Example 5

De la Vallee Poussin means and nested spaces

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

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

In this PDF the 3 D plots of the scaling and wavelet functins are removed to reduce the file size

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`"];
```

Both constructions, the generalized de la Vallée Poussin menas and their subspaces are inversti-gated in Chapter 4 in [1] (in german), up to their ability to form an MRA.

De la Vallé e Poussin mean

The de la Vallée Poussin means from the onedimensional case are given in theis Fourier coeffi-cients as samplings of a linear function, more precisely a pyramid function, for example using

In[4]:=
```
? pyramidFunction
```

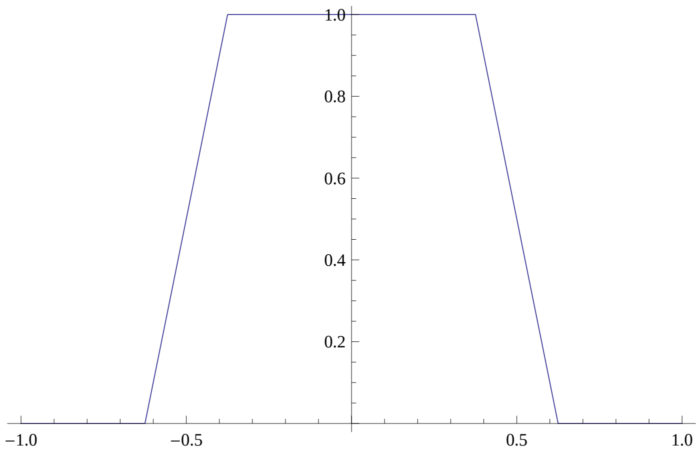
pyramidFunction[α,x]

The d–dimensional analog of the pyramid function, the de la Vallé e Poussin mean are build on. These are here shrunken for generality onto the symmetric unitcube, i.e. having a support from $-1/2-\alpha$ to $1/2+\alpha$ in each dimension.

α may be a nonnegative number less than 1/2 or an array of d elements containing such numbers, where d is the length of x.

In[5]:=
```
Plot[pyramidFunction[1 / 8, x], {x, -1, 1}]
```

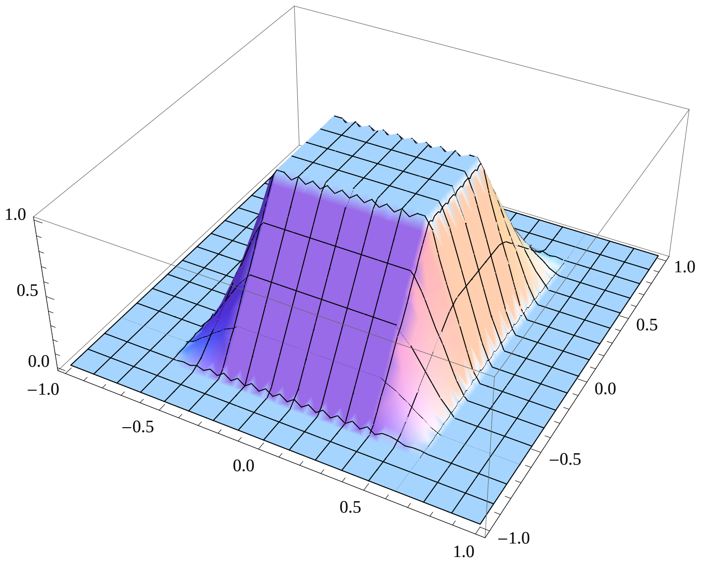
Out[5]=



and similar for higher dimensions, where they may have different decays, e.g.

In[6]:=
```
Plot3D[pyramidFunction[{1 / 8, 1 / 4}, {x, y}], {x, -1, 1}, {y, -1, 1}]
```

Out[6]=



These are then sampled at the points **Transpose[Inverse[mM]].k**, where k is an integer vector. For generality, the de la Vallée Poussin means can be obtained by any function g, as can be seen in the **::usage**. Though for the pyramidal cases, there is a shorthand: When g is a value or a vector representing α, where even more for the higher dimensions α may be a value indicating, that each dimension has the same decay. For these, the support is known and we don't have to specify that option.

Furthermore the function can also directly compute the corresponding Bracket sums:

In[7]:=
```
? delaValleePoussinMean
```

delaValleePoussinMean[g,mM]

Generate the de la Vallé e 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 $t \leq \frac{1}{2}$ which corresponds to
the pyramidal (tensor product of 1D de la Vallé e 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.

For simplicity lets first look at a diagonal matrix. The coefficients are computed and the array does at least cover the complete support. Let’s look at the image of the Fourier coefficients. As in the previous example, the origin is in the middle of the array

In[8]:=
```
mM = 32 IdentityMatrix[2];
```

In[9]:=
```
ckφM = delaValleePoussinMean[
   pyramidFunction[{1 / 14, 1 / 14}, #] &, mM, Support → {1 / 14, 1 / 14}];
max = (Dimensions[ckφM] - 1) / 2; origin = max + 1;
```

Where we specify the datarange

In[11]:=
```
ListPointPlot3D[ckφM, DataRange → Transpose[{-max, max}]]
```

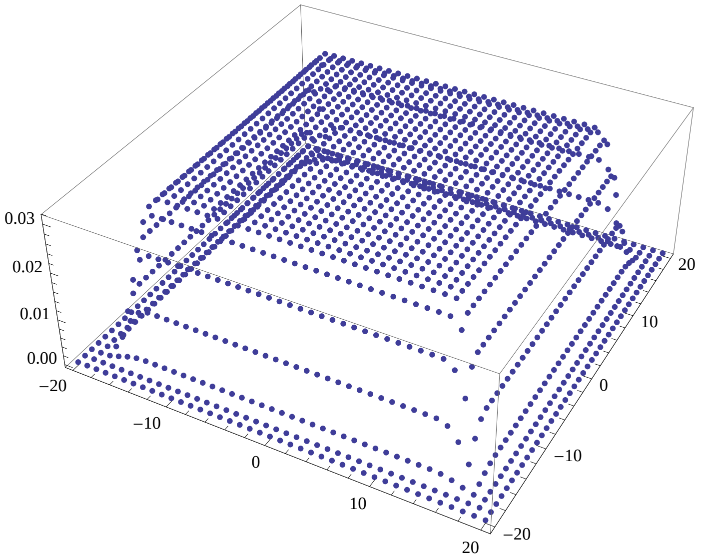
This is the same as using

In[12]:=
```
ListPointPlot3D[delaValleePoussinMean[{1 / 14, 1 / 14}, mM],
 DataRange → Transpose[{-max, max}]]
```

Or even shorter for this case due to the same decay in both dimensions

In[13]:=
```
ListPointPlot3D[delaValleePoussinMean[1 / 14, mM],
 DataRange → Transpose[{-max, max}]]
```

Out[13]=

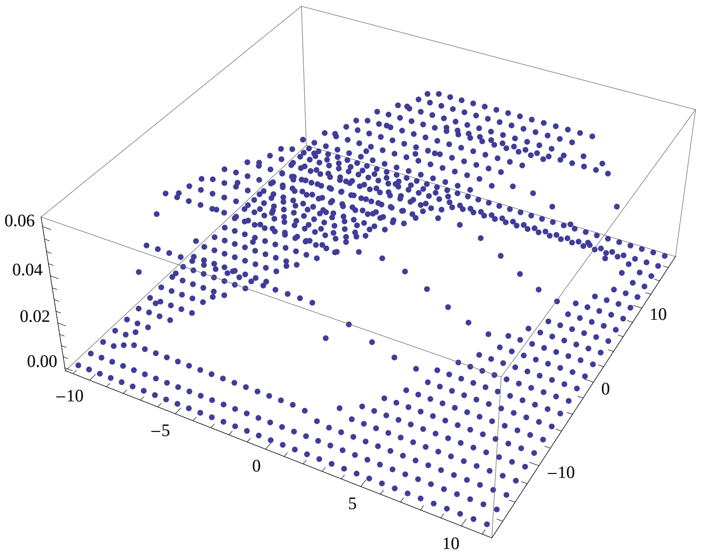


The same works for arbitrary matrices, where we ommit the axis

In[14]:=
```
M2 = {{16, 12}, {0, 16}};
ckφM2 = delaValleePoussinMean[1 / 14, M2];
 max2 = (Dimensions[ckφM2] - 1) / 2; origin2 = max2 + 1;
```

In[17]:=
```
ListPointPlot3D[ckφM2, DataRange → Transpose[{-max2, max2}]]
```

Out[17]=



In[18]:=
```
φ2[x_] := Sum[ ckφM2[[Sequence @@ ({k1, k2} + origin2)]] Exp[I {k1, k2}.x] ,
    {k1, -max2[[1]], max2[[1]]}, {k2, -max2[[2]], max2[[2]]}];
```

In[19]:=
```
φ2Term = Simplify[φ2[{x, y}]];
```

In[20]:=
```
Plot3D[Chop[φ2Term], {x, -π, π}, {y, -π, π}, PlotRange → All, MaxRecursion → 6]
```

Where the corresponding Dirichlet Kernel looks like the following (and is the limit case for α -> 0)

In[21]:=
```
ckDM2 = DirichletKernel[M2];
maxD = (Dimensions[ckDM2] - 1) / 2; originD = maxD + 1;
```

In[23]:=
```
d[x_] := Sum[ ckDM2[[Sequence @@ ({k1, k2} + originD)]] Exp[I {k1, k2}.x] ,
   {k1, -maxD[[1]], maxD[[1]]}, {k2, -maxD[[2]], maxD[[2]]}];
```

In[24]:=
```
dTerm = Simplify[d[{x, y}]];
```

In[25]:=
```
Plot3D[Chop[dTerm], {x, -π, π}, {y, -π, π}, PlotRange → All]
```

Subspaces

For any function φ_M M and a factorization of the regular matrix M = JN into interger matrices J and N, we would like to obtain a function φ_N which is in the space of translates,
i.e. span $\{\varphi_M(\circ - 2\pi y),\ y \in \mathcal{P}(M)\}$. These are characterized by either a set of coefficients with respect to these shifts and hence a value for each point of the **pattern[M]** or similarly (in their discrete Fourer transform) on the generating set. First,
the main matrices for factorization are in the dyadic case given by

In[26]:=
```
?dilationMatrix2D
```

dilationMatrix2D[letter]

represents the 2–dimensional dilation matrices available in the de la Vallée Poussin case. These are named by letters and an additional sign for some matrices, in total these are: “X”, “Y”, “D”, “Y+”, “Y–”, “X–” and “X+”

In[27]:=
```
Table[MatrixForm[dilationMatrix2D[L]],
 {L, {"X", "Y", "D", "Y+", "Y-", "X+", "X-"}}]
```

Out[27]= $\left\{\begin{pmatrix}2&0\\0&1\end{pmatrix}, \begin{pmatrix}1&0\\0&2\end{pmatrix}, \begin{pmatrix}1&-1\\1&1\end{pmatrix}, \begin{pmatrix}1&1\\0&2\end{pmatrix}, \begin{pmatrix}1&-1\\0&2\end{pmatrix}, \begin{pmatrix}2&0\\1&1\end{pmatrix}, \begin{pmatrix}2&0\\-1&1\end{pmatrix}\right\}$

We use

In[28]:=
```
mN = Inverse[dilationMatrix2D["X"]].M2
```

Out[28]= {{8, 6}, {0, 16}}

and define the smaller de la Vallée Poussin mean by its (discrete Fourier transform of the) space coefficients. again a function g models the decay of the coefficients of φN. Furthermore, in this dyadic case, we obtain one wavelet, whose translates (with respect to **pattern[mN]**) form the orthogonal complement of the translates of $\varphi_N$in the bigger space φ_M.

In[29]:=
```
? delaValleePoussinSubspaces
```

delaValleePoussinSubspaces[g,mM,mJ]

For any dyadic dilation matrix mJ and a function g fulfilling the 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.

this method divides the mM–invariant space V (having dimension m=|Det[mM]) into two orthogonal subspaces by returning two coefficient arrays, whose Fourier transforms are weights to the sum of translates of any function, that spans V with its translates. Both functions provide linear independence with respect to mN = Inverse[mJ].mM

For simplicity g might be given as a nonnegative number $t \leq \frac{1}{2}$ which corresponds to the pyramidal (tensor product of 1D de la Vallé e Poussin means) function, where the support is $-\frac{1}{2}-t$ to $\frac{1}{2}+t$ in each dimension.

Options

Orthonormalize → True | False
    Perform an orthonormalization of the translates of both functions with respect to mN.
Validate → True | False
    whether to perform a check (via isMatrixValid[mM]) on the matrix mM, mJ and mN.
File → None | {String,String}
    save the coefficients of both functions to a file each.
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.

Again these coefficients characterizing φ_N in φ_M are given in the frequency domain.

In[30]:=
```
{coeffS, coeffW} = delaValleePoussinSubspaces[1 / 14, M2, dilationMatrix2D["X"]];
```

we can reconstruct the Fourier coefficients using the function from the translation invariant spaces Example 4

In[31]:=
```
? 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[32]:=
```
ckφN = getFourierFromSpace[coeffS, ckφM2, origin2, M2];
```

In[33]:=
```
φN[x_] := Sum[ ckφN[[Sequence @@ ({k1, k2} + origin2)]] Exp[I {k1, k2}.x] ,
    {k1, -max2[[1]], max2[[1]]}, {k2, -max2[[2]], max2[[2]]}];
```

In[34]:=
```
φNTerm = Simplify[φN[{x, y}]];
```

In[35]:=
```
Plot3D[Chop[φNTerm], {x, -π, π}, {y, -π, π}, PlotRange → All, MaxRecursion → 6]
```

And the same for the corresponding wavelet

In[36]:=
```
ckψN = getFourierFromSpace[coeffW, ckφM2, origin2, M2];
```

In[37]:=
```
ψN[x_] := Sum[ ckψN[[Sequence @@ ({k1, k2} + origin2)]] Exp[I {k1, k2}.x] ,
    {k1, -max2[[1]], max2[[1]]}, {k2, -max2[[2]], max2[[2]]}];
```

In[38]:=
```
ψNTerm = Simplify[ψN[{x, y}]];
```

In[39]:=
```
Plot3D[Chop[ψNTerm], {x, -π, π}, {y, -π, π}, PlotRange → All, MaxRecursion → 6]
```

Literature

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