CIMPA-13 Notes , by HGFei

adapted from MACHA11 Notes, August 2011

prepared by Hans G. Feichtinger: www.nuhag.eu

1 Recalling Linear Algebra and Basic MATLAB^{TM}

Note: throughout these notes we will follow standard MATLAB routines, sometimes include NuHAG specific tools (which can be typically downloaded from `www.nuhag.eu`. Some of them, such as SHG.M are just convenience tools (calling figure(gcf); in this case).

1.1 Linear combinations and matrix multiplication

```
>> a1 = rand(4,1)
a1 =
    0.8147
    0.9058
    0.1270
    0.9134

>> a2 = rand(4,1); a3 = rand(4,1);

>> A = [a1,a2,a3]
    0.8147    0.9575    0.9572
    0.9058    0.9649    0.4854
    0.1270    0.1576    0.8003
    0.9134    0.9706    0.1419

>> x = rand(3,1)
    0.4218
    0.9157
    0.7922

>> A*x
    1.9787
    1.6501
    0.8319
    1.3864

>> x(1)*a1 + x(2)*a2 + x(3)*a3
    1.9787
    1.6501
    0.8319
    1.3864
```

NuHAG M-files: compnorm.m

```
>> compnorm(ans, A*x)
\% quotient of norms: norm(x)/norm(y) = 1
\% difference of normalized versions  = 0
ans =      0
>> compnorm(1.55 * A*x , A*x)
\% quotient of norms: norm(x)/norm(y) = 1.55
\% difference of normalized versions  = 2.22e-016
ans =   2.2204e-016
% what are the properties of these vectors in R^4?

>> rref(A)   % running Gauss Elimination
ans =
     1     0     0
     0     1     0
     0     0     1
     0     0     0
>> rank(A)
ans =      3
>> null(A)   % looking for the null-space
ans =    Empty matrix: 3-by-0
```

NuHAG M-files: COMPNORM : comparing the difference of normalized vectors or matrices (format independent, as long as the same dimension appears);

1.2 Ordinary Matrix Multiplication

Just let us recall that the usual rules of calculating the *product matrices*, i.e. to realize the formal action $\boldsymbol{C} := \boldsymbol{A} * \boldsymbol{B}$ is through the formula

$$c_{l,j} = \sum_{k=1}^{n} a_{l,k} \cdot b_{k,j} \qquad (1)$$

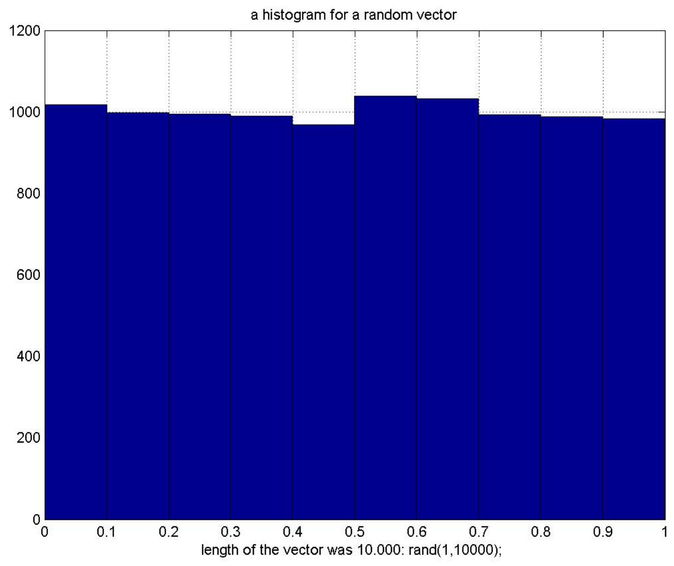
matr-mult

allows different interpretation. First of all the "domino rule" for dimensions (matrices can only be composed if the formats are correct) indicates that each term is in fact a *scalar product* (at least for real vectors). Secondly one can easily see that the action of $\boldsymbol{A}$ on $\boldsymbol{B}$ is columnwise, i.e. the columns of $\boldsymbol{C}$ (same number as the columns of $\boldsymbol{B}$!) are just the images of the columns of $\boldsymbol{B}$ (usually denoted by $\mathbf{b}_k$). In this way it is clear that the columns of $\boldsymbol{C}$ are linear combinations of the columns of $\boldsymbol{A}$.

Either by applying the same principle to the transpose case $\boldsymbol{C}^t = \boldsymbol{B}^t * \boldsymbol{A}^t$, or by reinterpreting (1) we can see that one can also fix l first and find out, that each row of $\boldsymbol{A}$ is undergoing a right matrix multiplication by $\boldsymbol{B}$, and hence the rows of $\boldsymbol{C}$ are different linear combinations of the rows of $\boldsymbol{B}$.

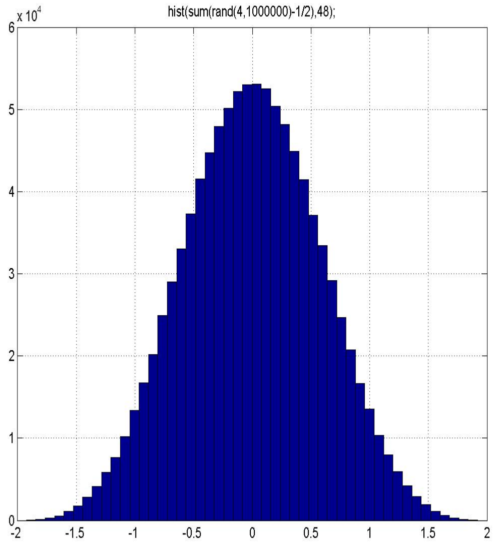
Clear enough, the row-space of $\boldsymbol{C}$ and of $\boldsymbol{B}$ are equal if $\boldsymbol{A}$ is (square and) invertible (this is what is used permanently during the Gauss elimination process, where one multiplies from the left with elementary matrices). Equivalently, multiplication from the right with invertible matrices $\boldsymbol{B}$ preserves the column space, i.e. the column space of $\boldsymbol{B}$ equals the column space of $\boldsymbol{A}$ in this case.

1.2.1 Random vectors: uniform distribution in the unit interval



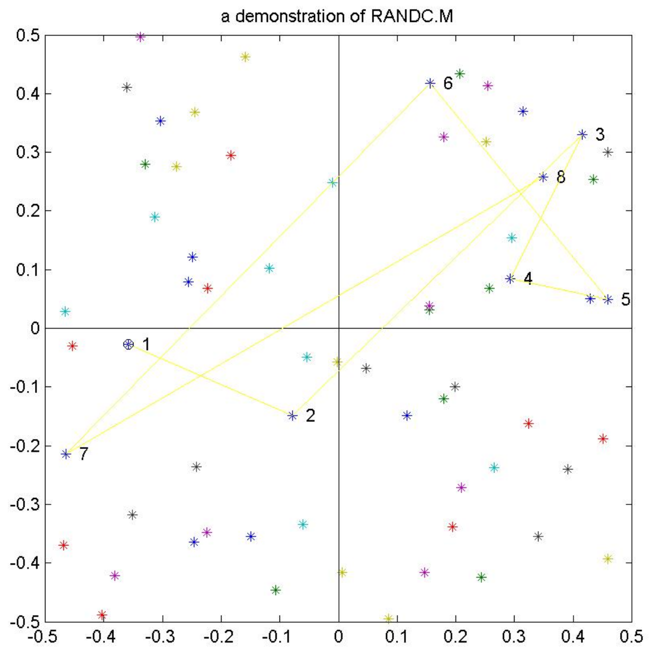
If we are interested in the histogram of the probability distribution of the sum of four independent variables, all of them with values equidistributed in the interval $[-1/2, 1/2]$ we can get a good approximation of this by building the following histogram obtained by the following one-line experiment:

```
hist(sum(rand(4,1000000)-1/2),48);
```



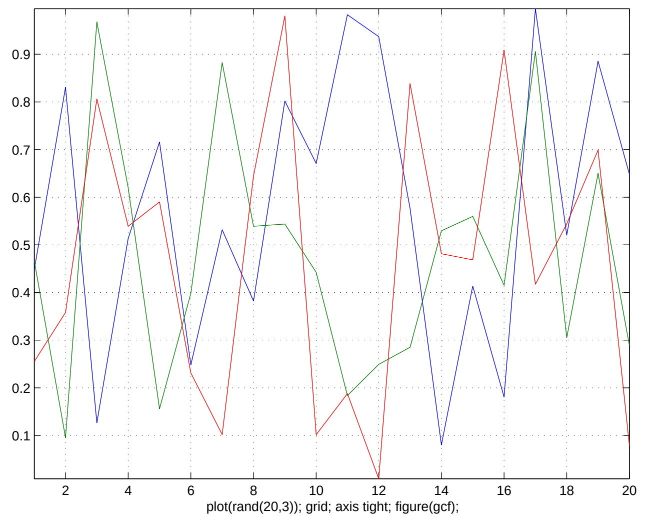
We will work with (random) complex-valued functions as well, and for this reason it is convenient to have a complex random-number generator, which we call RANDC.M[1].

```
plot(randcn(8),'*');
```



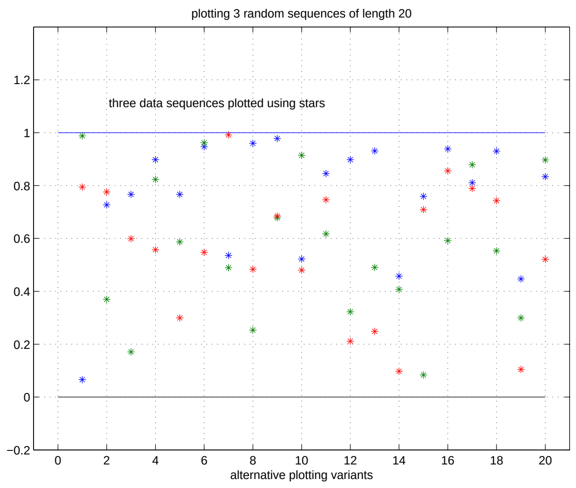
Another experiment:

```
plot(rand(20,3)); grid; axis tight; figure(gcf);
```



```
plot(rand(20,3),'*'); grid; axis([-1,21,-.2,1.4]);
hold on; plot(0:20,zeros(1,21),'k'); plot(0:20,ones(1,21),'b-'); hold off;
figure(gcf);   % or  figure(1);  if figure 1 was the last active figure
```

[1]The **c** will stand most of the time in the future, at the end of M-file names as an indication that this is a "c"entered version of the standard routine.



1.3 Linear algebra excercises: Studying $\vec{x} \mapsto \boldsymbol{A} * \vec{x}$

```
% diary C:\ml5\macha11
\subsubsection{Orthnormal basis and projection}
OA =  orth(A)
   -0.6211   -0.3528    0.6959
   -0.5610    0.1816   -0.4777
   -0.2150   -0.7366   -0.5265
   -0.5032    0.5477   -0.1015
PA = OA*OA'
    0.9946   -0.0480    0.0270    0.0487
   -0.0480    0.5759    0.2384    0.4303
    0.0270    0.2384    0.8660   -0.2418
    0.0487    0.4303   -0.2418    0.5635
norm(PA  - PA', 'fro')
ans = 0
norm(PA*PA - PA)
ans =   6.5192e-016
norm(PA*A - A)
ans =   1.1844e-015
disp('This shows that PA is THE orthonormal projection onto the column space');
This shows that PA is THE orthonormal projection onto the column space
b = rand(4,1);
xb = A\b
    1.0257
   -0.5661
    0.5551
norm(A*xb - b)
ans =     0.2938
norm(A*xb -  PA*b)
ans =   1.0370e-015
save macha00A
```

```
(b - PA*b)' * A
ans =   1.0e-015 *
    0.4545    0.4580   -0.0694
disp('this shows that  b - PA*b is perpendicular to the column space of A');
this shows that  b - PA*b is perpendicular to the column space of A
save macha00A
norm(xb - pinv(A)*b)
ans =   5.4328e-014
pinv(A)
  -14.4970   10.1652   10.7390    2.4523
   13.7472   -9.1951  -10.5685   -1.6743
   -0.3734    0.4958    1.4595   -0.3615
 norm(A*pinv(A) - PA)
ans =
  4.6214e-015
cao = (b - PA*b)/norm(b - PA*b);
cao' * A
ans =   1.0e-014 *
    0.1568    0.1624   -0.0333
size(cao)
ans =      4     1
 OA4 = [OA, cao] ;
OA4' * OA4
    1.0000    0.0000   -0.0000   -0.0000
    0.0000    1.0000   -0.0000    0.0000
   -0.0000   -0.0000    1.0000   -0.0000
   -0.0000    0.0000   -0.0000    1.0000
norm( OA4' * OA4  - eye(4))
ans =   2.2384e-015
disp('the ONB for  COL(A) combined with orthonormal vector is ONB for R^4');
the ONB for  COL(A) combined with orthonormal vector is ONB for R^4
```

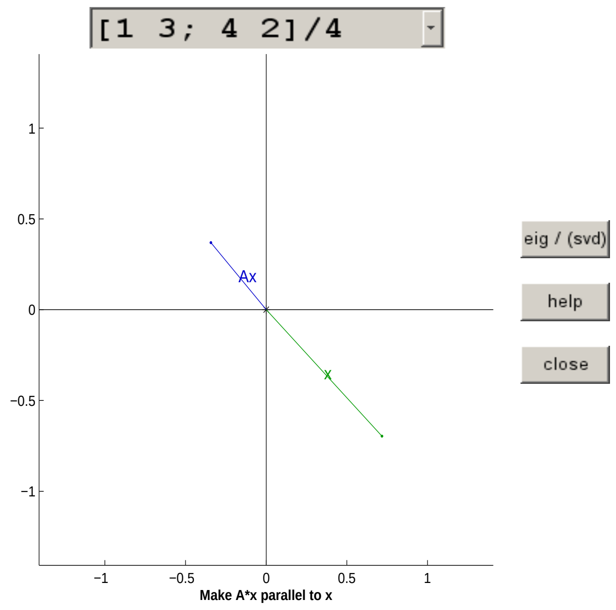
1.4 Alternative orthnormal bases for the COL(A)

```
    OA1 = orth(A(:,3 :  -1 : 1))
   -0.6211   -0.3528    0.6959
   -0.5610    0.1816   -0.4777
   -0.2150   -0.7366   -0.5265
   -0.5032    0.5477   -0.1015
norm(OA1 - OA)
ans =   1.3605e-015
OA1 = orth(A(:,[1,3,2]))
   -0.6211   -0.3528    0.6959
   -0.5610    0.1816   -0.4777
   -0.2150   -0.7366   -0.5265
   -0.5032    0.5477   -0.1015
```

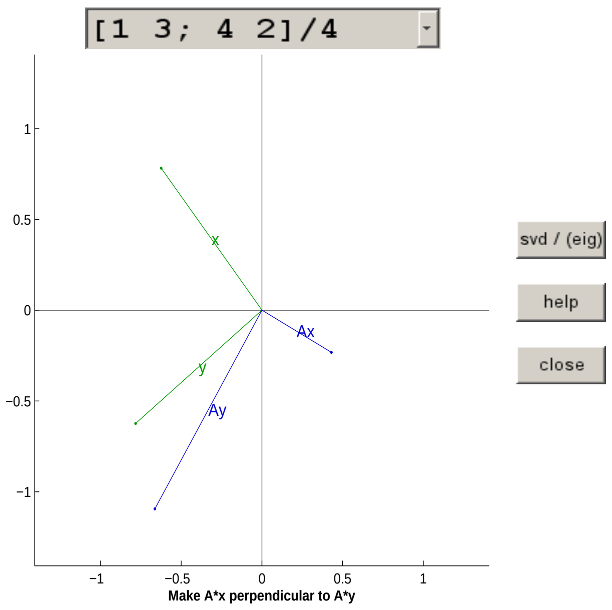
```
norm(OA1 - OA)
ans =   7.7586e-016
OAB = orthbest(A);
norm(OAB - OA)
ans =     2.0000
norm(OAB + OA)
ans =     1.6289
norm( OAB * OAB'  - PA)
ans =   3.0406e-015
norm( OA1 * OA1'  - PA)
ans =   1.4284e-015
ans =     0.5332    0.8109   -0.2
GA = gramsfei(A)
    0.5332    0.8109   -0.2295
    0.5928   -0.3628    0.3048
    0.0831    0.2326    0.8972
    0.5978   -0.3959   -0.2223
norm(GA*GA' - PA)
ans =   1.4886e-014
GAV = gramsfei(A(:,3 : -1 :1))
    0.7110    0.0757   -0.6952
    0.3605    0.4564    0.4875
    0.5945   -0.4974    0.5150
    0.1054    0.7339    0.1176
norm(GAV*GAV' - PA)
ans =   1.6398e-015
```

It is important to recall the most important concepts related to eigenvalues, the SVD (Singular value decomposition) and the

A nice demo is the MATLAB routine EIGSHOW, with the following result:



The option SVD gives additional insight:



Assignment: there should be an easy to make SVD plausible, by inspecting the image of an arbitrary orthonormal basis of the row-space, for a typical 3×3-matrix $\boldsymbol{A}$, such as $A = zeros(3); A(:) = 1:9;$.

It is contained in the NHGTB as `testmat(3)`.

$$\begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{pmatrix} \tag{2}$$

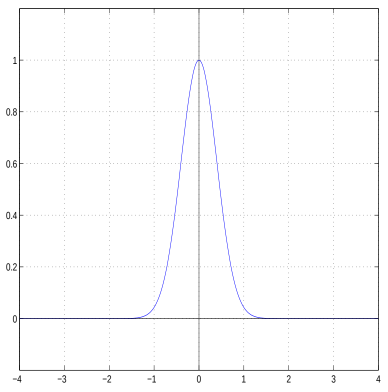
NuHAG M-files: TESTMAT

1.5 Ordinary plotting routines and FFT

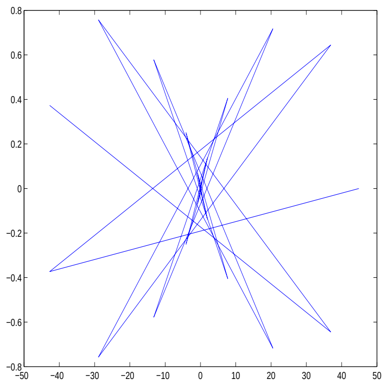
Trying to find out that the Fourier transform of the Gauss function is another Gauss function could be *naively be attempted* in the following way:

```
>> bas = linspace(-4,4,360);
>> gaubas = exp(-pi*bas.^2);
>> figfei;
>> plot(bas,gaubas);
>> grid; ax20; plotax; shg;
```

which produces the following plot:



Trying to plot the FFT (fast Fourier transform) in a naive way, i.e. by calling `plot(fft(gaubas));` results in the following (nice) desaster:

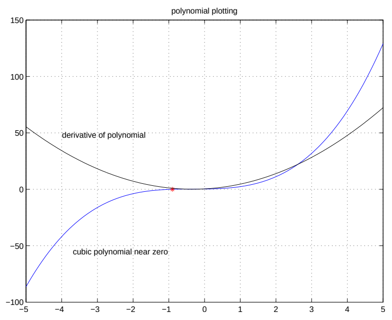


We will try to give detailed explanations later. Basically MATLAB is doing the plot (a curve) in the complex plane. The commands AX20 (adjusting the axes) and PLOTAX (adding the plot of the $x-y$-coordinate system in black) are convenient (and simple) NuHAG tools.
NuHAG M-files: AX20, PLOTAX

1.6 More on polynomials and plotting

Here `derb` are the coefficients of the derived (now quadr.) polynomial, cf. below:

```
>> figure;
>> plot(bas,polyval(b,bas)); hold on; plot(rootsb(1),0,'r*');
>> grid;  hold on; plot(bas,polyval(derb,bas),'k'); hold off; figure(gcf);
>> gtext(' derivative of polynomial ');
>> gtext(' cubic polynomial near zero ' );
>> title('polynomial plotting');
```



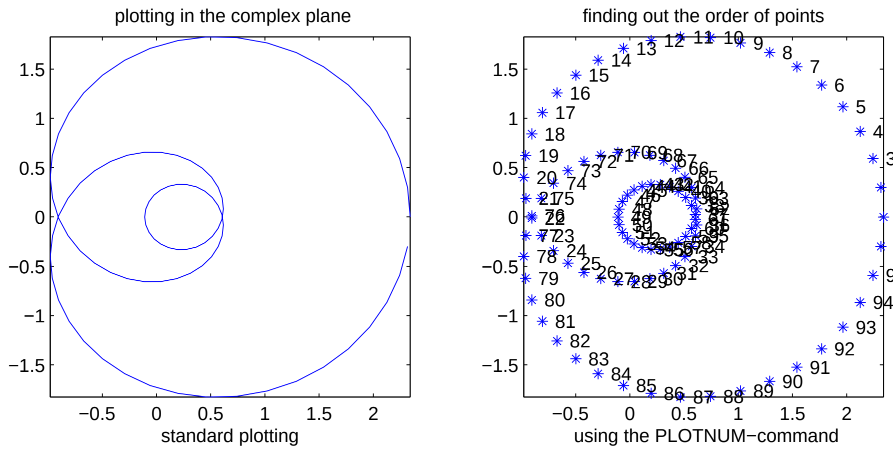
How has the derivative been determined (using the linear procedure of derivation):

```
>> DER =  diag(n-1 : -1 : 0)
DER =
     3     0     0     0
     0     2     0     0
     0     0     1     0
     0     0     0     0

>> DER =  DER([n, 1: n-1],:)

     0     0     0     0
     3     0     0     0
     0     2     0     0
     0     0     1     0

derb =  DER * b;
```

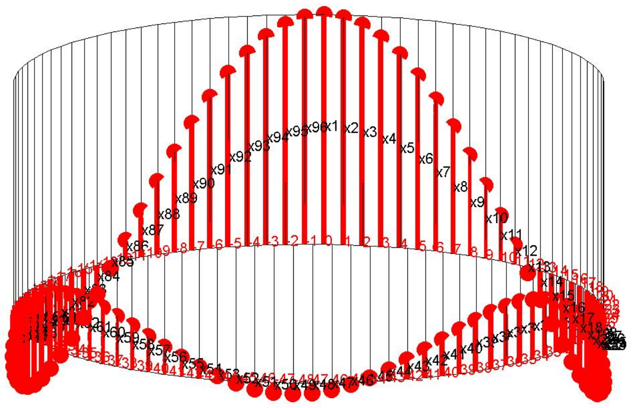
1.7 Complex numbers and plotting

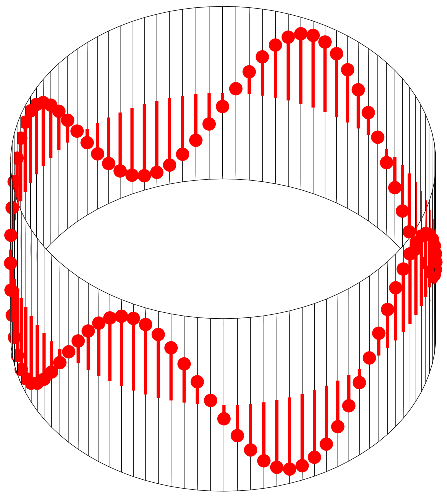
The polyval-command allows to evaluate polynomials over sequences of points in the complex point. Of course the values of a (real or complex-valued) polynomial function over the complex numbers gives complex values, and in this case the

This is a plot of our polynomial (with coefficients b as above) over the unit-roots of order 96:



Observe that the curve is not closed at the end, i.e. the points 96 (last point) and the first point (simply because MATLAB does/cannot know that we have periodic functions/sequences in mind).
NuHAG M-files: PLOTNUM

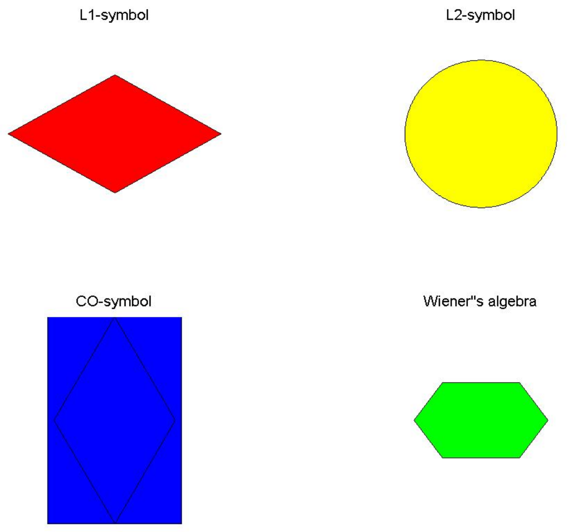




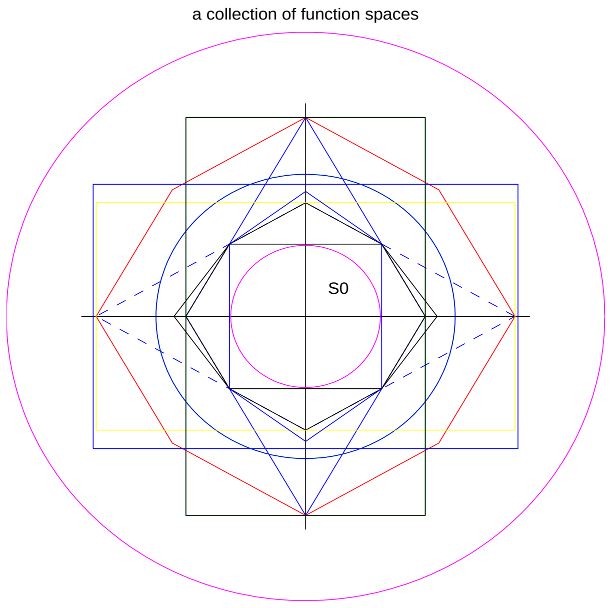
Using a recent NuHAG routine (called stemcyl: stem command, on the cylinder) one can show e.g. the real-part of the complex-valued function as a periodic function, i.e. plotted on a cylinder, so to say a graph on $\mathbb{Z}_{96} \times \mathbb{R}$, using the stem command[2].
NuHAG M-files: STEMCYL
Task: Check for other ways to plot a function on the unit circle...

[2]Here $\mathbb{Z}_{96}$ stands for the cylcic group of order 96

1.8 Function space plots



or in another form



The corresponding MATLAB files may not all be included in our distribution (right now).
NuHAG M-files: SOPLOT3 (standard), SOPLOT2 (min), SOPLOT4, SOSPACES, SOPLOTFLI, CFILL

```
[S0,L2,SOP,CO,LI, LINF,LIFLI,  WR, xfli, bas ] =  soplotfli;
>> cfill(LIFLI);  cfill(S0,'y');
```

1.9 Polynomial Multiplication and Convolution

The convolution operation is quite important in Harmonic Analysis. At first sight one can say that it corresponds to the multiplication of polynomials. For the continuous domain one can say that the convolution of f and g, each of them describing the probability density of a random variable (call it X and Y respectively), then $f * g$ describes the probability distribution of $X + Y$ (assuming that X and Y are independent variables).

1.10 Polynomials and Vandermode Matrices

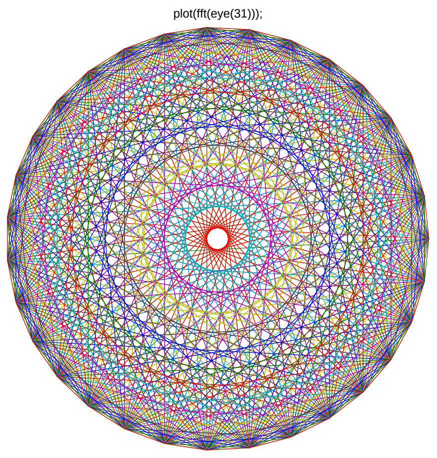
The MATLAB command `polyval` allows to evaluate the values of a polynomial not only over sequences of points over the real axis, but of course also in the complex domain. Clearly the mapping from the coefficients (NOTE: a polynomial of degree k is described by $k+1$ coefficients, with the following MATLAB convention: $p(t) = a_1 t^k + a_2 t^{k-2} + \dots a_{k+1}$, i.e. the ordered basis of monomials comes in descreasing order $\{t^k, t^{k-1}, \dots, t, 1\}$[3].

Soon we will see that there is a command for the multiplication of polynomials (the so-called *convolution* or the *Cauchy product*) which is very much comparable with the multiplication of long integer numbers.

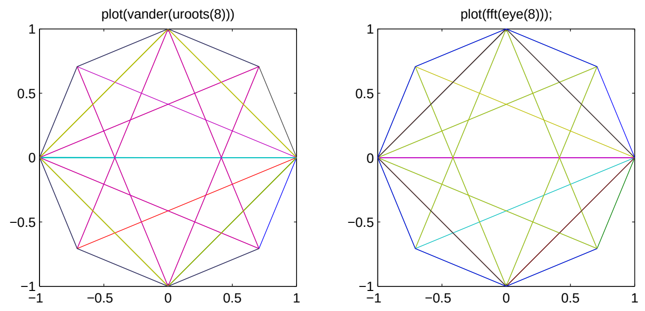
Experiment: conv([1,2,3],[1,2,3]);

The linear mapping from coefficients of a polynomial to values (at given complex positions) is a linear mapping, hence there is a matrix describing this linear mapping in the standard bases. For MATLAB the natural ordered bases describing these mappings is the (descreasing) monomial basis described above and the standard unit vectors in R^d, if d samples are taken. Of course the most important case is the situation, where the number of samples equals the number of unknown coefficients, i.e. order of $p(t) = k+1$. The corresponding matrix is know as Vandermode matrix: `V = vander(t)`, for $\vec{\mathbf{t}} = [t_1, \dots, t_{k+1}]$.

We will later see that the Fourier matrix (i.e. the matrix describing the FFT, the fast resp. DFT = discrete Fourier transform) is in fact a Vandermonde matrix, up to some minor modifications.



[3]Just like we are used in everyday life to think of 123 as *one* hundred and *two* times ten and *three* times $1 = 10^0$. The *position* of the digit determines its value. In fact, $123 == polyval([1,2,3],10)$!



Comparing the entries (for practical reasons the discussion of the case $k = 6$, or 7 coefficients, is most appropriate:

```
>> real(vander(conj(uroots(7))))
ans =
    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000
    0.6235   -0.2225   -0.9010   -0.9010   -0.2225    0.6235    1.0000
   -0.2225   -0.9010    0.6235    0.6235   -0.9010   -0.2225    1.0000
   -0.9010    0.6235   -0.2225   -0.2225    0.6235   -0.9010    1.0000
   -0.9010    0.6235   -0.2225   -0.2225    0.6235   -0.9010    1.0000
   -0.2225   -0.9010    0.6235    0.6235   -0.9010   -0.2225    1.0000
    0.6235   -0.2225   -0.9010   -0.9010   -0.2225    0.6235    1.0000
>> real(fft(eye(7)))
and =
    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000
    1.0000    0.6235   -0.2225   -0.9010   -0.9010   -0.2225    0.6235
    1.0000   -0.2225   -0.9010    0.6235    0.6235   -0.9010   -0.2225
    1.0000   -0.9010    0.6235   -0.2225   -0.2225    0.6235   -0.9010
    1.0000   -0.9010    0.6235   -0.2225   -0.2225    0.6235   -0.9010
    1.0000   -0.2225   -0.9010    0.6235    0.6235   -0.9010   -0.2225
    1.0000    0.6235   -0.2225   -0.9010   -0.9010   -0.2225    0.6235
```

This suggests that one has to arrange the sequence in opposite order, hence for comparison let us check:

```
>> compnorm( vander(conj(uroots(8))), fliplr(fft(eye(8))));
quotient of norms: norm(x)/norm(y) = 1
difference of normalized versions  = 6.076e-016
```

This `fliplr` command corresponds of course to the MATLAB convention of choosing the basis of monomials in descreasing order, which as a consequence requires the Vandermonde matrix to follow the same rule (in order to be compatible with the `polyval` command.

2 Fourier Analysis: pure frequencies

Obviously Fourier Analysis plays a central role in time-frequency analysis and Gabor analysis. In a way one can say that time-frequency analysis is a kind of localized Fourier analysis comparable with the analysis of a musical signal in terms of score. One is not interested in the sequence of samples (as in principle stored very efficiently, i.e. quantized and well coded, in order to make the reading of the CD robust against scratches, etc.) only, or in the frequency distribution of the whole piece of music, but rather in the *melody* (in the case of a flute with a pure spectrum it means, what is the dominant frequency at a given time, when the score is played), and of course also in the *timbre*, the characteristic distribution of frequencies (overtones, harmonic sounds) which depend on the instrument. In fact, one can even do a fine analysis of how the individual accords of an instrument are changing the energy-distribution over time (e.g. in a piano it decays after a percussive beginning, for an organ it is more or less constant and produces a sustained sound as long as the key is pressed, etc.).

The mathematical tool for Fourier analysis (which at the end is also in the background of the MP3 audio compression algorithm) of finite sequences is the Fourier transform, which can be seen as a (unitary) change of basis, from the standard orthonormal basis of unit-vectors in the signal domain to the basis of pure frequencies.

We will try to explain in this section how this system of pure frequencies is generated, what the analogue is for the $2D$-setting (namely plane waves), and how this can be understood from the algebraic (and of course computational) point of view. In fact, on any (finite, resp. locally compact) Abelian group the (continuous) eigenvalues of the shift/translation operator *are the characters* (also named pure frequencies in signal processing, in reference to the situation that we have in the $1D$ = audio context).

EXAMPLE of a SOUND!

In short (and it will become more clear later on) we view vectors in $\mathbb{R}^n$ resp. $\mathbb{C}^n$ (= *ordered n-tuples*, ignoring the view-point of row or column vectors) as functions on finite groups, preferably on the cyclic group of unit-roots of order n[4], denoted by $\mathbb{Z}_n$, resp. the quotient group $\mathbb{Z}/n\mathbb{Z}$.

The code for creating the unit-roots is quite simple:

```
% UROOTS.M - It is the unit roots of order N.
%
% Input         : N = order of the unit roots
%                 l = optional for circular solution ("one" is added again
%                     at the end
%
% Output        : u = unit roots of order N, in the natural (counter
%                   clockwise sense).
%
% Usage         : u = uroots(N,l);
%
% Comments      : the optional parameter "l" allows to "close the circle"
%
```

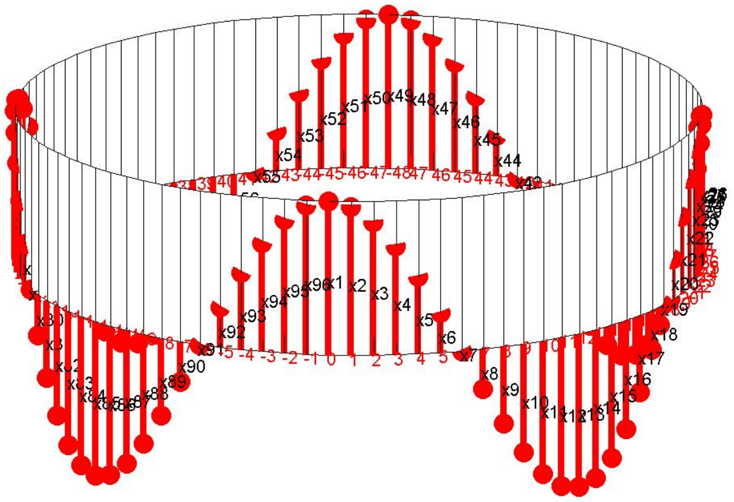
[4]Sometimes and without specific meaning our signal length will be N, or L, or anything reasonable!

```
% See also:  FFT, IFFT

% HGFei,  Oct. 1998
% Renewed by    :    Noha ElAmary,  08 . 1999

function    u = uroots(N,l);

if nargin == 0;  N = 8;  end;
bas = 0 : 1/N : (N-1)/N;
u =  exp(2 * pi * i * bas);
if nargin > 1;   u = [u,1]; end;

if nargin == 0;  plotnum(uroots(8,1)); axis square;
   grid;
   title('unit roots of order 8');
   hold; plot(uroots(8,1),'r*'); hold off;
   xlabel(' call:  uroots(8,1)' );
   figure(gcf);
 end;
```

In fact, the code coud be just one line: $u = exp(2 * pi * i * (0 : 1/N : (N-1)/N));$
The last paragraph is a typical NuHAG-feature: If the routine is called without input argument it displays a typical instance with a short description. The option of a second input arguments (enforcing a closed graph) is used to draw e.g. *closed circles*.

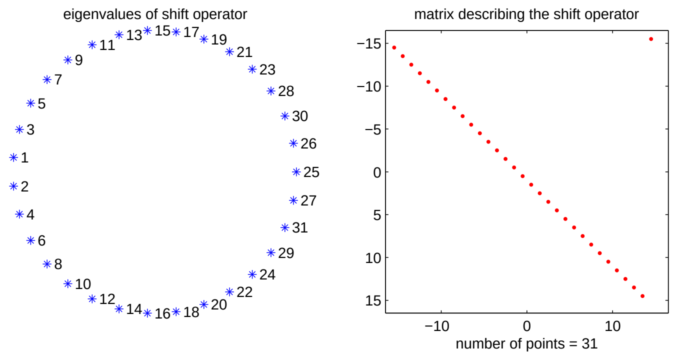
2.1 The monomials over the unit circle

Pure frequencies are the building blocks of Fourier Analysis: They are nothing but the monimials over the unit circle: Let us look at the pure frequency (four full oscillations) over the unit circle (here just unit roots of order 96):



In fact the pure frequencies arise as the eigenvectors of the shift operator T_1. Let us just first look at the eigenvalues. The matrix describing the a cyclic shift is analyzed (since it is unitary it is normal and hence it can be orthogonally transformed into a diagonal matrix with eigenvectors on the unit circle):

```
T1 = eye(31);  T1 = rot(T1,-1);
[V1,D1] = eig(T1);
plotnum(diag(D1));
```



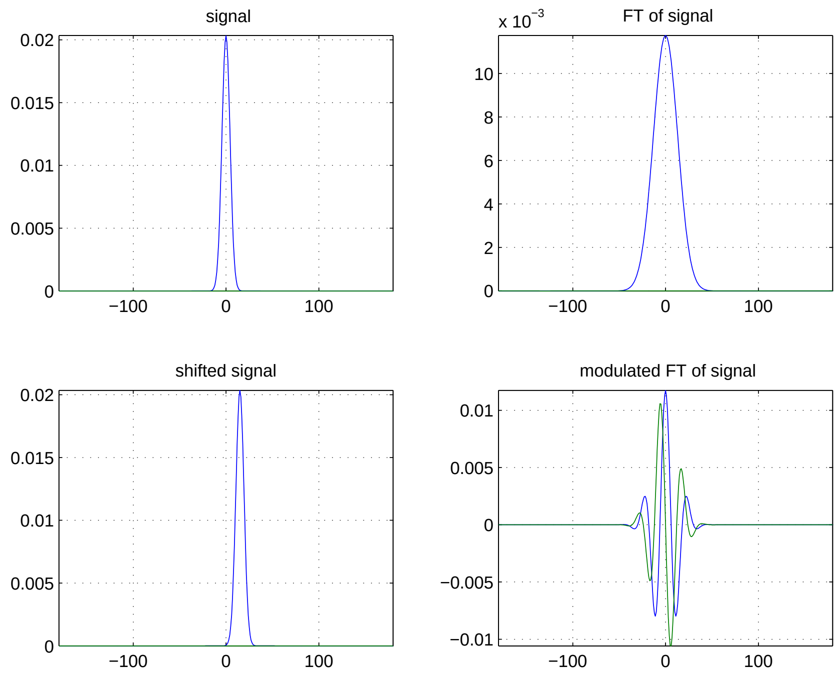
NuHAG M-files: ROT.M: the cyclic shift operator
UROOTS.M: creating the unit roots of order n, in the mathematical positive sense, starting at $\omega^0 = 1$.

2.2 Further MATLAB routines

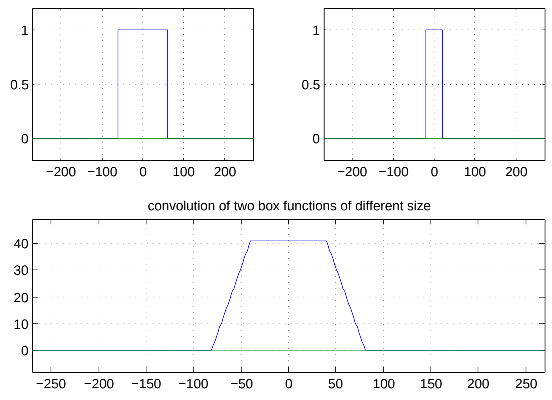
F2SP.M in the form `f2sp(xx1,xx2)` exhibits a signal and its Fourier transform.

It is a routine that is used frequently for the comparison of (1D)-signals, in order to see e.g. whether they are orthogonal (e.g. because of disjoint support on one or the other side), etc.

In addition to the shift operator we also have the modulation operators: multiply with a "pure frequency" (which corresponds to a frequency shift, i.e. a translation on the Fourier transform side).



Convolution of two BOX-functions:



2.3 Properties of the (unitary) Fourier transform

NuHAG M-files: FFTU

FFTU has two functionalities:

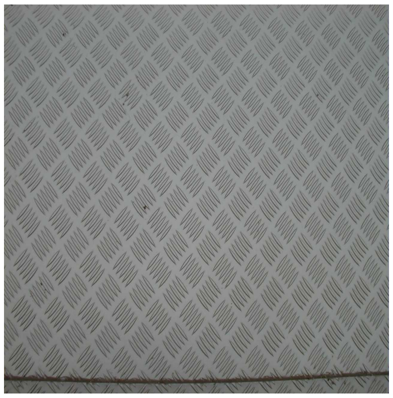
`y = fftu(x)` realizing the operator on a given signal, or `FU = fftu(n)`, providing the matrix, which describes the (normalized) version of the FFT, in fact, it is just $FU = fft(eye(n))/sqrt(n)$.

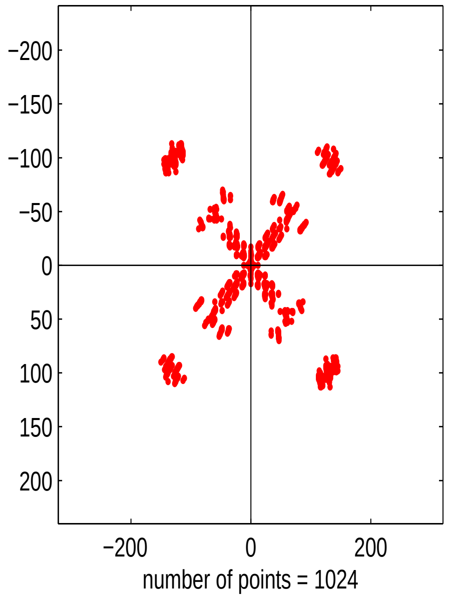
2.4 2D Fourier transform

The $2D$ Fourier transform is essentially an iterated $1D$ algorithm. There are natural reasons for this, namely the fact that the characters of a product group (group homomorphisms from $G_1 \times G_2$ into the unit circle or torus $\mathbb{U}$) are naturally (unique) products of the factor groups, i.e. $\chi \in \hat{G}$ if and only if

$$\chi(x, y) = \chi_1 \otimes \chi_2(x, y) := \chi_1(x) \cdot \chi_2(y)$$

for all $x \in G_1, y \in G_2$.



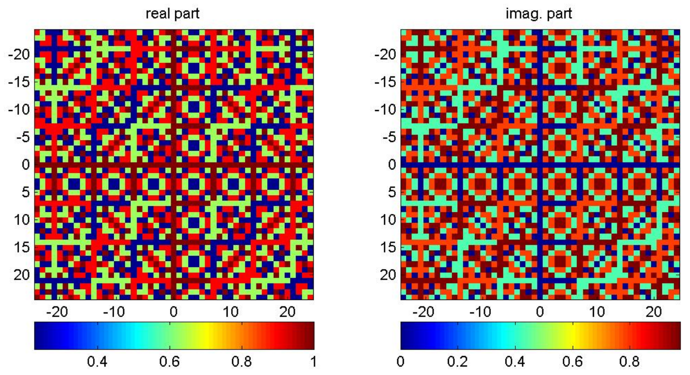


NuHAG M-files:

The standard $2D$-Fourier transform can be written as a row-wise and then columnwise (or vice versa, these operations commute!) application of $1D$-FFTs provide the answer.

```
Building the matrix for the action of FFT2 on  7 x 7 matrices in the standard basis
>> for jj= 1:49; U = zeros(7);U(jj) = 1; UF = fft2(U); F2MAT(:,jj) = UF(:); end;
>> size(F2MAT)   %  ans =     49    49
>> F7 = fft(eye(7));
>> compnorm(kron(F7,F7), F2MAT);
quotient of norms: norm(x)/norm(y) = 1
difference of normalized versions  = 0
```

Visualizing real and imaginary part of the corresponding 49×49 matrix gives the following two pictures.



3 Shift invariant spaces: spline-type spaces

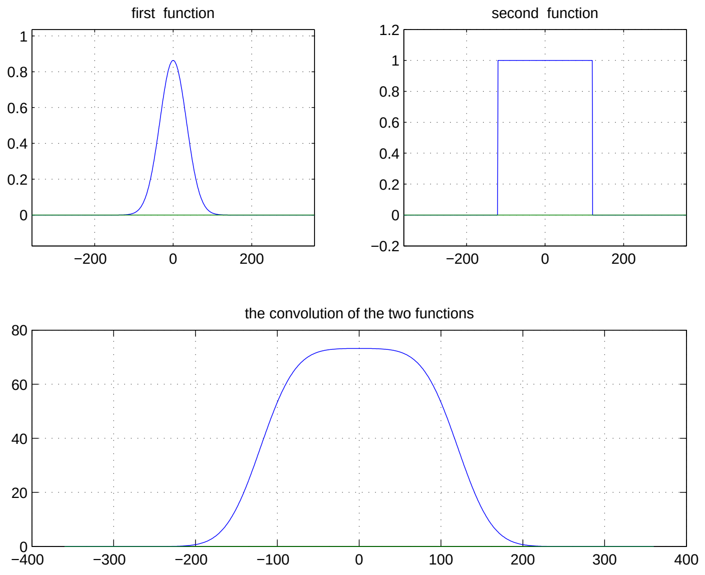
The theory of **shift-invariant spaces** is known under a variety of names[5] The original ground-breaking work in this direction was done by DeBoor, Ron, and others (more recent contributions by Bownik, etc.). We stick with the terminology (mostly standard in connection with the irregular sampling problem for such spaces) of **spline-type spaces**, in order to emphasize that space of cubic splines over the real line $\mathbb{R}$ are the most important case, with the B-spline-basis (shifted copies of basic splines, obtained via convolution from rectangular functions) obtained by shifting a fixed function along the integer lattice. An additional property is that BUPU property of this family (to be discussed later in a more general setting).

3.1 A general PINV-formula

$$pinv(A) = A' * pinv(A * A') = pinv(A' * A) * A.$$

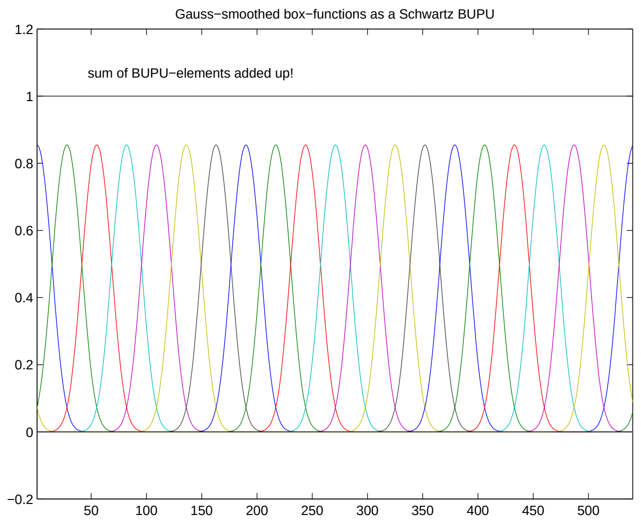
3.2 B-spline functions

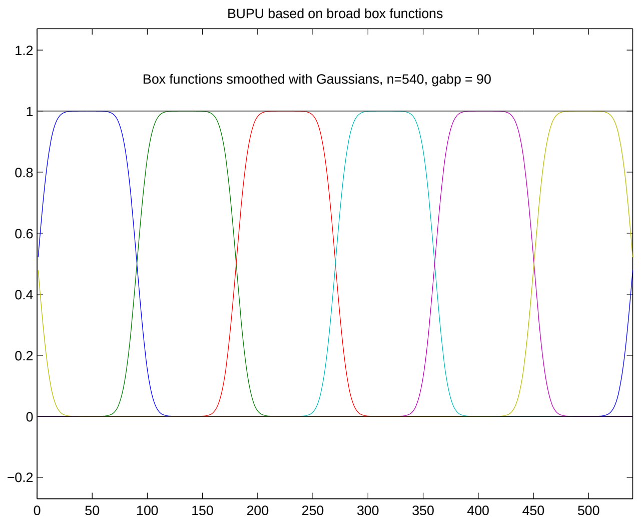
One can start with a BUPU (e.g. a sequence of shifted triangular functions) and convolve them with a Gauss function (which is normalized such that the sum of terms is adding up to one, so obtained by
`g = gaussnk(n); g1 = g/sum(k);`



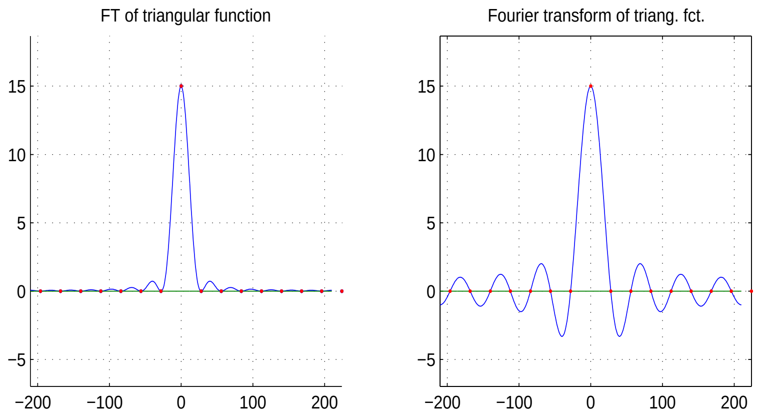
[5]One has to be careful with the terminology: while e.g. the space of band-limited functions, with spec$(f) := \mathrm{supp}(\hat{f}) \subseteq \Omega$ is a **translation invariant** subspace (any shifted/translated copy is in the same space, because shifting the function just corresponds to modulating its Fourier transform, a pointwise operation = multiplication with a pure frequency, which obviously preserving the support). In contrast, shift-invariant spaces are supposed to be only invariant under discrete subgroups of all possible shifts/translations, e.g. $\mathbb{Z}^d \subset \mathbb{R}^d$.

3.3 Plots showing BUPUs and



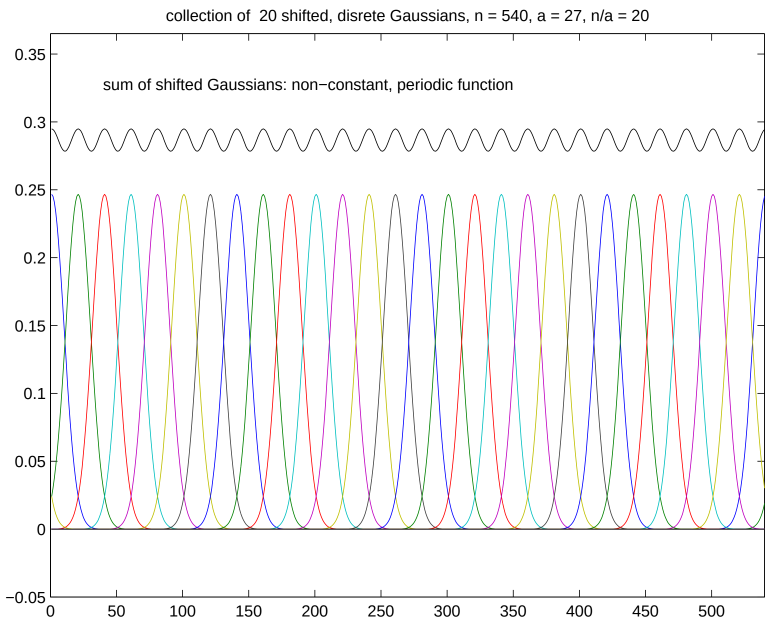


For the case that a is an odd divisor of the signal length n (here $n = 420 = 3{\cdot}4{\cdot}5{\cdot}7, b = 15$), we show that the FT of the window (box of triangular, obtained by convolving the box of size 15), e.g. `bx = flts(420,7)`



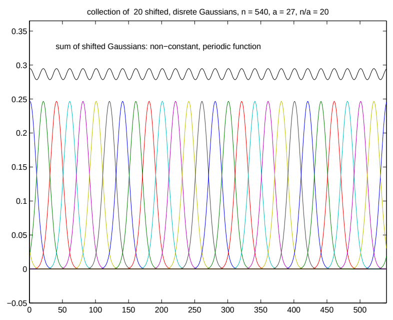
3.4 Shift invariant spaces: so-called spline-type spaces

Let us consider the collection of linear combinations of shifted copies of e.g. some Gauss function:



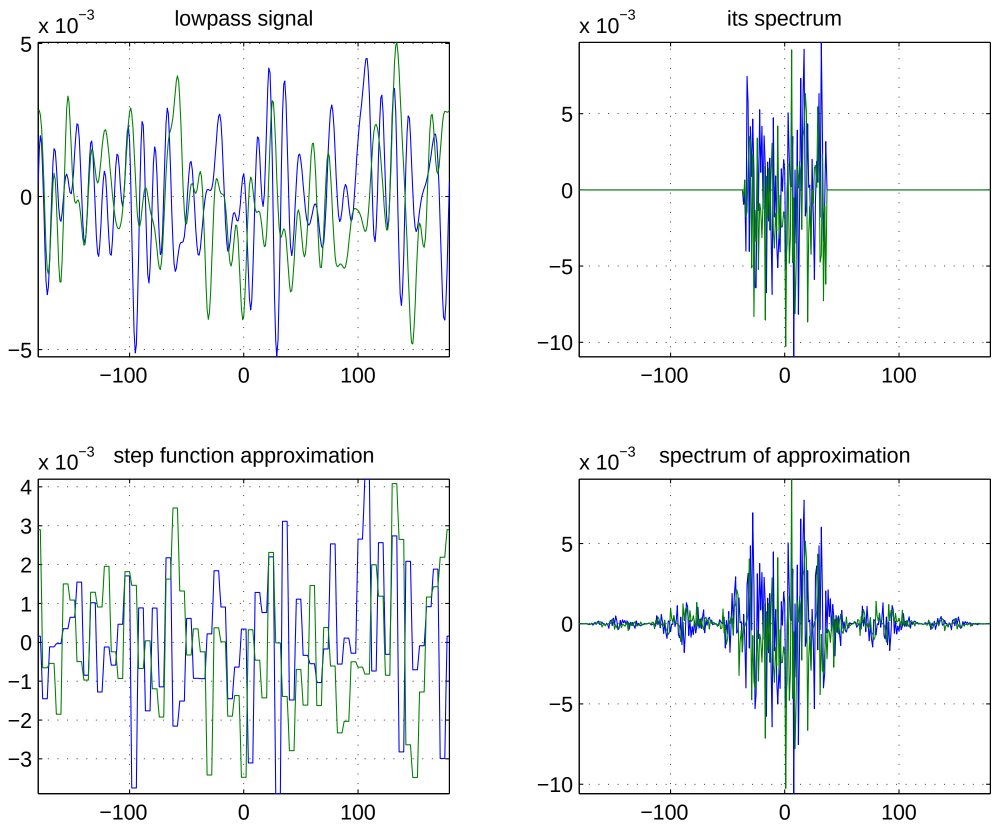
It is clear that the shift parameter a has to be an integer which divides the signal length n, in order to make sure that the family all shift operators involved forms a group (within the group of all possible cyclic translation parameters), i.e. to assume that $mod(a,n) == 0$. Correspondingly the group of shift operators $T_{ka}, 0 \leq k \leq (n/a) - 1$ is a group of operators, naturally isomorphic to $\mathbb{Z}_{n/a}$.

Just to conform the idea that such a family is invariant under translations by the elements of this group let us apply a cyclic shift by $3a$ on the above example, in order to see (!observe how the colored bump functions are moving to the right, resp. reappear on the left, as they drop out of the window of view to the right).



3.5 Approximation of smooth functions

It is no surprising to see that a smooth function is well approximated by step functions. The best approximation by a step function (the value over each interval is just the average value of the approximated function):

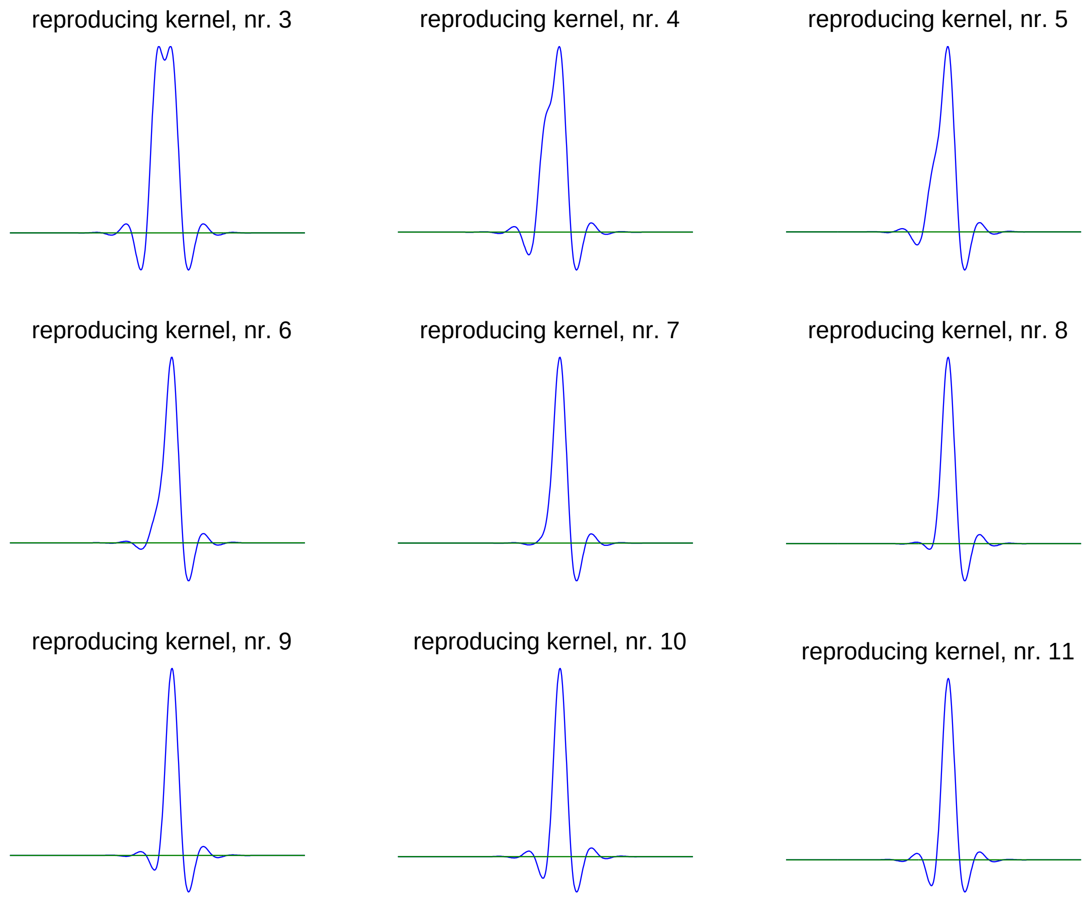


3.6 The projection operator

Obviously the operator describing the best approximation of a given signal by elements of such a spline-type space is a linear operator from $\mathbb{C}^n$ onto the n/a dimensional subspace of $\mathbb{C}^n$ generated by the (cyclic) shifts of the generator. Although - from an harmonic analysis point of view we look at this operator as a mapping from $\boldsymbol{\ell}^2(\mathbb{Z}_n)$ onto $\boldsymbol{V}_{\phi,\Lambda}$ it can be described by a matrix.

In order to get a better description of this operator we recall first that it can be computed in the general case via matrix multiplication with `pinv(TRL)`, where `TRL = trlbas(g,a)` describes the collection of shifted copies of the atom `g`.

There are hence also corresponding elements describing this projection operator. Since this projection operator (in our case `a = 18`) is translation invariant, it is just 18 elements. For reasons of symmetry of our (Gaussian) generator it is enough to represent those generators (in the sense of a reproducing kernel Hilbert space) for the first 9 positions:

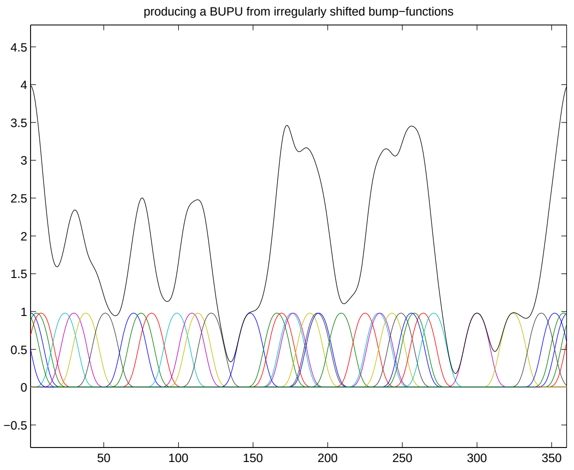


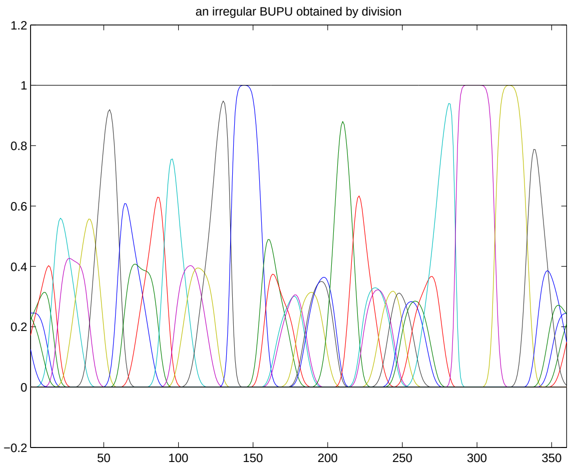
On the spline-type space itself the projection operator acts as identity (RKH-property). It is just a sum of 18 rank one operators. The question of reconstruction of a signal from irregular samples in such a spline-type space can therefore be translated back into a questions, whether the corresponding family of kernels is a generator for the family.

NuHAG M-files: TRLBAS.M: building a collection

Using irregular version of trlbas:

```
n = 360; a = 18; B = bupuspline(n,a,7,4);
xp = find(rand(1,n) > 0.9); TRLb3 = trlbas(b3,xp); TRLs3 = sum(TRLb3); min(TRLs3)
if this minimum is acceptable on can devide by the sum and obtain a BUPU:
lxp = length(xp); TRLphi3 =  TRLb3 .*  (ones(lxp,1)*oneover(TRLs3));
```

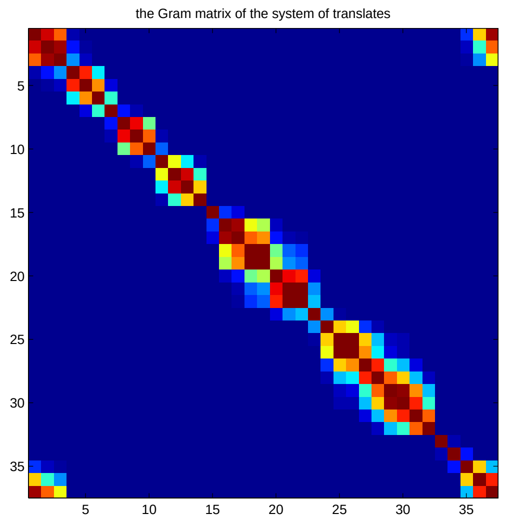


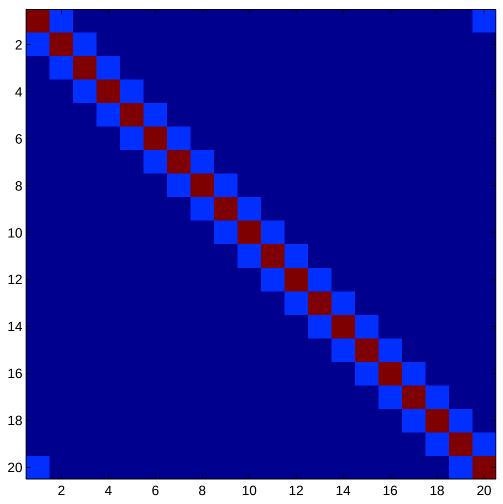


The condition number of the system displayed above is “not so good” (exactly

```
cond(TRLb3), ans = 161.1546
```

On the other hand the irregularity is seen easily on the Gram matrix, which shows variable thickness (broad at those indices, where many bump functions are living close to each other, i.e. are highly correlated). In contrast, the regular set of centers gives a circulant Gram matrix (to the right):





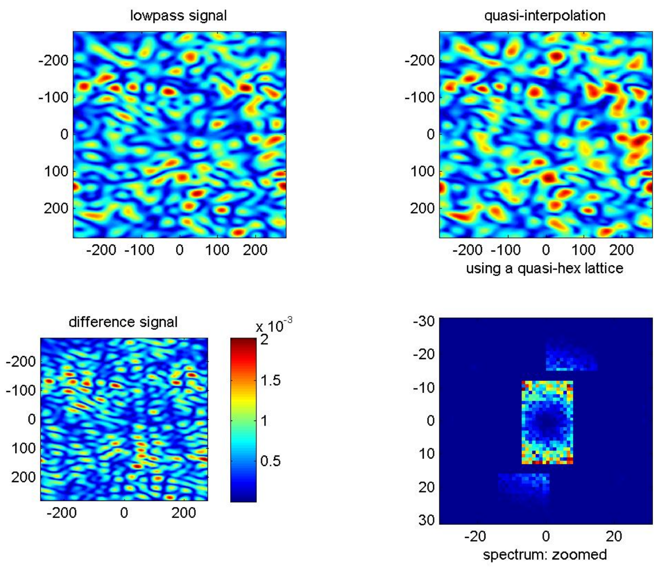
3.7 Quasi-Interpolation

The quasi-interpolation operator (comparable to Shannon's sampling theorem) is given in an abstract way as Given $h > 0$ and some prescribed function ψ on $\mathbb{R}^d$, such as a B-spline, the quasi-interpolation $Q_h f = Q_h^\psi f$ of a continuous function f on $\mathbb{R}^d$ is defined by

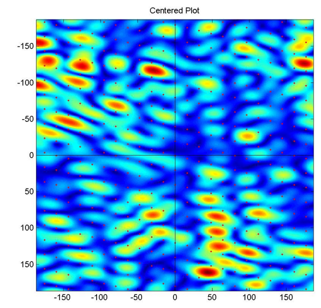
$$Q_h f(x) = \sum_{k\in\mathbb{Z}} f(hk)\psi(x/h - k), \qquad x \in \mathbb{R}^d. \tag{3}$$ eqdefQh

For suitable ψ, this formula describes an approximation to f from its samples on the fine grid $h\mathbb{Z}^d \subset \mathbb{R}^d$.

Of course it can be done on general lattice



The sampling structure (a discrete subgroup, similar to a hexagonal lattice, so we often speak of quasi-hexagonal lattices) is indicated in the subsequent image (zoomed in around the center):



In fact, we have adapted the $2D$ Gauss function in this setting in such a way that it modified to form a partition of unity (by the division through the periodized version of the function). Alternatively one could try best approximation (using `TRL2AP.M`), i.e. project the given (smooth) signal onto the linear span of the shifted copies of the standard $2D$ Gauss function (`g(:)*g(:).'`.

3.8 Poisson's formula and Shah distribution

One of the main principles of Fourier analysis is the fact that sampling one one domain (say the so-called time-domain) corresponds to periodization on the spectrum (the Fourier transform), so on the Fourier domain (and vice versa).

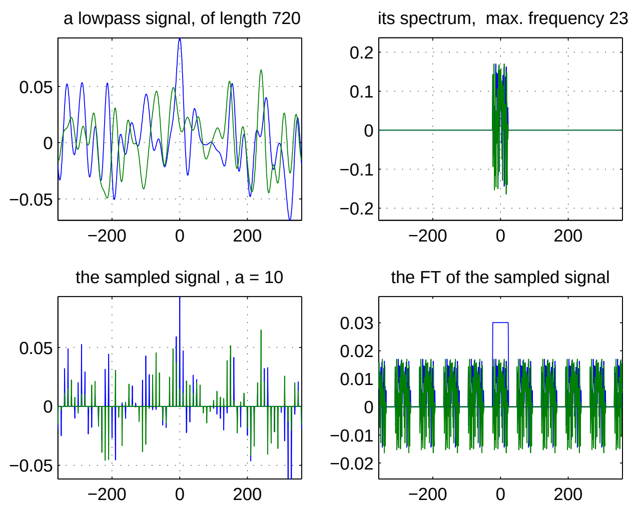
It is based on two things:

- The Shah-distributions goes to the Shah-distribution on the Fourier transform side;
- Convolution and pointwise multiplication change their role (the so-called convolution theorem of Fourier Analysis)[6]

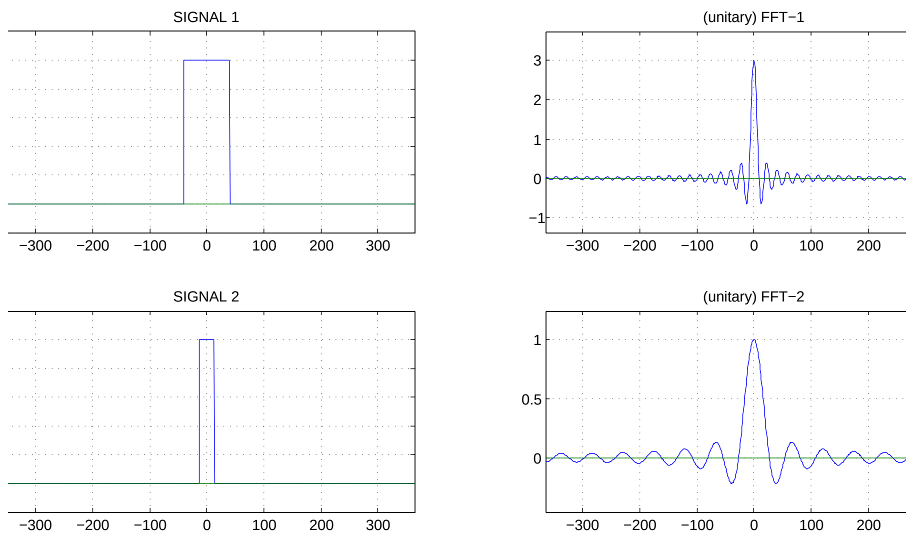
3.9 Shannon's Sampling Theorem

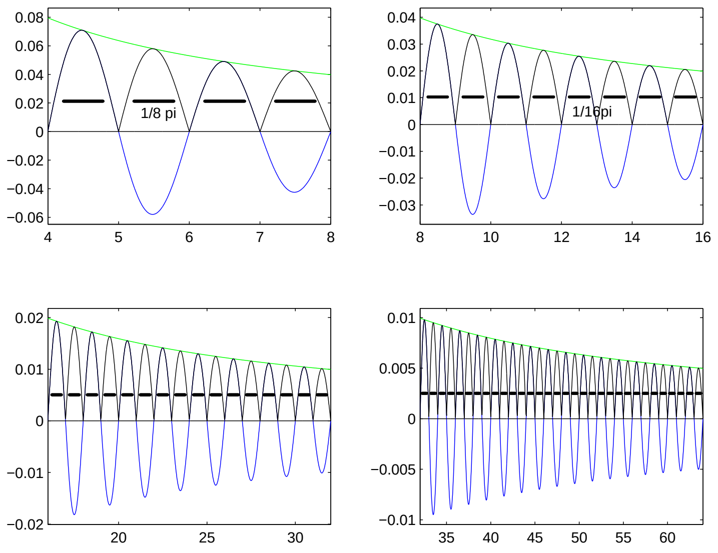
Bandlimited functions are exactly reconstructed from the samples using the so-called SINC-function.

First the effect of sampling in the frequency domain (namely periodization in the frequency domain) is demonstrated:

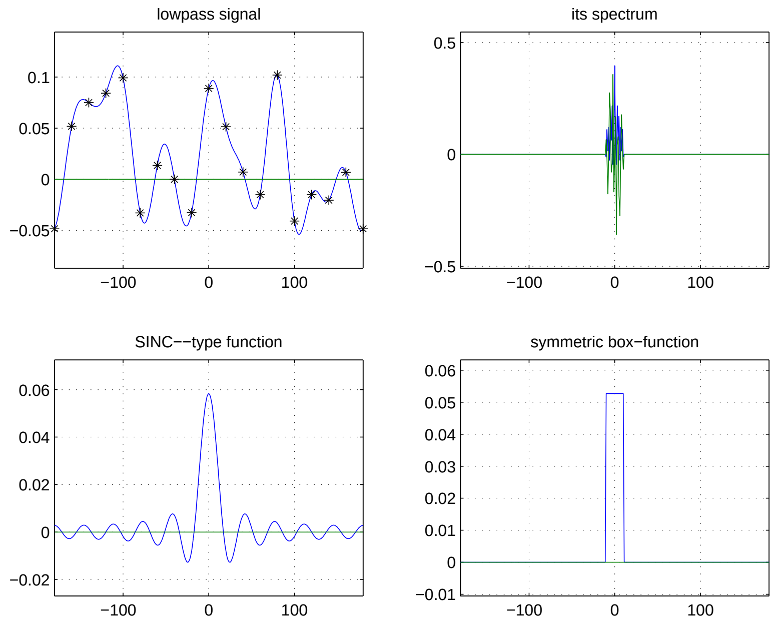


[6]The best way to understand it is via the *polynomial connection:* Convolution corresponds to do the multiplication of polynomial in the form of a *Cauchy product*, i.e. in the same way as you multiply integer number!, whereas on the Fourier transform side we do a pointwise multiplication of polynomial functions over the unit roots of order N.

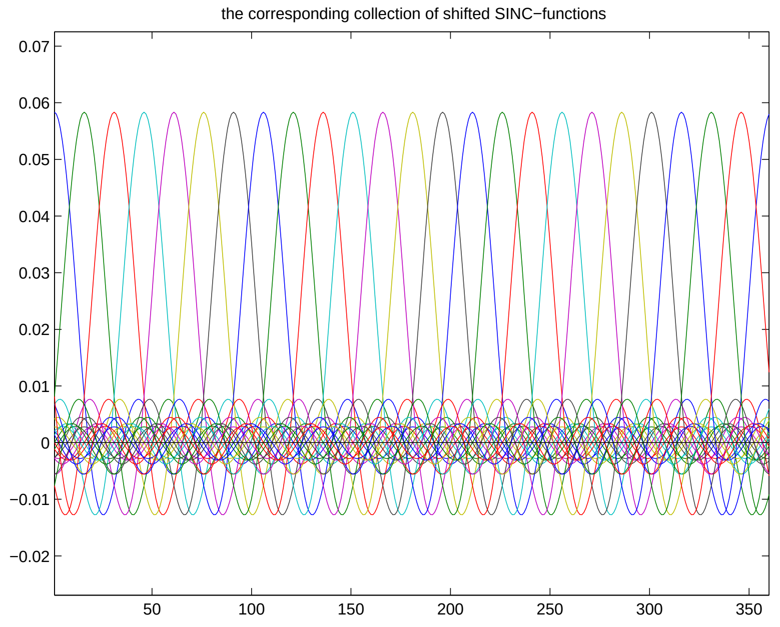




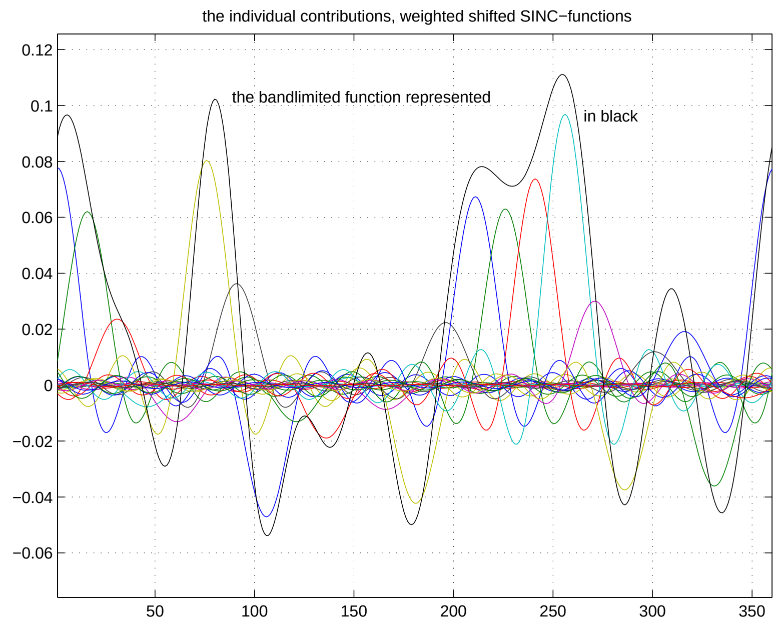
Let us now look at the situation describe by the Shannon sampling theorem: A (in this case real-valued) bandlimited function is given (signal length $n = 360$, maximal frequency $maxfr = 10$, hence 21 degrees of freedome. Sampling distance $a = 15$, hence we obtain $24 = n/a$ sampling points, marked in black in the plot.



Let us look at the corresponding family of shifted SINC-functions:

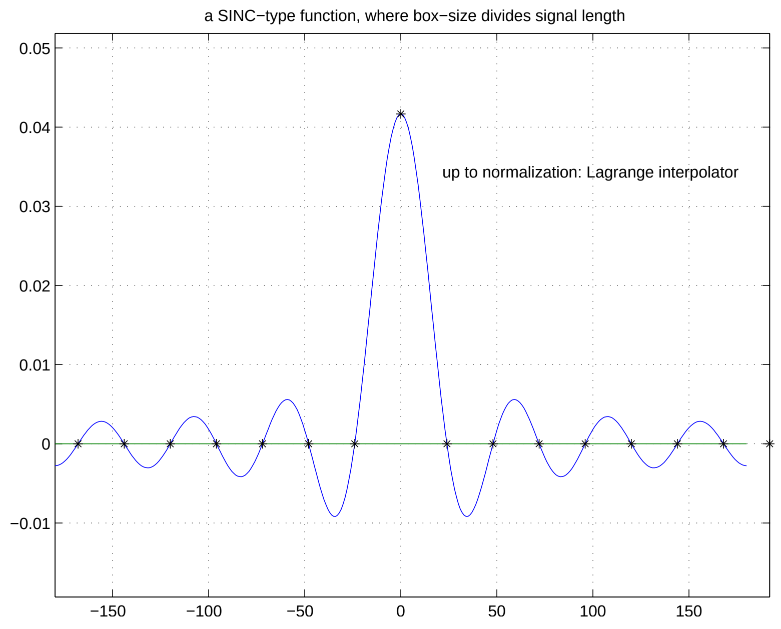


The weighted contribution making up the band-limited signal are as follows:

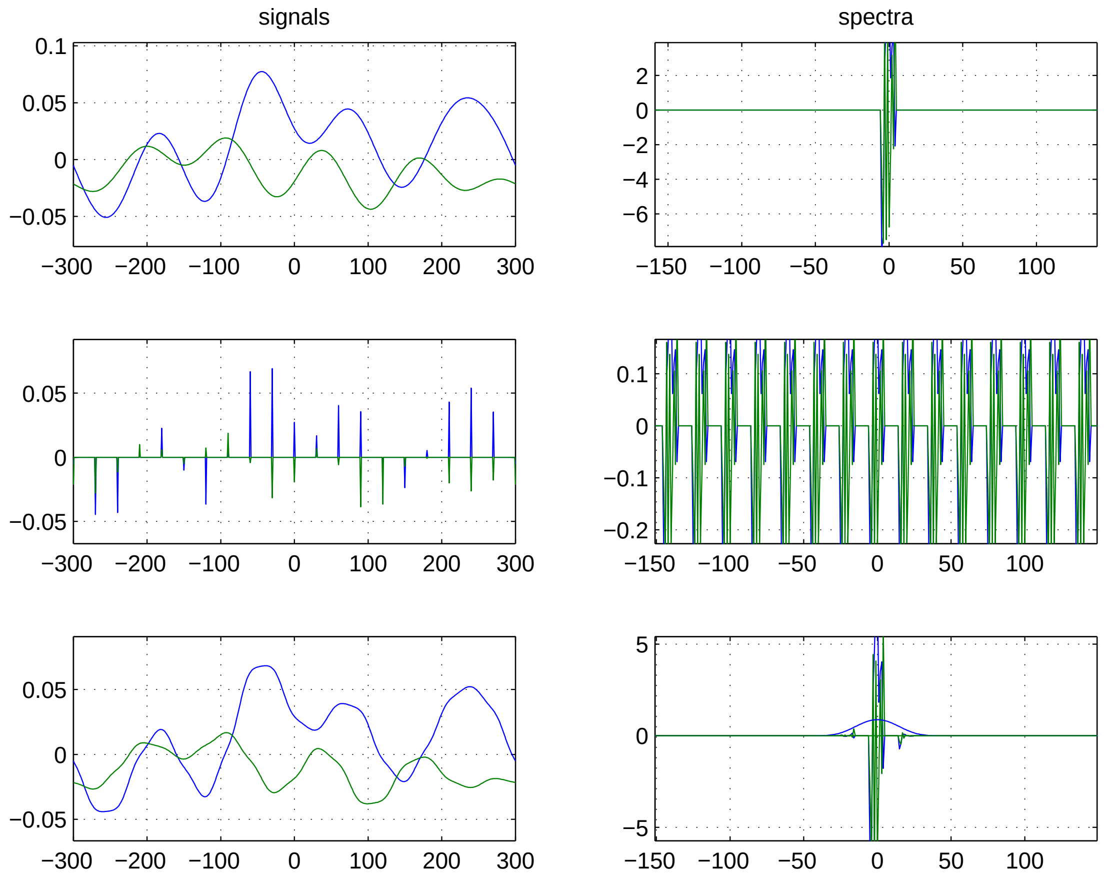


It is not surprising that this expansion is not at all very localized, due to the bad localization of the SINC function.

In the continuous case the SINC function has the property that it is a Lagrange interpolator, i.e. for the integer lattice one has $SINC(k) = \delta_{0,k}$ (Kronecker's delta). In the discrete (finite) case this is only the case if the length of the plateau function (which is $2maxfr$, so always an integer) is dividing the signal length. The example below shows the case $maxfr = 7$, and corresponding sampling rate $= n/15 = 24$.



This perfect reconstruction method is in fact much better than just "ordinary quasi-interpolation" (demonstrated by another example), showing the effect of sampling (periodization of the spectrum) followed by quasi-interpolation.



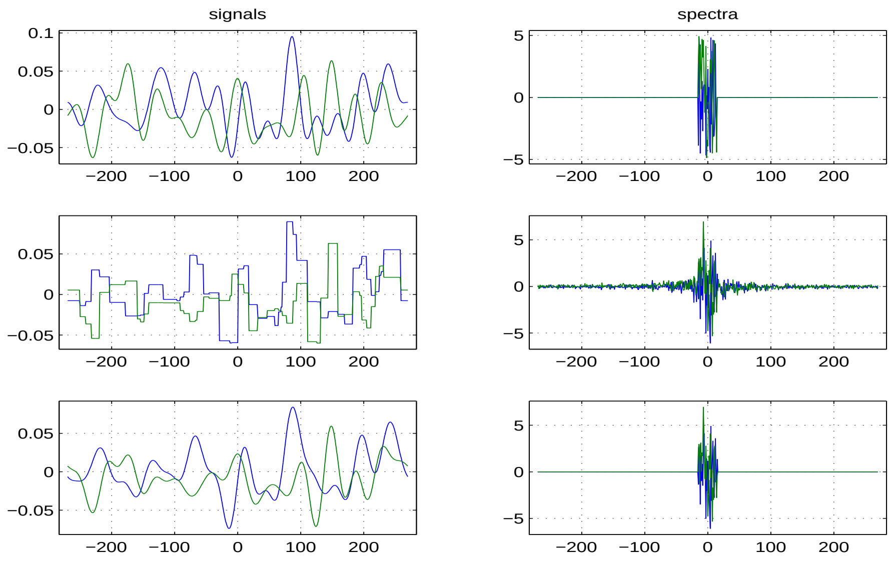
3.10 Better localization using smoothed box-functions

The poor decay of the SINC-functions (the inverse FT of the box function) suggests to perform the reconstruction (isolation of the main contribution of the series of shifted copies of the spectrum) using a smooth plateau- function for cutout.

Here I hope that Andreas will provide some input (or later Daniel), based on the BUPUSPLINE-routine...

3.11 Irregular Sampling

In practice it is often necessary to recover a signal not just from strictly regular samples, but one has to deal with irregular samples. A simple (Fourier based) "first generation" algorithm for recovery is the so-called Voronoi-method (based on nearest neighborhood interpolation). It is an iterative algorithm, where in each iteration the following procudure is applied: (a) sample (b) do step-function (Voronoi) interpolation (c) do a smoothing (d) check the remaining sampling error.



Similar methods can be applied in order to do the recovery of function in spline-type spaces from irregular samples. (cf. the work of Aldroubi, Feichtinger, Gröchenig).

More recent methods (like the ACT method) are based on ideas from frame theory and conjugate gradient methods.

M-file name	Action realized
TRL1AP	best approx. by translates
TRL1DU	dual generator for spline space
TRL1FR	S frame operator for frame TRL
TRL1INT	interpolating spline generator
TRL1ONB	orthonormal basis generator
TRL1REC	Lagrange type interpolation function .
TRL2AP	$2D$ best approximation by splines
TRL2BAS	family of atoms (stack) !?
TRL2DU	dual generator for spline space in $2D$
TRL2FR	S frame operator for frame TRL $2D$
TRL2INT	interpolating spline generator in $2D$
TRL2ONB	orthonormal basis generator in $2D$
TRL2REC	Lagrange type interpolation function
TRL2SYNT	reconstruction = synthesis from cofs.

4 Gabor Analysis

Gabor Analysis is concerned with the properties of families of functions which are generated from a single (or a finite set of) atom(s) by the application of a family of TF-shift operators.

Particularly important are *Gabor Riesz Bases* (which are imporant for mobile communication), and *Gabor Frames* (which are used in order to expand arbitrary signals).

For a long time Gabor analysis (names after the idea of D. Gabor, [**?**]) was considered an interesting concept allowing to interpret coefficients as the energy contained in a signal at a given time and at a certain frequency (similar to to a *musical score*), with the disadvantage of being difficult to compute and numerically instable.

There is a lot of algebra behind Gabor analysis, and in a discrete setting it is important to work with signal length which are rich of divisors and hence the fact that the discrete *phase-space* $\mathbb{Z}_N \times \widehat{\mathbb{Z}_N}$ allows for a large variety of subgroups.

4.1 Basic Time-frequency Operations

Time-shift operators (cyclic shifts in the case of finite, i.e. periodic sequences) and corresponding modulation operators (frequency shifts).

4.2 The Short-time Fourier transform

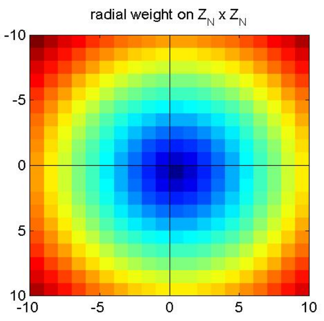
The **Short-time Fourier Transform** is probably the most important tool for the analysis of signals, by looking at the spectrogram. It is even built in in the MEDIA WAVPLAYER, or at a more sophisticated level in the ARI software package STX^{TM}.

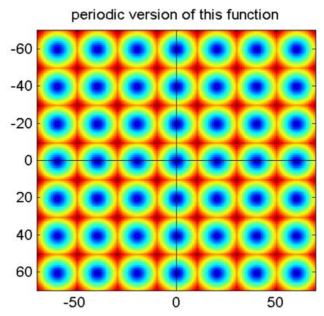
4.3 Regular and Irregular Gabor Families

We have different routines to build Gabor families.

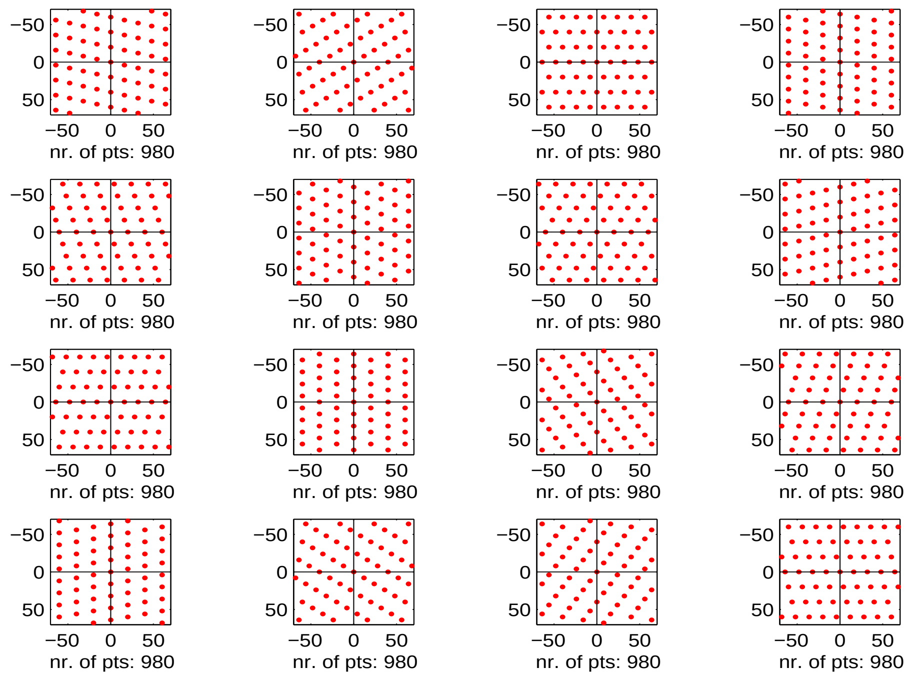
First of all let us recall that Gabor families arise by applying a family of TF-shift operators to a single (later a finite family) of so-called **Gabor atoms**, or *windows*.

In order to visualize the corresponding regular or irregular Gabor families let us first display them in the usually (centered) mode (using the `imgc`-command).

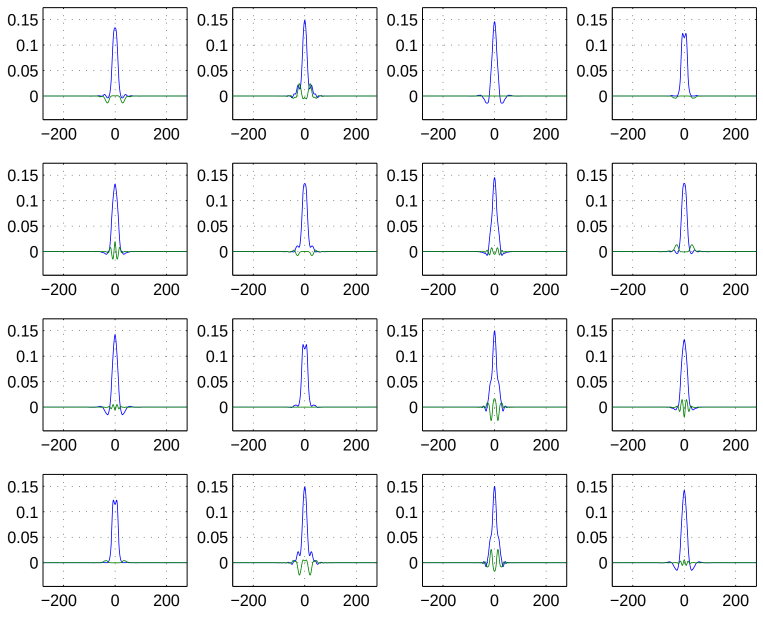




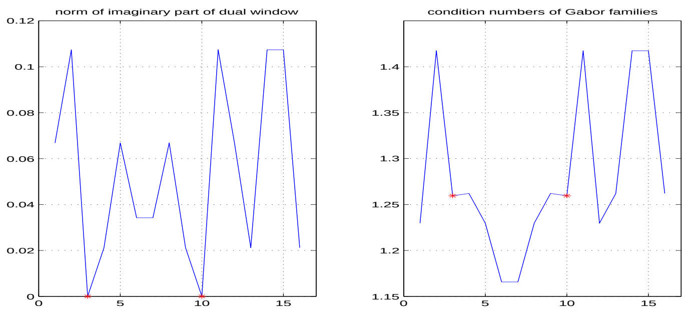
For a “good signal size” (such as $n = 560$ and redundancy $1.75 = 7/4$) one finds altogether 16 nice subgroups with 980 points. We show only a segment of the lattice, giving an idea about the orientation resp. similarity to hexagonal like atoms.



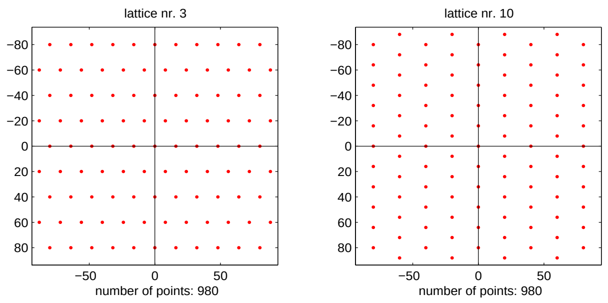
The corresponding collection of dual atoms looks as follows:



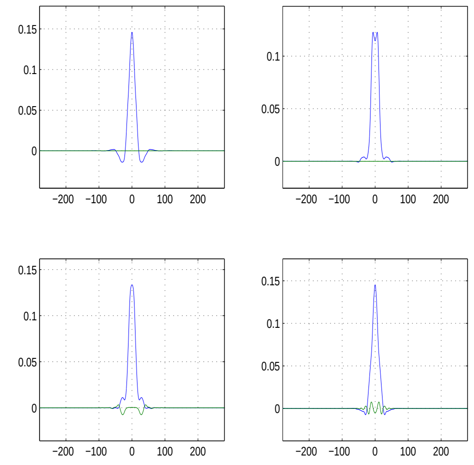
Looking at the norms of the imaginary part and the condition numbers we get



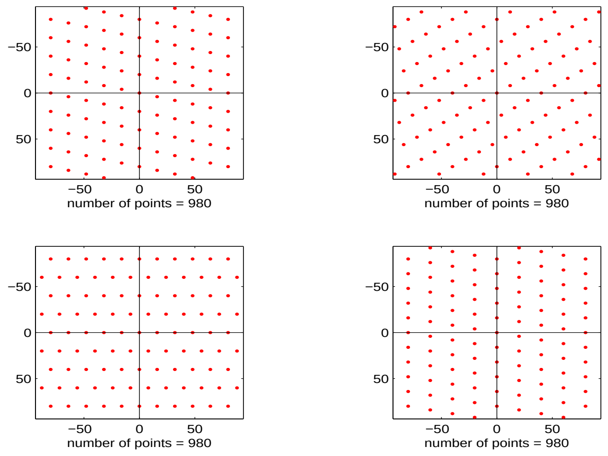
It shows that some (in fact most of those) dual windows are not real-valued, except for two of them. Let us therefore pick out those two lattices which induce real-valued dual windows. Maybe not completely surprising we find that these are exactly the lattices which are more or less like hexagonal lattices, in one or the other orientation:



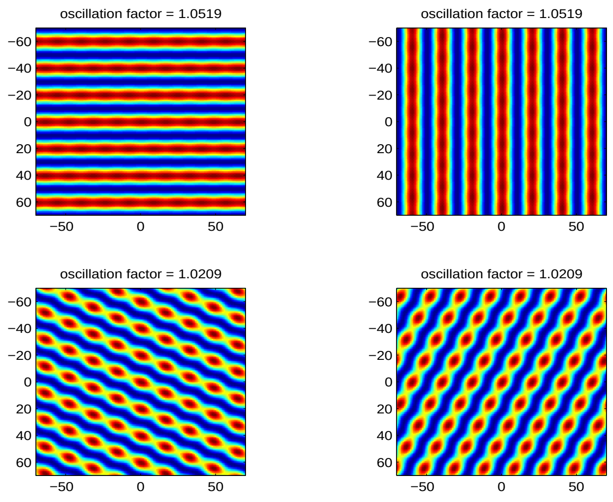
If we pick out the most interesting ones we find, that the duals (nr.3 and nr.10) which are real, and the two which provide the best condition number (nr.6 and nr.7):



and also to recall once more, let us look for the corresponding sampling pattern:



One possible way to estimate the quality is to find out, how the “Gaussian density” behaves, i.e. what the convolution between the given sampling pattern and a standard $2D$ Gauss function is behaving (whether it is closer to a constant function or having more oscillation).

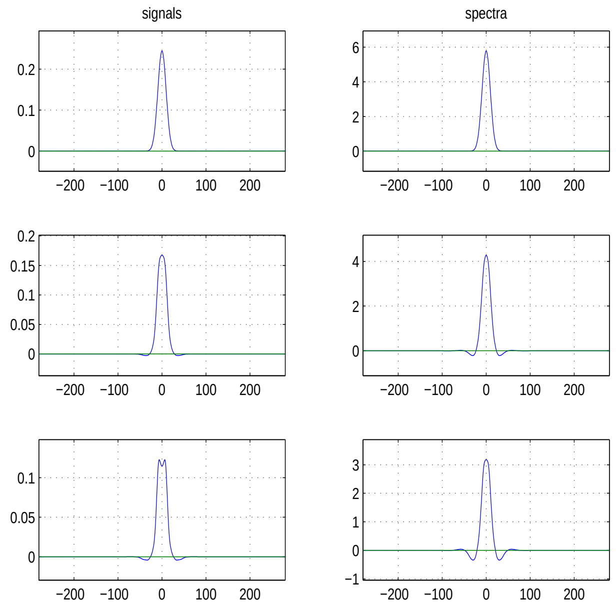


4.4 Gaborian Riesz Basis

Assume that a *regular Gabor family*, i.e. a family of TF-shifted copies of an atom along some lattice, is a Gaborian *Riesz basic sequence* (i.e. a Riesz basis for its closed linear span). Then it has a (canonical) biorthogonal system, which turns out to be another Gabor system.

4.5 Gabor Frames

Each Gabor family which is a frame has a dual frame which IS itself a Gabor frame. There is also another window which is tight, which is as well a Gabor frame.



These images show on the left hand side the (Gaussian) window, in the middle row the tight window, and in the third row the dual window. In the corresponding row on the right hand side one sees the FT of the corresponding window.

The fact that the Gaussian window is invariant under the unitary version of the FFT implies that one can also interpret the right hand side as describing the shape if the dual window for the case (b, a) instead of (a, b).

In the concrete case we have $a > b$ (not much), and hence the *inverse dip* in the center of the dual window, while $b > a$ introduces more and more sidelobes.
NuHAG M-files: F3SP, GABDDD or PPDW, GABTGTMH

4.6 The Wexler Raz relation

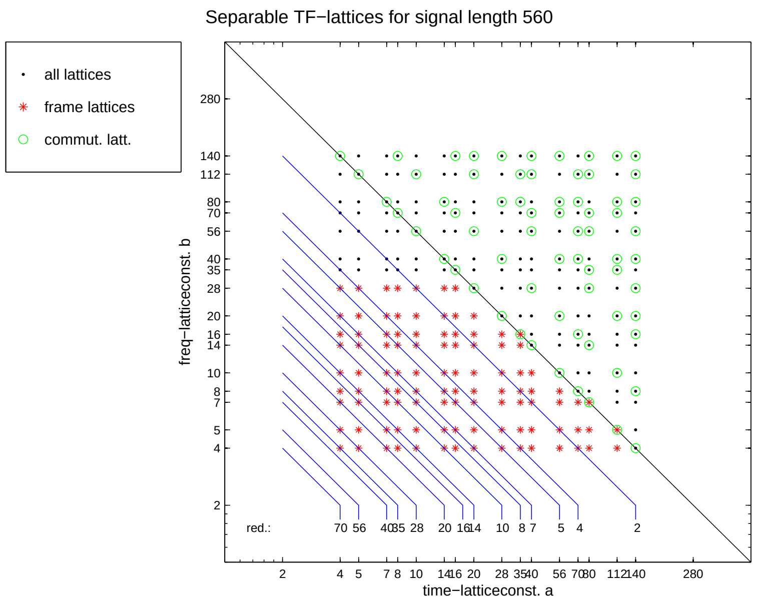
Theorem (Wexler-Raz Biorthogonality Relations). Assume that the synthesis operators D_g and D_γ are bounded on $\boldsymbol{\ell}^2(\mathbb{Z}^{2d})$. Then the following conditions are equivalent:

1. $S_{g,\gamma} = S_{\gamma,g} = \mathbf{I}$ on $\boldsymbol{L}^2(\mathbb{R}^d)$.

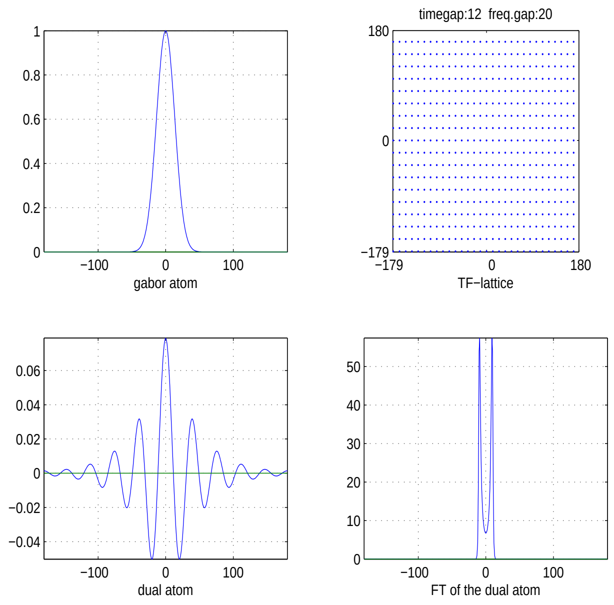
2. $(\alpha\beta)^{-d}\langle\gamma, M_{l/\beta}T_{n/\alpha}g\rangle = \delta_{l,0}\delta_{n,0}$ for $l, n \in \mathbb{Z}^d$.

Details are in Charly's book: [5], Theorem 7.3.1.

The following plot is the user-interface for a routine which allows to take a look at



Each read symbol $*$ represents a pair of lattice constants, which generates a Gabor frame for the Gauss-window. The dual window and its Fourier transform (equal to the dual window for the group with lattice constants (b, a) instead of (a, b)) as well as the TF-lattice itself are represented:



The shape of dual atoms is typical. The right lower window is typical for large a compared to small b, while the left lower (dual) window is typical for large b compared to a. For both a and b relatively small one has on the one hand high redundancy, but on the other hand dual windows which look very similar like a suitable normalized version of the original window itself.

Exercise: find out, what the optimal scaling of the window is, in order to use it as an approximate dual window (norms), etc..

NuHAG M-files: SIDEDIGM, SIDE2MAT

TOPIC: alternative (non-separable groups) are easily obtained by applying the *group automorphism* by the name SIDEDIGM (and its inverse, named SIDE2MAT) allow to generate non-separable lattices.

TEST: `compnorm( adjlat(sidedigm(LAM)), sidedigm(adjlat(LAM)));`
and consequently the same for the inverse transform:
`compnorm( adjlat(side2mat(LAM)), side2mat(adjlat(LAM)));`

4.7 The Janssen Representation

The frame operator $S = S_{g,\Lambda}$ satisfies an important commutation law, namely

$$\pi(\lambda) \circ S = S \circ \pi(\lambda), \forall \lambda \in \Lambda \tag{4}$$ `commutSLam`

This implies that the Gabor frame operator, or more generally operators of the form $x \mapsto \sum_{\lambda \in \Lambda} \langle x, g_\lambda \rangle \gamma_\lambda$ have a specific form, which is useful when it comes to solve the linear equation $S\tilde{g} = g$ for $\tilde{g}$ (because it means that S has an explicit sparsity structure).

In order to inspect the Gabor frame matrix let us consider $Sggam$ (the operator described above), which is (using right matrix multiplication) of the form $Sggam = G1' * G2$, where $G1$ and $G2$ are the collection of row vectors describing the Gabor family (for g and γ respectively).

4.7.1 Tight Gabor families

Especially for the applications *tight Gabor atoms* are of great value. They have a number of advantages over general windows. In particular, they allow to have a symbolic calculus (i.e. the *tranfer of properties* from the (so-called *upper* symbol of a Gabor multiplier to the properties of the corresponding operator (Anti-Wick operator with weight/symbol W, or STFT multiplier in the continuous case, or Gabor multiplier, in the phase-space discretized situation)[7].

It is interesting to observe that the composition of Gabor multipliers (i.e. the matrix multiplication of the corresponding matrices) is resulting in a non-commutative set of operators. So (in particular), not even the composition of an operator with weight W with another of the form $1/W$ (which is supposed to undo the first one) will give the identity operator, not even for Gaussian windows.

Let us therefore consider some interesting properties of the tight window, and how to obtain it.

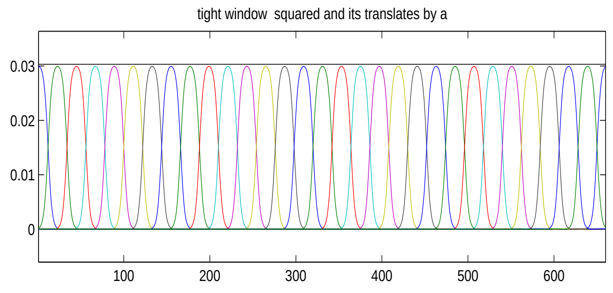
First of all let us note that there are many ways to obtain the tight window, given the lattice Λ. The canonical description is to say that one is applying the inverse square

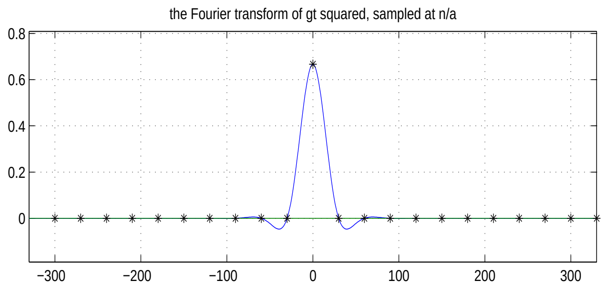
[7]Using the engineers view-point one is multiplying the part of the signal which is going through each of those channels by number which are allowed to vary continously over time: this is in fact exactly what an audio-engineer is doing, when he is moving the sliders of his device slowly over time, in order to adapt e.g. from one style (say rock) to another (say classic), or in order to bring out high frequencies better at a moment where the orchestra is playing a particular piece, with the main melody in the flutes.

root (or the square root of the inverse) of the frame operator, which is positive definite. There are many ways of doing so. The naive one builds the Gabor matrix and then takes the square root, applying it to g.

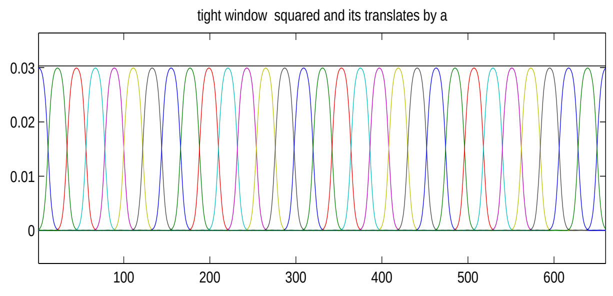
The command `f3sp(g,gt,gd);` which is part of the GABINIT routine shows how all three functions look like and confirms that `gt` is - in terms of shape - halfway between `g` and `gt`.

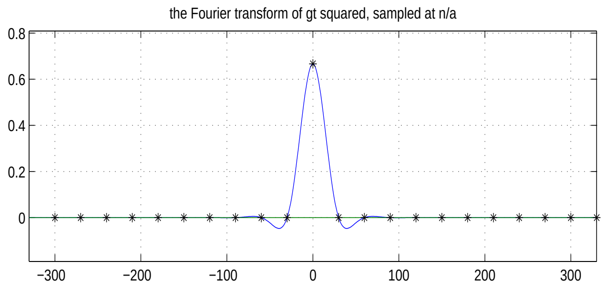
Since the frame operator has the particular property that its main diagonal is just the sum of the a-shifted copices of the atom g squared, this must add up to one. But the tight window for the lattice with parameters (a, b) is also a tight window for the lattice (b, a), if g is Fourier invariant (as it is the case of the Gaussian).





Since the BUPU property on the time side corresponds to an interpolating property on the frequency side, we also have, that the Fourier transform of `abs(gt).`^{2} is a Lagrange interpolator for the orthogonal lattice, i.e. is different from zero at zero an vanishes over the rest of the orthogonal lattice (which can be obtained in general using the [new] command `orthgrp`.





These plots use the following recent/new M-files:
NuHAG M-files: PLOTSUMS, PLOTCSMP, TRLBAS

There is one more connection between tight Gabor frames and orthonormal Gabor Riesz basis. While the tight Gabor family is obtained by the Gabor family, the corresponding family over the adjoint lattice can be normalized in the most stable way (implemented in the `ORTHBEST` routine, which is just a realization of the *Loewdin* orthonormalization.

For general frames one can show, that the approximation of a system of linear vectors (i.e. a Riesz basic sequence, or a Riesz basis for its closed span) by any orthonormal basis for the same space is achieved by making use of the inverse square root (= the square root inverse) of the Gramian matrix. Although in principle one can do this, using the `SQRTM`-command, one can think of the obtaining the

TEST: `GA = gabbasp(g,n/b,n/a); GAO = orthbest(GA); gat = GAO(1,:);` Then a comparison with `gt = gabtgtmh(g,a,b)` shows that - up to normalization - they are equal.

A typical timing would be:

```
>> tic; g*inv(sqrtm(S)); toc
Elapsed time is 1.526868 seconds.
>> tic; g*inv(sqrtm(S)); toc
Elapsed time is 1.506410 seconds.
>> tic; gabtgtmh(g,a,b); toc
Elapsed time is 0.122223 seconds.
>> tic; gabtgtmh(g,a,b); toc
Elapsed time is 0.045629 seconds.
```

Note that calling a routine twice tells you more about the execution time, because the first call may spend significant a fraction of time in searching for the variables and M-files involved in the command.

Timing can be sometimes (for single calls) be deceptive, as the following simple (but concrete) example indicates

```
>> tic; gabtight(gaussnk(720),18,18); toc
Elapsed time is 0.504355 seconds.
>> tic; gabtight(gaussnk(720),18,18); toc
Elapsed time is 0.050946 seconds.
>> tic; gabtight(gaussnk(720),18,18); toc
Elapsed time is 0.006483 seconds.
```

This shows that after two calls the procedure is almost 100 times faster!

4.7.2 Double preconditioning

The idea of the method of double preconditioning is based on the fact, that in many cases the diagonal part of the frame operator S is dominant (especially when b is not too big, resp. when the number of channels is high enough). Equivalently the Fourier version (the description of the matrix in the Fourier basis) can have the same property is a is relatively small. But this turns out to be equivalent to the idea that the Gabor frame matrix is well approximated by a convolution matrix. Since preconditioning in general is the idea of bringing a matrix closer to the identity operator by multiplying it with the inverse of some simple (simply to invert, to be more correct) matrix close to the orginal matrix, one can do this in the present case in two ways:

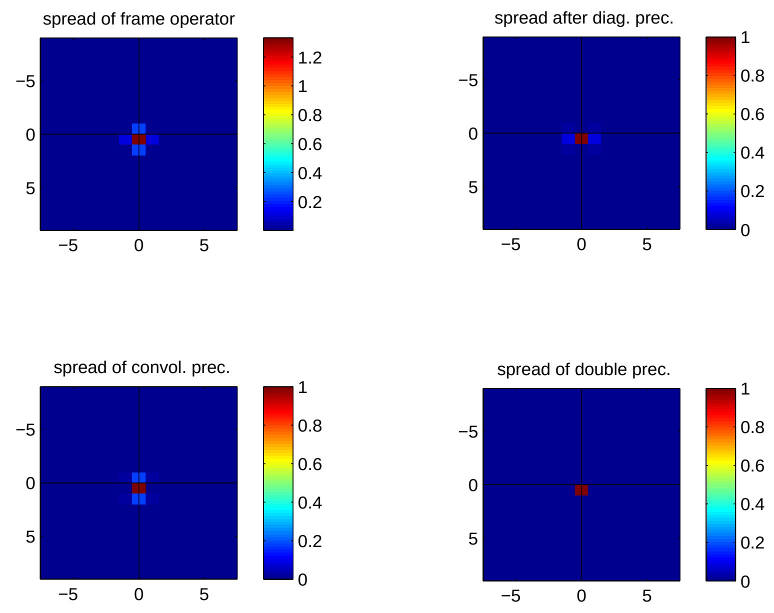
- approximate S by its diagonal part, which is in MATLAB nothing but `diag(diag(S))`, so we are forming (at least on paper) the matrix `S * diag(oneover(diag(S)))` ; but since one has explicit knowledge of the diagonal of the frame operator the direct way of doing this is to form `S * diag(oneover(maindiag(g,a,b)));` , which is shown in its spreading representation in `subplot(2,2,2)` below.

- The same can be done either with convolution (one has to invert the nearest convolution matrix, which is of course obtained by taking the mean over the side-diagonals, and then invert - using a small FFT - the corresponding circulant matrix, or *alternatively* does the diagonal preconditioning trick on the Fourier version of the frame operator, which one could - just for inspection, do as follows:
`FU = fftu(n); FSF = FU * S * FU'; ....`

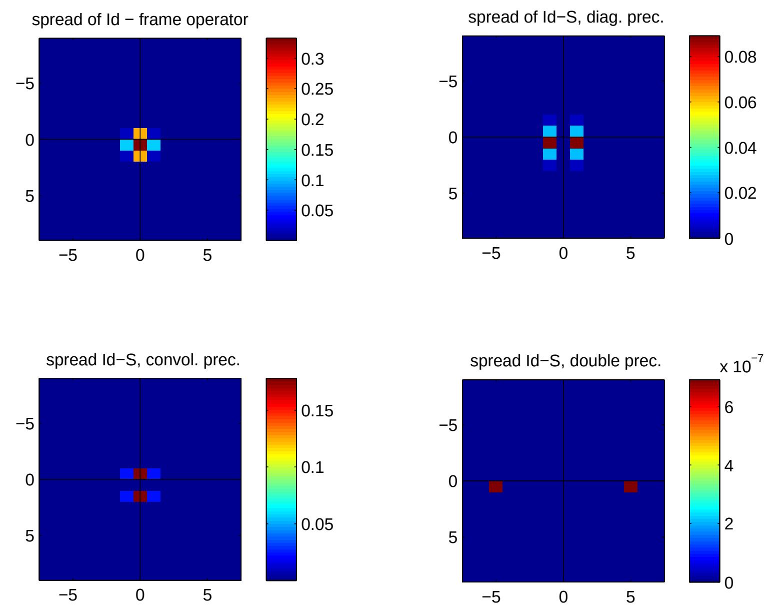
The idea of double preconditioning is then based on the observation that one can apply both preconditioners. Although the order is having some influence on the result (the procedures do not commute) the effect in terms of direct improvement is quite comparable and depends on the concrete situation (g, a, b). Since the action of the preconditioner is easily executed directly on the atom (by pointwise multiplication of the given atom on the time resp. the Fourier transform by the inverse of a suitable periodicfunction) this is a quite flexible way. Also, from the operator theoretic point of view the inversion makes use of commutative subfamilies within the family of all operators (like the Gabor frame operator) which commute with the family of TF-shifts from the original lattice Λ determined by (a, b).

Showing the change of the frame operator in the Janssen domain by single and double preconditioning[8].

[8] cf. below for the details on the spreading domain: The ideal case is the identity operator, which corresponds in the spreading domain to the Dirac at $(0, 0)$, which is the unitmatrix $unitmat(n, 1, 1)$.



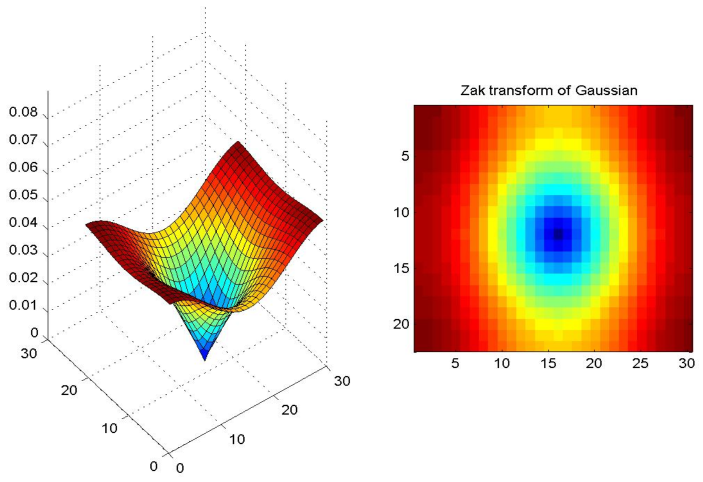
The difference to the identity operator is shown (again in the Janssen domain) looks like this:



This is obtained using the NuHAG routine `gabprec7.m`.

4.7.3 The Zak transform

One can/should note that for the extreme case $b = n/a$ one has *commutative Banach algebra* of matrices, hence by the abstract *Gelfand theory* a representation as a space of pointwise multipliers. This is in fact provided by the ZAK transform, which is a very good tool, included both in the LTFAT toolbox as well as in the NuHAG tools.



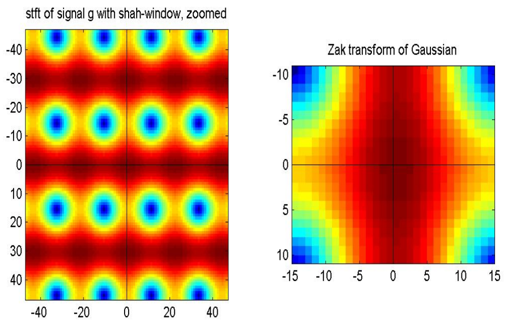
A verification that the action of the frame operator corresponding to the family $(g, a, n/a)$ is just multiplication with the absolute square of the Zak-transform of the atom g is done as follows (described in row mode):

```
xx = randcn(1,n); Zxx = zak(xx,a); Zga = zak(g,a);
Sa = gabfrmat(g,a,n/a); xxSa = xx * Sa;
[ here Sa == G'*G; for G = gabbasp(g,a,n/a); ]
ZxxSa = zak(xxSa,a);
>> compnorm( Zxx.* abs(Zga).^2, ZxxSa)
quotient of norms: norm(x)/norm(y) = 0.0015152
difference of normalized versions  = 3.8354e-016
ans =  3.8354e-016}
```

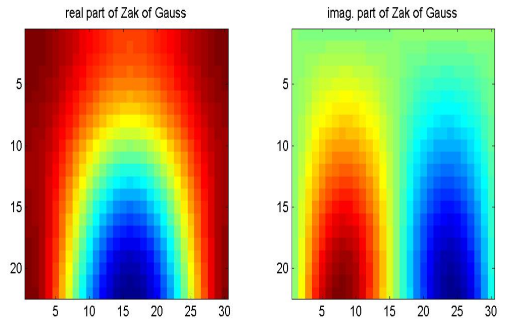
Note the the generators of the corresponding lattice commute:

```
Ta = tfmatr(n,a,0); Mb = tfmatr(n,0,n/a);
compnorm(Ta*Mb, Mb*Ta);
```

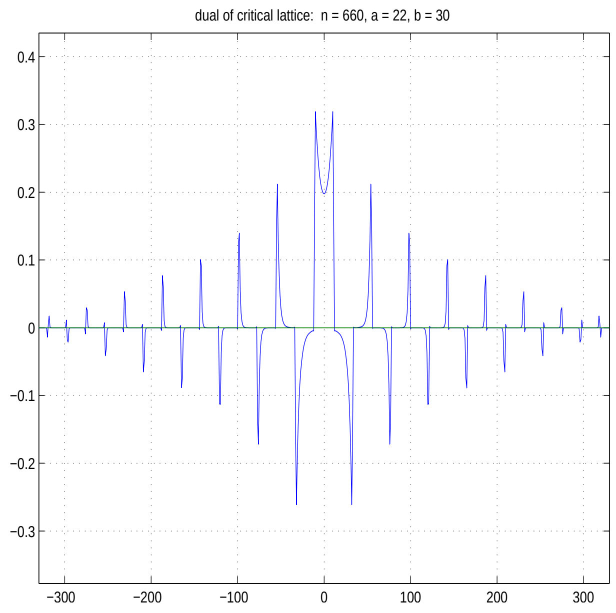
In fact, one can view the Zak-transform as a function on the whole phase space (TF-plane), namely as the Short-time Fourier transform with a rather unusual window, namely the SHAH-distribution for the lattice with lattice constant a (i.e. `sha = shah1(n,a);` in the NuHAG TB. Let us take a look at this:



The periodicity properties of the Shah-distribution go over into corresponding (quasi-)periodicity properties of the Zak-transform (which are not seen in the display of the absolute values). Let us therefore look at the phase behavior of the Zak transform (which has a zero at the midpoint):



The main disadvantage of the so-called critical case is the fact, that one cannot have a nice Gabor atom (say a Schwartz function) at critical density. In fact, the closer on gets to critical density the larger the condition number will be. The computation of the dual Gabor atom (as suggested originally by Martin Bastiaans, around 1978) is possible using the Zak transform, but the result is a rather unpleasant function (which is bounded, but not square integrable, not even in any of the $\boldsymbol{L}^p$-spaces for $p < \infty$. It looks essentially like the following example $(n = 660, a = 20, b = 33)$.



4.8 Spreading Representation of Matrices

The commutation relation has a very clear and helpful analogue in the spreading function.

For comparison, let us recall: a function is periodic if and only if its Fourier transform is concentrated on the so-called orthogonal subgroup of the frequency domain. Alternatively, we can say, a signal does not change under certain time-shifts (describing the periodicity) if and only if it is built up from frequencies only which share the same property: these are exactly those of the orthogonal group. Formally we have:

rthgroupdef **Definition 1.** DEFttt

For TF-analysis also the adjoint group is of relevance.

4.9 Kohn-Nirenberg Symbol of a Matrix

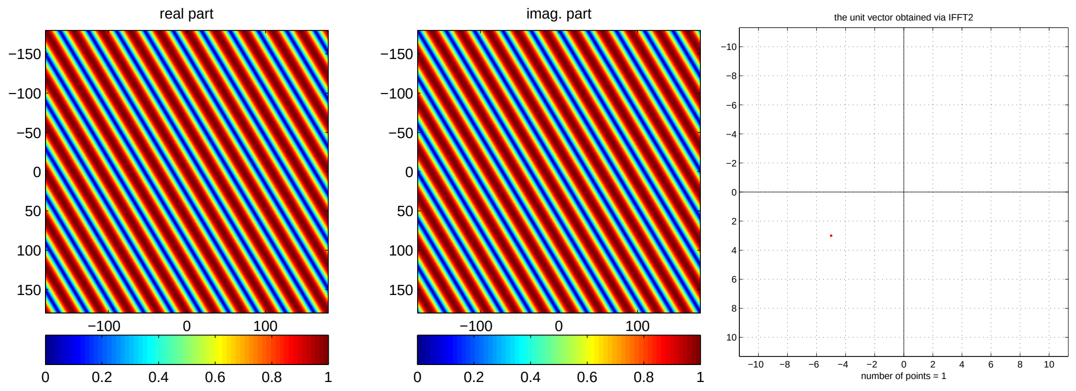
Due to the composition properties of TF-shift operators one has a particular pattern when intertwining a random matrix with a TF-shift-matrix.

Experimentally this can be verified as follows:

```
>> etaB = randcn(n);  B = spr2mat(etaB,n);
>> tf35 = tfmat(n,3,5);
>> B35 =  tf35 *  B * tf35';
>> etaB35 = mat2spr(B35);
>> compimri(etaB.*oneover(etaB35));
```

which shows clearly that there is a systematic behaviour: one of the symbols is obtain by a certain pointwise multiplication with a pure frequency. In order to find out which pure frequency it is one can apply the inverse (2D) Fourier transform to this plane wave one finds that it gives (up to normalization) a unitvector, at (toroidial) position $(-3, 5)$, vizualized by the command

```
spyc(abs(ifft2(etaB.*oneover(etaB35))) > .1);  grid; plotax; zoom(8);
```

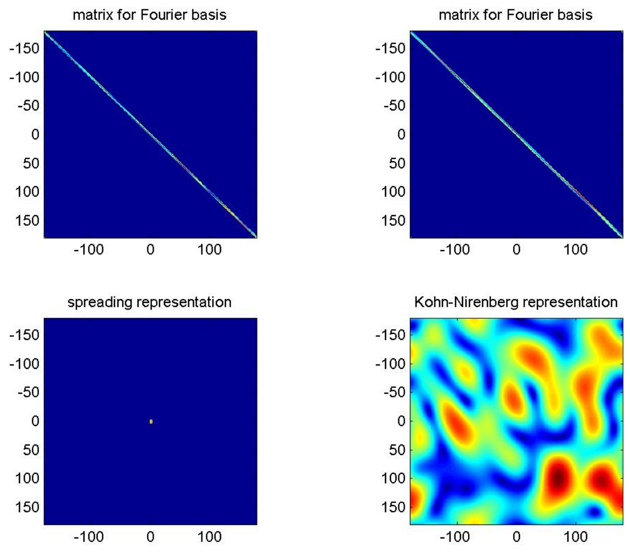


It is thus plausible to change the order and the sign of one of the entries (i.e. apply the operation $(5,-3) \mapsto (3,5)$) in order to get the position operator $(5,3)$ (in the $x-y$-sense) describing the plane wave. In fact, one has to use the symplectic Fourier transform instead of the ordinary (inverse) FFT2.

```
compnorm(ffts(etaB.*oneover(etaB35)) > .1,  unitmat(n,6,4));
quotient of norms: norm(x)/norm(y) = 1
difference of normalized versions  = 0
```

Note that the parameters $(6,4)$ mean that we are in the fifth row below the zero-row, and in the third column to the right of the zero-column!

Looking at the collection of different representations of an operator.

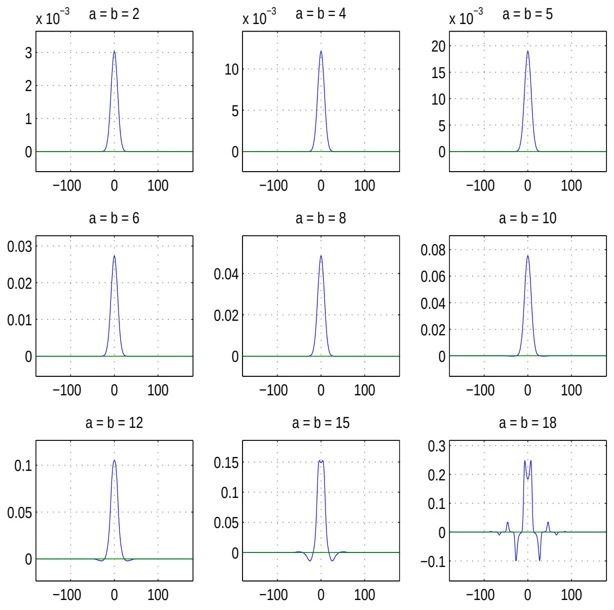


It exhibits concentration of the operator (both in the time and the Fourier domain) along the diagonal (with smooth, in fact band-limited side-diagonals). There is concentration to (in this case) a small rectangular box in the spreading domain, while (since one is going from spreading to KNS via a symplectic Fourier transform) the KNS-symbol of the underspread operator (representing a *slowly varying linear system*) is a smooth, in fact 2D bandlimited function over the TF-plane (also called phase-space).

In the case of the Gauss function once can find that the absolute value of the KNS of the projection operator $P_g : f \mapsto \langle g, g \rangle$ (assuming that $\|g\|_2 = 1$) is just the simple $2D$ Gaussian $g(:) * g(:).'$. begincenter includegraphics [width=8cm,height=8cm] PDFS KNSgauPg1.jpg endcenter

4.10 Changing the lattice constant

The plot below shows, how the (shape) of the dual window changes, when a runs through the sequence $2, 4, 5, 6, 8, 10, 12, 15, 18$, from the left upper to the right lower corner. The choice $a = 18 = b$ corresponds at a signal length $n = 360$ to a redundancy factor $n/324 = 1.1111$ or 11% (only).



4.11 Summary of Gabor commands

Action taken	**NuHAG filename**	**LTFAT filename**
Building a Gabor family	GABBAS, GABBASP	TFMAT('gabor') ??
Gabor analysis mapping	STFT	DGT
Gabor synthesis	GABSYN	IDGT
Dual Gabor window	GABDDD, PPDW	xxx
tight Gabor window	GABTGTMH	xxx
Gabor multiplier	GABMULHF, GABMULMH	xxx
spreading function	MAT2SPR	SPREADFUN
operator from spr. symb.	SPR2MAT	xxx
Kohn-Nirenb. symb.	MAT2KNS	xxx
matrix from KNS	KNS2MAT	xxx
Janssen coefficients	JANSCOF	xxx

4.12 Gabor Multipliers

Gabor multipliers are linear operators (resp. matrices) which correspond to multiplication operators in the STFT-domain.

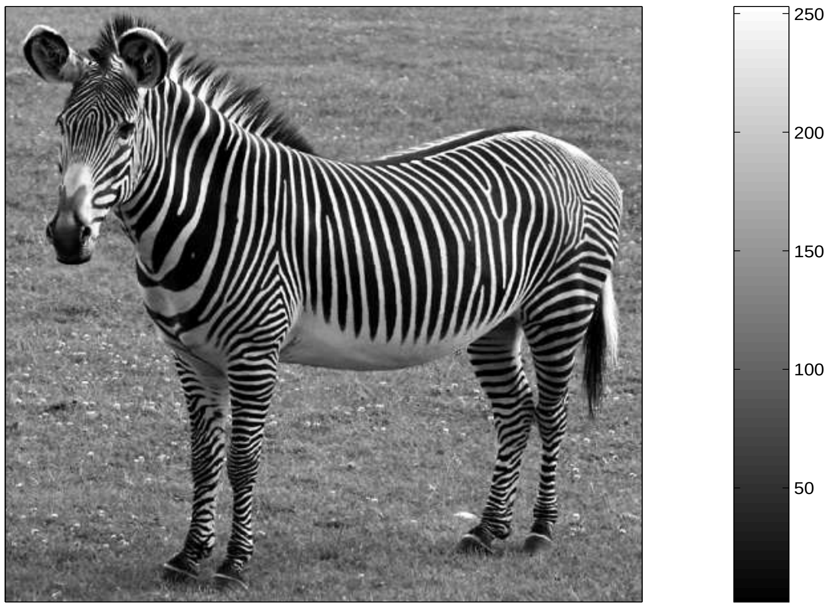
We will explain how the built them.

4.13 Gabor analysis of images

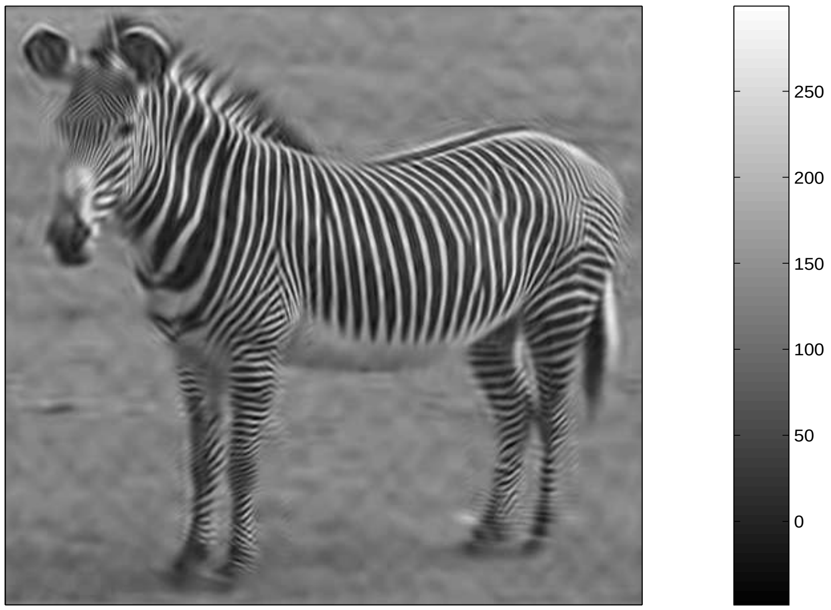
Reference to the master thesis of S. Paukner ([7]) and the article published in the Handbook of Imaging ([**?**], in [9]).

Typical example: the zebra image.

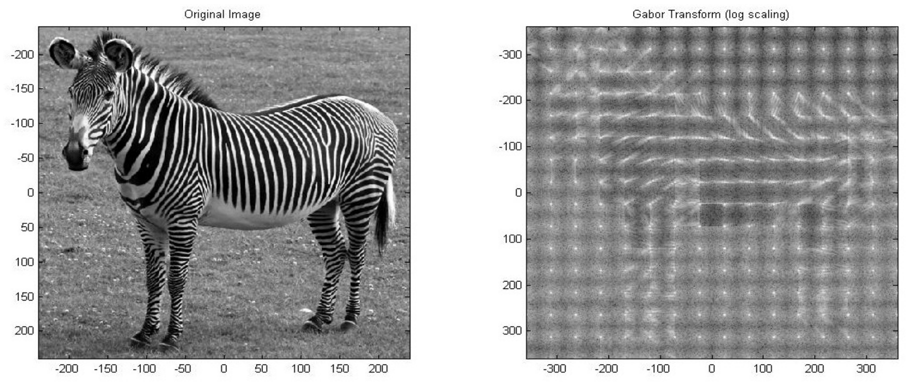
Topics: Problem of sorting the Gabor coefficients which are now labeled by a lattice in FOUR dimensions, namely the phase space $\mathbb{Z}^2 \times \widehat{\mathbb{Z}^2}$.



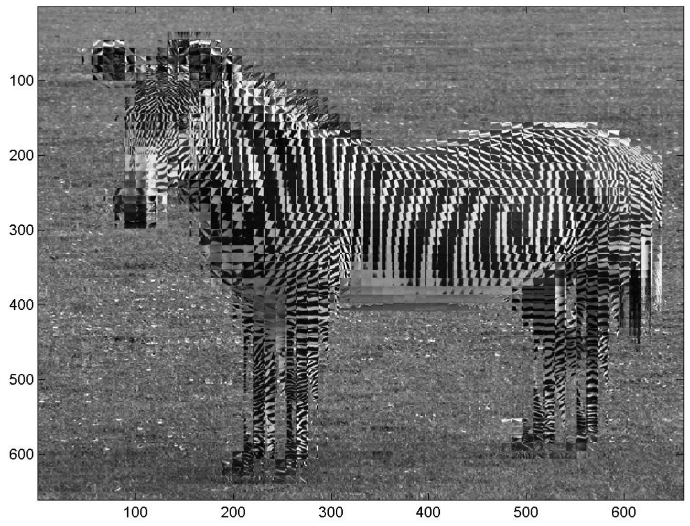
and after Gabor thresholding



Finally some impression about the Gabor coefficients of the zebra:

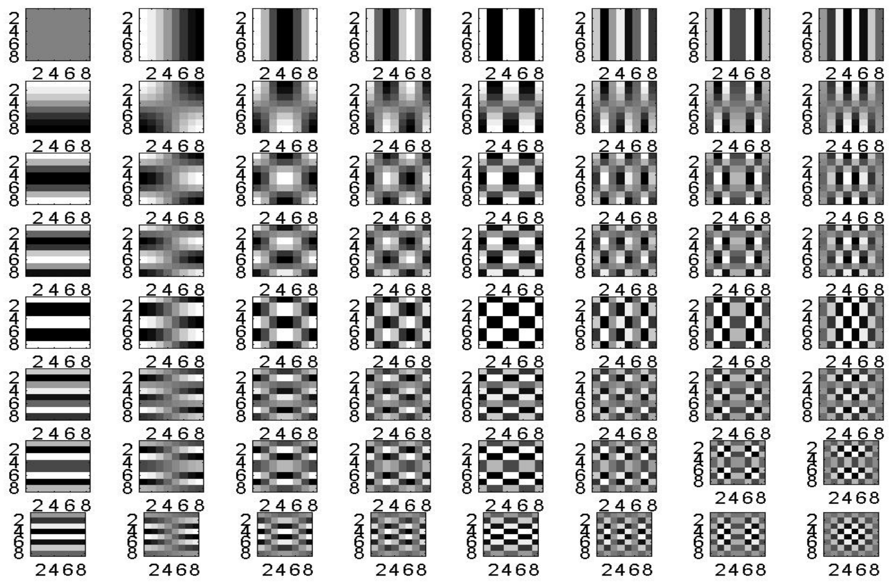


One can easily see that this compression is much more pleasant than the one obtained by the orthonormal system coming from box functions:

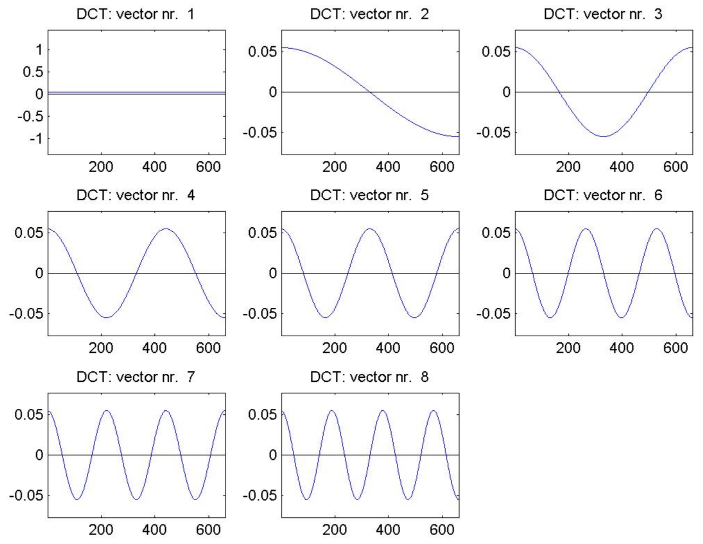


The standard compression method would be using the DCT (discrete cosine transform); The following plot can be obtained by observing that the building block of an orthogonal transform (like the DCT) is obtained by taking the inverse transform of a unit vector. In our case it is unit-matrices in the sense of a matrix (of size 8) having exactly one entry equal to 1 (and zero elsewhere), sorted according to positions.

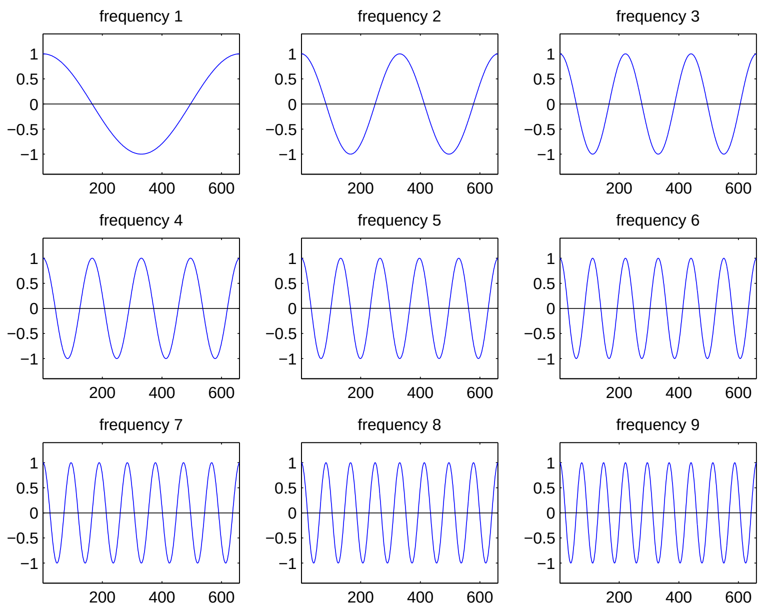
```
for jj = 1:8;
  for kk=1:8;
subplot(8,8, (jj-1)*8 + kk);
imagesc(abs(idct2(unitmat(8,jj,kk))));
  end;
end;
```



In fact, it is a separable transform based on the 1-dimensional DCT.



They are in fact obtained by pairing pure frequencies if length $2n - 1$ (the following plot shows the first 9 of them for larger n):



Just consider the first n coordinates of this family.

5 MATLAB toolboxes at NuHAG

In the course of this workshop we will use a number of MATLAB tools, which have been used at NuHAG for a long time, so most of them are very well tested. Some of them are just convenience tools, others are heavily used in order to do experimental work.

Detailed instructions of how to use them and where to download them will be given later.

NOTE: The initiator of these toolboxes (Hans G. Feichtinger, or for short hgfei) likes to work with collections of $1D$-signals given as rows of a matrix (natural way of inspecting a sequence of signals, one by one, and hence occasionally one will see right matrix multiplication. In fact, if $\boldsymbol{B}$ is a collection of row vectors with rows $r_1, \dot{;} r_m$ and $\mathbf{y} \in \mathbb{C}^m$ is another row vector, then matrix multiplication $\mathbf{y} * \boldsymbol{B}$ provides us with the linear combination of row vectors, namely the output $\sum_{k=1}^m c_k r_k$. In fact, everything done/known for column vectors can be transported into statements about row vectors by transposition (exactly), or (most of the time, equivalently, e.g. for the case of real-valued date), using the command $\boldsymbol{A} \mapsto \boldsymbol{A}'$ (MATLAB transpose conjugate), since obviously one has

$$(\boldsymbol{A} * \boldsymbol{B})' = \boldsymbol{B}' * \boldsymbol{A}'. \qquad (5)$$

transpmatrmul

[?, 5, 8]

5.1 Minor auxiliary routines

ROTRC.M rotates in a two-dimensional way:

$$\boldsymbol{A} = \begin{pmatrix} 1 & 6 & 11 & 16 & 21 \\ 2 & 7 & 12 & 17 & 22 \\ 3 & 8 & 13 & 18 & 23 \\ 4 & 9 & 14 & 19 & 24 \\ 5 & 10 & 15 & 20 & 25 \end{pmatrix} \quad (6)$$

`B = rotrc(A,1,2)  gives`

$$\boldsymbol{B} = \begin{pmatrix} 20 & 25 & 5 & 10 & 15 \\ 16 & 21 & 1 & 6 & 11 \\ 17 & 22 & 2 & 7 & 12 \\ 18 & 23 & 3 & 8 & 13 \\ 19 & 24 & 4 & 9 & 14 \end{pmatrix} \quad (7)$$

i.e. it *rotates* the matrix, viewed as a double periodic function (so to say a function on a finite version of the $2D$-torus) first in the direction of rows ("rotate rows and columns") and then in the direction of columns (to the right).

Observe that action of `rotrc(A,1,1)` is the same as conjugating with the matrix representing a shift operator:

```
T1 =
     0     0     0     0     1
     1     0     0     0     0
     0     1     0     0     0
     0     0     1     0     0
     0     0     0     1     0

     >> T1 * A * T1'
ans =
    25     5    10    15    20
    21     1     6    11    16
    22     2     7    12    17
    23     3     8    13    18
    24     4     9    14    19
>> rotrc(A,1,1)
ans =
    25     5    10    15    20
    21     1     6    11    16
    22     2     7    12    17
    23     3     8    13    18
    24     4     9    14    19
```

Checking for the compatibility with other opertions:

```
>> XX = randcn(n);
>> compnorm(  rotrc( invol(XX),-2,3), invol(rotrc(XX,2,-3)))
>> compnorm( fft2(invol(XX)), conj(fft2(XX)));
```

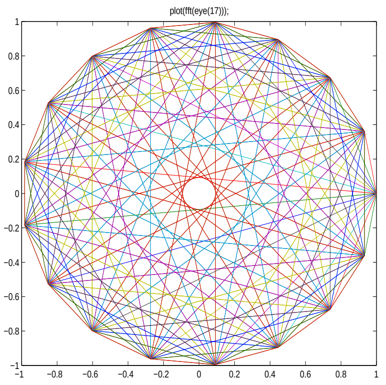
In fact, in 1D we have

```
>> compnorm(   conj(fliplr( rot(xx,-1))), invol(xx));
quotient of norms: norm(x)/norm(y) = 1
difference of normalized versions  = 0
```

5.2 SURPRISING OBSERVATIONS

```
    >> which shg
C:\Program Files\MATLAB\R2011a\toolbox\matlab\graphics\shg.m
>> help shg
    SHG.M    HGFei, 1998:   just replacement of  "figure(gcf)"
    works in the same war as the official file!  hgfei
>> type shg
%   SHG.M    HGFei, 1998:   just replacement of  "figure(gcf)"
figure(gcf);
```

TEST PLOT



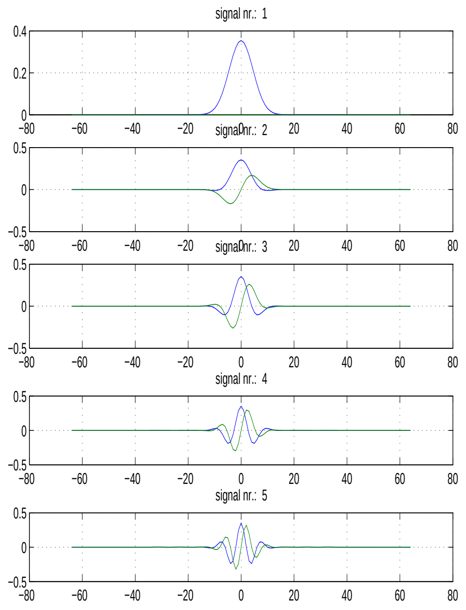
TESTCODE:

NuHAG M-files: FFTS.M
LTFAT M-files: DSFT.M

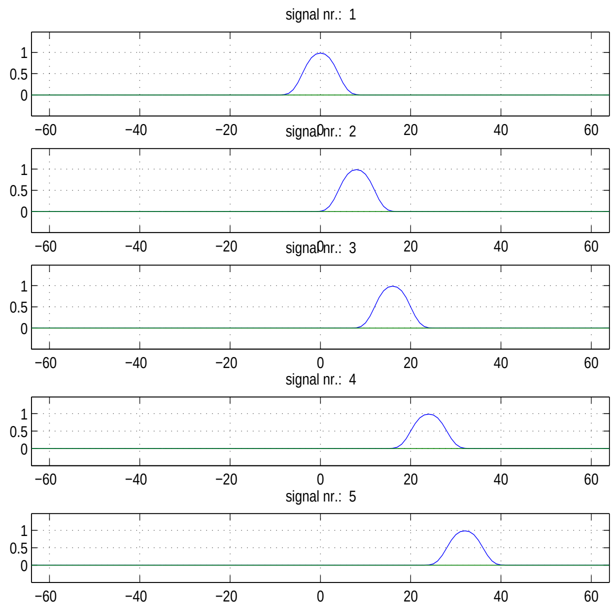
6 Tools and helper files

6.1 Rows versus columns

One of the perhaps disturbing conventions with the original NuHAG toolbox (as used by hgfei) is the fact that collections of $1D$ signals are presented as collection of row vectors. In this way the idea is, that the signals are viewed/displayed one after the other, as in the following plot:



Another choice (spline-type basis):



There are various advantages (and the main advantage, that this is in conflict with the usual interpretation of matrices as operators, which act from the left on column vectors).

In order to mitigate the problem[9] we have introduced a GLOBAL variable named ROWCOLMODE.
NuHAG M-files: RCR.M, RCC.M, RCM.M

- RCR: set ROWCOLMODE to rows
- RCC: set ROWCOLMODE to cols
- RCM: show the ROWCOLMODE

In order to make code independent from these two interpretations we are typically using code of the following form:

```
global ROWCOLMODE
if isempty(whos('global','ROWCOLMODE')) == 0;
   else
ROWCOLMODE = 'cols' ;
end;
if strcmp(ROWCOLMODE,'cols') == 1;
   trl = trl.';  % act of transposition
end;
```

Sometimes it is required to add some information to all the subplots in a given rectangular display, e.g. of format $hh \times ww$. For such situations (e.g. putting colorbars by setting $todo =' colorbar'$)

[9]But as a mental exercise we think it is a healthy training, like driving occasionally in UK or New Zealand.

```
%  DOALL.M  HGFei    Marburg, Aug. 2011
%
%  Usage:   doall('todo',hh,ww);

function  doall(todo,hh,ww);

disp(todo);
for jj=1:hh*ww;
subplot(hh,ww,jj);
eval(todo);
end;
figure(gcf);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

6.2 Rules for help parts

There are many ways to make programs useful for easy use, either by the person who has written the code, or for another person using it later. In many cases even our own code looks strange after a longer period. Hence good documentation and certain basic rules are very important for the long-term usability of code (in fact, independent of the software used).

Some of the recommendations I (hgfei) can give, are the following ones:

- provide not only a written documentation in the help-part of the M-file, but provide also (ideally close to the end of the M-file) a line of the format
  `Usage: output = actionfile(input1,input2);`
  so that, whenever standard names are used, the file can be called using copy-and-paste (instead of retyping a long list of complicated variable names);

- It is highly important to use the theoretical background, which requires a number of (strict = analytic) equalities. If different implementations of the same operator (for example) provide the same (consistent) output for random (complex) signals, then there are good chances, that the formulas are correct;

- If M-files produces a signal, it may be more instructive to *show* the output in the format of an image, e.g. by plotting the function; (cf. LOWSIGN below)

- use filenames which are allowing some systematic *reading*, so they should not be too short (there are few short names, and after a short while confusion arises), nor to long (that requires too much typing);

- provide DEMOS and TUTORIALS

6.3 File names and their actions/use

There is a rich variety of MATLAB files, see

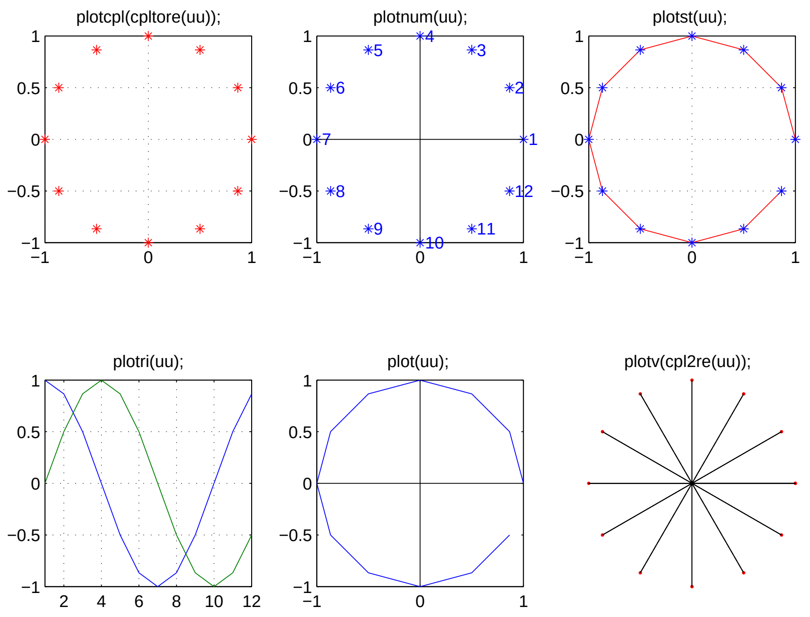
`http://www.mathworks.de/help/techdoc/ref/importdata.html` providing information as part of **R2011a Documentation** for MATLAB.

We have the following NuHAG specific tools to offer:

M-file name	Action realized	Comment and Application
PLOTC.M	plot in centered mode	functions on $\mathbb{Z}_N$
IMGC.M	absolute image, centered	function on $\mathbb{Z}_N \times \mathbb{Z}_M$
SPYC.M	centered SPY command	inspecting point sets
F2SP.M	comparing 2 signals	shows functions and spectra
COMPNORM.M	compare normed signals	format independent
SECMATC.M	show SECtions of matrix	showing RC by RC
SECSTF.M	show STF of SECtions	one by one-mode
SECSTFM.M	show STF of SECtions	!Multiple plot-windows

M-file name	Action realized	Comment and Application
PLOTC.M	plot in centered mode	functions on $\mathbb{Z}_N$
PLOTRI.M	plot real/imag. part	standard plot, index $1 \dots N$,
PLOTNUM.M	plots numbers with values	...
PLOTST.M	plots stars	
PLOTV.M	useful to show frames in $\mathbb{R}^2$	zero to point in $\mathbb{C}$

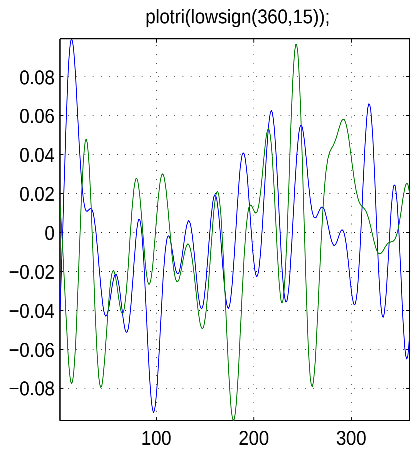
Demonstration of these different plot commands:

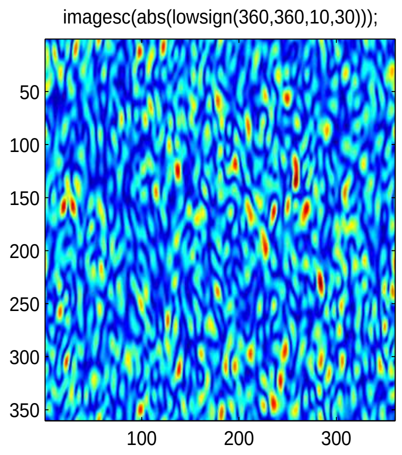


6.4 Generating Functions and Signals

M-file name	Action realized	Comment and Application
GAUSSNK	Gauss function (a la NK)	FFT-u invariant
FLTS	filter - symmetric	I: length and half-width
LOWSIGN	low-pass signal	I: length and max. frequ.
LOWSIGN	low-pass signal (2D)	smooth images (four par.)
PLATCOS	cos-slope plateau	constant one on a set
FLTRECS	filter rectang. symm.	standard rect. filter
FLTC	circular filter in $2D$	..
UNITMAT	unit matrix	standard basis for matr.
UNITVEC	unit vector	standard basis for $1D$

Example of the output with empty argument (cf. section xxx above):
`lowsign;` delivers as output an *illustration*





6.5 Generating Families of Signals

Building Riesz bases for spline-type spaces, or Gabor families is achieved using the following commands:

M-file name	Action realized	Comment and Application
TRLBAS	collection of shifted	generator for SP-T spaces
BUPUSPLINE	B-spline BUPUS	B-splines shift bases
TRL1NBAS	multiwindow spline basis	generator for MW-SP-T
GABBASP	collection of TF-shifts	product lattice
GABBAS	collection of TF-shifts	irregular case

6.6 Generating Matrices and Operators

Transforms and Operators

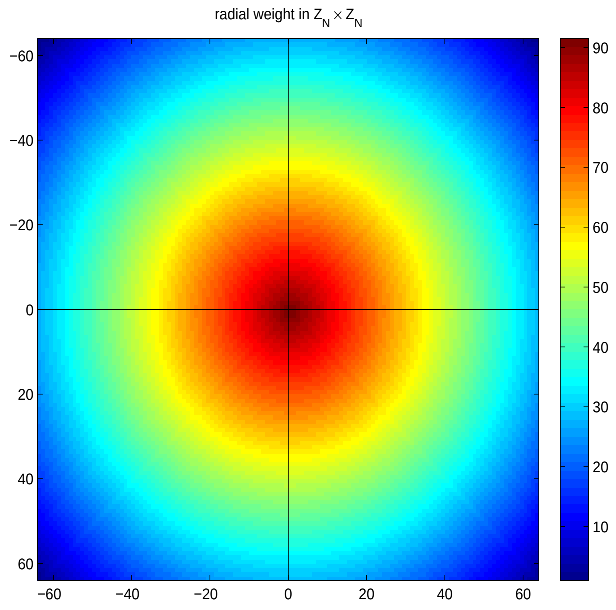
M-file name	Action realized	Comment and Application
Unitary FFT	FFTU	DFT
Symplectic FFT	FFTS	DSFT
Convolution matrix	CONVMAT	
Circulant rotation	ROT	
Modulation	MODUL	
Time-frequency-shift	ROTMOD	
TF-shift matrix	TFMATR	

6.7 Refined auxiliary tools

M-file name	Action realized	Comment and Application
MAT2TEX	MATLAB to LATEX	convenient transfer
SAVEJPG	saving to PDF-dir	convenience tool
SAVEEPSC	saving as EPSC-file	includes conversion
DEPFUNWRITE	dependent M-file listing	concrete paths
FIGPLACE	opening a window at given position	
TESTSUITExxx	various test suites	maybe not yet provided
SHOWEX	shows number of examples	good for series experiments
SHOW10	provides info regularly	during long computations
SHOW7	shows 7×7 submatrix	depends on resolution

6.8 Hermite Functions as eigenvalues of STFT-multipliers

We need the following type of *radial weight* on the (discrete) $2D$-torus:

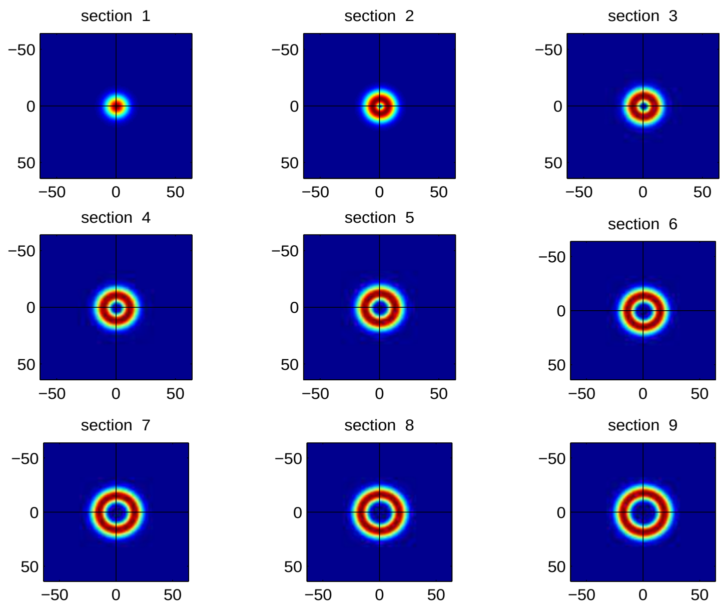


NuHAG M-files: RADWGH.M

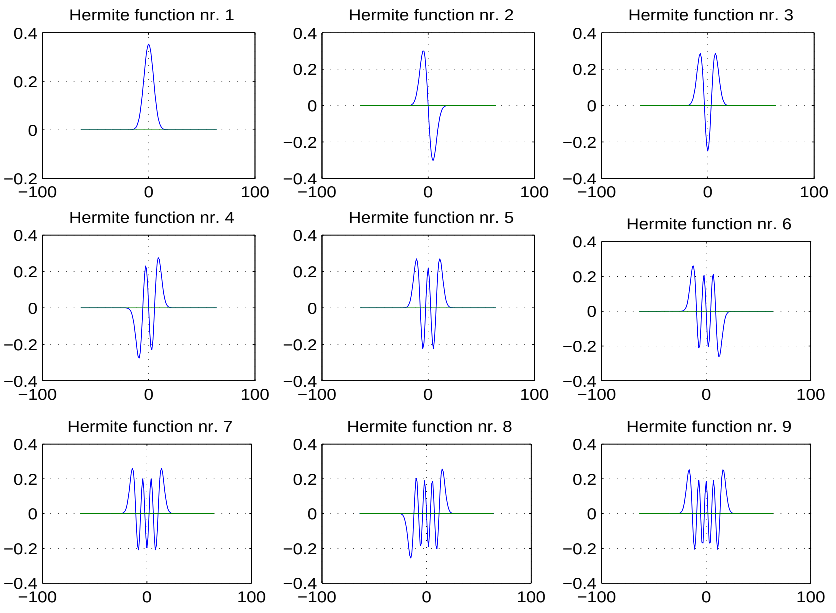
The code is in fact quite simple, once one has understood the Euclidean structure of the metric:

```
function radMat = radwgh(m,n)
if nargin < 2;  n = m; n = n; end;
dm=min(0:m-1, m:-1:1);    % calculating minimum distance along the row
dn=min(0:n-1, n:-1:1);    % calculating minimum distance across column
radMat= sqrt((ones(m,1)*dn).^2+((dm(:)*ones(1,n)).^2))+1;
```

The following image shows the TF-concentration of these Hermite functions:

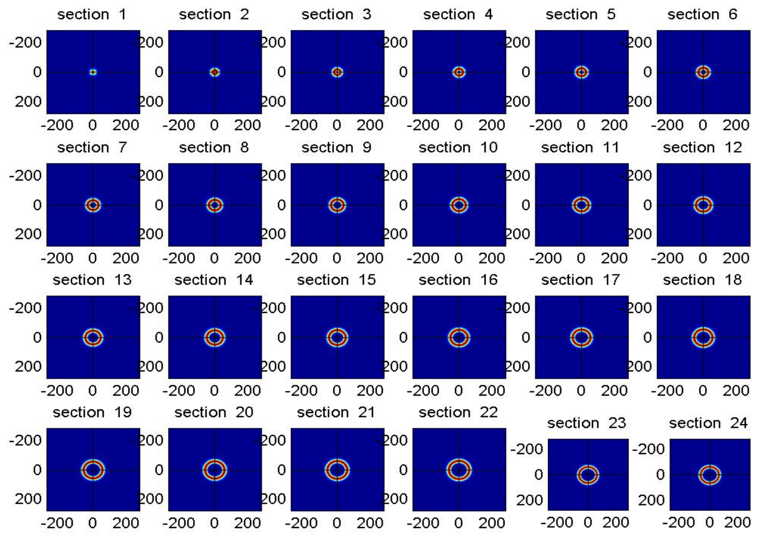


and they are real-valued functions with an increasing

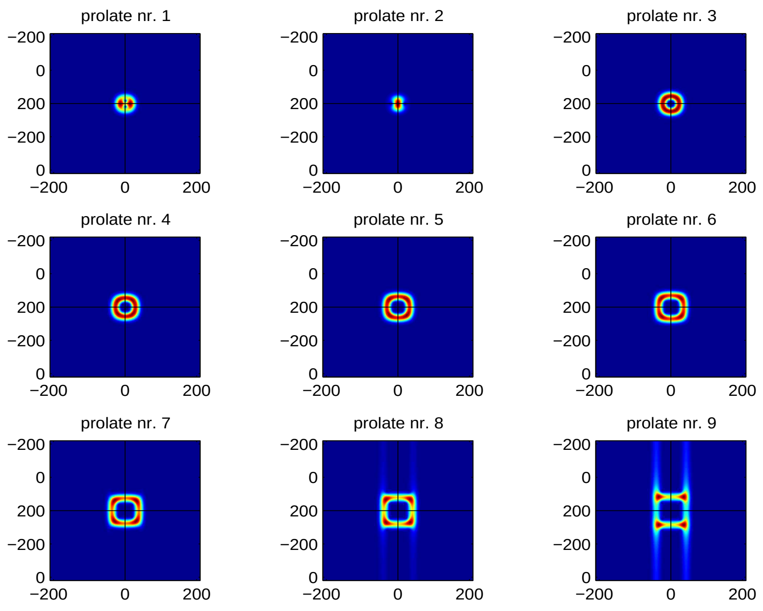


They are eigenvectors for the Fourier transform and they can used in order to define a fractional Fourier transform, which corresponds more or less to a rotation of the spectrogram with respect to a discrete Gauss-function.

Let us take a look at the the first 24 Hermite functions once more:

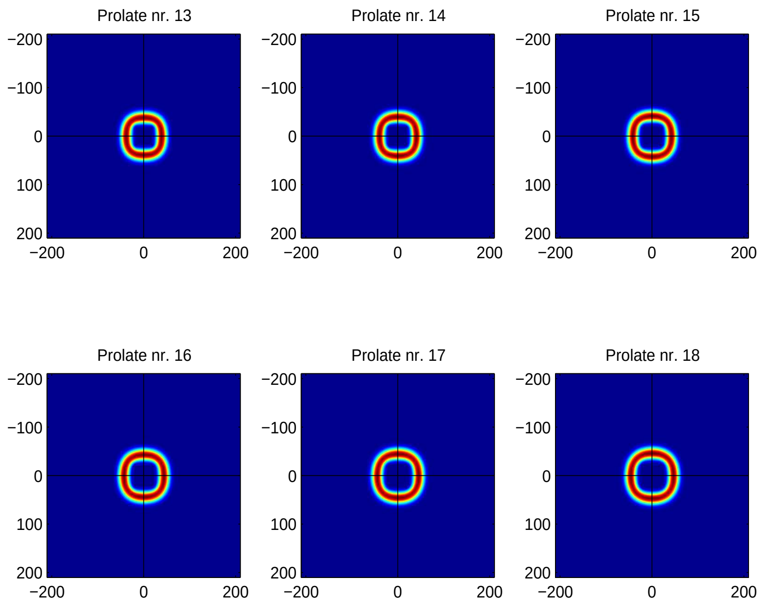


They are quite similar to the eigenvectors of certain product-convolution operators, which concentrated in a similar way, and known as **prolate spheroidal functions**: (citations!, to come). They look roughly like this:



Similar functions appear as eigenvectors of rectangular TF-filter!

```
A = gabmulhf(fltrecs(n,n,21,21),g);
```



Using the following code:

```
FILT = flts(n,21);   % a box function, with 43 = 2*21+1 ones
FU = fftu(n);  % the unitary version of the Fourier matrix
 OP =  diag(FILT) * FU * diag(FILT) * FU * diag(FILT);
```

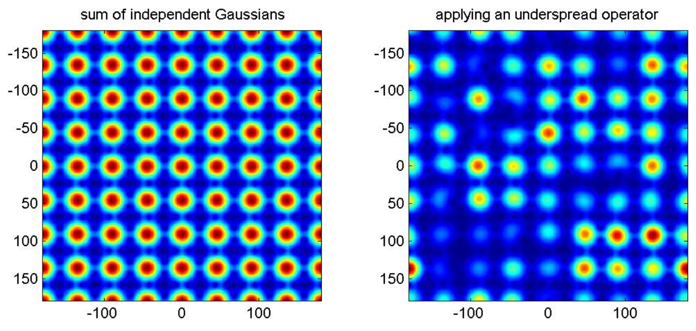
6.9 Refined Linear Algebra Tools

M-file name	Action realized	Comment and Application
EIGSORT	eigenvalues in decreasing order	second output: eigenvalues
ORTHBEST	symmetric orthonormalization	Loewdin method
GRAMSFEI	Gram-Schmidt	geometric view (hgfei)

6.10 Further comparison tools

M-file name	Action realized	Comment and Application
COMPSPYC	compare two pointsets	centered SPY-mode
COMPIMRI	compare complex images	Re/Im-part separately
SHOWTFTF	time, frequency and TF	different representations
SHOWMAT4	showing matrix: 4 FORMS	KNS, spreading, etc.
CONTC	centered contour plot	comparable with IMGC
IMAGGRAY	gray image display	standard black & white
SPIRAL3	3-D plot of complex fct.	graph in $\mathbb{C} \times \mathbb{R}$
————	————	————
ZOOMAX	zoom into maximum of signal	real/imag/abs. value

Demonstration of the action of an underspread operator on a collection of well separated Gauss functions (as they are used in mobile communication systems):



The overlap of closely related signals with random phase may cause serious cancelations in some places, as seen from the image.

6.11 Auxiliary functions

6.11.1 Finding dependent files

Finding out, what the file are which may/can be called from the file names GADPRECLFR.M we have/use a script file named DEPFUNWRITE.M which is based on the (MATLAB) DEPFUN Routine, e.g. filename = 'gadprclfr' , or 'gadprclfr.m', then calling >> depfunwrite, gives:

```
EXAMPLE:
---------- DEPFUNWRITE.TXT
    'C:\ml5\gabml\wks01\gadpreclfr.m'
    'C:\ml5\gabml\wks01\gapavg.m'
    'C:\ml5\gabml\wks01\preclf.m'
    'C:\ml5\gabml\wks01\preclfr.m'
    'C:\ml5\gabml\wks01\preclfr1.m'
    'C:\ml5\gabml\wks01\preclfr2.m'
    'C:\ml5\gabml\wks01\rotfr.m'
    'C:\ml5\gabml\wks01\trlbasfr.m'
    'C:\ml5\select\gaussnk.m'
    'C:\ml5\select\modul.m'
    'C:\ml5\select\oneover.m'
    'C:\ml5\select\rot.m'
    'C:\ml5\select\rotmod.m'
    'C:\ml5\select\rotrc.m'
```

WEXRPL.M REQUIRES

```
    ---------- DEPFUNWRITE.TXT
    'C:\ml5\gabml\wks01\wexrpl.m'
    'C:\ml5\gabml\gabtb1\blockn.m'
    'C:\ml5\gabml\gabtb1\blocknon.m'
    'C:\ml5\gabml\gabtb1\gaus.m'
    'C:\ml5\gabml\gabtb2\fullfull.m'
    'C:\ml5\gabml\gabtb2\gabddd.m'
    'C:\ml5\select\alldiv.m'
    'C:\ml5\select\block.m'
    'C:\ml5\select\pz.m'
    'C:\ml5\select\rot.m'
    'C:\ml5\select\schach.m'
    'C:\ml5\select\showmat.m'
    'C:\ml5\select\suniq.m'
```

In order to find out, how much time is used within a given routine, the MATLAB **profiler** is quite useful.

```
profile on;
```

```
ACTION ...

profile off;
profile report;
```

provides a list of files that have been called together with a statistics (CPU-times, etc.) spent within the different routines. It is a very good method to find out, which parts of the algorithm may need most of the time. Sometimes it appears, that a mathematically interesting step can be improved by some very sophisticated twist, but then it may turn out, that the effect of this improvement is minimal, because the part of the code in questions requires only a minimal part of the overall timing.

6.12 Saving pictures and plots

NuHAG M-files: SAVEJPG, SAVEEPSC

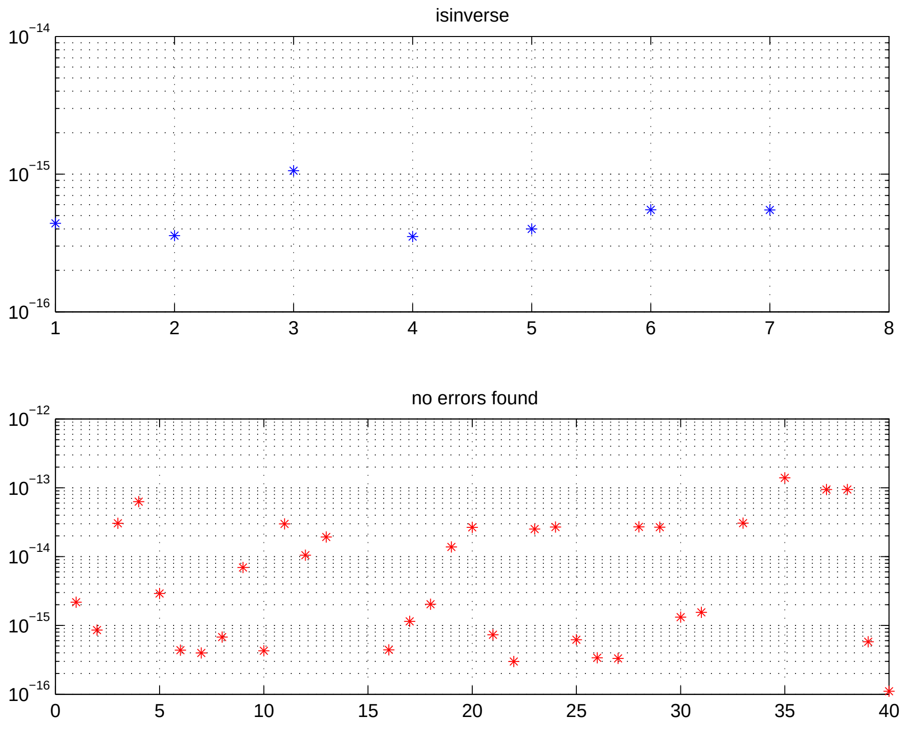
These two simple routines make sure, that one does not need to click a number of timers in order to create an image within a standard directory, but allows to do so from the command line. A typical call is simple

```
savejpg('filename2');
```

but it produces also an output-string (ready for copy and paste) to be used in the corresponding LATEX environment.

6.13 Consistency tests based on identities

OUTPUT of the first test-suite (due to Vlad and Nico, 2004):



The scaling indicates that all errors are at the level of the numerical precision of the system, so they are fine, and since ALL the tests are successful the file indicates that *everything is OK* at the end.

Otherwise it gives the tests which are a problem, and one could easily (in most cases) find the reason, i.e. the file which is causing the problem (assuming that the test is OK).

NOTE: We should agree on a standard format, so that only a sequence of individual TEST-blocks has to be created that one could use for individual tests and contributions to the overall system, and then put it together to TEST-suites, like this one.

7 LTFAT comments and useful things

Comparing the NuHAG tools with LTFAT (P.Soendergaard):

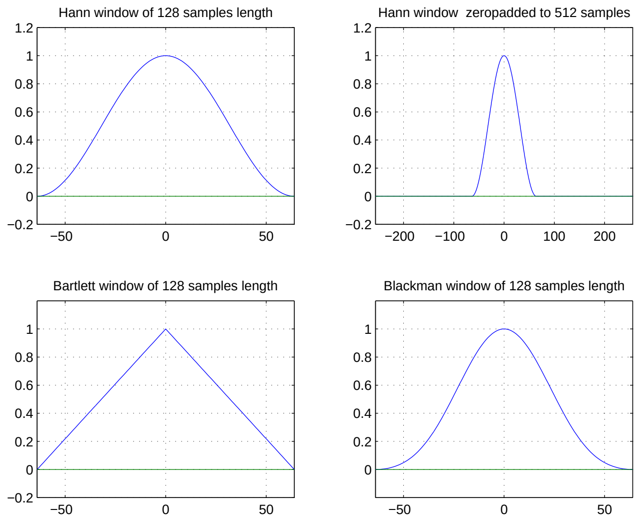
```
>> compnorm(pgauss(n), gaussnk(n));
quotient of norms: norm(x)/norm(y) = 1
difference of normalized versions  = 2.2176e-016
```

NuHAG M-files: GAUSSNK
LTFAT M-files: PGAUSS (allows additional paramters)

In the sequel you see the MATLAB command, based on LTFAT.

```
g1 = firwin('hann',128);
g2 = fir2long(g1,512);
g3 = firwin('bartlett',128);
g4 = firwin('blackman',128);
subplot(2,2,1); plotc(g1); title('Hann window of 128 samples length');
subplot(2,2,2); plotc(g2); title('Hann window zeropadded to 512 samples');
subplot(2,2,3); plotc(g3); title('Bartlett window of 128 samples length');
subplot(2,2,4); plotc(g4); title('Blackman window of 128 samples length');
axfour20;  % adjustment of plots: hgfei
```

OUTPUT produced in this way:



There have been also some name-conflicts between LTFAT and the NuHAG tools, which have mostly been eliminated. For the future we think that it is a good idea to keep names separately. So far in those few cases where the output was different we have decided to respect the LTFAT terminology and choose variations of our original names.

7.1 Name conflicts

In principle the subsequent remarks are only relevant for those who might be involved (run, read, etc.) earlier NuHAG MATLAB code and find a conflict with LTFAT.

By example: LTFAT uses the name for matrices representing a transform by a given name, as described in the help file:

```
    function F=tfmat(ttype,p2,p3,p4,p5)
%TFMAT Matrix of transform / operator
%   Usage:  F=tfmat('fourier',L);
%           F=tfmat('dcti',L);
%           F=tfmat('dgt',g,a,M);
%           F=tfmat('dwilt',g,M);
%           F=tfmat('wmdct',g,M);
%           F=tfmat('zak',L,a);
%           F=tfmat('gabmul',sym,a);
%           F=tfmat('spread',c);
```

In contrast, TFMAT in the NuHAG context was referring to the MATRIX representing a TF-shift, so to say the analog of the abstract symbol $\pi(\lambda) = M_\omega T_t$, for $\lambda = (t, \omega) \in \mathbb{R}^d \times \widehat{\mathbb{R}}^d$ in the abstract descriptions. Consequently the name (in NHGTB) has been changed to TFMATR.M.
NuHAG M-files: TFMATR.M
LTFAT M-files: TFMAT.M

7.2 Timing of algorithms

Timing is always a critical and sometimes important business.

7.3 APPENDICES: Concrete CODE for various M-files

7.4 APPENDIX: GABINIT.M Initialization routine

NOTE: This is a NuHAG file which works in ROW-mode, i.e. collections of signals are collections of *row-vectors* [this has various advantages, but also the disadvantage of incompatibility with the Standard MATLAB convention (Matrices are collections of *column vectors*, and plotting etc. is done from this point of view, also eigenvectors are delivered as orthonormal collections of column vectors, etc.).

GABINIT

```
    % GABINIT.M  Gabor Initialization    hgfei, last modified 22.7.2010 NZ
%----------------------------------------------------------------
% Setting the stage for quick Gabor demonstrations.
% Typically  n  is given, and the rest is computed, like a natural
% choice of (a,b), with reasonable redundancy (but typically a different
% from b, to show effects from this), dual and tight Gabor atom, frame
% operator S,  Gabor family G etc.
%----------------------------------------------------------------
```

```
%
% AUTHOR(s) : Hans G. Feichtinger Nov. 1st, 2007 (upload)
%
% COPYRIGHT : (c) NUHAG, Dept.Math., University of Vienna, AUSTRIA
%             http://nuhag.eu/
%             Permission is granted to modify and re-distribute this
%             code in any manner as long as this notice is preserved.
%             All standard disclaimers apply.
% EXTERNALS :
% SEE ALSO  :

% GABINIT.M        - Gabor initialization file, assumes only  n = length to be given
%
% Input:            Man kann vorher n,a,b festlegen - Programm nimmt Bezug darauf
%                   Vorsicht vor unbeabsichtigten definierten Variablen "a" oder "b"
%
% Output:           G     GD    S     alph  b     f     fs    gd    mx1   red   xpa
%                   GA    GT    a     ans   divs  ffs   g     gt    n     s     xpo
%             new   ldiv  ndrich
%
% Usage:            n vorher festlegen; gabinit2?
% Man bentigt die Packete: gabmin, gabtb1, gabtb2, gabtgt

if exist('n') == 0;  n = 480;  else disp(['n = ', num2str(n)]);  end;
for jj=1:3*n; ldiv(jj) = length(alldiv(jj)); end;
ndrich = find(ldiv > 16);
alph = alldiv(n); alph = alph(:)';
mx1 = max(find(alph <= sqrt(n)));
divs = alph(1:mx1),
  if exist('a') ==0; a = alph(mx1), else
     disp(['a = ', num2str(a) ]); end;
  if exist('b') ==0; b = alph(mx1-1),   else
     disp(['b = ', num2str(b) ]);  end;
red = n/(a*b)
xpo = lattp(n,a,b);
xpa = lattp(n,n/b,n/a);
if abs(length('g') - n) > 0; g = gaussnk(n);  end;
gd = ppdw(g,a,b);
disp('calculating dual atom and tight atom');
gt = gabtgtmh(g,a,b);
disp('using Mario Hampejs routine for tight Gabor atom');
disp('alternative method would be  gabtgtgj, G.Janssen-method');
disp('');
who ,
figure('position', [400,80,900,550]);
f3sp(g,gt,gd);
```

```
disp('now establishing G,GD,GT and other matrices'); %  pause(1);
if red*n < 1111;   G = gabbasp(g,a,b);
GD = gabbasp(gd,a,b);  GT = gabbasp(gt,a,b); S = G'*G;
GA =  gabbasp(g,n/b,n/a);
else     disp('..');
disp('.................!!  red*n is too large to automatically establish matrices');
   disp('..');
end;
```

8 Guide to the literature

Of course the book by Charly Gröchenig is the fundamental reference for those who are interested in the full theory, the analytic background, etc. ([5]).

Those who want to start from the linear algebra and come to Gabor analysis from this point of view are well served by browsing through [4]. It is downloadable from
`http://univie.ac.at/nuhag-php/bibtex/open_files/feluwe05_FGabsingFinal.pdf`

The necessary background to return from the linear algebra case to the case of continuous variables is provided by the method of Banach Gelfand Triples (which are already a good tool to describe the *Fourier transform* in a simple, distributional setting, but also the (unitary) mapping between the kernel of an operator (our matrix), endowed with the standard Euclidean structure of $\mathbb{C}^{N^2}$. A summary on this is provided by [**?**]. It is downloadable from
`http://univie.ac.at/nuhag-php/bibtex/open_files/felu05_0503364.pdf`

Those who are more interested in the *algebraic* aspects of Gabor analysis we can recommend the paper: *Gabor Analysis over finite Abelian groups* ([2]). It s downloadable from `http://univie.ac.at/nuhag-php/bibtex/open_files/133_fekolu08a.pdf` .

The role of the underlying function spaces for the validity of integral identities (such as Wexler-Raz, etc.) is described in detail in the paper: *Wiener amalgam spaces for the Fundamental Identity of Gabor Analysis* ([3]). It is downloadable from
`http: http://univie.ac.at/nuhag-php/bibtex/open_files/felu05_0503364.pdf`

There are also books concerning *Fourier Analysis on Finite Abelian Groups*, such as the one by Bao Luong ([6]), but his approach (also making use of basic principles of linear algebra and group theory) is still rather abstract and might be well complemented by some of the ideas (or MATLAB experiments) as presented in this course.

9 Excercises and Small Projects

For now the numbers are not yet fixed!!

1. In order show how natural the FFT2 is, show that the set of eigenvectors for (suitable) 2D-shifts on $\mathbb{Z}_8 \times \mathbb{Z}_9$ are just the plane waves (shift operators are realized using `ROTRC.M`). Questions: which shift parameters can be used (and why exactly those), and what is specific about the numbers 8 and 9 (aside from the fact that both are not too big). LIT: [1];

2. Show/verify experimentally, that the adjoint group Λ° describes just the set of all points in phase space $\mathbb{Z}_N \times \widehat{\mathbb{Z}_N}$ which commute with all the TF-shifts from the original lattice Λ (or equivalently, a set of generators from Λ. The routine `adjlat.m` calculates the adjoint lattice using the symplectic Fourier transform;

   NuHAG M-files: ADJLAT.M

3. y

10 Algebraic Afterthoughts

In this section we plan to mostly present material related to ideas from algebra. Examples:

The pairing between a sub-group H in G, and the corresponding *orthogonal* group $H^{\perp}$ in $\hat{G}$, or the (symmetric) relationship between subgroups H of $G \times \widehat{G}$ and their adjoint subgroups H° in $G \times \widehat{G}$. SSSS

It is so to say "trivial" that isomorphic groups have isomorphic dual groups and hence being able to carry out TF-analysis on one realization allows (at least in principle) to do it in the other context (the note [1] is an early example of this principle).

11 APPENDIX: The Code of TESTGABFORMULAF.M

```
 % TESTGABFORMULAF.M
%  TESTGABFORLULAC.M - tests the NuHAG Gabor formulas...
%
% ----------------------------------------------------------------
%
% COPYRIGHT : (c) NUHAG, Dept.Math., University of Vienna, AUSTRIA
%             http://nuhag.eu/
%             Permission is granted to modify and re-distribute this
%             code in any manner as long as this notice is preserved.
%             All standard disclaimers apply.
%
% TESTGABFORLULAC.M - tests the NuHAG Gabor formulas...

% Nicolae Tecu \& Vlad Vicol
% Created: July 2004

% please reffer to the PDF file for formulas...
i = sqrt(-1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% check ->contains all the results of the tests.
% wrong -> contains all the elements of check that are bigger than tresh.
% isinverse -> checks whether the routines ar consistent with their inverses

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% INITIAL DATA

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
check = zeros(1,39);
wrong = zeros(1,7);
tresh = 1000 * eps;
isinverse = zeros(1,10);
n = 144;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% THE ISINVERSE TESTS

X = randcn(n);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% WHICH FUNCTIONS ARE WE USING ?

which ffts

which stft
which istft

which mat2spr
which spr2mat

which mat2kns
which kns2mat

which modrot
which rotmod

which tfconj
which tfconjdir
which tfmat
which rotrc

which inp
which unitmat
which purfr
which purfr2

which isunitm
which ispurfr

which twistcon
which lattp
which testadjlat
which gabmulmh
which gabmul


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
isinverse(1) = compnorm(X, ffts(ffts(X)));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
f = randcn(1,n);
g = rand(1,n); % it does not work for complex window yet

isinverse(2) = compnorm(f, istft(stft(f,g),g));


isinverse(3) = compnorm(f, istft(stft(f,g,3,4),g));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
isinverse(4) = compnorm(X, mat2spr(spr2mat(X)));
isinverse(5) = compnorm(X, spr2mat(mat2spr(X)));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
isinverse(6) = compnorm(X, mat2kns(kns2mat(X)));
isinverse(7) = compnorm(X, kns2mat(mat2kns(X)));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% SECTION 1
% TESTS REGARDING THE TF-OPERATORS, TF-CONJUGATION,...

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

freq = floor(rand(1) *(n-1) ) ;

shift = floor(rand(1) * (n-1) );

x = randcn(1,n);

disp('testing formula (1)');

check(1) = compnorm( exp(-2*pi*i*freq*shift/n) * rotmod(x,shift,freq) , modrot(x,freq,shift) );

%%%%%%%%%%%%%%

M = randcn(n);
N = tfmat(n,shift,freq);
P = rotmod(M,shift,freq);

disp('testing formula (2)');

check(2) = compnorm(M*N, P);

%%%%%%%%%%%%%

freq2 = floor(rand(1) * n ) - 1;
shift2 = floor(rand(1) * n ) - 1 ;

mat1 = rotmod(rotmod(M,shift,freq), shift2,freq2);
mat2 = exp( - 2 * pi * i * shift2 * freq /n) * rotmod(M, shift+shift2, freq+freq2) ;

disp('testing formula (3)');

check(3) = compnorm(mat1, mat2);

%%%%%%%%%%%%
mat1 = rotmod(rotmod(M,shift,freq), shift2,freq2);
mat2 = rotmod(rotmod(M,shift2,freq2), shift,freq);

mat2 = mat2 * exp( 2 * pi * i * shift * freq2 /n) * exp( - 2 * pi * i * shift2 * freq /n) ;

disp('testing formula (4)');

check(4) = compnorm(mat1,mat2);
%%%%%%%%%%%%%

mat1 = tfconj(tfconj(M,shift,freq),shift2,freq2);
mat2 = tfconj(M,shift+shift2, freq + freq2);

disp('testing formula (5)');
```

```
check(5) = compnorm(mat1, mat2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% SECTION 2
% TESTS REGARDING THE SYMPLECTIC FOURIER TRANSFORM (FFTS)...

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

A = randcn(n);
disp('testing formula (6)');
check(6) = compnorm(A ,ffts(ffts(A)));

%%%%%%%%%%%%%%
f = randcn(1,n);
g = randcn(1,n);

gh = fft(g.');
A = gh*f;
B = ffts(A);

disp('testing formula (7)');

check(7) = compnorm(B , fft(f.')*g);
%%%%%%%%%%%%%

a = randcn(n);
b = randcn(n);

disp('testing formula (8)');

check(8) = compnorm(ffts(ifft2(fft2(a).*fft2(b))),ffts(a).*ffts(b));
%%%%%%%%%%%%%

a = randcn(n);
b = randcn(n);

fa = ffts(a);
fb = ffts(b);

disp('testing formula (9)');

check(9) = compnorm( inp(a,b), inp(fa,fb)) ;% where inp is the L2 inner product for 2-dim

%%%%%%%%%%%%%%

f = randcn(n);

Fsf = ffts(f);
F2f = fft2(f);
F2fnew = zeros(n);

for j = 1:n;
    F2fnew(:,j) = F2f(mod(-(j-1),n)+1,:).';
end;

disp('testing formula (10)');
check(10) = compnorm(Fsf, F2fnew);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% SECTION 3
% TESTS REGARDING THE SPREADING FUNCTION, KOHN-NIERENBERG SYMBOL, SYMPLECTIC FOURIER TRANSFORM (FFTS)...

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

disp('testing formula (11)');
check(11) = compnorm(inp(a,b),inp(mat2kns(a),mat2kns(b))) ;

disp('testing formula (12)');
check(12) = compnorm(inp(a,b),inp(mat2spr(a),mat2spr(b))) ;% check normalization

disp('testing formula (13)');
check(13) = compnorm(inp(mat2kns(a),mat2kns(b)),inp(mat2spr(a),mat2spr(b))) ;% check normalization of mat2spr it needs and 1/sqrt(n) factor

disp('testing formula (14)');
check(14) = compnorm(mat2kns(eye(n)) , ones(n)) ;%check normalization

disp('testing formula (15)');
check(15) = compnorm(mat2spr(eye(n)) , unitmat(n,1,1)); % check normalization

%%%%%%%%%%%%%%

disp('testing formula (16)');

a1 = spr2mat(randcn(n));

check(16) = compnorm( ffts(mat2kns(a1)), mat2spr(a1) );
%%%%%%%%%%%%%%

disp('testing formula (17)');
check(17) = compnorm(mat2spr(tfmat(n,shift, freq)), unitmat(n,freq+1,shift+1));
%%%%%%%%%%%%%%%%
```

```
disp('testing formula (18)');
check(18) = compnorm(mat2kns(tfconj(M,shift, freq)), rotrc(mat2kns(M),freq,shift));
%%%%%%%%%%%%%%%%%%

disp('testing formula (19)');
l1 = floor( rand(1) * (n-1) ) + 1;
l2 = floor( rand(1) * (n-1) ) + 1;

H = randcn(n);
HA = mat2spr(tfconj(H,l1,l2));
HC = mat2spr(H);
%HB = zeros(n);
%HD = zeros(n);

% %%%%%%%%%%%%%%%%%%%% pairwise %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% % also work but is slower
%
% for j=1:n
%     for k=1:n
%         HB(j,k) = exp( 2* pi * i * ((l2)*(k-1) - (l1)* (j-1)) / n)  * HC(j,k);
%         %% why does the role of k and j exchange here ?
%         HD(j,k) = exp( 2* pi * i * ((l2)*(k-1) - (l1)* (j-1)) / n);
%     end;
% end;
%
% check19b = compnorm(HA,HB)
% %%%%%%%%%%%%%%%%%%%% using purfr2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% [ok,row,col,quot] = ispurfr(D);
%
% check19b = 1-ok

HE = purfr2(n,n,n-l1,l2);
%% why n-l1 and not n-l1+2 (which should correcpond to  -l1 in theory)

check(19) = compnorm(HA,HE.*HC);
%%%%%%%%%%%%%%%

f = randcn(1,n);
g = randcn(1,n);

R = zeros(n);
gf = fft(g);

for j =1:n;
    for k = 1:n;
        R(k,j) = f(j)*conj(gf(k)*exp(2*pi*i*(k-1)*(j-1)/n));
    end;
end;

disp('testing formula (20)');
check(20) = compnorm( R , mat2kns(g' * f) );
%%%%%%%%%%%%%%%%%%%%%%%%

G = randcn(n);
H = randcn(n);

disp('testing formula (21)');
check(21) = compnorm( mat2spr(G*H), twistcon(mat2spr(G),mat2spr(H)));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% SECTION 4
% TESTS REGARDING THE SHORT-TIME FOURIER TRANSFORM & CONNECTIONS TO THE
% ABOVE....

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

g = randcn(1,n);
f = randcn(1,n);

disp('testing formula (22)');
check(22) = compnorm(mat2spr(g'*f), stft(f,g))  ;

%%%%%%%%%%%%%%%
f = randcn(1,n);
g = randcn(1,n);

R = zeros(n);
gf = fft(g.').';

for j =1:n;
    for k = 1:n;
        R(k,j) = f(j)*conj(gf(k)*exp(2*pi*i*(k-1)*(j-1)/n));
    end;
end;

disp('testing formula (23)');
check(23) = compnorm(R, ffts(stft(f,g))) ;

% please note that this formula works, but in the
% definition of R above, the k,j are reversed as compared with the formula
% given in the second column on the second page, in the Gabor Formulae. If
% we put it like there it doesn't work anymore. Maybe it is because in
% Matlab row actually means frequency, and column means time....
%%%%%%%%%%

% testing further formulae for STFT

disp('testing formula (24)');
f = randcn(1,n);
g = randcn(1,n);

STF1 = stft(conj(f),g);
STF2 = stft(g,conj(f));
R = zeros(n);

for k = 1:n;
    for j = 1:n;
        R(k,j) = conj(exp(2*pi*i*(k-1)*(j-1)/n) * STF2(mod(-(k-1),n)+1,mod(-(j-1),n)+1));
    end;
end;

check(24) = compnorm(STF1, R);
%%%%%%%%%%%%%%%%%%%%%%%%

a = randcn(1,n);
b = randcn(1,n);
c = randcn(1,n);
d = randcn(1,n);

f1 = stft(a,b);
f2 = stft(c,d);

z1 = f1(:)' * f2(:);
z2 = conj(a*c');
z3 = b*d';

disp('testing formula (25)');
check(25) = compnorm(z1/n , z2 * z3);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

f = randcn(1,n);
g = randcn(1,n);

ftilde = f(mod(-([1:n]-1),n)+1);
gtilde = g(mod(-([1:n]-1),n)+1);


s1= stft(ftilde,g);
s21= stft(f,gtilde);
s2 = s21;

for k = 1:n;
    for j = 1:n;
        s2(k,j) = s21(mod(-(k-1),n)+1,mod(-(j-1),n)+1);
    end;
end;

disp('testing formula (26)');
check(26) = compnorm(s1, s2);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

s1 = stft(conj(f),g);
s21 = stft(f,conj(g));
s2 = zeros(n);

for j = 1:n;
    s2(j,:) = s21(mod(-(j-1),n)+1,:);
end;

disp('testing formula (27)');
check(27) = compnorm(s1 , conj(s2));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% testing fundamental time-frequency identity%%%
f = randcn(1,n);
g = randcn(1,n);

STF1 = stft(f,g);
STF2 = stft(fft(f),fft(g))/n;
STF2new = zeros(n);
FACTSTF =  zeros(n);

disp('testing formula (28)');
for j = 1:n;
          STF2new(:,j) = STF2(mod(-(j-1),n)+1,:).' .* (exp(-2*pi*i*(j-1)*[0:n-1]/n)).';
          FACTSTF(:,j) =  (exp(-2*pi*i*(j-1)*[0:n-1]/n)).';
end;

check(28) = compnorm(STF1,STF2new);
check(29) = compnorm(fft(eye(n)),FACTSTF);

%%%%%%%%%%%%%%%%%%%%

a = randcn(1,n);
b = randcn(1,n);
```

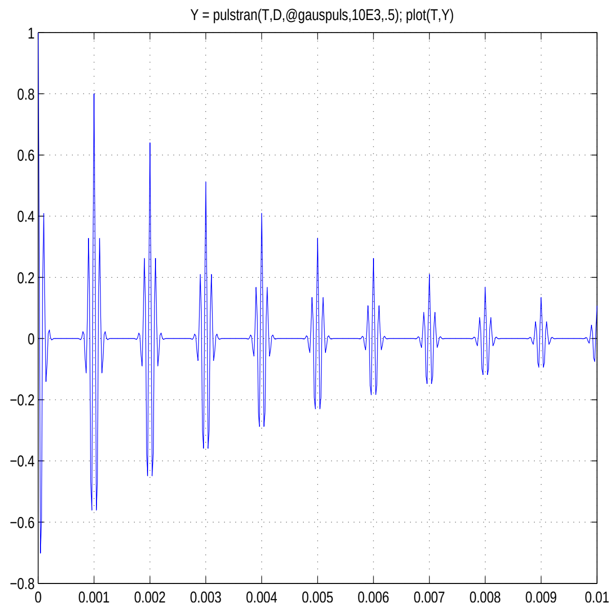
```
c = randcn(1,n);
d = randcn(1,n);

disp('testing formula (29)');
check(30) = compnorm(twistcon(stft(a,b),stft(c,d)),a*d'*stft(c,b));

disp('testing formula (30)');
check(31) = compnorm(twistcon(stft(a,b),stft(c,d)),mat2spr((b'*a) * (d'*c)));


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SECTION 5
% ADDITIONAL TESTS -> MAT2SPR, ADJLAT, GABMULMH

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


A = randcn(n);

a = floor(rand(1) * (n-1));
b = floor(rand(1) * (n-1));

x = mat2spr(tfmat(n,a,b));
y = unitmat(n,b+1,a+1);

check(32) = 1- isunitm(x);
check(33) = compnorm(x,y);

p = mat2spr(A);
r = mat2spr(tfconj(A,a,b));

q = p./r;

[ch,rows, cols,qu] = ispurfr(q);

check(34) = 1 - ch;

s = purfr2(n,n,rows,cols);

check(35) = compnorm(q,s);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

xpo = lattp(144,12,16);
xp1 = adjlat3(xpo);

check(36) = compnorm(xp1,lattp(144,144/16,144/12));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% same test as above, but more general lattice, i.e. xpo =
% sidedigm(lattp(n,a,b)) for two functions...

check(37) = norm(testadjlat(24,6,8,'adjlat'));

check(38) = norm(testadjlat(24,6,8,'adjlat3'));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

g = randcn(1,n);
a = 1; b = 1;

W = randcn(n,n) .* randcn(n,n);
A = W(1:b:n,1:a:n);
disp(' comparison between GABMULMH, GABMUL ');
% GM=gabmulmh(W,g,a,b);  % old version
% new version:
GM = gabmulmh(g,W(1:b:n,1:a:n));
% size(W),
% GM2 = gabmul(W(1:b:n,1:a:n),g,g,a,b);   %HGFei: does not work!!!
% check(41) = compnorm(GM,GM2);
%disp('check 41 has been removed!!!!');
%check(41) = 0;
% pause;

sprGM = mat2spr(GM);
fW = ffts(W);
fW(1:b:n,1:a:n);
check(39) = compnorm(sprGM , fW(1:b:n,1:a:n).*stft(g,g,a,b));
% this works, but _only_ for a = b = 1; I think it should work
% for general a = b, even for general a, b.

M = randcn(12);
disp('comparing  TFCONJ  and TFCONJDIR ');

check(40) = compnorm( tfconjdir(M,3,5),  tfconj(M,3,5))

disp('check41 has been already done: GABMUL versus GABMULMH ');

n =    480; a = 6; b =8;
W = randcn(n);   g1 = randcn(1,n);
tic;GM1 = gabmul(W(1:b:n,1:a:n),g1,g1,a,b);toc
% tic;GM2 = gabmulmh(W,g1,a,b); toc
% tic; GM2 = gabmulmh(g,W(1:b:n,1:a:n)); toc;
```

```
%% check(42) = compnorm(GM1,GM2);
% removed HGFei

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
wrong = find( check > tresh & check-1 >0 );
% finds all the checks that are bigger than tresh and that are not 1 (i.e.
% ispurfr(x) )
% figure;
subplot(211); semilogy(isinverse,'b*');
%stem(check);
title('isinverse');  grid;
subplot(212);
semilogy(check,'r*'); grid;
wrong
if length(wrong) > 0
title(['Errors occur at coordinatds:  ' vec2str(wrong)]);
else
title(' no errors found' );
end;

figure(gcf);

FU = fftu(n); F = fft(eye(n)); IF = F'/n; XX = randcn(n);

compnorm(ffts(XX),  F*(IF*XX).');
% quotient of norms: norm(x)/norm(y) = 1
% difference of normalized versions  = 8.2583e-016
%
compnorm(ffts(XX),  FU*(FU'*XX).');
% quotient of norms: norm(x)/norm(y) = 1
% difference of normalized versions  = 8.2976e-016
%
compnorm(ffts(XX).',  FU'*XX*FU.');
% quotient of norms: norm(x)/norm(y) = 1
% difference of normalized versions  = 8.3003e-016
%
compnorm(ffts(XX).',  IF*XX*F.');
% quotient of norms: norm(x)/norm(y) = 1
% difference of normalized versions  = 8.3473e-016
```

12 Left over material

```
T = 0 : 1/50E3 : 10E-3;
D = [0 : 1/1E3 : 10E-3 ; 0.8.^(0:10)]';
Y = pulstran(T,D,@gauspuls,10E3,.5); plot(T,Y)
```

provides the following plot:



Using `saveepsc.m` it took about 3 minutes to produce this page (hgfei, Aug. 19th, 2011).

DIGITAL METHODS in our daily life!
TO BE REPLACED by low resolution image!

Literatur

feka97 [1] H. G. Feichtinger and N. Kaiblinger. 2D-Gabor analysis based on 1D algorithms. In *Proc. OEAGM-97 (Hallstatt, Austria)*, 1997.

fekolu09 [2] H. G. Feichtinger, W. Kozek, and F. Luef. Gabor Analysis over finite Abelian groups. *Appl. Comput. Harmon. Anal.*, 26(2):230–248, 2009.

felu06 [3] H. G. Feichtinger and F. Luef. Wiener amalgam spaces for the Fundamental Identity of Gabor Analysis. *Collect. Math.*, 57(Extra Volume (2006)):233–253, 2006.

feluwe07 [4] H. G. Feichtinger, F. Luef, and T. Werther. A guided tour from linear algebra to the foundations of Gabor analysis. In *Gabor and Wavelet Frames*, volume 10 of *Lect. Notes Ser. Inst. Math. Sci. Natl. Univ. Singap.*, pages 1–49. World Sci. Publ., Hackensack, 2007.

gr01 [5] K. Gröchenig. *Foundations of Time-Frequency Analysis.* Appl. Numer. Harmon. Anal. Birkhäuser Boston, Boston, MA, 2001.

lu00-1 [6] S. Luo. Deforming Gabor frames by quadratic Hamiltonians. *Integral Transforms Spec. Funct.*, 9(1):69–74, 2000.

pa07 [7] S. Paukner. Foundations of Gabor Analysis for Image Processing. Master's thesis, 2007.

rest00 [8] H. Reiter and J. D. Stegeman. *Classical Harmonic Analysis and Locally Compact Groups. 2nd ed.* Clarendon Press, Oxford, 2000.

sc11 [9] O. Scherzer. *Inverse Problems - Methods.* Springer Berlin, 2011.