

## 1. Allgemeine Befehle

### 1.1. Standardbefehle

Befehl	Funktion
<code>save(filename, variable)</code>	speichert <i>variable</i> in matfile
<code>load(filename)</code>	lädt Variable aus matfile
<code>clear variable</code>	löscht <i>variable</i>
<code>clear all</code>	löscht alle Variablen im Workspace
<code>clc</code>	löscht Inhalt des Kommandofensters
<code>doc expression</code>	Hilfdatei zu <i>expression</i>
<code>help expression</code>	Kurzhilfe zu <i>expression</i>

### 1.2. Allgemeine Rechenoperationen

Befehl	Funktion
<code>mod(x,y)</code>	x modulo y (immer positiv)
<code>rem(x,y)</code>	x modulo y (vorzeichenabhängig)
<code>sqrt(x)</code>	$\sqrt{x}$
<code>exp(x)</code>	$e^x$
<code>log(x)</code>	Natürlicher Logarithmus $\ln(x)$
<code>floor(x)</code>	Abrunden auf Integer
<code>ceil(x)</code>	Aufrunden auf Integer
<code>sum(x)</code>	Summe über Werte des Vektors x
<code>prod(x)</code>	Produkt über Werte des Vektors x
<code>min(x)</code>	kleinster Wert des Vektors x
<code>max(x)</code>	größter Wert des Vektors x
<code>all(x)</code>	1 für keine 0 in Vektor x
<code>any(x)</code>	1 für eine Nicht-0 in Vektor x
<code>mean(x)</code>	Mittelwert des Vektors x
<code>sign(x)</code>	Vorzeichen von x (0 wenn x=0)
<code>abs(x)</code>	Betrag von x

### 1.3. Operatoren

Operator	Funktion
<code>a&gt;b</code>	Gibt logisch 1 zurück wenn $a > b$ , sonst 0
<code>a&gt;=b</code>	Gibt logisch 1 zurück wenn $a \geq b$ , sonst 0
<code>a==b</code>	Gibt logisch 1 zurück wenn $a = b$ , sonst 0
<code>a~=b</code>	Gibt logisch 1 zurück wenn $a \neq b$ , sonst 0
<code>a*b</code>	Matrix-Matrix-Multiplikation von a und b
<code>a.*b</code>	Elementenweise Multiplikation von a und b
<code>a(a&gt;=0)</code>	Gibt Teilvektor von a zurück, wo der Wert $\geq 0$

### 1.4. Komplexe Zahlen

Befehl	Funktion
<code>complex(a,b)</code>	$a + jb$
<code>real(z)</code>	Realteil von z
<code>imag(z)</code>	Imaginärteil von z
<code>abs(z)</code>	Betrag/Komplexe Amplitude von z
<code>angle(z)</code>	Phase von z
<code>conj(z)</code>	konjugiert komplex von z

### 1.5. Trigonometrische Funktionen

Befehl	Funktion
<code>sin(x)</code> , <code>cos(x)</code> , <code>tan(x)</code>	x in Bogenmaß
<code>sind(x)</code> , <code>cosd(x)</code> , <code>tand(x)</code>	x in Grad
<code>asin(x)</code> , <code>acos(x)</code> , <code>atan(x)</code>	Arcusfunktionen (Rad)
<code>asind(x)</code> , <code>acosd(x)</code> , <code>atans(x)</code>	Arcusfunktionen (Grad)

## 2. Matrizenrechnung

### 2.1. Rechenoperationen

Befehl	Funktion
<code>[a b c]</code>	Zeilenvektor (bzw. vertikales aneinanderreihen)
<code>[a; b; c]</code>	Spaltenvektor (bzw. horizontales aneinanderreihen)
<code>[a b c; d e f; g h i]</code>	3x3-Matrix
<code>[a b] = size(A)</code>	Dimensionen der Matrix (a Zeilen, b Spalten)
<code>inv(A)</code>	inverse Matrix von A
<code>A'</code>	$A^T$ (transponiert konjugiert komplex)
<code>A.'</code>	$A^T$ (transponiert)
<code>A \ b</code>	löst $A \cdot \vec{x} = \vec{b}$
<code>A(m,n)</code>	Element $A_{m,n}$
<code>A(m,:)</code>	m-te Zeile
<code>A(:,n)</code>	n-te Spalte
<code>A(Bedingung)</code>	Alle Element in A, auf die die Bedingung zutrifft
<code>find(A)</code>	lokalisiert Nicht-Null-Elemente (Indizes)
<code>det(A)</code>	Determinante von A
<code>rank(A)</code>	Rang (Anzahl unabhängiger von A)
<code>[V,D]= eig(A)</code>	Eigenwerte (D) und Eigenvektoren (V) von A
<code>(a:b:c)</code>	Vektor von a bis c mit Schrittweite b
<code>linspace(a,b,n)</code>	n Werte im gleichen Abstand von a bis b
<code>norm(x)</code>	eukl. Norm des Vektors x
<code>sum(A)</code>	Summer der Werte über Spaltenwerte
<code>sum(A,2)</code>	Summer der Werte über Zeilenwerte
<code>mean(A)</code>	Mittelwert der Spaltenwerte
<code>mean(A,2)</code>	Mittelwert der Zeilenwerte
<code>numel(A)</code>	Anzahl der Elemente in A

Komponentenweises Rechnen durch einen Punkt vor einem Operator  
Bsp:  $A.^2$  quadriert jedes Element der Matrix A  
Inlinefunktion:  $@(x)(f(x))$

### 2.2. Spezielle Matrizen

Befehl	Funktion
<code>eye(m,n)</code>	$m \times n$ Einheitsmatrix
<code>zeros(m,n)</code>	$m \times n$ 0-Matrix
<code>ones(m,n)</code>	$m \times n$ 1-Matrix
<code>diag(x)</code>	Diagonalmatrix mit den Werten von x
<code>rand(m,n)</code>	$m \times n$ Zufallsmatrix (Werte: 0-1)
<code>randi(imax,m,n)</code>	integer Zufallsmatrix mit max. imax
<code>magic(n)</code>	$n \times n$ magisches Quadrat

## 3. Schleiflab

while:	for:
<code>while expression</code>	<code>for i=0:1:20</code>
<code>statements</code>	<code>statements</code>
<code>end</code>	<code>end</code>

Schleife vorzeitig verlassen mit break

## 4. Plotten

Befehl	Funktion
<code>plot(x,y,'prop')</code>	Plottet x und y mit Farbe/Symbol 'prop'
<code>stem(y)</code>	Plottet diskrete y Werte (nicht verbunden)
<code>bar(x,y)</code>	Plottet x und y in einem Balkendiagramm
<code>xlim([a b])</code>	Begrenzt Bereich der x-Achse auf [a,b]
<code>ylim([c d])</code>	Begrenzt Bereich der y-Achse auf [c,d]
<code>axis([a b c d])</code>	Kombination aus xlim + ylim
<code>xlabel('Name')</code>	Benennt x-Achse
<code>ylabel('Name')</code>	Benennt y-Achse
<code>title('Titel')</code>	Tituliert den Plot
<code>subplot(H,B,Position)</code>	Selektiert Plot in Figure mit H x B Plots

Beispiel:

```
figure(1); % new figure
clf; % clear old figures
plot(x, y, 'k'); % plot y(x) in black 'k'
hold on; % more plots in same figure
plot(x, z, 'ro'); % plot z(x) in red circles
legend('y', 'z'); % names of plots
hold off;
```

## 5. Stochastische Zufallsvariablen

### 5.1. Realisierung von Standardmodellen

Befehl	$m \times n$ -Realisierung einer ...
<code>rand(m,n)</code>	gleichverteilte ZV (Werte: 0-1)
<code>randn(m,n)</code>	Standardnormalverteilung $\mathcal{N}(0, 1)$
<code>gamrnd(k,t,m,n)</code>	gammaverteile ZV, shape k, scale t
<code>binornd(n,p,m,n)</code>	Binomialverteilung mit Parameter n, p
<code>binornd(1,p,m,n)</code>	Bernoullierteilung mit Wahrscheinlichkeit p
<code>geornd(p,m,n)</code>	Geometrische Verteilung mit Wahrsch. p
<code>exprnd(1/lambda,m,n)</code>	Exponentialverteilung mit Parameter $\lambda$

Beispiele:

Befehl	Ergebnis
<code>2*rand+1</code>	gleichverteilte ZV im Bereich [1,3]
<code>plot(y,unifpdf((y-1)/2)/2)</code>	plottet PDF der oberen ZV
<code>plot(x,unifpdf(2*x)*2)</code>	PDF einer gv. ZV im Bereich [0,0.5]
<code>sigma*randn+mu</code>	Realisierung der Normalv. $\mathcal{N}(\mu, \sigma^2)$

### 5.2. Wahrscheinlichkeitsdichtefunktion (PDF) $f_X(x)$

Befehl	PDF an den Stellen x einer ...
<code>unifpdf(x,a,b)</code>	Gleichverteilung im Intervall [a,b]
<code>poisspdf(x,lambda)</code>	Poissonverteilung mit Parameter $\lambda$
<code>normpdf(x,mu,sigma)</code>	Normalverteilung mit Parameter mu und sigma
<code>gampdf(x,k,t)</code>	Gammaverteilung mit shape k und scale t

### 5.3. Kommulative Verteilungsfunktion (CDF) $F_X(x)$

Befehl	CDF an den Stellen x einer ...
<code>unifcdf(x,a,b)</code>	Gleichverteilung im Intervall [a,b]
<code>poisscdf(X,lambda)</code>	Poissonverteilung mit Parameter $\lambda$
<code>normcdf(X,mu,sigma)</code>	Normalverteilung mit Parameter $\mu$ und $\sigma$
<code>cdfplot(X)</code>	Schätzt und plottet CDF von X

## 6. Analyse von Zufallsvariablen

### 6.1. Schätzung von Paramtern einer Zufallsvariable X x ist ein Vektor von Realisierungen von X

Befehl	Funktion
<code>mean(x)</code>	Schätzt Erwartungswert $\mu$ der ZV X
<code>var(x)</code>	Schätzt Varianz $\text{Var}(X)$ der ZV X
<code>std(x)</code>	Schätzt die Standardabweichung $\sigma$ der ZV X
<code>length(x)</code>	Anzahl der Realisierungen der ZV X
<code>mean(x.^3)</code>	Schätzung des 3. Moments der ZV X

### 6.2. Histogramm

Befehl	Funktion
<code>hist(A)</code>	Teilt A in 10 gleiche Bereiche (Bins) und zählt die Elemente im jeweiligen Bin
<code>hist(A,b)</code>	Teilt A in b gleiche Bereiche (Bins) und zählt die Elemente im jeweiligen Bin
<code>hist(A,centers)</code>	Zählt die Elemente in den Bins um den Einträgen in centers
<code>[n,centers]=hist(...)</code>	Gibt in Anzahl der Elemente in n zurück und die Position der Bins in centers
<code>bar(hist(...))</code>	Erstellt ein Balkendiagramm des Histogramms

## 7. Signalverarbeitung

### 7.1. Audiosignalverarbeitung

Befehl	Funktion
<code>[a,b]=audioread('....')</code>	Liest Audiodatei in Vektor a ein (Abtastrate b)
<code>sound(x,b)</code>	Spielt das Signal in Vektor x (Abtastrate b) ab
<code>buffer(x,n)</code>	Teilt Signalvektor x in nichtüberlappende Frames der Länge n
<code>buffer(x,n,p)</code>	Teilt Signalvektor x in Frames der Länge n die sich um p überlappen

Beispiel:

```
buffer(1:12,5,3)
%liefert ...
ans =
    0    0    2    4    6    8
    0    1    3    5    7    9
    0    2    4    6    8    10
    1    3    5    7    9    11
    2    4    6    8    10    12
```

### 7.1.1 Lineare Systeme

Befehl	Funktion
<code>conv(x,y)</code>	Faltung zwischen Vektor x und y
<code>conv(x,y)</code>	gibt einen Vektor der Länge $\text{length}(x) + \text{length}(y) - 1$ zurück

## 8. Begriffe

i.i.d.  
Independent and identically distributed (unabhängig und gleichverteilt)

### Ergodisch

Eine Zufallsfolge heißt ergodisch, wenn Mittelwerte über die Zeit (für eine einzelne gegebene Realisierung der Folge) zum gleichen Ergebnis führen wie Erwartungswertbildung (für einen einzelnen Zeitpunkt).

### Klingonisch

Die klingonische Sprache ist die von Klingonen gesprochen Sprache. Sie ist im gesamten Klingonischen Imperium verbreitet.

## 9. Kapitel 1

### 9.1. Histogramm einer gleichverteilten ZV plotten

```
function x=hist_rand(N)
% Vektor x mit N Realisierungen von X (Gleichverteilung)
x=rand(N,1);
centers=0+1/40:1/20:1-1/40;
counts=hist(x,centers);
% Normierung von counts fuer eine PDF
counts = counts/N*20;
bar(centers,counts);
xlabel('x')
ylabel('h(x)')
title(sprintf('N = %01d',N));
```

## 10. Kapitel 2: Quantisierung und Transformation von ZV

### 10.1. Beispielaufgabe

Gegeben sei eine Zufallsvariable  $X$ , die im Intervall  $[0, 1]$  gleichverteilt ist und eine Zufallsvariable  $Y = g(X)$ , deren WDF mit der folgenden Funktion ausgewertet werden kann:

```
function f = mypdf(y)
f=unifpdf(exp(y)).*exp(y);
```

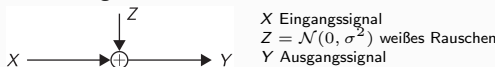
#### Folgerungen:

Die Funktion  $g = g(x) = \ln(x)$

1000 Realisierungen von  $Y$  erzeugen:  $y=\log(\text{rand}(1000,1));$

## 11. Kapitel 3: Bedingte Verteilung

### 11.1. Histogramm eines AWGN-Kanals



```
function hist_out(N,p,sigma)
binwidth=0.025;
centers=-2*binwidth/2:binwidth:3*binwidth/2;
figure
subplot(311)
```

```
% X 0 oder 1, Y = X + normalverteiltes Rauschen
x=binornd(1,p,N,1);
y=awgn_channel(x,sigma);
```

```
counts=hist(y,centers);
bar(centers,counts/(binwidth*sum(counts)),1);
ylim([0 2])
xlabel('y')
ylabel('h(y)')
title(sprintf('N=%01d',N))
```

```
subplot(312)
% Fall X ist immer 1
x=ones(N,1);
y=awgn_channel(x,sigma);
```

```
counts=hist(y,centers);
bar(centers,counts/(binwidth*sum(counts)),1);
ylim([0 2])
xlabel('y')
ylabel('h(y|x=1)')
```

```
subplot(313)
% Fall X ist immer 0
x=zeros(N,1);
y=awgn_channel(x,sigma);
```

```
counts=hist(y,centers);
bar(centers,counts/(binwidth*sum(counts)),1);
ylim([0 2])
xlabel('y')
ylabel('h(y|x=0)')
```

### 11.2. Maximum-Likelihood-Detektion

$$\max_{\hat{x} \in \{0,1\}} f_Y | X(y|\hat{x})$$

In unserem Fall:  $\hat{x} = \begin{cases} 0 & y \leq \frac{1}{2} \\ 1 & y > \frac{1}{2} \end{cases}$  Nachteil: Ignoriert das Wissen über die Eingangsverteilung.

```
function xhat = ml_detector(y)
xhat=(y>0.5);
```

**Nachteil:** Ignoriert Wissen über die Eingangsverteilung

### 11.3. Maximum-A-Posteriori-Detektion

$$\max_{\hat{x} \in \{0,1\}} p_X | Y(\hat{x}|y)$$

mit

$$p_X | Y(x|y) = \begin{cases} \frac{p_{f_Z}(y-1)}{f_Y(y)} & x = 1 \\ \frac{(1-p)f_Z(y)}{f_Y(y)} & x = 0 \\ 0 & \text{sonst} \end{cases} = \frac{f_Y | X(y|x)p_X(x)}{f_Y(y)}$$

```
function xhat = map_detector(y,p,sigma)
xhat=(p*normpdf(y-1,0,sigma)>(1-p)*normpdf(y,0,sigma));
```

**Nachteil:** Die Verteilung der Zufallsvariable am Eingang muss bekannt sein

**Spezialfall** der diskreten Gleichverteilung  $p_X(x) = \frac{1}{|\Omega_X|} \forall x \in \Omega_X$ :

$p_X$  für alle  $x$  gleich  $\rightarrow$  kann aus der Entscheidungsregel gestrichen werden. ML äquivalent zu MAP

## 12. Kapitel 4: Standardmodelle, Erwartungswert und Varianz

### 12.1. Modellierung für Mobilfunknetz

Überlagerung von Nutzern im Hotspot-Bereich (Normalverteilung)  $X_h, Y_h$  und anderen Nutzern (Gleichverteilung)  $X_u, Y_u$  mit der Bernoulli-verteilten ZV  $B$ .

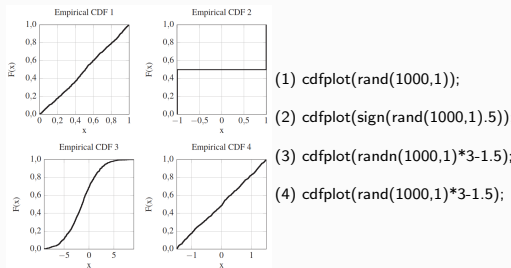
$$X_m = \begin{cases} X_h & \text{wenn } B = 1 \\ X_u & \text{wenn } B = 0 \end{cases} \quad Y_m = \begin{cases} Y_h & \text{wenn } B = 1 \\ Y_u & \text{wenn } B = 0 \end{cases}$$

```
function M=mixed_positions(N,L,muX,muY,sigma,q)
M=zeros(N,2); % Matrix initialisieren
```

```
H=hotspot_positions(N,L,muX,muY,sigma);
U=uniform_positions(N,L);
b=binornd(1,q,N,1);
```

```
M(b==1,:) = H(b==1,:);
M(b==0,:) = U(b==0,:);
```

### 12.2. Empirische Realisierung von CDFs



(1) `cdfplot(rand(1000,1));`

(2) `cdfplot(sign(rand(1000,1).5));`

(3) `cdfplot(randn(1000,1)*3-1.5);`

(4) `cdfplot(rand(1000,1)*3-1.5);`

## 13. Kapitel 5: Zufallsfolgen

### 13.1. Realisierung einer Zufallsfolge

$X_{n+1} = X_n + V_n$   $Y_{n+1} = Y_n + W_n$   
 $V_n$  und  $W_n$  sind i.i.d. Gleichverteilung mit Intervall  $[-\delta; \delta]$   
Folgender Code aktualisiert die Positionen  $(x_i, y_i)$  im Vektor `pos`

```
function pos=update_positions(pos,delta)
pos=pos+2*delta*(rand(size(pos))-0.5);
```

## 14. Kapitel 6: Zufallsfolgen und lineare System

### 14.1. Signalgenerierung

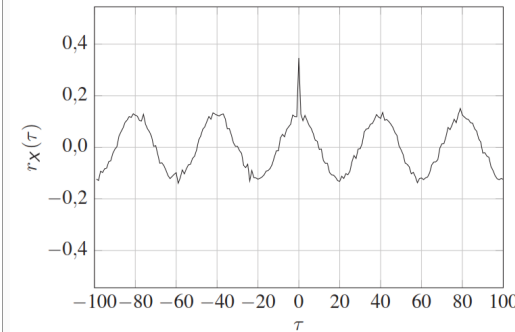
```
%Erstellt Sinussignal der Frequenz f1 mit der Amplitude A1,
%ueberlagert mit weissen Rauschen mit Parameter sigma
function x=create_signal(fS,T,f1,A1,sigma)
t=1:T*fS;
x=A1*sin(2*pi*f1/fS*t) + sigma*randn(1,fS*T);
```

### 14.2. Geschätzte Autokorrelationsfolge

Gegeben:

$$X_n = A_1 \sin(2\pi \frac{f}{f_s} n + \varphi_0) + Z_n$$

wobei  $\varphi_0$  im Intervall  $[0, 2\pi]$  stetig gleichverteilt ist.



Geschätzte Autokorrelationsfolge von  $X_n$

#### Folgerung

- Schätzung der Frequenz:  $f = \frac{f_s}{40}$
- Der Peak bei  $\tau = 0$  ist auf  $Z_n$  zurückzuführen
- $E[Z_k Z_l] = 0$  für  $k \neq l$