NuHAG Numerical Harmonic Analysis Group

A Fresh Look at Fourier Analysis Between Linear Algebra and Distribution Theory

Hans G. Feichtinger
hans.feichtinger@univie.ac.at
.
WEBPAGE: www.nuhag.eu

Aveiro, April 4th to 7th, 2016
Workshop HARMONIC ANALYSIS and INVERSE PROBLEMS

Goals of my presentation

- Offer a completly new approach to Fourier Analysis
- establish stronger ties between LA, FA and HA
- provide a new approach to distribution theory
- promote the idea of conceptual harmonic analysis
- make use of MATLAB in order to support claims
- provide an easy explanation to standard facts (e.g. convolution theorem, Shannon sampling etc.);
- have a (versatile, technical) basis for abstract HA;
- switch properly between finite and continous setup;

Possible Topics to be covered

- Banach modules (Factorization theorem)
- homogeneous Banach spaces, solid BF-spaces
- Wiener amalgam spaces (and function spaces)
- **Banach Gelfand Triples !!**
- Coorbit Theory (including metric approximation prop)
- the role of BUPUs and BAPUS
- decomposition spaces, clusters, coverings, etc.
- irregular sampling and reconstruction
- spline-type spaces (and sampling)
- Gabor analysis (foundations and applications)
- Gabor multipliers (theory and applications)
- from linear algebra to distribution theory
- generalized stochastic process from a FA viewpoint

The role of MATLAB/OCTAVE (math.softw.)

It is one of my side-goals to emphasize the possibility of *teaching Fourier Analysis* starting from Linear Algebra (I plan to give a short presentation at the MATLAB EXPO in Muenich on this topic, coming up: May 10th 2016).
MATLAB (by def.: A MATrix LABoratory) FOR ME: **linear algebra in a box** (a software covering more or less all the aspects of linear algebra, and certainly more than most people can teach in a two semester course). Using MATLAB allows to make things more concrete and even clarify concepts (like dual spaces) in a very practical sense (can be discussed separately with those interested in this subject).
At NuHAG we have plenty of software (and a PhD thesis) on this subject, simulations, exercises, demostrations, tutorials.

The role of MATLAB/OCTAVE II

The different roles of MATLAB in my work:

1. Experimental Mathematics, checking for numbers
2. Building finite models which are correctly implementing Fourier and TF-analysis over finitegroups;
3. Visualize facts
4. Verify/estimate condition numbers, eigenvalues
5. more to come...

INTERMISSION !

THIS IS MATERIAL FOR THE PRESENTATION AT THE MATLAB EXPO, MAY 10TH

Overview over my presentation

- Persönlicher Hintergrund
- Kurzer historischer Abriss
- MATLAB und Lineare Algebra
- Polynome und die DFT/FFT
- Anwendungen der FFT, TILS
- Unkonventionelle Anwendungen
- Zusammenfassung

Kurzer Lebenslauf von Hans G. Feichtinger

- Lehramt Mathematik und Physik, Univ. Wien;
- Habilitation im Fach Mathematik 1979 in Wien;
- Bis zur Pensionierung (Dez. 2015) an der Univ. Wien;
- diverse Gastprofessuren weltweit;
- Seit 1992 Aufbau der Numerischen Harmonischen Analyse Gruppe NuHAG (www.nuhag.eu) in Wien;
- Sehe mich selbst als Forscher und akademischer Lehrer; (27 PhD students laut Mathematical Genealogy).
- ab 2000 Chief Editor: J. Fourier Analysis & Applications;
- zahlreiche theoretische und angewandte Projekte.

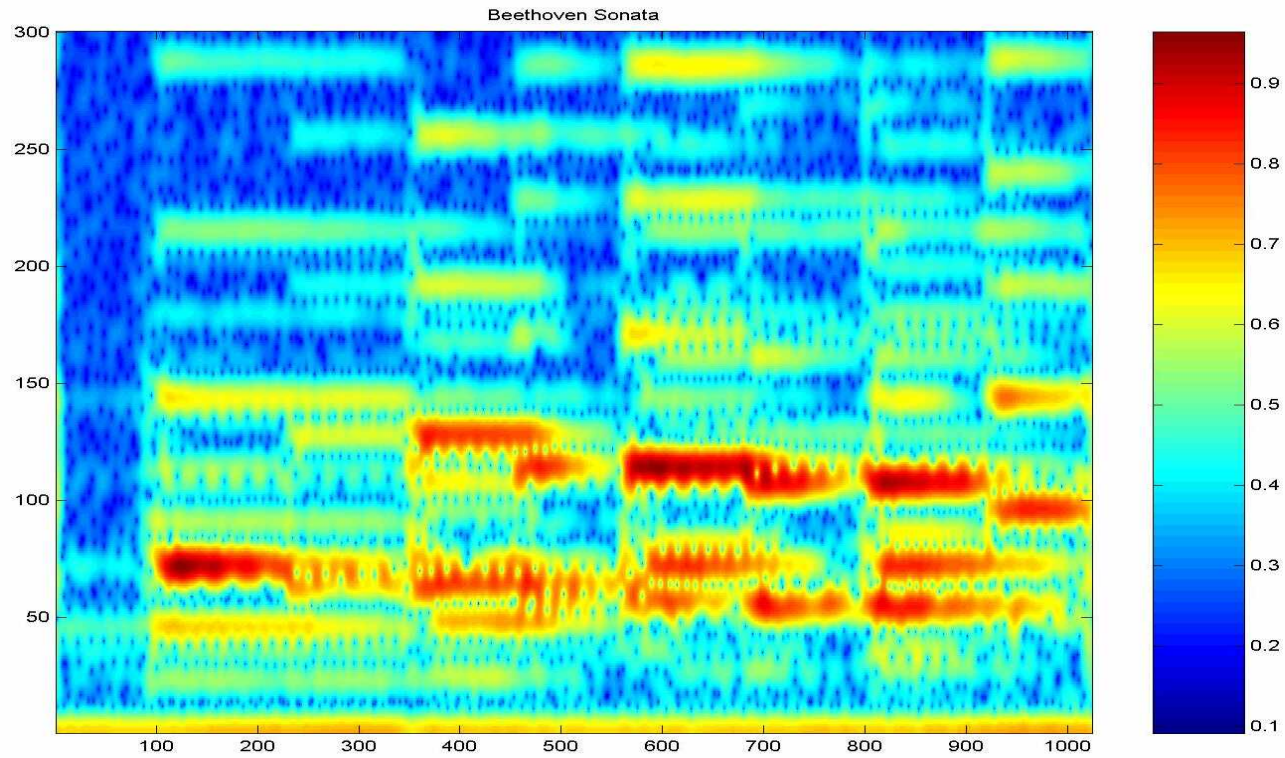
HGFei: Forschungsprofil

Ich bezeichne mich als **anwendungsorientierten Mathematiker**, mit starkem Interesse für die mathematischen Grundlagen der Signal- und Bildverarbeitung (die als WAV-files, JPEG-Bildern, MP3-Dateien zum täglichen Lebens gehören).

Genauer, sehe ich mich als **anwendungsorientierter Vertreter der Numerischen Harmonischen Analyse**, der Konzepte und Algorithmen entwickelt und propagiert.

Mein besonderes Interesse gilt dem Begriff der **Conceptual Harmonic Analysis**, die Distributionentheorie aber auch Numerische Fourier Analysis umfaßt. Gabor Analysis und Funktionenräume gehören zu meinen Spezialgebieten.

Gabor Analysis: Beethoven Piano Sonata



Das Spektrogramm einer Beethoven Sonate!

Das klassische Fourier Programm

1. J. B. Fourier: 1822: Jede periodische Funktion hat eine Darstellung als eine unendliche Summe von "harmonischen Schwingungen": $f(t) = \sum_{k\in\mathbb{Z}} c_k e^{2\pi ikt}$ (Fourier Reihe);
2. Die Klärung der Konvergenzfrage führte zu vielen wichtigen Begriffen der Analysis, z.B. Lebesgue Integral;
3. zufriedenstellende Theorie f.d. Hilbertraum $\left(L^2(\mathbb{U}), \|\cdot\|_2\right)$;
4. durch Übergang zur unendlichen Periode (und dementsprechend konintuierliches Frequenzspektrum) entsteht die *Fourier Transformation*;
5. Fouriertransformation und Umkehrformel für $L^1(\mathbb{R}^d)$; Satz von Plancherel: $\|f\|_2 = \|\hat{f}\|_2$;
6. DFT = Discrete Fourier Transf., FFT Algorithmus (1965).

Die Definition der Diskreten Fourier Tranformation (DFT)

Wir schreiben $f = [f(0), \cdots, f(L-1)] \in \mathbb{C}^L$ (komplexe Vektoren der Länge L), und betrachten f als **periodisches, diskretes Signal** (mit Periode L), mit $i = \sqrt{-1}$ wie üblich:

$$\widehat{f}(k) = \sum_{l=0}^{L-1} f(l) e^{-\frac{2\pi i k \cdot l}{L}} \qquad k = 0, \cdots, L-1 \qquad (1)$$

$$f(n) = \frac{1}{L} \sum_{l=0}^{L-1} \widehat{f}(l) e^{\frac{2\pi i n \cdot l}{L}} \qquad n = 0, \cdots, L-1 \qquad (2)$$

Die DFT in MATLAB Beschreibung

Da Vektoren keine "nullte Komponente haben", ist es besser (wenn man die DFT naiv aufgrund der Formel in MATLAB realisieren will) folgende Konvention zu verwenden: Für $\mathbf{x} = [x[1], \cdots, x[L]] \in \mathbb{C}^L$ ist die DFT (Discrete Fourier Transform) $\mathbf{y} = DFT(\mathbf{x})$ definiert als:

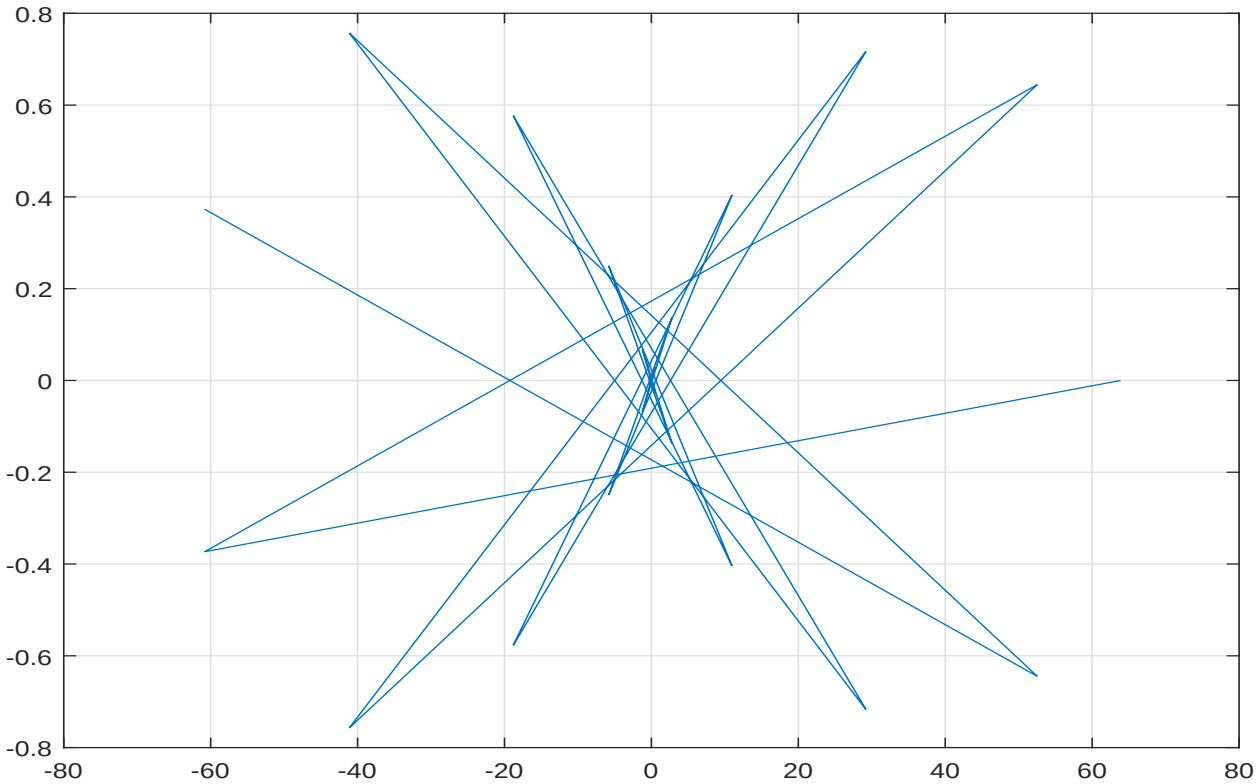
$$y[k] = \sum_{l=1}^{L} x[l] e^{-\frac{2\pi i (k-1)\cdot(l-1)}{L}} \qquad k = 1, \cdots, L. \tag{3}$$

$$x[n] = \frac{1}{L} \sum_{k=1}^{L} y[l] e^{\frac{2\pi i (n-1)\cdot(k-1)}{L}} \qquad n = 1, \cdots, L. \tag{4}$$

VORTEIL: Korrekte Indizierung, aber schwierige Interpretation der Koeffizienten! Außerdem gibt es ein Plotproblem!

Diskrete, Fourier-invariante Gauss-Funktion

Die Befehlsfolge: $bas = linspace(-4, 4, 512); gdisc = exp(-pi * bas.^2); plot(fft(gdisc))$, kann nur Verwirrung stiften.



Enorme Zeitunterschiede!

```
>> xx = rand(1,4410); d.h.  1/10 Sekunde Audiosignal
>> F = fft(eye(4410));
tic; myDFT(xx); toc; :NAIVE IMPLEMENTATION
tic; F*xx(:); toc
tic; fft(xx); toc
Elapsed time is 130.442276 seconds.
Elapsed time is 0.038570 seconds.
Elapsed time is 0.000194 seconds.

>> norm(fft(xx(:))  - F*xx(:))
ans = 3.7100e-12
>> yx = myDFT(xx); norm(yx - fft(xx))
ans = 1.8122e-09

>> tic; fft(eye(4410)); toc
Elapsed time is 0.315484 seconds.
```

Alternativen zum klassischen Programm?

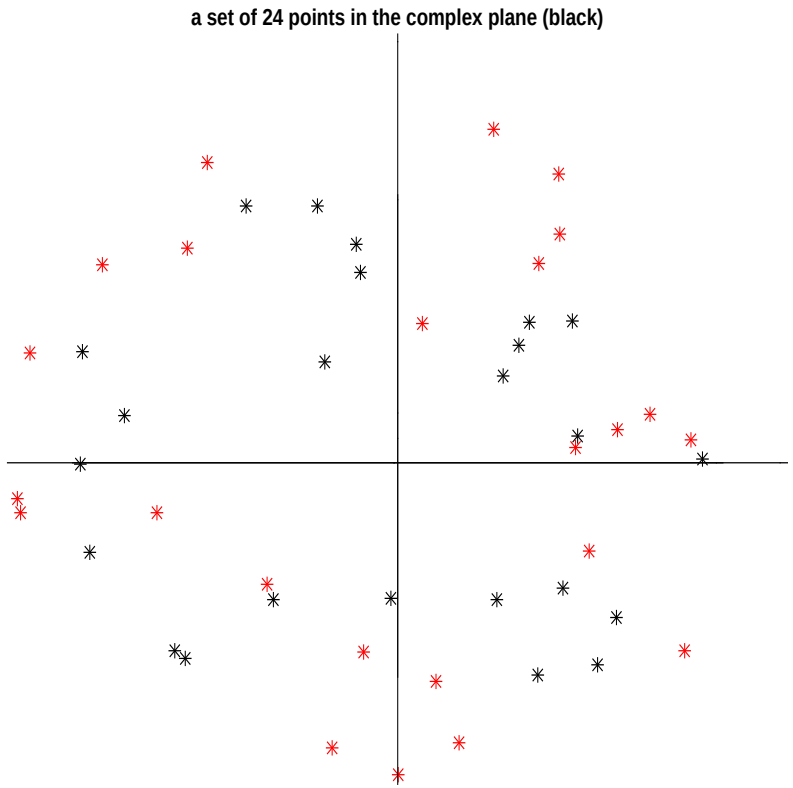
Ich behaupte, dass **MATLAB** eine Möglichkeit bietet, *von der Seite der linearen Algebra* den Einstieg in die wesentlichen Aspekte der Fourier Analysis auf einem anwendungsorientierten, aber ebenso mathematisch korrekten Weg klarzumachen.

So wie die reellen Zahlen als Grenzwerte von einfachen rationalen Zahlen beschrieben werden können, sollte man die kontinuierliche Theorie nur als Grenzfall von diskreten Signalen gesehen werden (so wie Pixel Bilder eine gute Approximation von kontinuierlichen Bildern sind!).

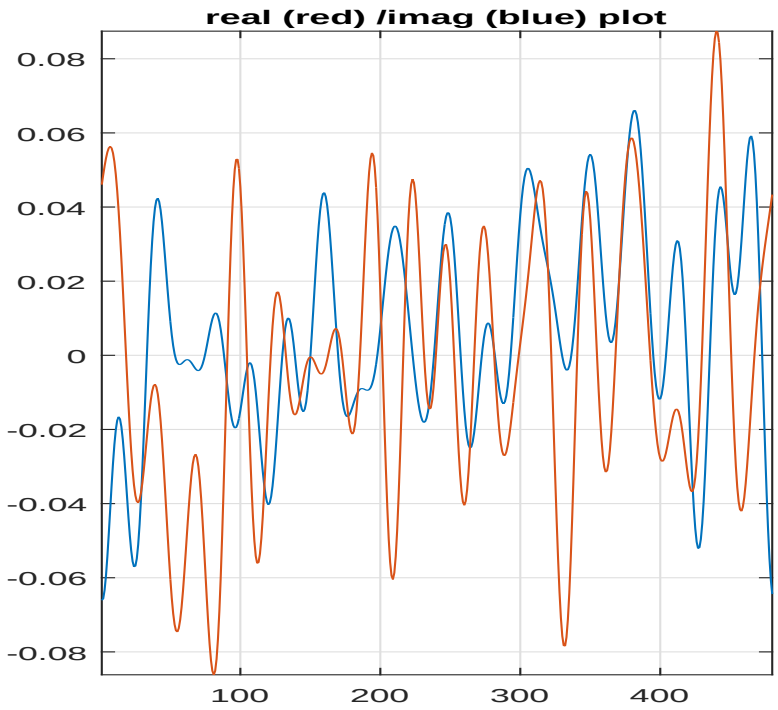
Zur Erkärung wollen wir ein paar Fakten aus der Linearen Algebra (für komplex-wertige Vektoren und Matrizen) in Erinnerung rufen!

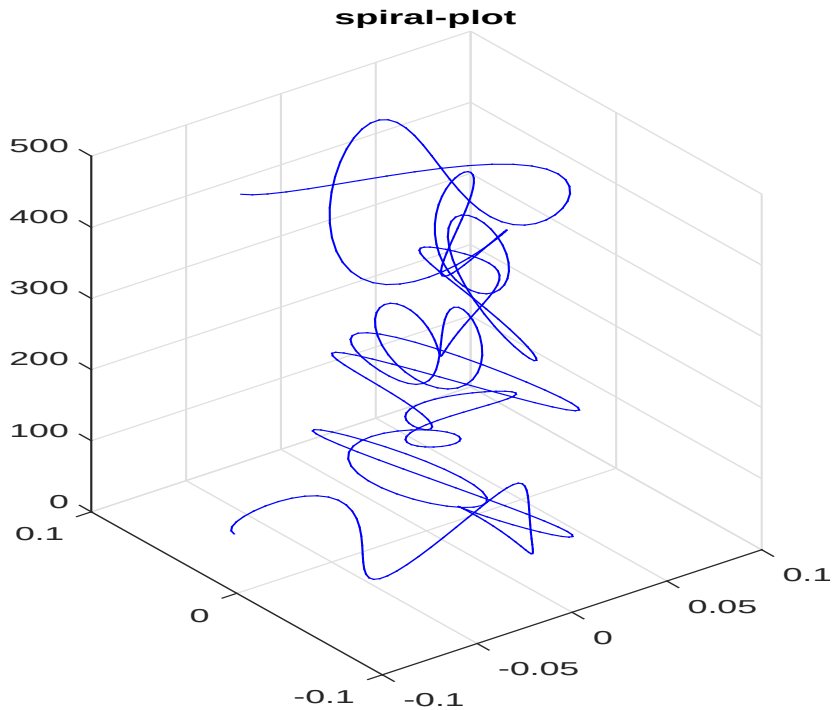
Multiplikation von komplexen Zahlen

Die rote Punktmenge ist durch eine komplexe Multiplikation aus der schwarzen hervorgegangen! Mit WELCHEM Faktor in $\mathbb{C}$? Erinnerung: Multiplikation entspricht einer Drehstreckung (Polardarstellung der komplexen Zahlen!).

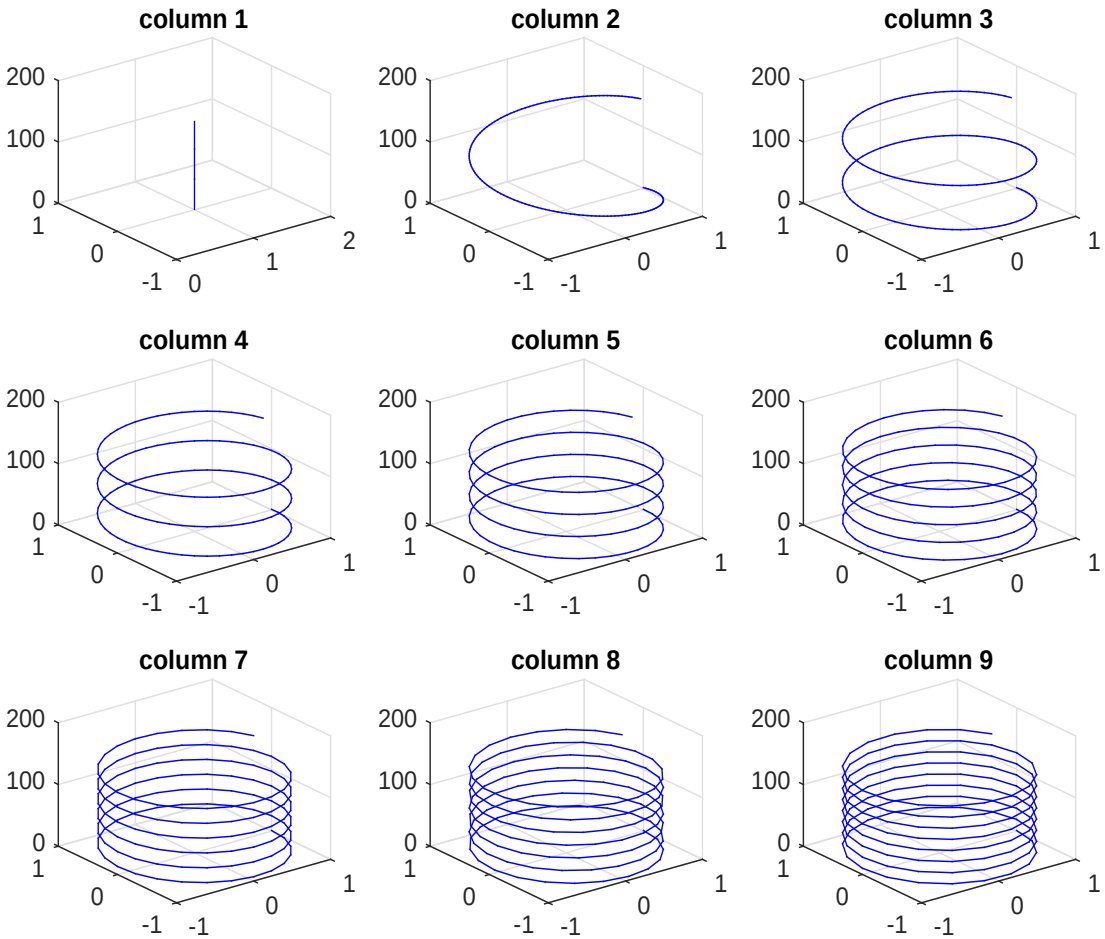


Plot von komplexwertigen Funktionen der Zeit





Spalten der Fourier Matrix: Spiralen



Linearen Algebra und MATLAB I

Erinnern wir uns an die Grundbegriffe der Linearen Algebra und ihre Realisierung mittels MATLAB, d.h. die Rolle von Matrizen und Matrix-Multiplikation:

Definition

Ein Vektor $\mathbf{b}$ ist eine Linearkombination, d.h. von der Form $\mathbf{b} = x_1\mathbf{a}_1 + \cdots x_n\mathbf{a}_n$ ist gleichbedeutend mit $\mathbf{b} = \boldsymbol{A} * \mathbf{x}$, wobei $(\mathbf{a}_k)_{k=1}^n$ die Spalten von $\boldsymbol{A}$ in $\mathbb{C}^m$ (oder $\mathbb{R}^m$) sind.

Dementsprechend sind Linear-Kombinationen von Linear-Kombinationen wieder Linear Kombinationen (der ursprünglichen Vektoren), und $\boldsymbol{B} = \boldsymbol{A}_2 * \boldsymbol{A}_1$ sagt uns, welche Koeffizienten man braucht.

Linearen Algebra und MATLAB II

Definition

Eine Abbildung T von $\mathbb{C}^n$ nach $\mathbb{C}^m$ heisst *linear*, wenn sie Linearkombinationen respektiert, d.h. wenn gilt

$$T\left(\sum_k c_k \mathbf{b}_k\right) = \sum_k c_k T(\mathbf{b}_k).$$

Konsequenterweise kennt man eine linear Abbildung, solange man sie nur auf den Basis Elementen kennt, und es ist naheliegend, T durch eine Matrix zu beschreiben, die einfach die (Koordinaten) der Bilder der Basisvektoren enthält, also durch eine $m \times n$ Matrix, mit $\mathbf{a}_k = T(\mathbf{e}_k), 1 \leq k \leq n$.

Verträglich mit Matrix Komposition und Inversion!

Linearen Algebra und MATLAB III

Es gibt in MATLAB keinen eigenen Befehl für das Skalarprodukt (in $\mathbb{C}^n$), weil $\langle \mathbf{x}, \mathbf{y} \rangle$ ohnehin durch den Befehl $\mathbf{y}' * \mathbf{x}$ realisiert werden kann. Die bekannt Formel

$$\langle \boldsymbol{A} * \mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{x}, \boldsymbol{A}' * \mathbf{y} \rangle$$

ergibt sich so aus der Assoziativität der Matrix-Multiplikation und der bekannten Regel

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

Insbesondere ist $\mathbf{y} \mapsto \boldsymbol{A}' * \mathbf{y}$ für $\mathbf{y} \in \mathbb{C}^m$ gleichbedeutend mit

$$\mathbf{y} \mapsto (\langle \mathbf{y}, \mathbf{a}_k \rangle)_{k=1}^{n}.$$

Linearen Algebra und MATLAB IV

Daraus folgt wieder unmittelbar (mit ein wenig Theorie betreffend inverse Elemente) dass eine $n \times n$-Matrix $\mathbf{U}$, deren Spalten ein Orthonormalsystem bilden, d.h. mit $\mathbf{U}' * \mathbf{U} = \mathrm{Id}_{\mathbf{n}}$, auch

$$\mathbf{U} * \mathbf{U}' * \mathbf{x} = \mathbf{x}, \quad \mathbf{x} \in \mathbb{C}^n$$

erfüllen, d.h. es gilt für $\mathbf{x} \in \mathbb{C}^n$:

$$\mathbf{x} = \sum_{k=1}^{n} \langle \mathbf{x}, \mathbf{u}_k \rangle \mathbf{u}_k,$$

m.a.W.: die eindeutigen Koeffizienten bzgl. der Basis $(\mathbf{u}_k)$ sind durch die Skalarprodukte mit diesen Vektoren gegeben!

Linearen Algebra und MATLAB V

Ein wichtiges Beispiel ist die Abbildung, die den Koeffizienten $\vec{a}$ eines Polynoms (entsprechend den MATLAB Konventionen)

$$p(z) = a_1 z^{n-1} + a_2 z^{n-2} \dots a_n$$

mittels `polyval(a,z)` die Werte an den Stellen z_k in $\mathbb{C}$ zuordnet. Diese ist genau dann invertierbar ist, wenn wir n verschiedene Stellen betrachten, wenn wir also Daten $\vec{d} \in \mathbb{C}^n$ haben.

Dann gilt für die Vandermonde Matrix `vander(z)` mit $\vec{z} \in \mathbb{C}^n$

$$\vec{d} = polyval(a, z) = vander(z) * a;$$

bzw.

$$a = inv(vander(z)) * d.$$

Von der Linearen Algebra zur Fourier Transformation

Da sich die DFT (die Diskrete FT), implementiert in Form der FFT (Fast FT) durch Lineare Algebra und folglich durch Matrizen-Rechnung verständlich gemacht werden kann, schlage ich einen Zugang von dieser Seite, unter Zuhilfe von MATLAB vor: Dazu kann man von der **FFT-Routine** ausgehen, und **experimentell einige Grundfakten verifizieren**.

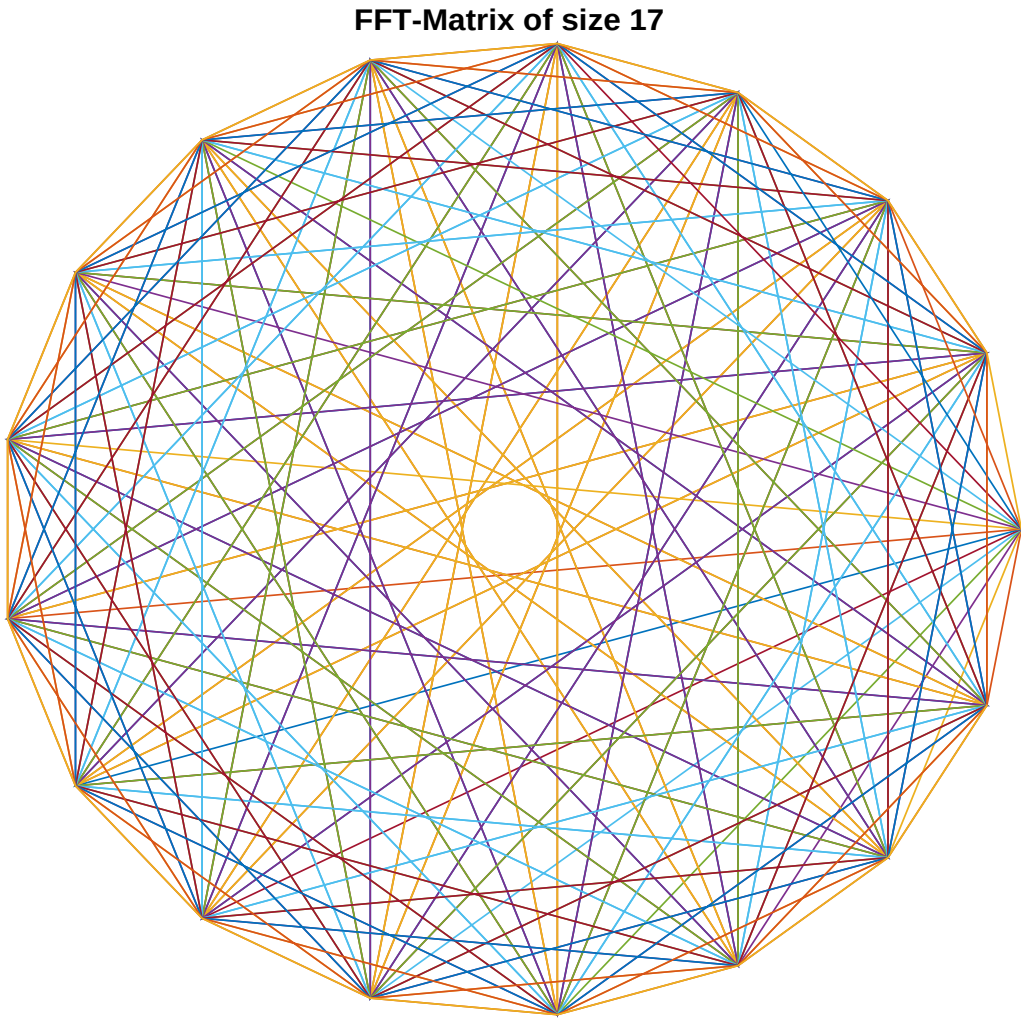
Da der *output* der FFT-Routine offensichtlich eine komplexwertige Folge ist, wollen wir die FFT gleich als eine Abbildung von $\mathbb{C}^N$ nach $\mathbb{C}^N$ ansehen (offensichtlich ist das output Format gleich dem input Format!), wie ein einfacher TEST zeigt:

```
fft(rand(1,5)), fft(rand(4,1)), fft(rand(3,4)),
```

Von der Linearen Algebra zur Fourier Transformation II

1. **Die FFT bewahrt die (euklidische) Norm von Vektoren**, bis auf einen Faktor $\sqrt{N}$, der von der Länge abhängt.
   TEST: `x = rand(N,1);`
   `norm(fft(x))/(sqrt(N)*norm(x))`,
2. **die FFT ist linear**, wie durch folgenden Test "verifizert" wird: `A = rand(3,4); x = rand(4,1);`
   `norm(fft(A)*x - fft(A*x))`,
   Nach den Konventionen **agiert die FFT auf einer Matrix spaltenweise**. Also zeigt der obige Test, dass die FFT-Routine (jedenfalls eine zufällige) Linear-Kombinationen respektiert!
3. `F = fft(eye(4)), norm(fft(x) - F * x)`,
   zeigt, dass die Matrix Multiplikation mit F tatsächlich der Anwendung der FFT Routine entspricht.

Illustration der FFT-Matrix der Größe 17



Von der Linearen Algebra zur Fourier Transformation III

Da nun $x \mapsto fft(x)$ eine lineare Abbildung ist, ist es interessant, die zugehörige Matrix weiter zu analysieren:

```
N = 17; F17 = fft(eye(N)); plot(F17); axis square;
```

Diese Matrix ist auch (reell) symmetrisch:

```
>> norm(F - F.', 'fro'), ans = 1.4316e-14
```

Andererseits ist es interessant, sich die Eintragungen dieser Matrix spaltenweise anzusehen. Etwa für $N = 144$:

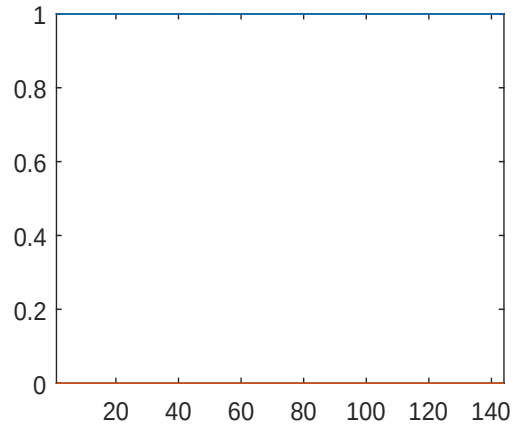
```
>> N = 144; F = fft(eye(N));
>> for jj=1:6; plot(1:N,real(F(:,jj)),1:N,
imag(F(:,jj))); shg; pause; end;
```

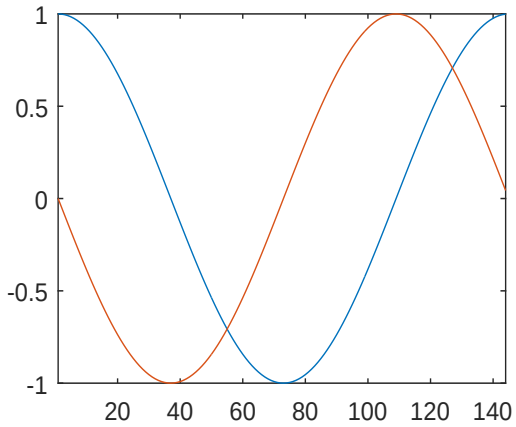
Die Spalten dieser Matrix haben alle gleiche Länge (: $\sqrt{N}$)

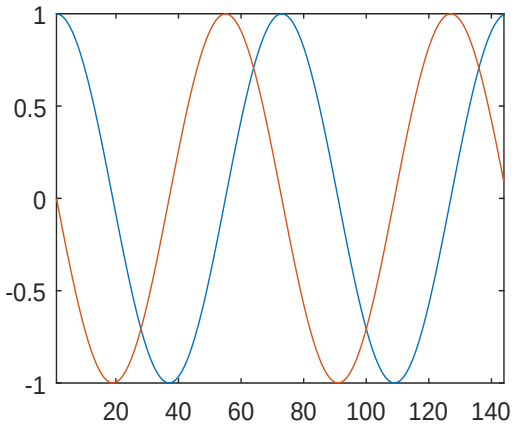
```
for jj=1:N; lF(jj)=norm(F(:,jj)); end;
norm( ones(1,N)*sqrt(N) - lF),
```

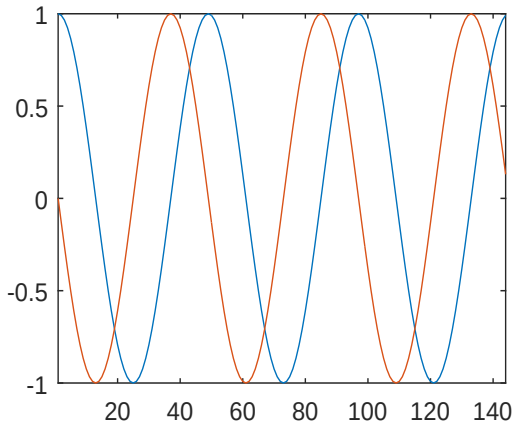
Die ersten Spalten der Fourier Matrix

Die Spalten der Fourier Matrix: die reinen Frequenzen!









Reine Frequenzen und Fourier Synthese

Die Spalten (= Zeilen) der Fourier Matrix sind die soeben gezeigten "reinen Frequenzen".
Nach der **Eulerschen Formel** sind sie von der Form

$$\chi_k(t) := e^{2\pi jkt} = \cos(2\pi kt) + j\sin(2\pi kt).$$

Diese sind insoferne "natürliche Bausteine", als sie (mit komplexen Eigenwerten) Eigenvektoren der Translationsoperatoren sind, und zwar als Folge des Exponentialgesetzes:

$$T_x\chi_k(t) := \chi_k(t-x) = e^{2\pi jk(t-x)} = e^{-2\pi jkx} \cdot \chi_k(t).$$

Von der Linearen Algebra zur Fourier Transformation IV

Natürlich kann man sich erhoffen, dass die FFT-Routine einem **orthogonalen Basis-Wechsel** entspricht. Zur Erinnerung:

```
A = rand(N); U = orth(A);
norm(U'*U - eye(N)), norm(U*U' - eye(N)),
```

Da die Spalten von F (und daher auch die Spalten von $F.' = F$) alle Länge $\sqrt{N}$ haben, kann man nur erwarten, dass **bis auf den Normierungsfaktor N die Matrix F' die inverse zu F ist**:

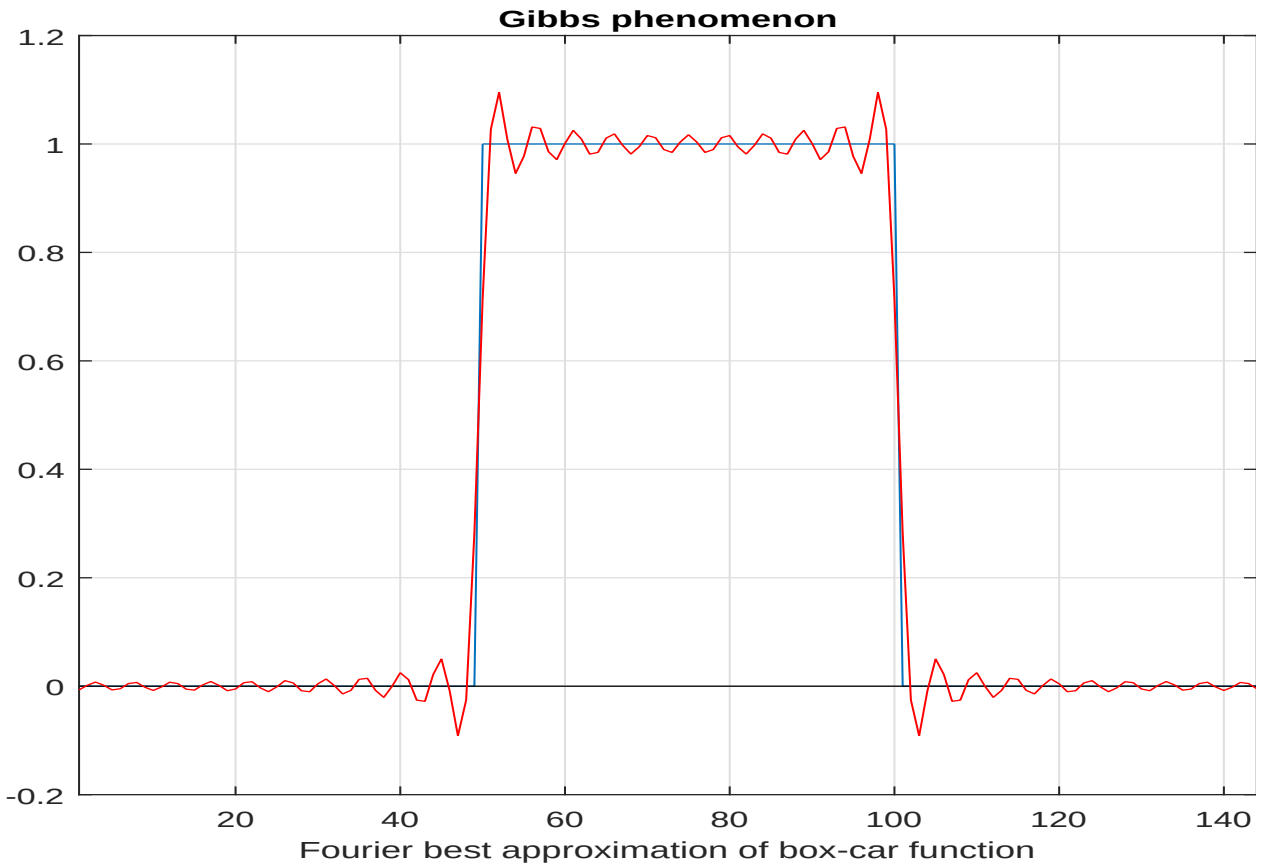
```
norm(F*F'/N - eye(N))
```

Die Bestapproximation durch gewisse Frequenzen, wobei J eine Teilmenge von $1 : N$ sei:

```
bx = zeros(1,N); bx(50:100) = 1; plot(bx);
ax20; grid; plotax;
```

Fourier Approximation of Box-car function

Die Best-Approximation (im Euklid. Sinne) der Rechtecksfunktion durch 61 Fourier-Koeffizienten (max. Frequenz 30):



Reine Frequenzen und Fourier Synthese

$$\mathbf{x} = F' * (F * \mathbf{x}/N), \quad \text{with} \quad F = fft(eye(N)),$$

zeigt also, dass jedes Signal als Summe der reinen Frequenzen geschrieben wird, und dass (bis auf den Faktor $1/N$) die Fourier Koeffizienten gerade die eindeutigen Koeffizienten sind.
Man spricht also bei der Darstellungsformel

$$\mathbf{x} = \sum_{k=0}^{N-1} c_k \chi_k$$

von der **Fourier Synthese**, und die Koeffizienten sind gerade von der Form $\mathbf{c} = fft(\mathbf{x})/N$. Dementsprechend ist $\mathbf{x} \mapsto fft(\mathbf{x})$ die **Fourier Analyse**.

Von der Linearen Algebra zur Fourier Transformation V

Einige der wichtigen Eigenschaften der Fourier Transformation ergeben sich daraus, dass sie die *Faltung* in punktweise Multiplikation überführt, und so dazu dient, *translations-invariante lineare Systeme* als Multiplikations-Operatoren darzustellen (*Transfer Funktion*).
Dazu beobachten wir die Wirkung einer *zyklischen Rotation*, in Matrix Format ($N = 7$, shift um 2 samples):

$$S2 := \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \tag{5}$$

Von der Linearen Algebra zur Fourier Transformation VI

Konjugiert man die Standard-Matrix (an jeder Position steht die Koordinaten-Nummer, von 1 bis 49) mit $S2$, d.h. bildet man $S2 * T7 * inv(S2)$, so bekommt man:

$$\begin{pmatrix} 41 & 48 & 6 & 13 & 20 & 27 & 34 \\ 42 & 49 & 7 & 14 & 21 & 28 & 35 \\ 36 & 43 & 1 & 8 & 15 & 22 & 29 \\ 37 & 44 & 2 & 9 & 16 & 23 & 30 \\ 38 & 45 & 3 & 10 & 17 & 24 & 31 \\ 39 & 46 & 4 & 11 & 18 & 25 & 32 \\ 40 & 47 & 5 & 12 & 19 & 26 & 33 \end{pmatrix} \tag{6}$$

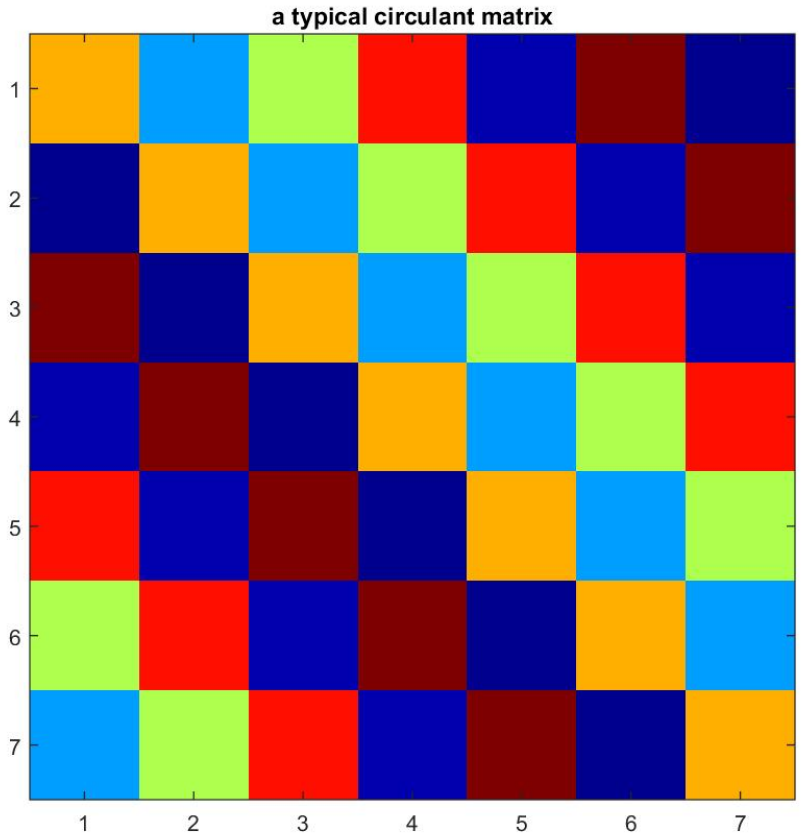
d.h. alle Eintragungen wandern entlang der Hauptdiagonale um zwei Einheiten nach unten.

Von der Linearen Algebra zur Fourier Transformation VII

Entsprechend kommutiert eine 7×7 Matrix mit allen (zyklischen) Verschiebungen genau dann, wenn sie (im zyklischen Sinne) konstant entlang er Hauptdiagonalen ist.
Wir verwenden dazu einen eigenen Befehl: `c = rand(1,7);`
`convc = convmat(c),`

$$\begin{pmatrix} 0.7133 & 0.5072 & 0.6387 & 0.7903 & 0.3971 & 0.8663 & 0.3760 \\ 0.3760 & 0.7133 & 0.5072 & 0.6387 & 0.7903 & 0.3971 & 0.8663 \\ 0.8663 & 0.3760 & 0.7133 & 0.5072 & 0.6387 & 0.7903 & 0.3971 \\ 0.3971 & 0.8663 & 0.3760 & 0.7133 & 0.5072 & 0.6387 & 0.7903 \\ 0.7903 & 0.3971 & 0.8663 & 0.3760 & 0.7133 & 0.5072 & 0.6387 \\ 0.6387 & 0.7903 & 0.3971 & 0.8663 & 0.3760 & 0.7133 & 0.5072 \\ 0.5072 & 0.6387 & 0.7903 & 0.3971 & 0.8663 & 0.3760 & 0.7133 \end{pmatrix} \quad (7)$$

Eine typische zirkulante Matrix



Diagonalisierung von zirkulanten Matrizen

```
F7 = fft(eye(7)); D7 = F7' * convc * F7;
imagesc(abs(D7)); shg
```

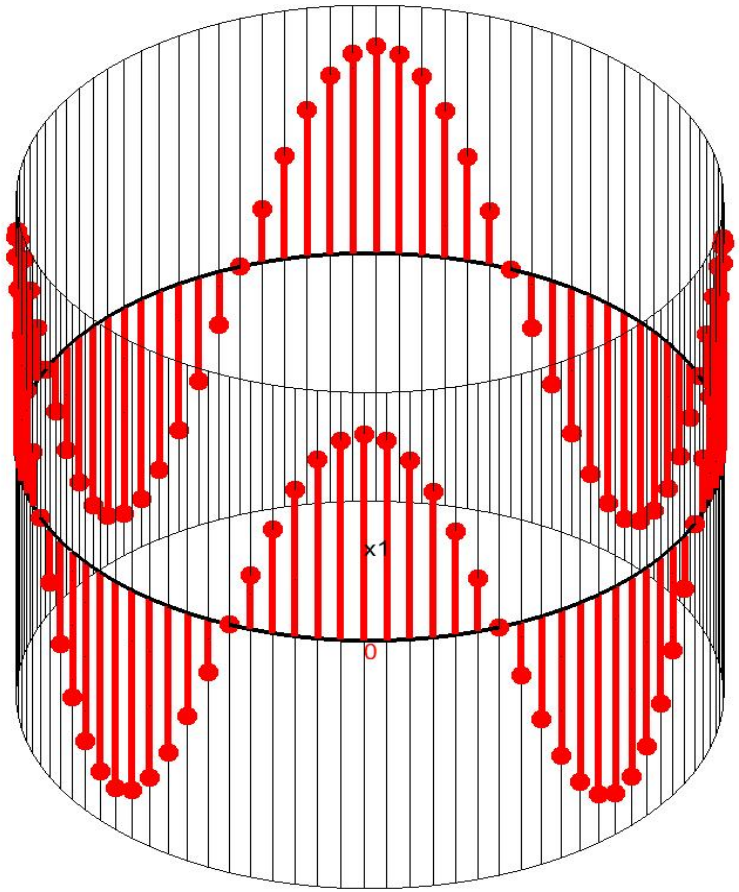
illustriert die Tatsache, dass die Fourier-Matrix als Basis-Wechsel gedacht, eine (jede!) zirkulante Matrix diagonalisiert!

`norm(D7 - diag(diag(D7))),`
zeigt, dass $D7$ tatsächlich eine Diagonalmatrix ist!
`diag(diag(A))` ist der "reine Diagonalteil" der Matrix!

Weiters ist die Eintragung in der Diagonale genau die Fourier Transformierte des erzeugenden Vektors c:
`norm(fft(c(:))*7 - diag(D7))`

Warum "zyklische" Translation?

Der zyklische Translations-Operator ist in Wahrheit ein Verschiebungs-Operator für (unendliche) *periodische* Folgen:



Wichtige Eigenschaften der FFT/DFT

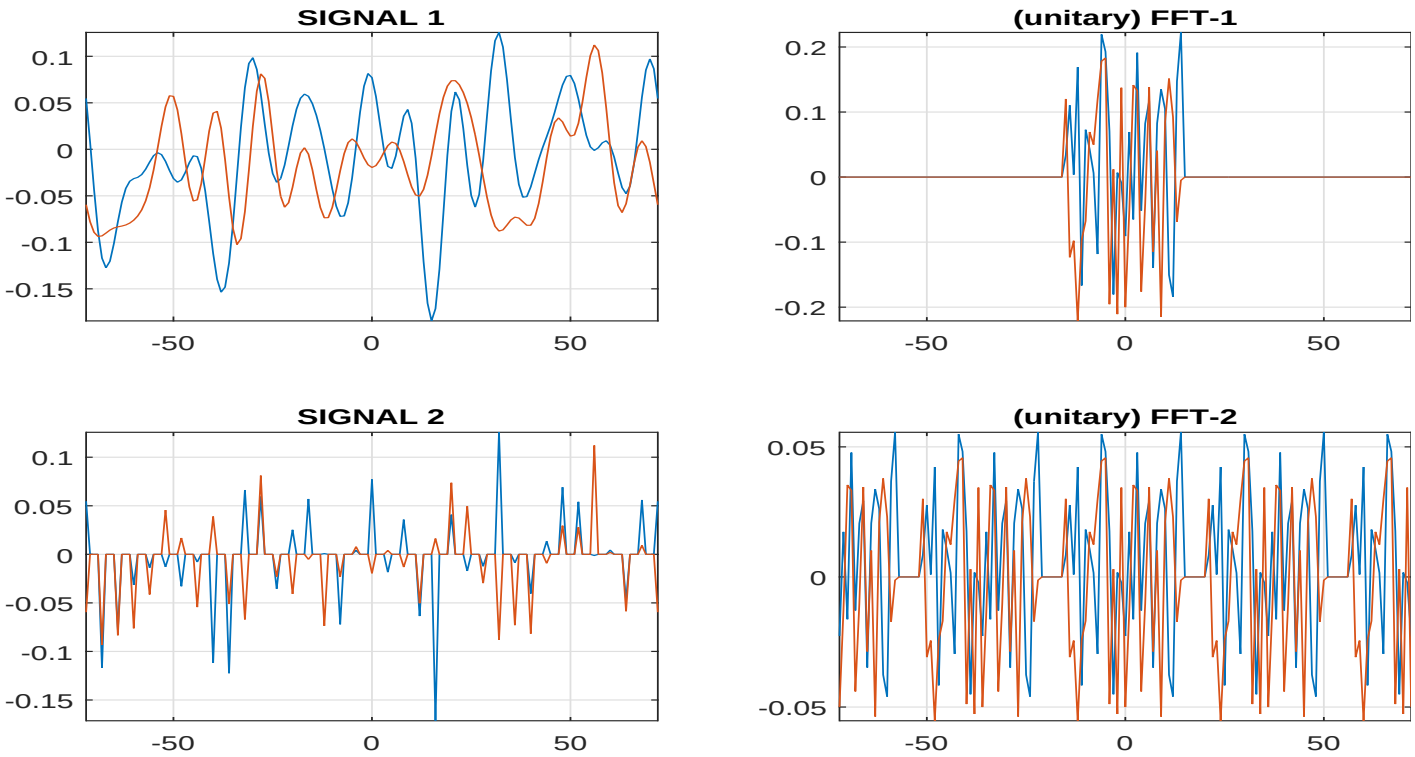
Da eine (zyklische) Verschiebung der Koeffizienten einfach eine Erhöhung der Potenz bedeutet, entspricht dies eine Multiplikation der Fourier Koeffizienten mit einer reinen Frequenz.

Ebenso ist die Prozedur des *downsampling* gut erklärbar. Ersetzt man die Folge $[c_1, \ldots, c_{2n}]$ durch die Folge

$$[c_1, 0, c_3, 0, c_5, \ldots c_{2n-1}, 0]$$

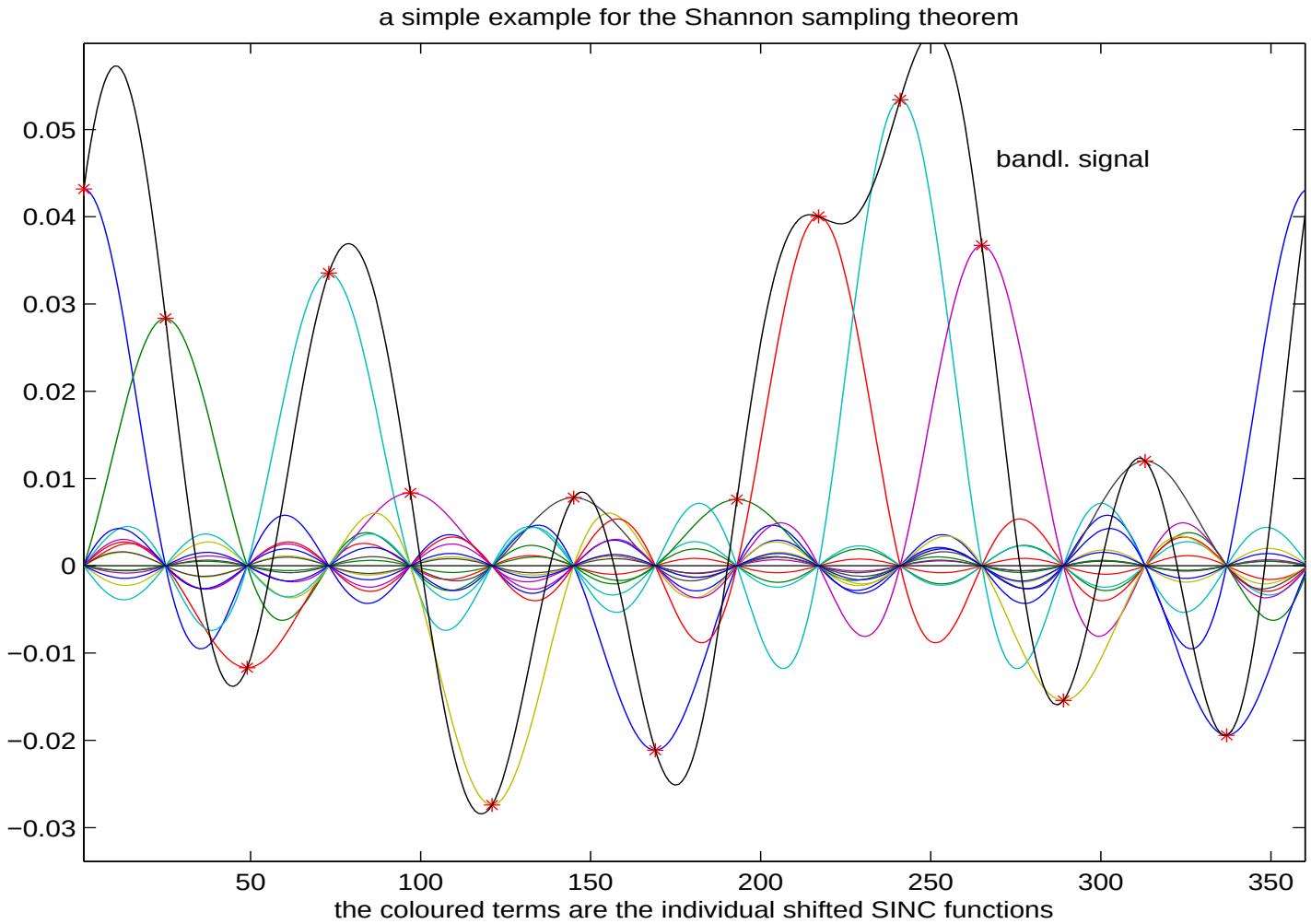
so entspricht die Anwendung der FFT Routine der Auswertung eines Polynoms $q(z)$. Setzt man $d_k := c_{2k-1}, 1 \leq k \leq n$ so gilt offenbar $p(z) = q(z^2)$, mit $q(y) = d_1 + d_2 y + \ldots d_n y^{n-1}$.

Shannon Abtast Theorem



Wie man *sieht* kann man den zentralen Block (rund um Null) durch Multiplikation mit einem Filter zurückgewinnen.

Rekonstruktion mittels Shifted SINC-Funktionen



Typische Fehler and Tricks

Es gibt einige Bücher und Hinweise in der angewandten Literatur, die auf "Probleme" und spezielle Tricks hinweisen, die notwendig sind, wenn man von der kontinuierlichen FT zur diskreten FT übergeht. "Angeblich" herrschen im Diskreten "andere Verhältnisse"!??

Beispielsweise kann man sich fragen, inwieferne die wichtige Eigenschaft der normierten Gauss-Funktion $g_0(t) := e^{-\pi|t|^2}$, nämlich invariant unter der (Integral-Form der) Fourier Transformation zu sein, also $\mathcal{F}(g_0) = g_0$ zu erfüllen, ein Analogon in der endlich-diskreten Situation zu haben.

Richtiges Abtasten und Approximation durch Vektoren

Wir sehen also, dass die "naive Form" der Anwendung der FFT *nicht zu einer FFT-invarianten* diskreten Gauss-Folge führt (wir sehen in Wirklichkeit einen Plot in der komplexen Ebene!).

Auch das Prinzip, dass "symmetrische Folgen" reelle FFTs haben sollten, muss mit Vorsicht behandelt werden. Was heißt es für eine Folge $(c_k)_{k=1}^N$ symmetrisch bzw. hermitsch zu sein?

Eine zirkulante Matrix ist genau dann hermitsch wenn:
$c_1 \geq 0$, und $c_{k+1} = conj(c(N+1-k))$ für $1 \leq k \leq N/2$.

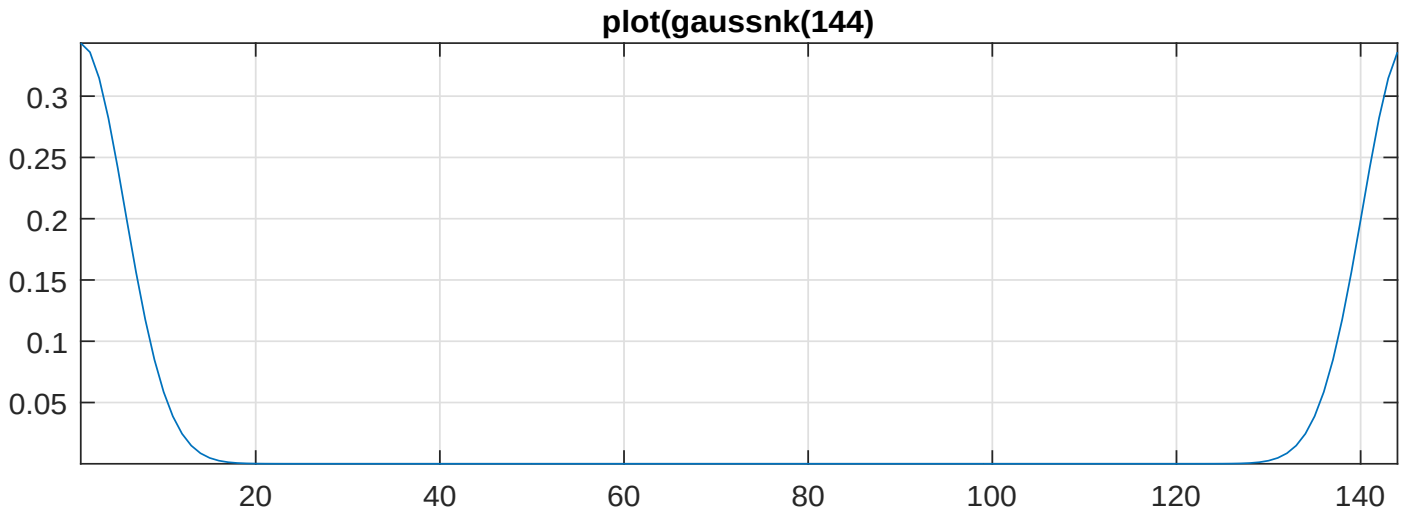
REZEPT: **Sampeln und Periodisieren**, idealerweise Samplingrate $1/k$ und Periode k, mit $k \in \mathbb{N}$.

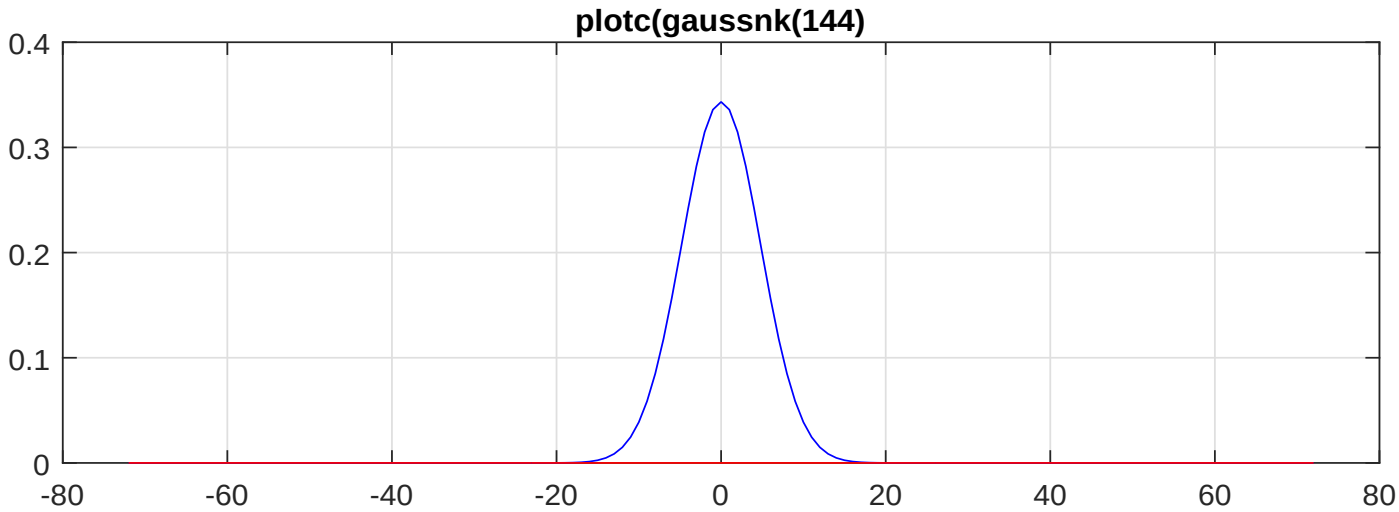
Datenstruktur und Visualisierung

Wir haben gesehen, dass die DFT Vektoren der Länge N auf Funktionswerte (einer Polynomfunktion) auf der Gruppe der N-ten Einheitswurzeln überträgt.
Aber auch die Koeffizentenfolge selbst kann als Funktion auf der Gruppe $\mathbf{U}_N$ angesehen werden, denn es gilt für Elemente $z \in \mathbf{U}_N$: $z^{k+N} = z^k$. Insbesondere wird ja bei der Polynomauswertung (jetzt der *math.* Tradition folgend!) dem ersten Koeffizienten c_1 der konstante Term zugeordent, und es gilt $1 = z^0$!

Dementsprechend muss man zur Visualisierung eines Vektors die ersten Koordinate “in den Mittelpunkt rücken.

Korrektes Ploten einer zentrierten Gaussfunktion





Ausblick auf das kontinuierliche Setting

Geht man nun wieder zurück (ich würde sagen, weiter) zum setting von $\mathbb{R}^d$ (kontinuierliche, nicht-periodische Signale), dann sollte man die Situation so beschreiben, dass die Idee erhalten bleibt, dass die Fourier Transformation die *reinen Frequenzen* (z.B. plane waves im Falle $d = 2$) in "Einheitsvektoren" üeberführt.
Obwohl es naheliegend erscheint (siehe Literatur!) die Fourier Transformation als *Integral-Transformation* zu beschreiben, und daher diese nur auf $f \in \boldsymbol{L}^1(\mathbb{R}^d)$ oder $f \in \boldsymbol{L}^2(\mathbb{R}^d)$ (Satz von Plancherel) anzuwenden, ist es wichtig, auch die Charakterie $\chi_s(t) = exp(2\pi i \langle s, t \rangle)$ sowie die *Dirac Maßse* δ_s einzubringen.
Eine solche Betrachtungsweise erfordert die Einführung von *Distributionen* (verallgemeinerten Funktionen), aber das ist eine anderer Geschichte!

Fourier Basis ist kein Zufall

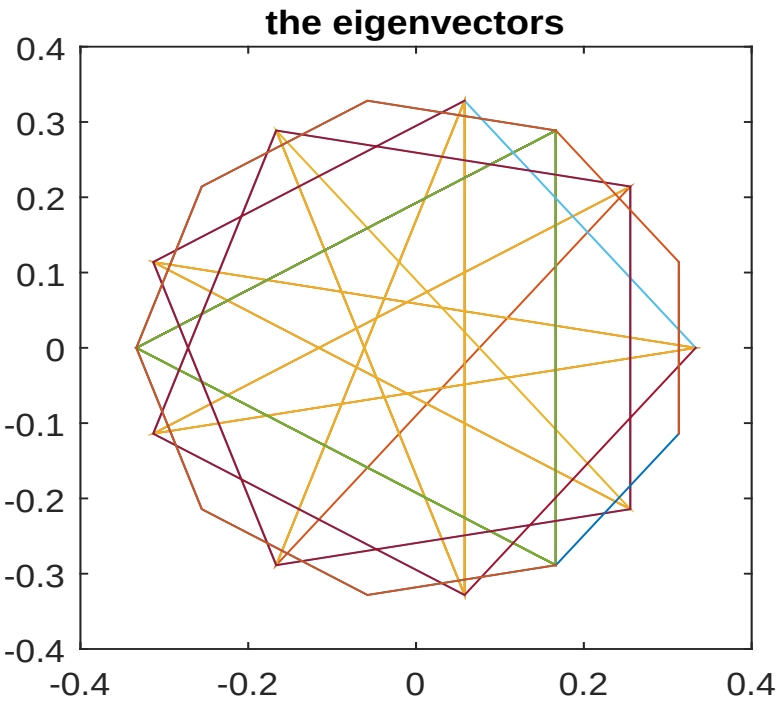
Die Wahl der Fourier Basis ist keine Zufall!!

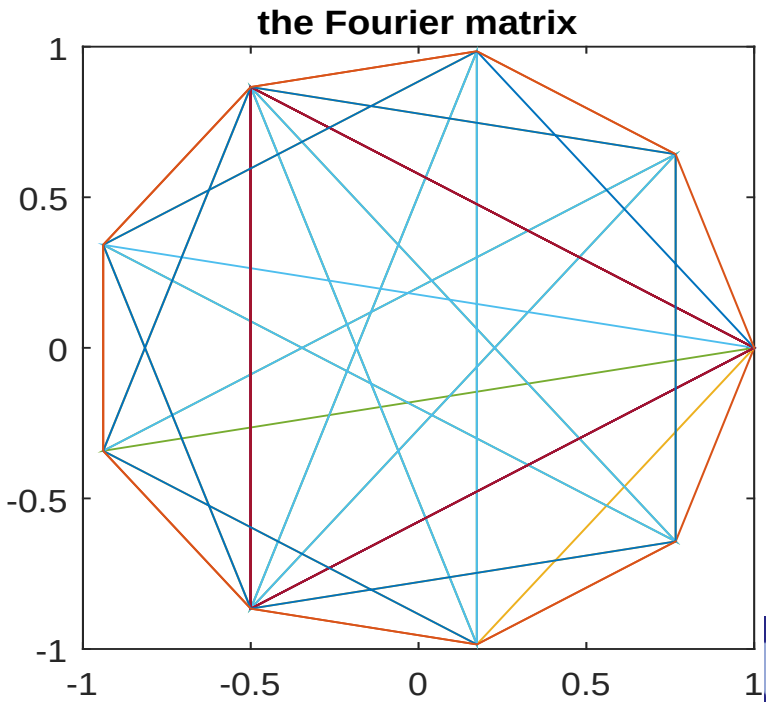
Es ist nicht nur so, dass die Fourier Basis aufgrund des Exponential-Gesetzes alle zyklischen Translationsoperatoren in Multiplikationsoperatoren überführt, sie diagonalisieren vielmehr *alle translationsinvarianten linearen Systeme*! Die Begründung: die System-Matrizen sind Summen von reinen Translationsmatrizen.

Es gilt aber auch die Umkehrung. Mehr oder minder jedes (zufällige) translationsinvariante linear System (m.a.W., jede positiv definite zirkulante Matrix) hat als Orthonormalbasis von Eigenvektoren (!!bis auf das Vorzeichen) genau die reinen Frequenzen!

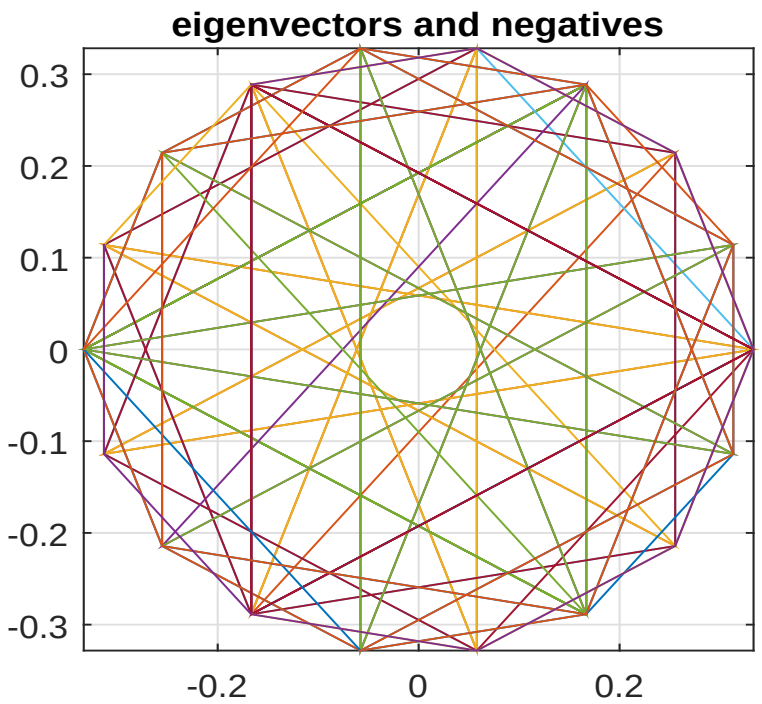
Eigenvektoren von Zufalls-Systemen

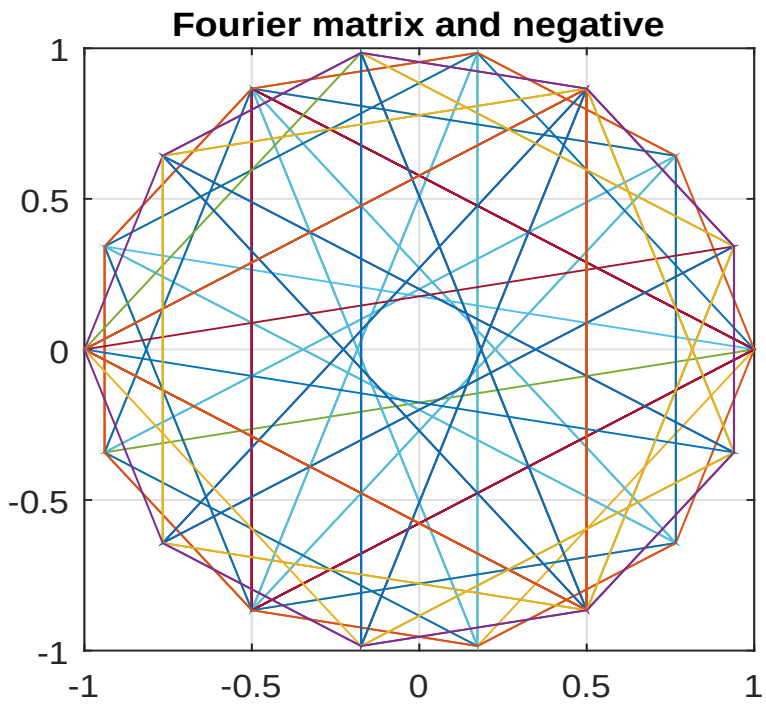
Visualisieren wir die Eigenvektoren einer solchen Matrix (der Einfachheit halber f.d. Fall $N = 9$):





Eigenvektoren von Zufalls-Systemen





CONV und Cauchy-Produkt der Koeffizieten

Die Polynomfunktionen bilden nicht nur ein Vektorraum, welcher auch noch invariant bzgl. Translationen und Streckungen ist, sie bilden auch eine *kommutative Algebra*.

Die Bedeutung des *Grades eines Polynoms* liegt darin, dass sich beim Multiplizieren der Polynomfunktionen die Grade addieren!

Kubische Polynomfunktionen sind klarerweise vom Grad $= 3$, bzw. von der *Ordnung* (Zahl der Koeffizienten) $= 4$. In der

Beschreibung auf der Ebene der Koeffizienten (CONV!) kann die Multiplikation mit Hilfe des sogenannten **Cauchy-Produktes** charakterisiert werden (ausmultiplizieren und neu anordnen!)

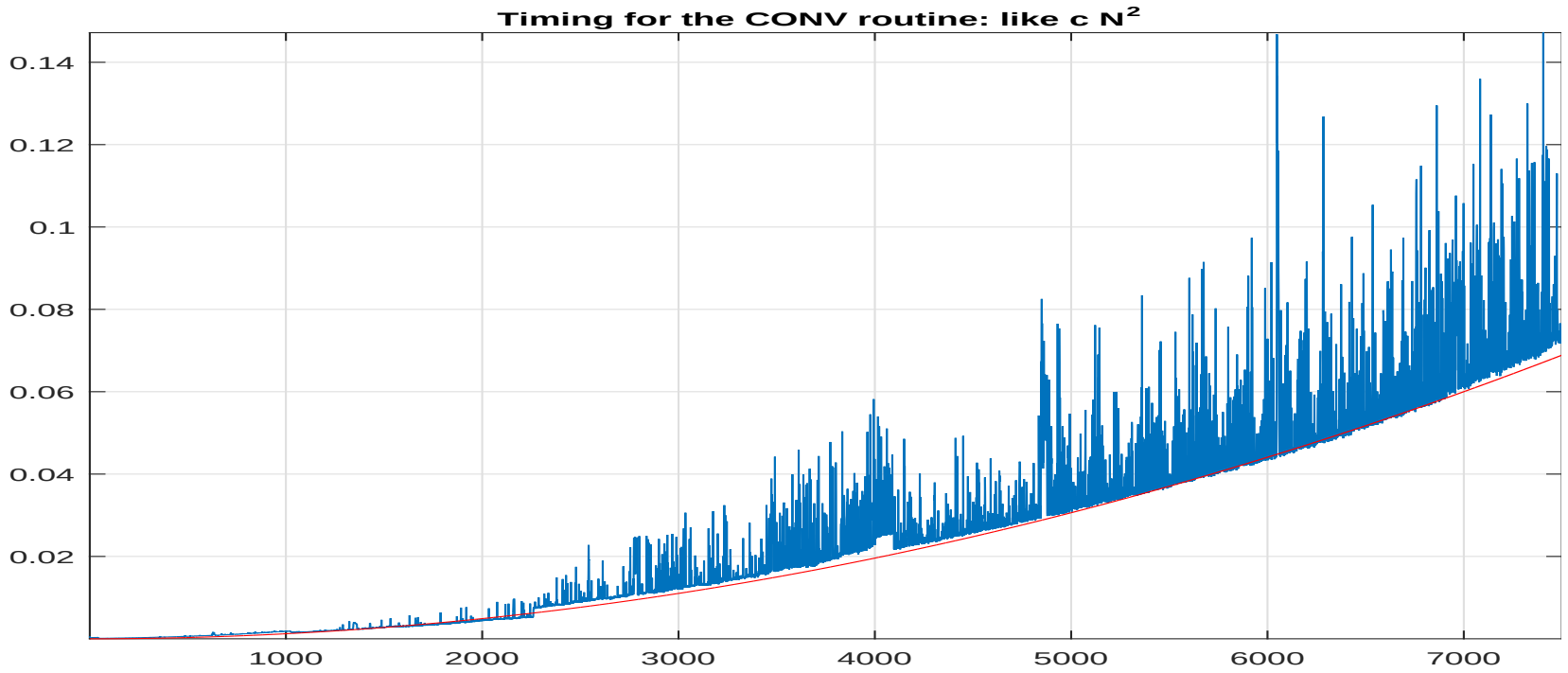
$$c_k = \sum_{k=0}^{r+s} a_k b_{r+s-k}.$$

test Der CONV-Befehl und das PASCAL Dreieck

```
>> format compact >> aa = [1,1]; bb = aa; >> for
jj=1:5; bb = conv(bb,aa), end
bb = 1 2 1
bb = 1 3 3 1
bb = 1 4 6 4 1
bb = 1 5 10 10 5 1
bb = 1 6 15 20 15 6 1
```

Liefert offenbar die Koeffizenten des Polynoms mit den Koeffizienten $[1,1]$, also $p(x) = 1 \cdot x + 1 = x + 1$ bzw. $(x+1)^k, k = 2, \ldots, 6$.

Dauer des CONV-Befehls für wachsendes N



Unkonventionelle Anwendungen

1. Multiplikation von sehr langen Zahlen
   (Polynome an der Stelle $t = 10$);
2. Summen von zufälligen Variablen mit diskreten Verteilungen;
3. Demonstration des zentralen Grenzwertsatzes;
   Binomial-Verteilungen exakt;
4. Multiplikation und Division von Polynomen in 2 Variablen;

Nicht-periodische Funktionen: die STFT

Normalerweise wird argumentiert, dass man bei der Behandlung von *nicht-periodischen* Signalen integrierbare oder wenigstens quadratisch integrierbare Funktionen nehmen muss. Die übliche Argumentation betrachtet Fourier-Reihen-Entwicklungen mit immer dichterem Spektraum (die Grund-Periode wird größer!). Das Ergebnis ist eine durch in Integral definiert koninuierliche Fourier-Transformation

$$\hat{f}(s) = \int_{\mathbb{R}^d} f(t) e^{2\pi i\, s\bullet t}\, dt, \quad s \in \mathbb{R}^d.$$

In diesem Setting kann (Satz von Plancherel) die FT als unitärer Automorphismus von $\left(\boldsymbol{L}^2(\mathbb{R}^d), \|\cdot\|_2\right)$ aufgefaßt werden, mit

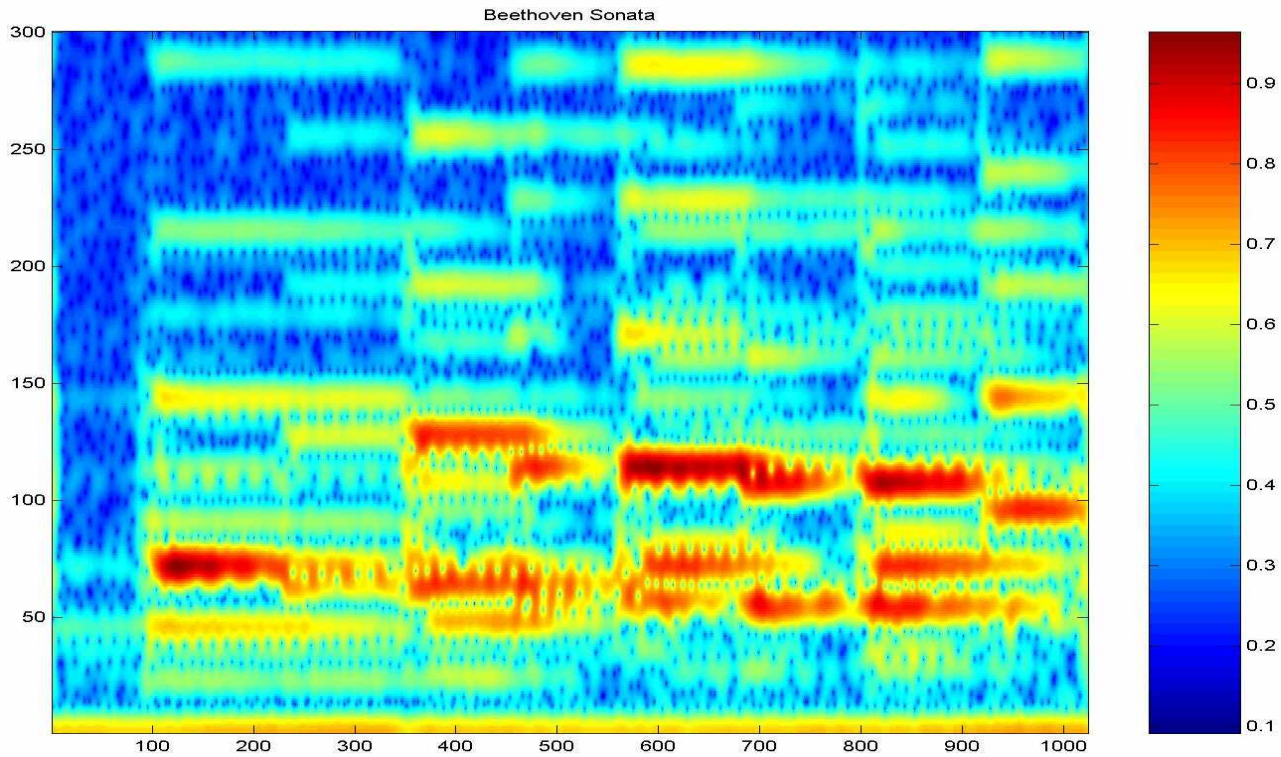
$$\|f\|_2 = \|\hat{f}\|_2, \quad f \in \boldsymbol{L}^2(\mathbb{R}^d).$$

die Kurzzeit-Fourier-Transformation und Anwendungen

Aber diese Verallgemeinerung der endlichen Fourier-Transformation enthält keinerlei Information, wann welches Sound-Ereignis stattfindet. Kurz gesagt, es erlaubt keine Aussage, über die *Melodie* die dem analysierten Signal zugrundelag. Dies kann nur mit Hilfe der sogenannten **Kurzzeit-Fourier-Transformation** bzw. **Sliding Window Fourier Transform** bewerkstelligt werden. Auch diese läßt sich mit Hilfe von MATLAB gut mathematisch erklären.

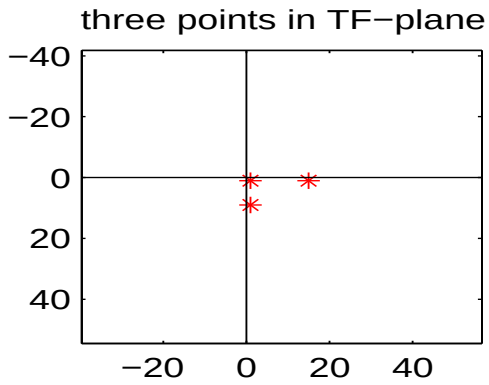
Es gibt zwei Interpretationen der sog. **Gabor Analysis**. Entweder geht es um eine Rekonstruktion einer STFT aus einer diskret abgetasteten Version, andererseits geht es um eine *atomare Darstellung* von Signalen mit Hilfe von zeit-frequenz-verschobenen Varianten eines *Gabor Atoms*.

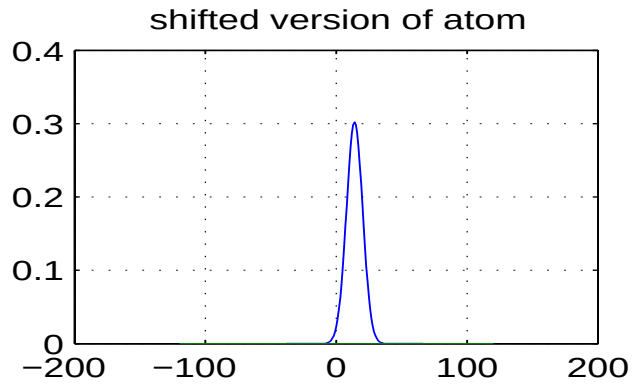
Gabor Analysis: Beethoven Piano Sonata

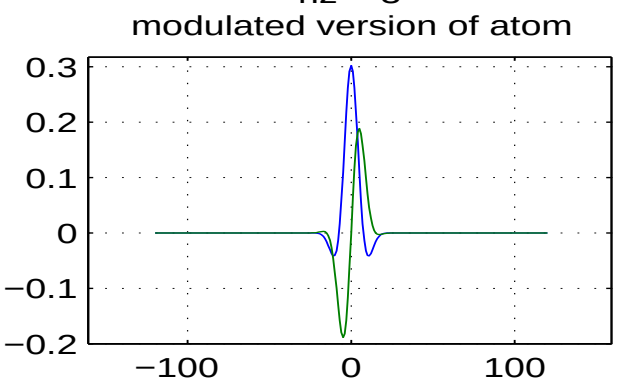


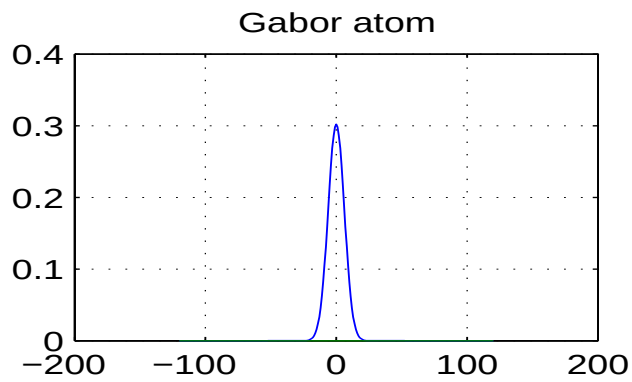
Das Spektrogramm einer Beethoven Sonate!

Zeit-Frequenz Verschiebungen

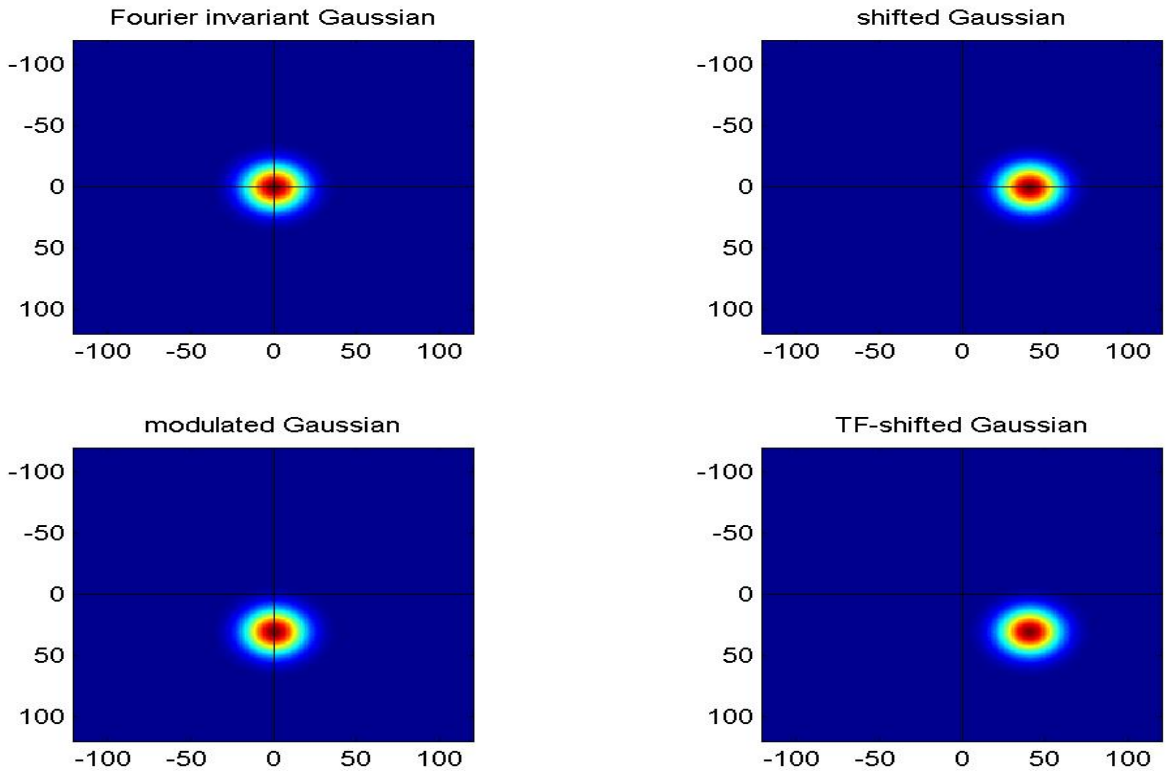




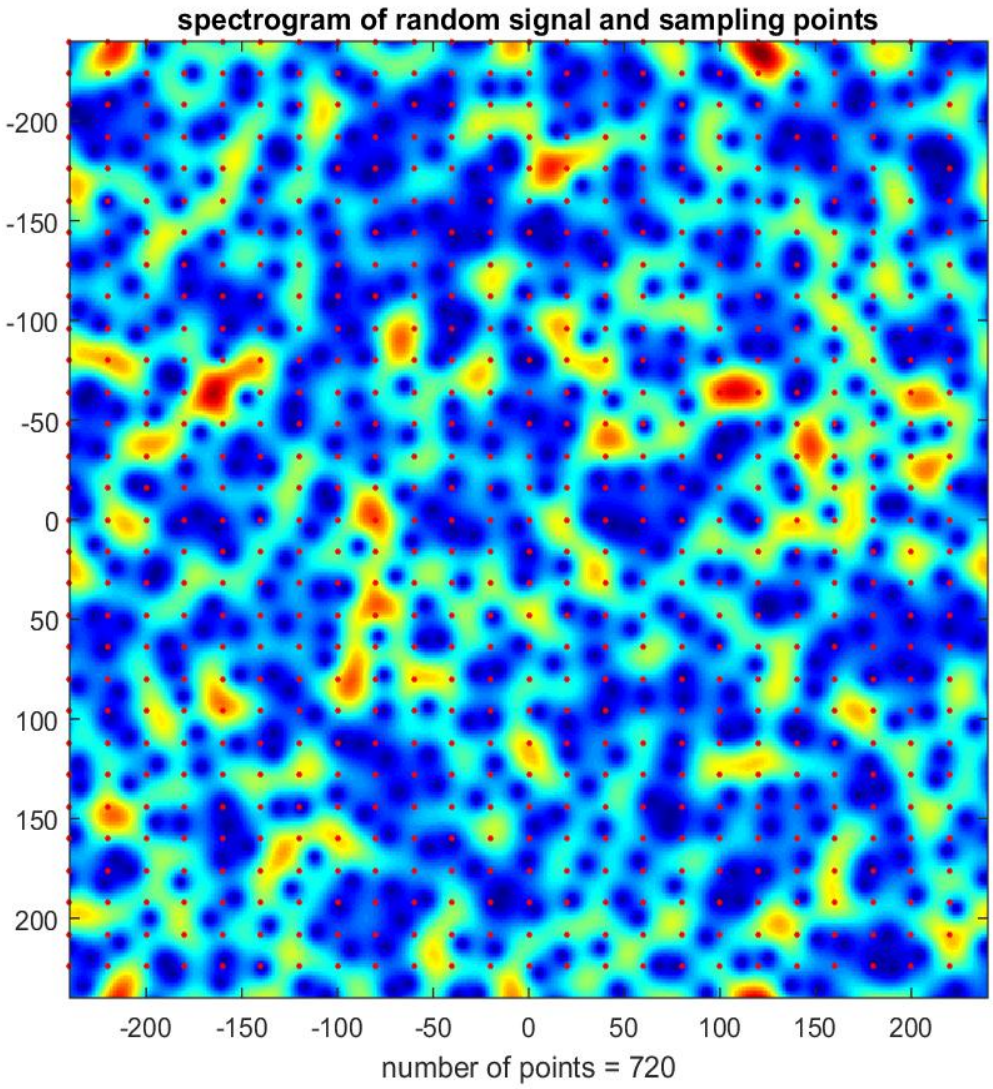




Zeit-Frequenz Verschiebungen im Spektrogramm

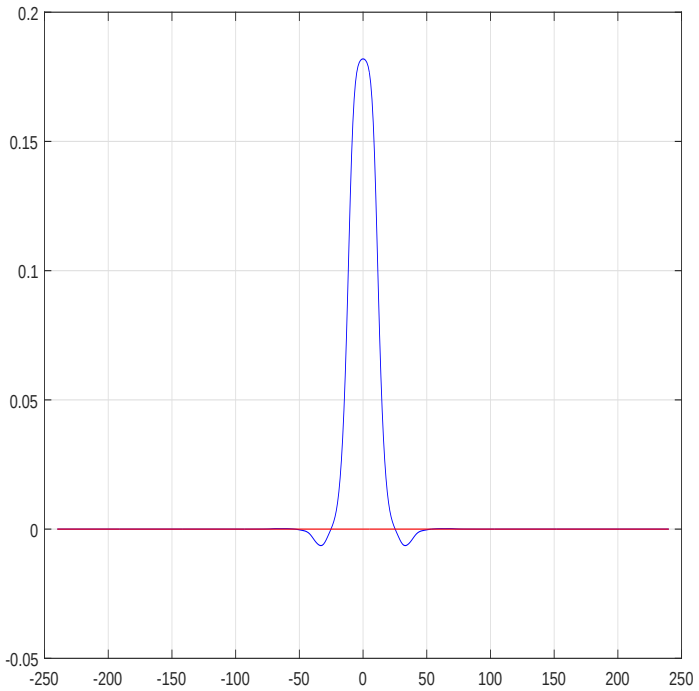


Spektrogramm und Sampling-Punkte



Spektrogramm und Gabor Reihen-Darstellung

Mathematisch können wir die Situation so beschreiben: Wählt man das Atom g passend, etwa so:



Tight Gabor Frames, Fortsetzung

Dann ist die *Gabor Familie* $(\pi(\lambda)g)_{\lambda\in\Lambda}$ ein "straffer Gabor frame", d.h. mit der Notation $g_\lambda := \pi(\lambda)g$ gilt:

$$f = \sum_{\lambda\in\Lambda} \langle f, g_\lambda\rangle g_\lambda,$$

und bis auf einen festen Faktor $C = C(\Lambda, g)$ gilt: die Länge des Koeffizienten-Vektors $(\langle f, g_\lambda\rangle)$ entspricht der Norm $\|f\|_2$ (in $\mathbb{C}^n$).

Solche "tight frames" sind natürlich nicht gleich einem Orthonormalsystem, weil sie linear abhängig sind (sie sind jedoch für viele Zwecke gleichwertig!).

Sog. *Gabor Multiplier* operieren (multiplikativ) auf den Gabor Koeffizienten, und sind die Basis des MP3-Verfahrens.

Praktische Hinweise

Wir werden eine Zusammenstellung der Routinen, die im Laufe des Vortrages sowie die verwendeten NuHAG M-files zur Verfügung stellen. Entsprechende Anfragen bitte an den Autor richten: `hans.feichtinger@univie.ac.at`

Sie sollen auch unter `www.nuhag.eu` im Internet bereitgestellt werden. HGFei