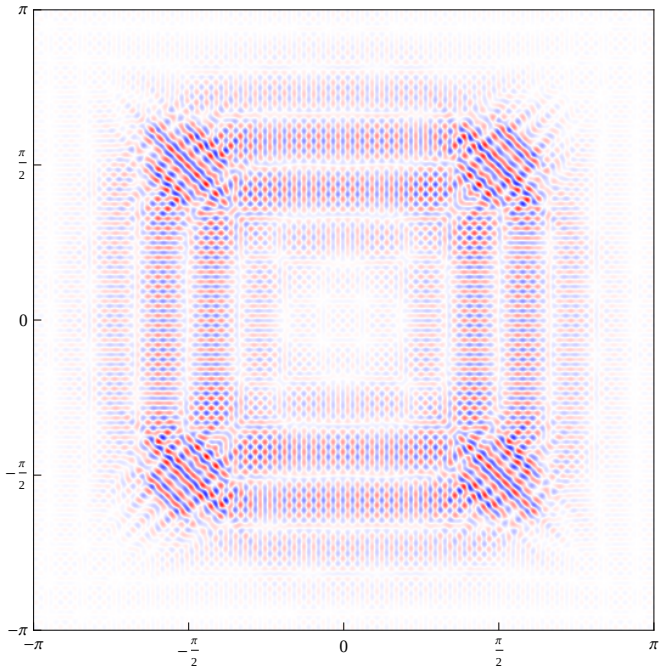
Padding Data...

Fourier transforming...

Shifting back...

Creating Color Image...

The range of Values is from -6.75981×10^{-6} to 8.02176×10^{-6}.



Decompose $\begin{pmatrix} 64 & 64 \\ -64 & 64 \end{pmatrix}$ (D) with Y

Loading subspace coefficients from "
example6/dlVPDY-S.dat" and "example6/dlVPDY-W.dat"...

failed. Compute from scratch.

Computing the scaling function coefficients...

Scaling function coefficients computed in 14.840464 seconds.

Computing the wavelet function coefficients...

Wavelet function coefficients computed in 1.069546 seconds.

Orthonormalizing coefficients...

Orthonormalization took 3.477588 seconds.

Orthonormalizing coefficients...

Orthonormalization took 3.704940 seconds.

Performing the Wavelet transform...

Performing the Wavelet transform took 3.888510 seconds.

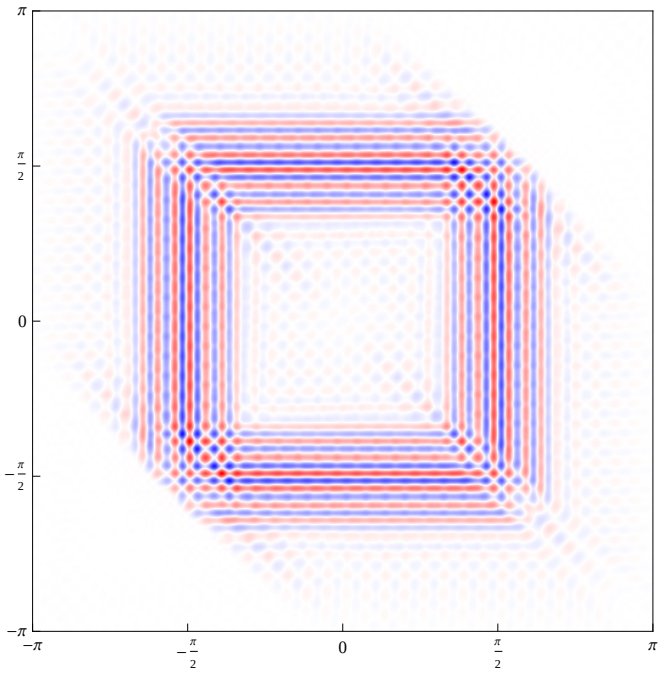
Padding Data...

Fourier transforming...

Shifting back...

Creating Color Image...

The range of Values is from -1.68114×10^{-4} to 1.81674×10^{-4}.



Decompose $\begin{pmatrix} 64 & 64 \\ -32 & 32 \end{pmatrix}$ (DY) with Y

Loading subspace coefficients from "
example6/dlVPDYY-S.dat" and "example6/dlVPDYY-W.dat"...

failed. Compute from scratch.

Computing the scaling function coefficients...

Scaling function coefficients computed in 7.494190 seconds.

Computing the wavelet function coefficients...

Wavelet function coefficients computed in 0.536572 seconds.

Orthonormalizing coefficients...

Orthonormalization took 1.731466 seconds.

Orthonormalizing coefficients...

Orthonormalization took 1.824763 seconds.

Performing the Wavelet transform...

Performing the Wavelet transform took 1.981914 seconds.

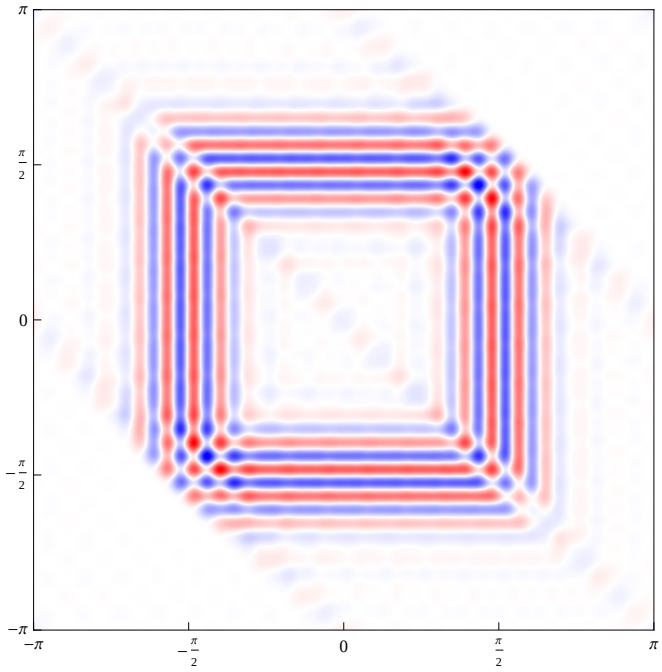
Padding Data...

Fourier transforming...

Shifting back...

Creating Color Image...

The range of Values is from -8.73122×10^{-4} to 8.22183×10^{-4}.



Decompose $\begin{pmatrix} 64 & 64 \\ -16 & 16 \end{pmatrix}$ (DYY) with Y

Loading subspace coefficients from "example6/dlVPDYYY-S.dat" and "example6/dlVPDYYY-W.dat"...

failed. Compute from scratch.

Computing the scaling function coefficients...

Scaling function coefficients computed in 3.722008 seconds.

Computing the wavelet function coefficients...

Wavelet function coefficients computed in 0.270692 seconds.

Orthonormalizing coefficients...

Orthonormalization took 0.866323 seconds.

Orthonormalizing coefficients...

Orthonormalization took 0.897381 seconds.

Performing the Wavelet transform...

Performing the Wavelet transform took 1.056554 seconds.

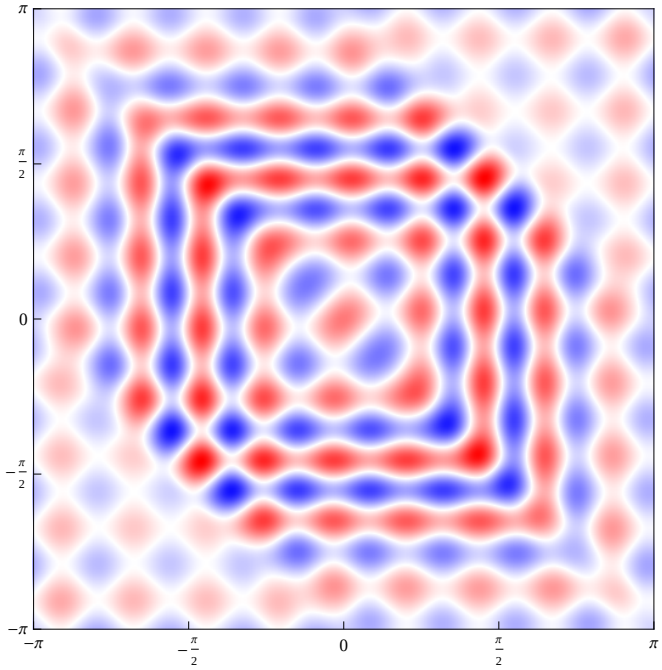
Padding Data...

Fourier transforming...

Shifting back...

Creating Color Image...

The range of Values is from -4.55974×10^{-3} to 4.88993×10^{-3}.



Decompose $\begin{pmatrix} 64 & 64 \\ -8 & 8 \end{pmatrix}$ (DYYY) with X

Loading subspace coefficients from "
example6/dlVPDYYYX-S.dat" and "example6/dlVPDYYYX-W.dat"...

failed. Compute from scratch.

Computing the scaling function coefficients...

Scaling function coefficients computed in 1.847099 seconds.

Computing the wavelet function coefficients...

Wavelet function coefficients computed in 0.136159 seconds.

Orthonormalizing coefficients...

Orthonormalization took 0.433609 seconds.

Orthonormalizing coefficients...

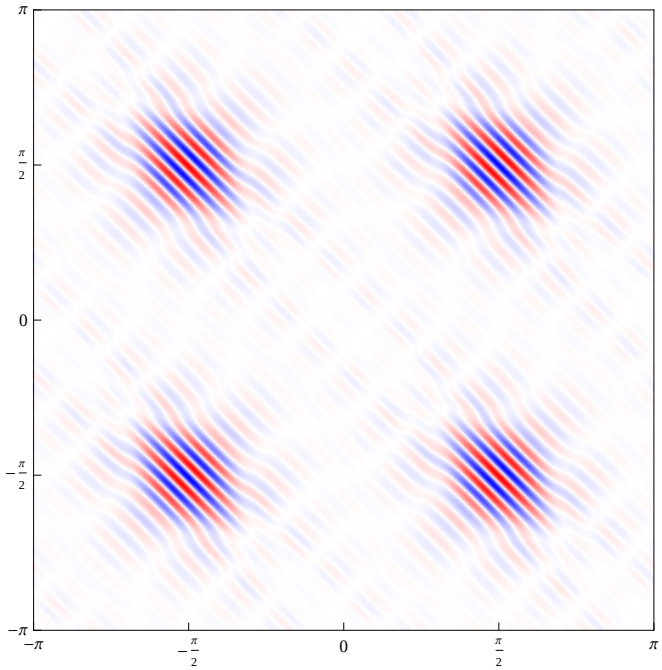
Orthonormalization took 0.457497 seconds.

Performing the Wavelet transform...

Performing the Wavelet transform took 0.574935 seconds.

Padding Data...

```
Fourier transforming...

Shifting back...

Creating Color Image...

The range of Values is from -3.54501 × 10^-5 to 3.54501 × 10^-5.
```



Here, the length of the lines is well reproduces, while the separation is not, but there are also just 512 translates of the last wavelet which is 1/16 of the diagonal case in the first decomposition step.

Literature

[1] Bergmann, R., *Translationsinvariante Räume multivariater anisotroper Funktionen auf dem Torus*, Dissertation, University of Lübeck, 2013.

[2] 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.