Example 4

Sampling and Interpolation

Taking a linear function and Box Spline, sampling it and projecting into a certain space of translates.

```
 Author:         Ronny Bergmann
Created:         15.08.2013
Last Changed:    15.08.2013
```

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`"];
```

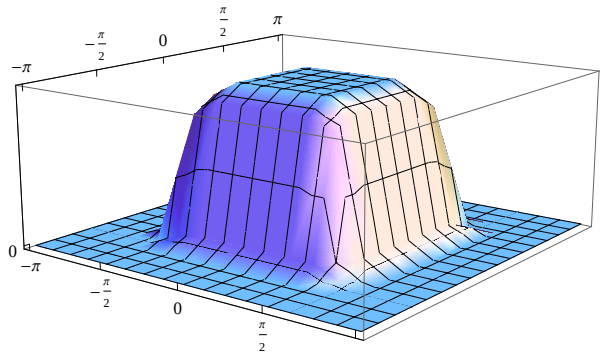
Sampling a (centered) Box spline

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]:=
```
f1[X_] := evaluateBoxSpline[N[mΞ], N[X + ct]]
```

In[8]:=
```
Plot3D[f1[{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[8]=



We will first use sampling, that corresponds to an quadratic image of the Box spline

In[9]:=
```
mM = 64 IdentityMatrix[2]; MatrixForm[mM]
```

Out[9]//MatrixForm=

$$\begin{pmatrix} 64 & 0 \\ 0 & 64 \end{pmatrix}$$

Sampling the function using Sample including the possibility to save the samples to a file and load them from there, if that the samples can be loaded. The Debug can also provide times. Due to the long time, at least for Box splines, the samples are provided.

In[10]:=
```
? 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[11]:=
```
data = sampleFunction[mM, f1,
   Debug → "Text&Time", File → "example4/sampling-f1.dat"];
```

```
Loading data from file "example4/sampling-f1.dat"...
```

Using the Dirichlet - Kernel, we can also obtain the Bracket Sums at the same time by using the Option **BracketSums**.

In[12]:=
```
? 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.

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[13]:=
```
{ckDM, DMBS} = DirichletKernel[mM, BracketSums → True];
```

These coefficients are again symmetric, as in Example 3, and hence the origin is in the center of the (in all dimensions) odd dimension lengths.

In[14]:=
```
origin = (Dimensions[ckDM] + 1) / 2;
```

In[15]:=
```
max = origin - 1;
```

Then, we can perform a change of basis, which is just a multiplication in the discrete Fourier domain. This can be done using **changeBasis[]**, see e.g. Section 2.1 in [1].

In[16]:=
```
? 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[17]:=
```
coeffs = changeBasis[mM, data, DMBS, Debug → "Text&Time", Input → "Time"];
```

```
The Fourier transform of the input took 0.010227 seconds.
The change of basis took 0.003800 seconds.
```

In[18]:=
```
ckf1a = getFourierFromSpace[coeffs, ckDM, origin, mM];
```

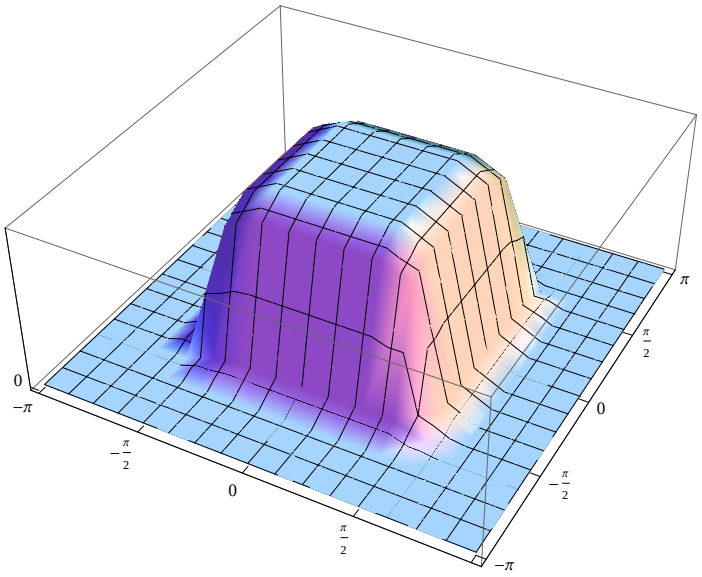
In[19]:=
```
f1approx[x_] := Sum[ ckf1a[[Sequence @@ ({k1, k2} + origin)]] Exp[I {k1, k2}.x] ,
   {k1, -max[[1]], max[[1]]}, {k2, -max[[2]], max[[2]]}];
```

In[21]:=
```
f1aTerm = Simplify[f1approx[{x1, x2}]];
```

We use Chop to exclude the small imaginary parts, that occur during computations

In[22]:=
```
Plot3D[Chop[f1aTerm], {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[22]=

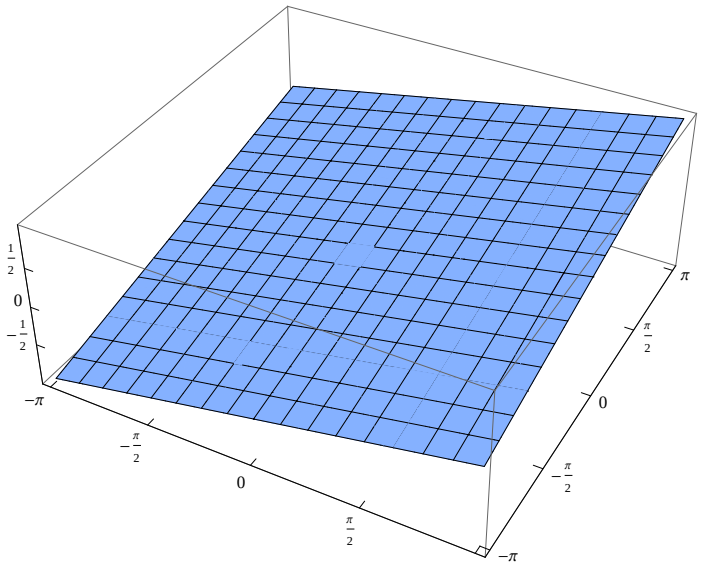


Sampling a linear function

In[23]:=
```
f2[{x_, y_}] := (x + y) / (2 π)
```

In[24]:=
```
Plot3D[f2[{x1, x2}], {x1, -π, π}, {x2, -π, π},
 Ticks → {{-π, -π / 2, 0, π / 2, {π, ""}}, {-π, -π / 2, 0, π / 2, π}, {-1 / 2, 0, 1 / 2}}]
```

Out[24]=



In[25]:=
```
data2 = sampleFunction[mM, f2,
   Debug → "Text&Time", File → "example4/sampling-f2.dat"];
```

```
Loading data from file "example4/sampling-f2.dat"...
```

Here, we can again use the Bracket sum from the first function to perform a change of basis.

In[26]:=
```
coeffs2 = changeBasis[mM, data2, DMBS, Debug → "Text&Time", Input → "Time"];
```

```
The Fourier transform of the input took 0.008372 seconds.

The change of basis took 0.003116 seconds.
```

In[27]:=
```
ckf2a = getFourierFromSpace[coeffs2, ckDM, origin, mM];
```

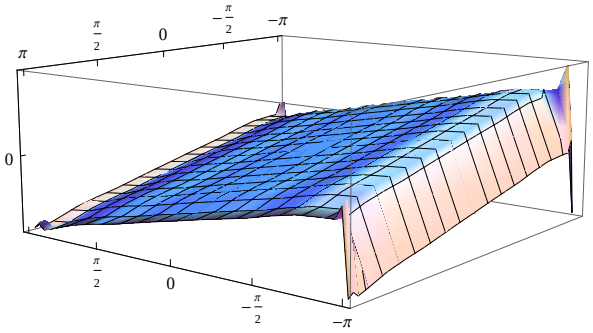
In[28]:=
```
f2approx[x_] := Sum[ ckf2a[[Sequence @@ ({k1, k2} + origin)]] Exp[I {k1, k2}.x] ,
   {k1, -max[[1]], max[[1]]}, {k2, -max[[2]], max[[2]]}];
```

In[29]:=
```
f2aTerm = Simplify[f2approx[{x1, x2}]];
```

Here, we see that the discontinuities at the boundary are approximated by a finite Fourier series and hence continuous.

In[32]:=
```
Plot3D[f2aTerm, {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[32]=



Literature

[1] R.Bergmann, *Translationsinvariante Räume multivariater anisotroper Funktionen auf dem Torus*, Ph.D.thesis, University of Lübeck, 2013 (german).