Lineare Algebra und Numerische Mathematik

Prof. R. Hiptmair

R. Hiptmair
SAM, ETHZ

Draft version 25. November 2013, SVN rev.

URL: http://www.sam.math.ethz.ch/~hiptmair/tmp/LANM13.pdf

Inhaltsverzeichnis

R. Hiptmair
SAM, ETHZ

R. Hiptmair
SAM, ETHZ

Allgemeine Informationen

- Link zur Webseite der Vorlesung im Herbstsemester 2013

- MATLAB-Codes sind verfügbar unter: http://www.sam.math.ethz.ch/~hiptmair/tmp/LANM_MATLAB/

- Vorlesungs-WIKI

Dieses Wiki dient hauptsächlich zur Meldung von Fehlern in den Vorlesungsunterlagen, in Übungsaufgaben oder Musterlösungen. Es ist öffentlich lesbar, schreibbar nur für registrierte Nutzer (Registieru Login mit NETHZ-Kennung und -Passwort).

1 Vektoren

1.1 Vektoren: Beispiele

1.1.1 Vektorpfeile in 2D und 3D

R. Hiptmair
SAM, ETHZ

1.1.2 Zeitdiskrete Signale

Example 1.1.1 (Endliches zeitdiskretes Audiosignal).

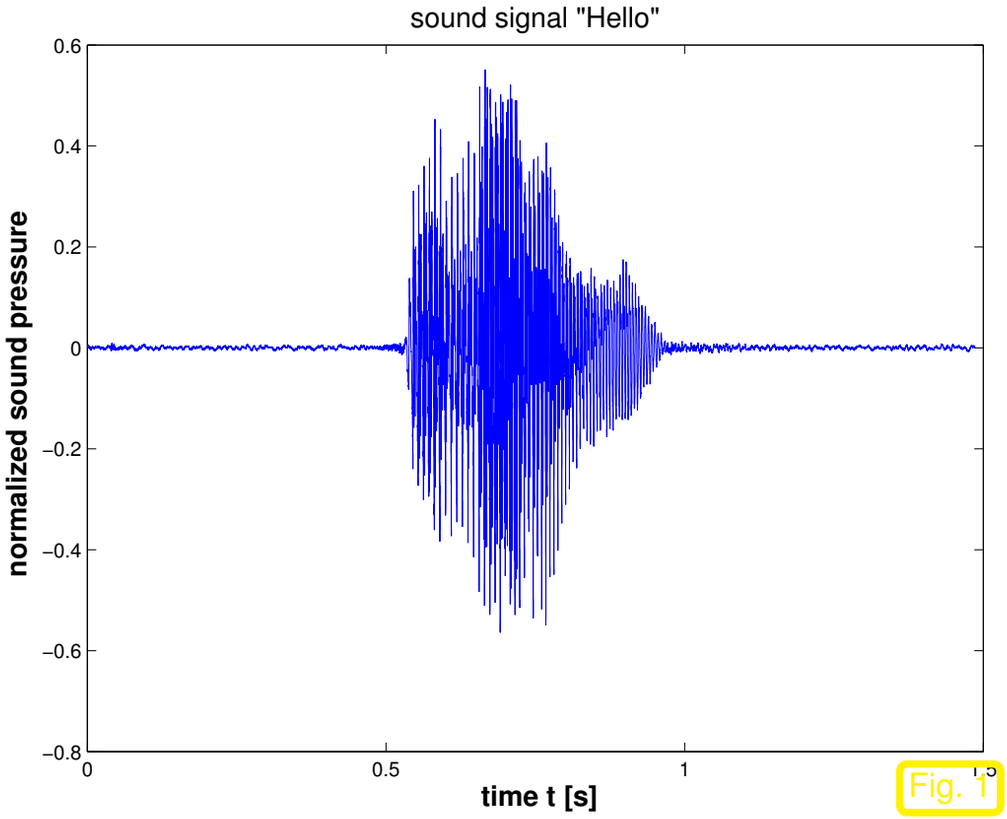


Fig. 1

Schalldruck wird in gleichen Zeitabständen abgetastet:

▶ `.wav`-Fileformat

◁ 44100 Abtastwerte pro Sekunde
(Abtastintervall $\Delta t = 2.2676 \cdot 10^{-5}\mathrm{s}$)

Code 1.1.2: Lesen und Abspielen von rohen Audiodaten in MATLAB

```
% MATLAB script: Frequency filtering of audio signal
% Read signal into column vector y. Also read sample rate, number of bits
% per sample
[y,Fs,nbits] = wavread('hello.wav');
% Next, play sound through the computer's audio output device
soundsc(y,Fs,nbits);
```

R. Hiptmair
SAM, ETHZ

Audiosignal mit $n \in \mathbb{N}$ Abtastwerten

$$\Updownarrow$$

vector $\mathbf{y} \in \mathbb{R}^n$: $\mathbf{y} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{pmatrix}$

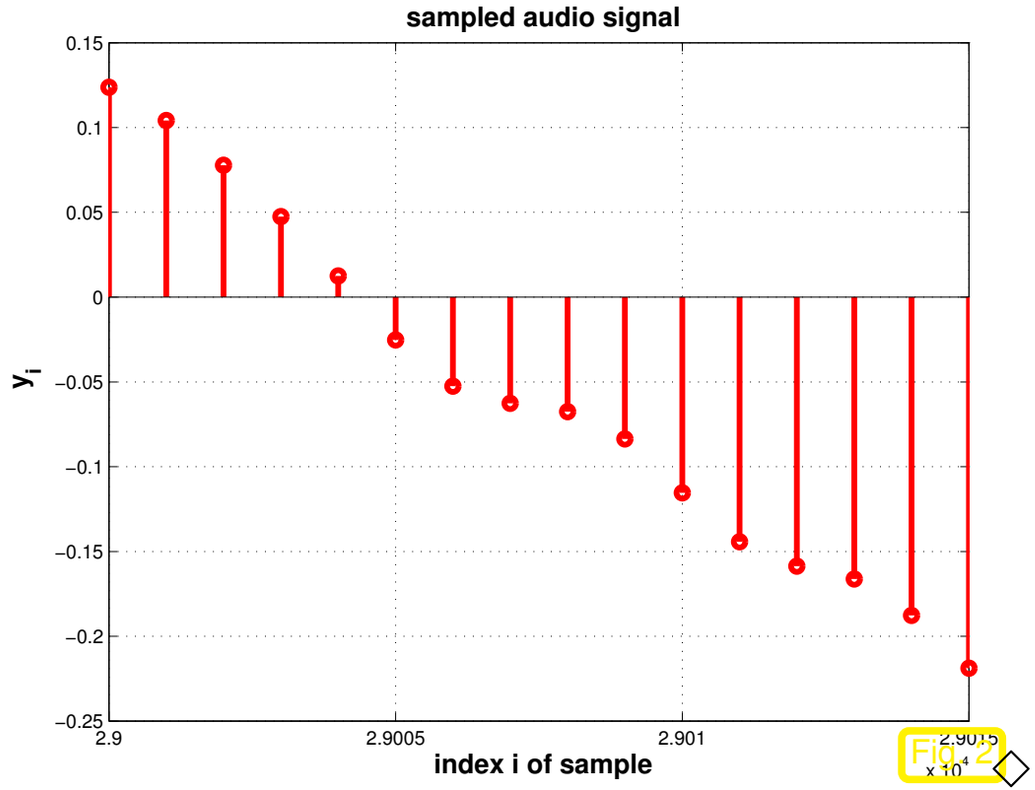


Fig. 2

R. Hiptmair
SAM, ETHZ

1.1.3 Spaltenvektoren aus $\mathbb{R}^n$

1.2 Vektorräume

1.3 Linearkombinationen und Unterräume

Es sei $(V, +, \cdot)$ ein Vektorraum

Def. 1.3.A. Gegeben seien Vektoren $\mathbf{v}^1, \ldots, \mathbf{v}^m \in V$ und Skalare $\alpha_1, \ldots \alpha_m \in \mathbb{R}$. Dann heisst die Summe

$$\sum_{j=1}^{m} \alpha_j \mathbf{v}^j = \alpha_1 \mathbf{v}^1 + \cdots + \alpha_m \mathbf{v}^m \in V$$

eine Linearkombination (LK) der Vektoren $\mathbf{v}^1, \ldots, \mathbf{v}^m$ mit Koeffizienten $\alpha_1, \ldots, \alpha_m$.

R. Hiptmair
SAM, ETHZ

Def. 1.3.B. Gegeben $\mathbf{v}^1, \ldots, \mathbf{v}^m \in V$ heisst die Menge

$$\mathrm{Span}\{\mathbf{v}^1, \ldots, \mathbf{v}^m\} \stackrel{\text{def}}{=} \{\sum_{j=1}^{m} \alpha_j \mathbf{v}^j,\ \alpha_1, \ldots, \alpha_j \in \mathbb{R}\} \subset V$$

aller möglicher Linearkombinationen von $\mathbf{v}^1, \ldots, \mathbf{v}^m$ der Span oder das Erzeugnis von $\mathbf{v}^1, \ldots, \mathbf{v}^m$.

Veranschaulichung: Geraden/Ebenen durch $\mathbf{0}$ im $\mathbb{R}^n$.

Def. 1.3.C. $U \subset V$ heisst Unterraum (UR), wenn gilt

- $\mathbf{v}, \mathbf{w} \in U \quad \Rightarrow \quad \mathbf{v} + \mathbf{w} \in U$,
- $\mathbf{v} \in U \quad \Rightarrow \quad \alpha \cdot \mathbf{v} \in U$ für jedes $\alpha \in \mathbb{R}$.

Satz 1.3.D. Für gegebene Vektoren $\mathbf{v}^1, \ldots, \mathbf{v}^m \in V$ ist $\mathrm{Span}\{\mathbf{v}^1, \ldots, \mathbf{v}^m\}$ ein Unterraum.

Die vielen Gesichter von '+':

- $\alpha + \beta$ für $\alpha, \beta \in \mathbb{R}$
- $\mathbf{v} + \mathbf{w}$ für Vektoren $\mathbf{v}, \mathbf{w} \in V$
- **Addition eines Vektors zu einer Menge $M \subset V$!**

$$\mathbf{v}\in V \quad , \quad M \subset V\colon \quad \mathbf{v} + M \stackrel{\text{def}}{=} \{\mathbf{x} \in V : \ \mathbf{x} = \mathbf{v} + \mathbf{m} \text{ für ein } \mathbf{m} \in M\} \subset V \ .$$

- **Addition zweier Mengen $U, W \subset V$!**

$$U, W \subset V\colon : \quad U + W \stackrel{\text{def}}{=} \{\mathbf{x} \in V : \ \mathbf{x} = \mathbf{u} + \mathbf{w} \text{ für irgendwelche } \mathbf{u} \in U, \ \mathbf{w} \in W\} \subset V \ .$$

R. Hiptmair
SAM, ETHZ

Def. 1.3.E. Eine Menge $A \subset V$ heisst affiner Raum, wenn es einen Vektor $\mathbf{v} \in V$ und einen Unterraum $U \subset V$ so gibt dass

$$A = \mathbf{v} + U \ .$$

Veranschaulichung im $\mathbb{R}^n$: Geraden/Ebenen in allgemeiner Lage

Satz 1.3.F. Seien $U, W \subset V$ zwei Unterräume von V. Dann sind auch die Teilmengen

$$U \cap V \qquad \text{und} \qquad U + W$$

Unterräume von V.

Veranschaulichung: Schnitt von Ebenen durch $\mathbf{0}$ ist eine Gerade durch $\mathbf{0}$.

1.4 Skalarprodukt

Def. 1.4.B. Das Euklidische Skalarprodukt zweier Vektoren im $\mathbf{v}.Vw \in \mathbb{R}^n$, $n \in \mathbb{N}$, ist definiert durch

$$\langle \mathbf{v}, \mathbf{w} \rangle \stackrel{\text{def}}{=} \sum_{j=1}^{n} v_j w_j \in \mathbb{R} \, .$$

Def. 1.4.B. Zwei Vektoren $\mathbf{v}, \mathbf{w} \in \mathbb{R}^n$ heissen orthogonal (senkrecht), wenn $\langle \mathbf{v}, \mathbf{w} \rangle = 0$.

Def. 1.4.D. Eine Verknüpfung $\langle \cdot, \cdot \rangle : V \times V \to \mathbb{R}$ auf einem Vektorraum $(V, +, \cdot)$ heisst Skalarprodukt, wenn sie erfüllt

$\langle \mathbf{v}, \mathbf{w} \rangle = \langle \mathbf{w}, \mathbf{v} \rangle$	für alle $\mathbf{v}, \mathbf{w} \in V$	(kommutativ)	(SP1)
$\langle \alpha \cdot \mathbf{v}, \mathbf{w} \rangle = \alpha \langle \mathbf{v}, \mathbf{w} \rangle$	für alle $\mathbf{v}, \mathbf{w} \in V,\ \alpha \in \mathbb{R}$	(distributiv)	(SP2)
$\langle \mathbf{v} + \mathbf{w}, \mathbf{u} \rangle = \langle \mathbf{v}, \mathbf{u} \rangle + \langle \mathbf{w}, \mathbf{u} \rangle$	für alle $\mathbf{v}, \mathbf{w}, \mathbf{u} \in V$	(distributiv)	(SP2)
$\langle \mathbf{v}, \mathbf{v} \rangle > 0$	für alle $\mathbf{v} \in V \setminus \{\mathbf{0}\}$	(positiv definit)	(SP4)

1.5 Längen und Winkel

R. Hiptmair
SAM, ETHZ

In diesem Abschnitt sei $(V, +, \cdot))$ ein Vektorraum, versehen mit einem Skalarprodukt $\langle \cdot, \cdot \rangle$.

Def. 1.5.A. Die Länge/Norm eines Vektors $\mathbf{v} \in V$ ist

$$\|\mathbf{v}\| := \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle}\ .$$

Eingenschaften:

$$\|\mathbf{v}+\mathbf{w}\| \leq \|\mathbf{v}\| + \|\mathbf{w}\| \quad \text{für alle } \mathbf{v},\mathbf{w} \in V \ ,$$
$$\|\alpha\mathbf{v}\| = |\alpha|\,\|\mathbf{v}\| \quad \text{für alle } \mathbf{v} \in V,\ \alpha \in \mathbb{R} \ .$$

Satz 1.3.C (Cauchy-Schwarz-Ungleichung)

$$\langle \mathbf{v},\mathbf{w}\rangle \leq \|\mathbf{v}\|\,\|\mathbf{w}\| \quad \text{für alle } \mathbf{v},\mathbf{w} \in V \ .$$

Def. 1.5.B. Der Winkel $\angle(\mathbf{v},\mathbf{w}) \in [0,\pi]$ zwischen Vektoren $\mathbf{v},\mathbf{w} \in V$ ist definiert durch

$$\cos\angle(\mathbf{v},\mathbf{w}) = \frac{\langle \mathbf{v},\mathbf{w}\rangle}{\|\mathbf{v}\|\,\|\mathbf{w}\|} \ .$$

R. Hiptmair
SAM, ETHZ

R. Hiptmair
SAM, ETHZ

1.6 Vektorprodukt

1.7 (Abstrakte) Analytische Geometrie

1.7.1 Entfernung Punkt-Gerade

1.7.2 Entfernung Punkt-Ebene

1.7.3 Dreiecksgeometrie

1.8 $\mathbb{R}^n$ auf dem Computer

1.8.1 Vektoren in MATLAB

R. Hiptmair
SAM, ETHZ

Code 1.8.1: Verlängerung von Spaltenvektoren in MATLAB

```
function fib = fibonaccivector(N)
% Storing the first N terms of the Fibonacci sequence in the components of
% a column vector.
% Example for successively increasing the size of column vectors

% Initialize the first two terms and take care of the special case N = 1
if (N==1), fib = [1]; else fib = [1;1]; end
for i=3:N % a loop in MATLAB; if N < 2, then the loop body is skipped
   fib = [fib;fib(i-1)+fib(i-2)]; % Grow vector
end
% Return value is stored in fib, because this is the return
% variable specified in the function header!
```

Code 1.8.2: (vecaccess.m) Zugriff auf Vektorkomponenten in MATLAB

```
% MATLAB script: accessing vector components
v = fibonaccivector(10),
% Note: line ending with a comma makes MATLAB print result

% Length of vector
N = length(v),

% Simple access by ()-operator
k = 7; vk = v(7),
% Subvector access by feeding integer vector or range into ()-operator
```

R. Hiptmair
SAM, ETHZ

```
subvec = v([1;7;8]), % access components 1, 7, and 8
evenfib = v(2:2:10), % access components from 2 to 10 with stride 2
lastfib = v(N-3:N), % access last four components
altlastfib = v(end-3:end), % another way the access the last four components
flipfib = v(end:-1:1), % revert order of vector components
```

Code 1.8.3: (specvec.m) Spezielle Vektoren in MATLAB

```
% Initializing special vectors in MATLAB
n = 5;
v = zeros(n,1), % A zero vector 0
v = ones(n,1), % A vector with components all 1
v = rand(n,1), % A vector with random entries even distributed betweeen 0 and 1.
```


1.8.2 Vektoroperationen in MATLAB

Code 1.8.4: (vecop.m) Vektoroperationen in MATLAB

```
% MATLAB script: vector operations in MATLAB
v = [1;2;3;4;5], % Initialize a column vector with five components
alpha = 0.5; alpha*v, % Scalar multiplication
w = [v(1:2:end);-v(2:2:end)], % Build another vector from v.
```

```
u = v + w;  % Addition of vectors
z = alpha*v + (1-alpha)*w,  % Scalar multiplication and vector addition

% Many MALAB functions can be applied to a vector and the result is the
% vector, whose components give the function evaluated for the components
% of the input vector
s = arrayfun(@(x) sin(x),v),  % Compute [sin(1);sin(2);...;sin(5)]
s = sin(v),  % the same for elementary mathematical functions
m = abs(w),  % take the modulus of each component

sp = dot(v,w),  % Scalar product
cp = cross([1;2;3],[4;5;6]);  % vector product = cross product
dot(cp,[1;2;3]), dot(cp,[4;5;6]),

% Componentwise operations, preceded by a dot:
p = v.*w,  % p(k) = v(k)*w(k)
p = v./w,  % p(k) = v(k)/w(k)
p = v.^w,  % p(k) = v(k)^w(k): taking the power (ger.: Potenz)

% Special functions that operate on vectors
mx = max(w)  % Maximum of components
mn = min(w)  % Minimum of components
```


R. Hiptmair
SAM, ETHZ

1.8.3 Visualisierung in MATLAB

Code 1.8.5: (visvecs.m) Visualisierung in MATLAB

```
% MATLAB script: demonstration plot() functions

% Plots in 2D
figure;
% Plot a point in 2D, marker is a red plus sign
x = 0.5; y = 0.7; plot(x,y,'r+');
axis([-1 1 -1 1]); % Set axis
hold on; % Use the same coordinate system for following plot commands
% Plot a number of points
vx = [1;2;3;4;5]/10; wy = vx.^2; plot(vx,wy,'m*');
% Plot a blue closed polygon
N = 10;
x = []; y = []; z = []; % Initialize empty vectors
for i=1:N
  x = [x;cos((i-1)/N*2*pi)]; y = [y;sin((i-1)/N*2*pi)]; z =
    [z;i/N];
end
X = 0.5*[x;x(1)]; Y = 0.5*[y;y(1)]; % Ensure that polygon is closed
plot(X,Y,'b-');

pause; % Wait for a key being pressed

```

R. Hiptmair
SAM, ETHZ

```
% Plot vector arrows
quiver(vx(1:end-1),wy(1:end-1),...
       vx(2:end)-vx(1:end-1),wy(2:end)-wy(1:end-1),'r');

pause; % Wait for a key being pressed

% Plots in 3D: a red spiral
figure; plot3(x,y,z,'r-*');
```

1.8.4 Rundungsfehler

Code 1.8.6: (roundoffvecs.m) Addition von Eckenvektoren eines regulären Polygons

```
function sv = roundoffvec(N)
% MATLAB function demonstrating the impact of roundoff by adding 2N vectors
% that should sum up to zero in exact arithmetic
N = 2*N;
x = []; y = []; sv = [0;0] % Initialize empty vectors
for i=1:N
  px = cos((i-1)/N*2*pi);
  py = sin((i-1)/N*2*pi);
  x = [x;px]; y = [y;py];
```

```
  sv = sv + [px;py];
end
figure; quiver(zeros(N,1),zeros(N,1),x,y,'r');
title('Corners of regular polygon','fontsize',14);
xlabel('{\bf x}','fontsize',14);
ylabel('{\bf y}','fontsize',14);
```

Code 1.8.7: (orthprojvec.m) Normalisierte Orthogonalprojektion

```
% MATLAB script: normalized orthogonal projection
v = [3;0.4]; w = [3;0.400001];     % the vectors
z = w - (dot(v,w)/dot(v,v))*v,% orthogonal projection of w onto v
z = z/norm(z),                     % normalize vector
dot(v,z), % Test for orthogonality
```

Code 1.8.8: Eingabefehler und Rundungsfehler

```
>> format long;
>> a = 4/3; b = a-1; c = 3*b; e = 1-c
e = 2.220446049250313e-16
>> a = 1012/113; b = a-9; c = 113*b; e = 5+c
e = 6.750155989720952e-14
>> a = 83810206/6789; b = a-12345; c = 6789*b; e = c-1
e = -1.607986632734537e-09
>> s = sin(10^16*pi)
s = -0.375212890012334
```


R. Hiptmair
SAM, ETHZ

Code 1.8.9: Nullstellenberechnung für eine Parabel

```
function z = zerosquadpol(alpha,beta)
% MATLAB function computing the zeros of a quadratic polynomial
% ξ → ξ^2 + αξ + β by means of the familiar discriminant
% formula ξ_{1,2} = 1/2(−α + √(α^2 − 4β)). However
% this implementation is vulnerable to round-off! The zeros are
% returned in a column vector
D = alpha^2-4*beta; % discriminant
if (D < 0), z = []; % No real zeros
else
    % The famous discriminant formula
    wD = sqrt(D);
    z = 0.5*[-alpha-wD;-alpha+wD];
end
```

Code 1.8.10: (compzeros.m) Nullstellenberechnung stabil und instabil

```
% MATLAB script for testing the computation of the zeros of a parabola
res = []; % array for storing results of computations.
gammavec = 2:10:1000; % Define test cases
for gamma=gammavec
    % Polynomial p(ξ) = (ξ − γ)(ξ − 1/γ)
    alpha = -(gamma + 1/gamma); % compute coefficients of polynomial
    beta = 1.0;
    z1 = zerosquadpol(alpha,beta); % Unstable way to compute zeros
    z2 = zerosquadpolstab(alpha,beta); % Stable implementation
```

R. Hiptmair
SAM, ETHZ

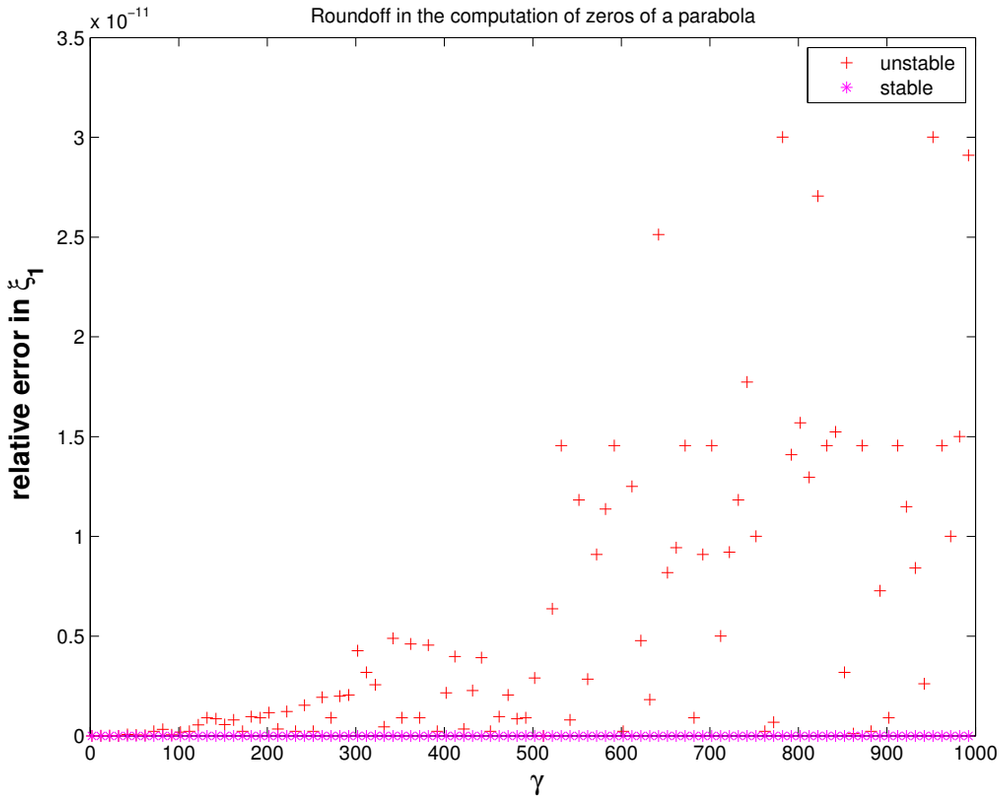
```
10    % Compute relative errors of the left zero
11    ztrue = 1/gamma;
12    res = [res; gamma, abs((z1(1)-ztrue)/ztrue),...
13        abs((z2(1)-ztrue)/ztrue)];
14 end
15 % Graphical output of relative errors
16 plot(res(:,1),res(:,2),'r+',res(:,1),res(:,3),'m*');
17 legend('unstable','stable');
18 xlabel('{\bf \gamma}','fontsize',14);
19 ylabel('{\bf relative error in \xi_1}','fontsize',14);
20 title('Roundoff in the computation of zeros of a parabola');
21 print -depsc2 '../../LANMFigures/zqperr.eps';
```

Code 1.8.11: Nullstellenberechnung für eine Parabel

R. Hiptmair
SAM, ETHZ

```
1  function z = zerosquadpolstab(alpha,beta)
2  % MATLAB function computing the zeros of a quadratic polynomial
3  % ξ → ξ^2 + αξ + β by means of the familiar discriminant
4  % formula ξ_{1,2} = (1/2)(−α + √(α^2 − 4β)) and
5  % Vieta's theorem. This is a stable implementation.
6  D = alpha^2-4*beta; % discriminant
7  if (D < 0), z = [];
8  else
9      wD = sqrt(D);
10     % Use discriminant formula only for zero far away from 0 in order to
11     % avoid cancellation (German: Auslöschung). For the other zeros
```

```
12      % use Vieta's formula.
13      if (alpha >= 0)
14          t = 0.5*(-alpha-wD);
15          z = [t;beta/t];
16      else
17          t = 0.5*(-alpha+wD);
18          z = [beta/t;t];
19      end
20  end
```



◁ Graphische Ausgabe von Code 1.8.9.

R. Hiptmair
SAM, ETHZ

2 Basen und Matrizen

2.1 Begriff der Basis

Es sei $(V, +, \cdot)$ ein Vektorraum.

Def. 2.1.A. Eine endliche Menge $\{\mathbf{b}^1, \ldots, \mathbf{b}^n\} \subset V$ heisst Basis von V, wenn es für jedes $\mathbf{v} \in V$ eindeutige Koeffizienten $v_i \in \mathbb{R}$ so gibt, dass

$$\mathbf{v} = \sum_{j=1}^{n} v_i \mathbf{b}^i \ .$$

Satz 2.1.B. Alle Basen eines Vektorraums haben die gleiche Anzahl von Elementen.

R. Hiptmair
SAM, ETHZ

Def. 2.1.C. Die Anzahl der Elemente einer Basis eines Vektorraums V heisst die Dimension von V, geschrieben $\dim V$.

2.2 Linearkombinationen und Matrizen

Link zu den handgeschriebenen Präsentationsfolien

2.2.1 Matrix-Vektor-Produkt

2.2.2 Matrizen in MATLAB

Initialisierung: Eckige Klammern $[\ldots]$

➢ Verallgemeinerung der Initialisierung von Vektoren, siehe Abschnitt 1.8.1.

R. Hiptmair
SAM, ETHZ

Code 2.2.1: (mathinit.m) Initialisierung von Matrizen in MATLAB

```
% Example code for initialization of (dense) matrices in MATLAB
% I: direct initialization of small matrices. Here ; is the row separator
% ("line feed"), the comma the column separator ("space"). In fact, the
% comma can be replaced with a space
A = [1,2,3;4,5,6;7,8,9],
A', % Transpose a matrix by the ' postfix operator!

% II: special matrices. Whenever two integer size arguments are given, the
% first is the number of rows, the second the number of columns
I = eye(3),       % identity matrix
Z = zeros(4,5),   % zero matrix
E = ones(4,3),    % matrix containing all ones
D = diag([1,2,3,4]); % diagonal matrix with prescribed diagonal
R = rand(2,5)  % matrix with i.i.d. random entries in [0,1]

% III: blockwise assembly of matrices, generalizing I
M = [0.5      , ones(1,4);...
     ones(4,1), D          ],

% IV: direct initialization of matrix entries, where the matrix size is
% increased automatically, which can be confusing and dangerous.
B(3,2) = 1; B(4,1) = -1; B,
% Gobbles up the memory:
```

R. Hiptmair
SAM, ETHZ

```
I = 9873; J = 3452; C(I,J) = 0;
whos C; % prints memory required by a variable
```


Zugriff auf Matrixeinträge: Runde Klammern $(\ldots)$

☞ Zwei Argumente: Zielen- und Spaltenindex

Wie bei Vektoren: Indexbereiche möglich

Code 2.2.2: (mataccess.m) Zugriff auf Matrixelemente und Untermatrizen

```
% Example code for accessing entries or parts of matrices
A = gallery('minij',5), % One of MATLAB's built in matrices
% acess operations
i = 2; j=3; A(i,j), % access individual entry
A([1,2],[3,4]), % bracket operator with ranges: sub-matrices
A(end-1:end,1:2),
A(:,3), A(2,:), % access to columns and rows
```

R. Hiptmair
SAM, ETHZ

Dadurch inspiriert:

✎ Notation: $\mathbf{A}(:,j) \hat{=} j$. Spalte der Matrix $\mathbf{A} \in \mathbb{R}^{n,m}$, $1 \leq j \leq m$

$\mathbf{A}(i,:) \hat{=} i$. Zeile der Matrix $\mathbf{A} \in \mathbb{R}^{n,m}$, $1 \leq i \leq n$

Matrix-Vektor-Produkt in MATLAB: $\star$-Operator: `v = A*c`

Rechenaufwand für das Matrix-Vektor-Produkt:

Def. 1.2.B. Der Rechenaufwand für eine Berechnung ist die Anzahl der dafür erforderlichen elementaren Operationen +,-,*,/ in $\mathbb{R}$.

$$\mathbf{A} \in \mathbb{R}^{n,m}, \mathbf{c} \in \mathbb{R}^m \quad \blacktriangleright \quad \text{Rechenaufwand}(\texttt{v = A*c}) = O(nm)$$

Zur "O-Notation" (Landau-Symbol für asymptotisches Verhalten einer Funktion):

Man schreibt für einen Term $T(n,m)$, $n,m \in \mathbb{N}$,

$$T(n,m) = O(nm) \quad \text{, falls} \quad \frac{T(n,m)}{nm} \sim \text{const} \quad \text{für grosse} \quad n,m \ .$$

Beispiele:

- $T(n) = 3n^3 - n^2$ erfüllt $T(n) = O(n^3)$ für grosse n

R. Hiptmair
SAM, ETHZ

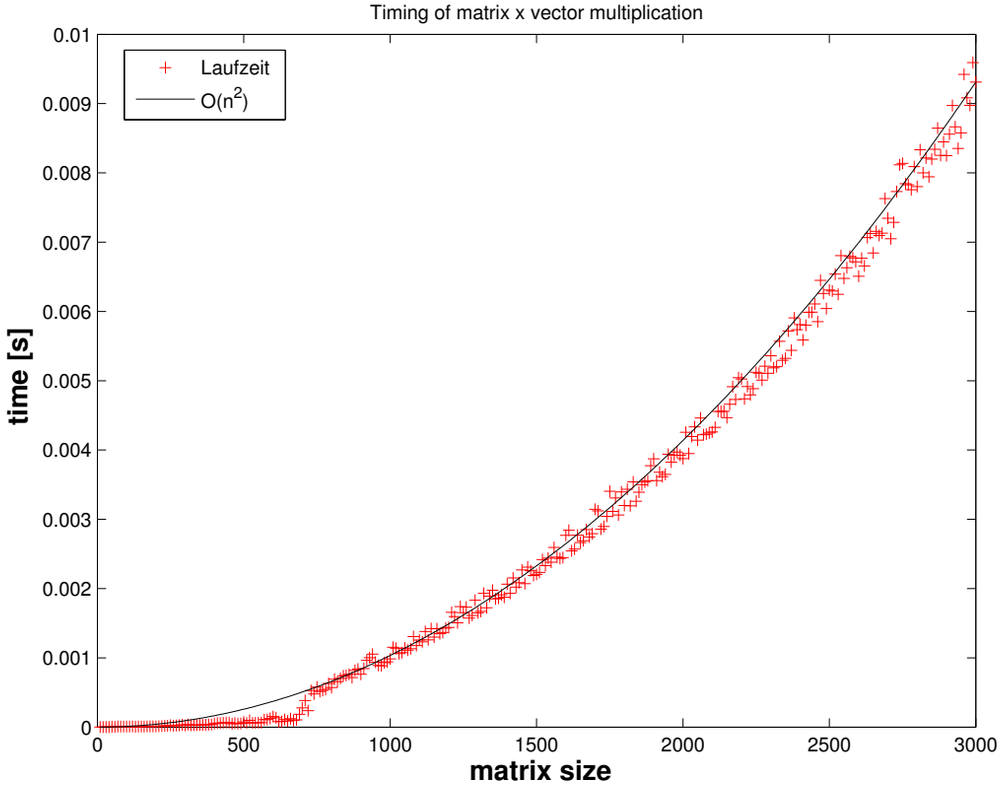
- $T(n,m) = \sqrt{n^2+m^2} - \log n$ erfüllt $T(n,m) = O(n+m)$ für grosse n, m.
- $T(n) = \log(\sqrt{n}) + \frac{\log(n^2)}{\sqrt{n}}$ erfüllt $T(n) = O(\log n)$ für grosse n

Grobe Korrelation von Rechenaufwand und Rechenzeit:

Code 2.2.3: (matvectiming.m) Rechenzeit für Matrix×Vektor

```
% Timing of matrix×vector operation for dense matrices
N = 3000; % maximum size of matrix
A = rand(N,N); % initialize random matrix
v = rand(N,1); % initialize random column vector

res = []; % matrix for collecting the results
for i=10:10:N
    B = A(1:i,1:i); % extract submatrix
    w = v(1:i);
    t = realmax;
    for j=1:3, tic; z = B*w; t = min(toc,t); end
    res = [res; i, t];
end

figure; plot(res(:,1),res(:,2),'r+',...
    res(:,1),res(:,1).^2/(res(end,1)^2)*res(end,2),'k-');
xlabel('{\bf matrix size}','fontsize',14);
```

```
ylabel('{\bf time [s]}','fontsize',14);
title('Timing of matrix x vector multiplication');
legend('Laufzeit','O(n^2)','location','best');

print -depsc2 '../../LANMFigures/matvectiming.eps';
```



◁ Rechenzeit für Matrix×Vektor-Multiplikation, Code 2.2.2

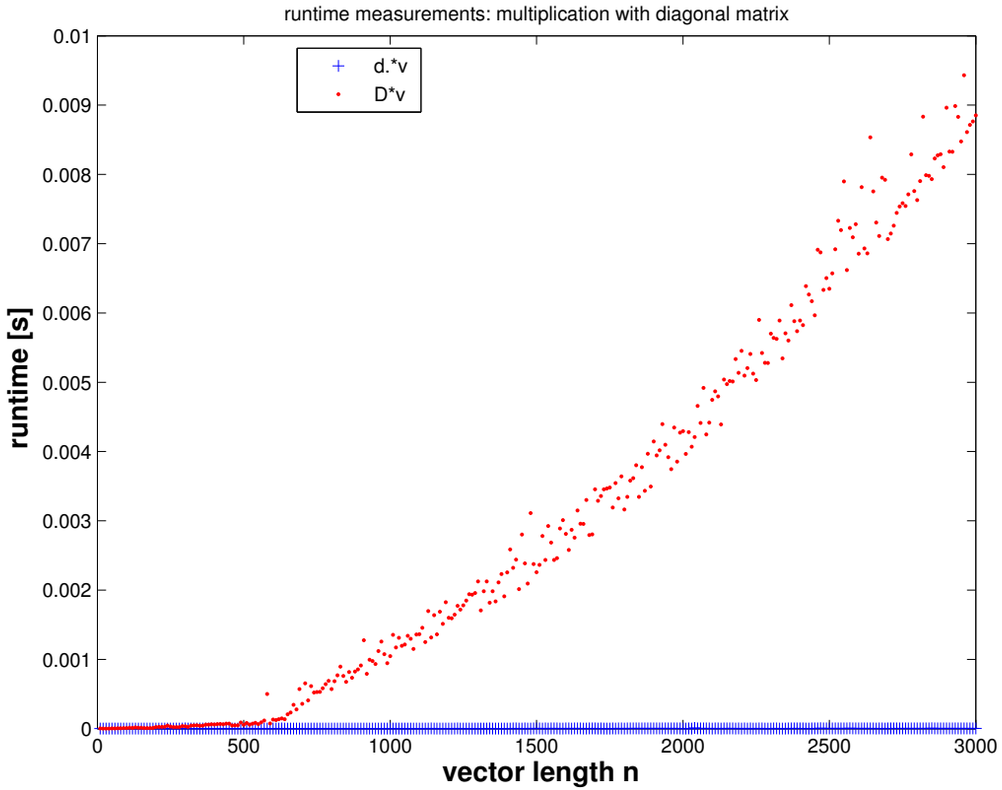
Mac OS X 10.6, MATLAB R2012a,
Intel Core i7, 2.66 GHz,

Spezielle Matrizen erlauben effiziente Implementierung von Matrix×Vektor:

Code 2.2.4: (diagmatvec.m) Effiziente Multiplikation mit Diagonalmatrix

```
% MATLAB script: runtime measurements for wasteful and efficient multiplication
```

```
% of a diagonal matrix with a vector.
res = []; % matrix for recording runtimes
for n=10:10:3000
   d = rand(n,1); v = rand(n,1); % Initialize random column vector
   D = diag(d);    % Build dense diagonal matrix
   t1 = realmax; for j=1:3, tic; z1 = d.*v; t1 = min(toc,t1); end
   t2 = realmax; for j=1:3, tic; z2 = D*v; t2 = min(toc,t2); end
   norm(z1-z2),
   res = [res; n,t1,t2];
end

% Graphical output of runtimes (see MATLAB documentation)
figure;
plot(res(:,1),res(:,2),'b+',res(:,1),res(:,3),'r.');
xlabel('{\bf vector length n}','fontsize',14);
ylabel('{\bf runtime [s]}','fontsize',14);
title('runtime measurements: multiplication with diagonal matrix');
legend('d.*v','D*v','location','best');

print -depsc2 '../../LANMFigures/diagmatvec.eps';
```



◁ Rechenzeiten: Multiplikation mit Diagonalmatrix, Code 2.2.3

Mac OS X 10.6, MATLAB R2012a,
Intel Core i7, 2.66 GHz,

Code 2.2.5: (onetriuatvec.m) Effiziente Multiplikation mit “1-Dreiecksmatrix”

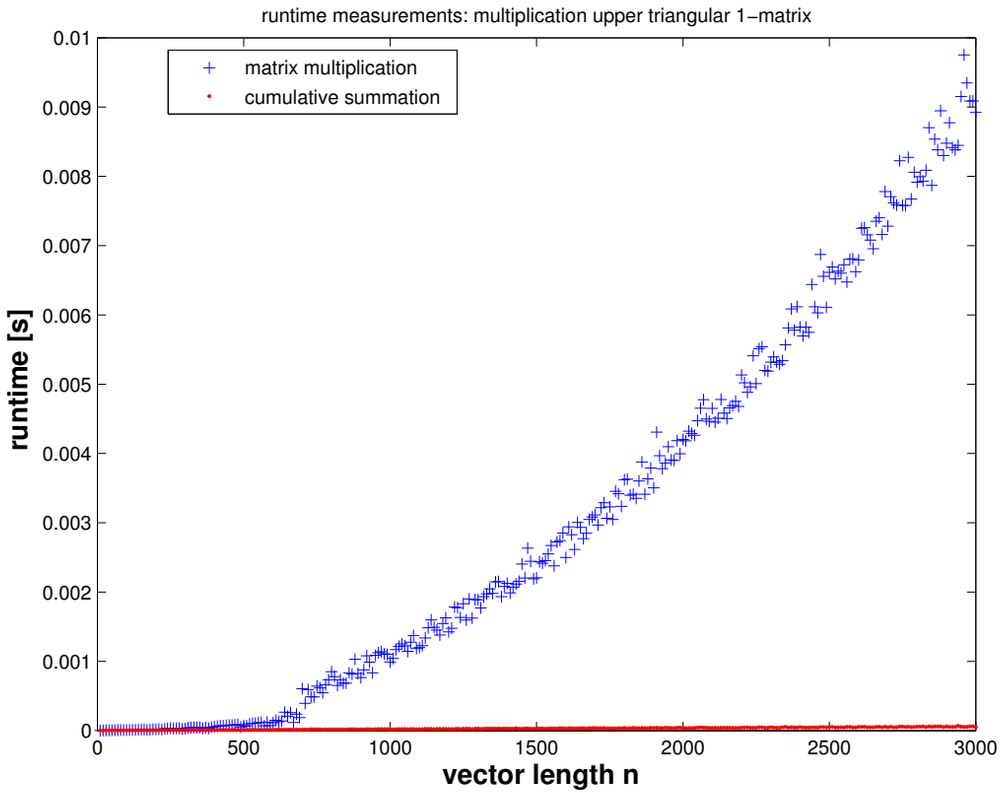
```
%MATLAB script: Wasteful and efficient matrix-vector multiplication with a
%matrix whose upper triangle is all 1: runtime measurements
N = 3000;
% Build dense matrix, triu extracts upper triangular part of a
% matrix (see MATLAB documentation for details)
T = triu(ones(N,N));
v = rand(N,1); % Initialize random vector

res = []; % Matrix for storing runtimes
for n=10:10:N
```

R. Hiptmair
SAM, ETHZ

```
    M = T(1:n,1:n); % obtain n×n submatrix by index range access
    w = v(1:n);     % isolate first n vector components
    % Direct multiplication with matrix, effort = O(n^2) !
    t1 = realmax; for j=1:3, tic; z1 = M*w; t1 = min(t1,toc); end
    % Plain summation of vector components, effort = O(n) !
    t2 = realmax;
    for j=1:3
        tic;
        z2 = zeros(n,1); z2(n) = w(n);
        for k=n-1:-1:1, z2(k) = z2(k+1)+w(k); end
        t2 = min(t2,toc);
    end
    norm(z1-z2), % Check whether the results are the same
    % Record runtimes in the matrix res
    res = [res; n,t1,t2];
end

% Graphical output of runtimes
figure;
plot(res(:,1),res(:,2),'b+',res(:,1),res(:,3),'r.');
xlabel('{\bf vector length n}','fontsize',14);
ylabel('{\bf runtime [s]}','fontsize',14);
title('runtime measurements: multiplication upper triangular
  1-matrix');
```


R. Hiptmair
SAM, ETHZ

```
4 legend('matrix multiplication','cumulative
    summation','location','best');
5 % Store plot in vector graphics format (EPS)
6 print -depsc2 '../../LANMFigures/onetriumatvec.eps';
```



◁ Rechenzeiten: Multiplikation mit spezieller Matrix, Code 2.2.4

Mac OS X 10.6, MATLAB R2012a,
Intel Core i7, 2.66 GHz,

2.3 Matrixprodukt

Link zu den handgeschriebenen Präsentationsfolien

Code 2.3.1: (matprod.m) Matrixprodukt

```
function matprod
% MATLAB demo: Various ways to implement to matrix product
A = rand(5,5); B = rand(5,5); % Initialize random matrices
C1 = mlmatprod(A,B);     % MATLAB matrix multiplication
C2 = loopmatprod(A,B);
C3 = dotmatprod(A,B);
norm(C1-C2), norm(C3-C2), % Check agreement of results

end

% Standard matrix product in MATLAB
function C = mlmatprod(A,B)
    C = A*B;
end

% Nested loop implementation of matrix multiplication
function C = loopmatprod(A,B)
  [n,k] = size(A); % A is an n × k-matrix
```

R. Hiptmair
SAM, ETHZ

```
  [kb,m] = size(B); % B is an k×m-matrix
  if (k ~= kb), error('Mismatch of matrix dimensions'); end
  C = zeros(n,m);
  for i=1:n
    for j=1:m
      C(i,j) = 0;
      for l=1:k
        C(i,j) = C(i,j)+ A(i,l)*B(l,j);
      end
    end
  endw
end

% Direct computation of entries of C by forming scalar products of columns
% of B and rows of A.
function C = dotmatprod(A,B)
  [n,k] = size(A); % A is an n×k-matrix
  [kb,m] = size(B); % B is an k×m-matrix
  if (k ~= kb), error('Mismatch of matrix dimensions'); end
  C = zeros(n,m); % Mainly allocate memory for matrix
  for i=1:n
    for j=1:m
      % Note: access to rows and columns through range operator :
      C(i,j) = dot(A(i,:),B(:,j));
```


R. Hiptmair
SAM, ETHZ

```
    end
  end
end
```


$\mathbf{A} \in \mathbb{R}^{n,m}, \mathbf{S} \in \mathbb{R}^{m,k}$ ⟹ Rechenaufwand(A*B) = $O(mnk)$

Speziell: $\mathbf{A}, \mathbf{S} \in \mathbb{R}^{n,n}$ ⟹ Rechenaufwand(A*B) = $O(n^3)$

Code 2.3.2: (matprodtiming.m) Verschieden effiziente Implementierungen des Matrixprodukts

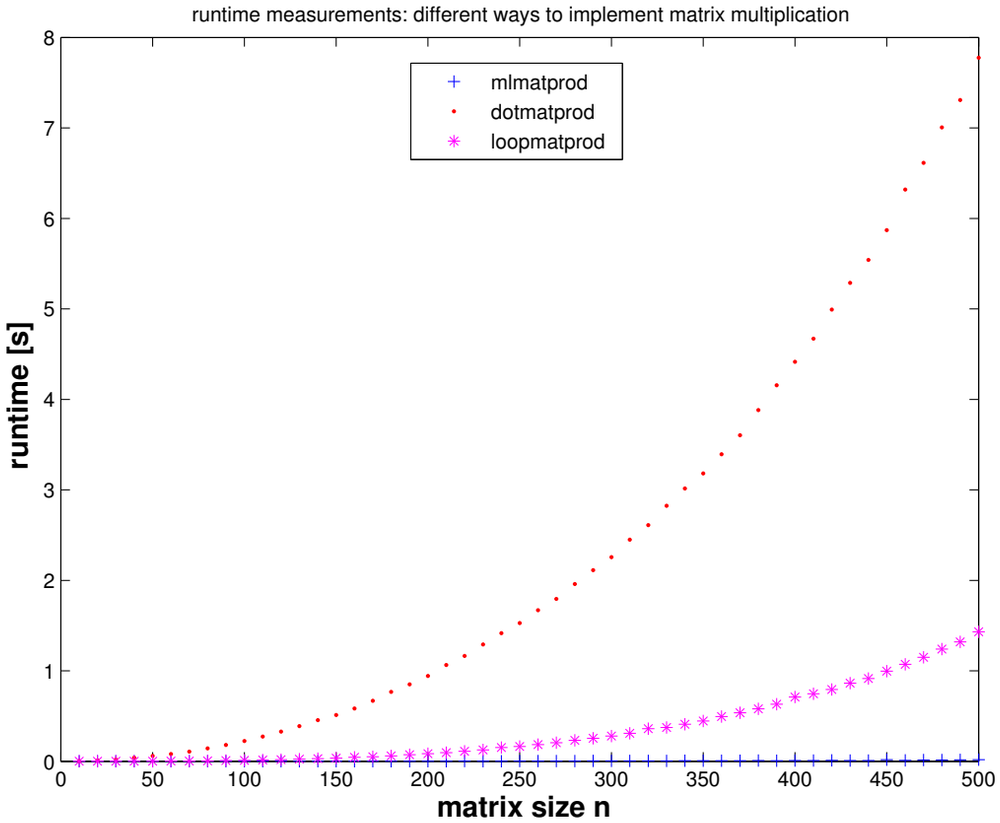
```
% Measurement of runtimes for different implementations of matrix products
N = 500;
A = rand(N,N); B = rand(N,N); % Initialize random matrices

res = [];
for n=10:10:N
    An = A(1:n,1:n); Bn = B(1:n,1:n);
    t1 = realmax; for j=1:3, tic; C1 = mlmatprod(An,Bn); t1 =
      min(toc,t1); end
    t2 = realmax; for j=1:3, tic; C2 = dotmatprod(An,Bn); t2 =
      min(toc,t2); end
    t3 = realmax; for j=1:3, tic; C3 = loopmatprod(An,Bn); t3 =
      min(toc,t3); end
    fprintf('%d: %f, %f\n',n,norm(C1-C2), norm(C1-C3));
```

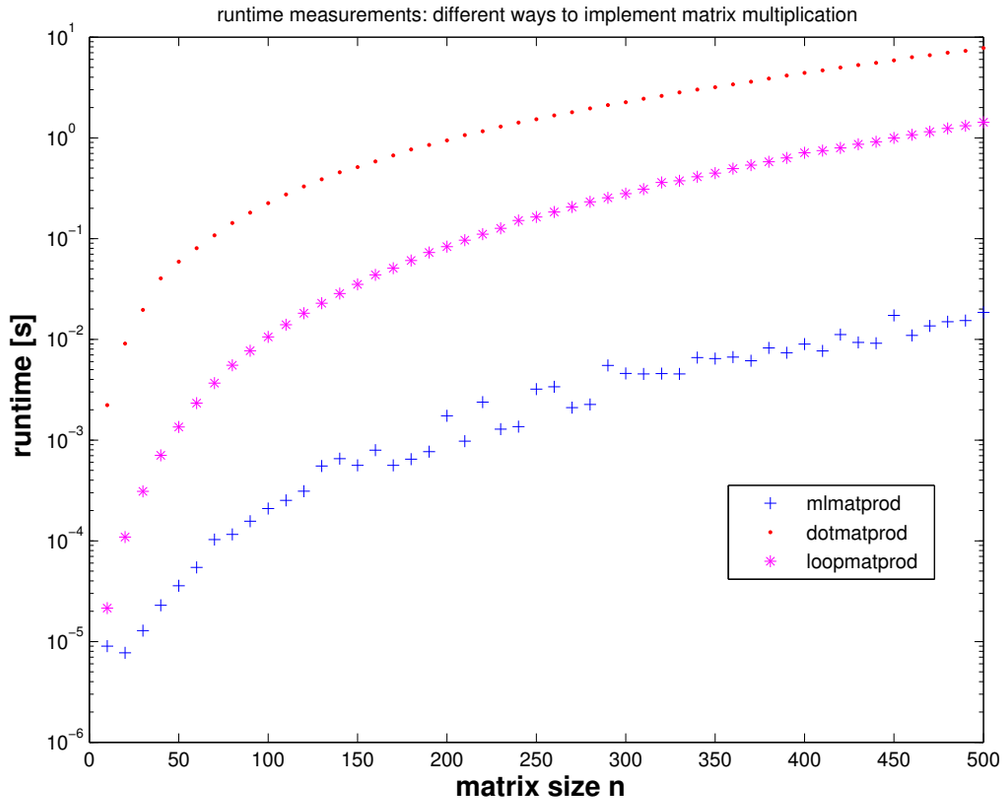
R. Hiptmair
SAM, ETHZ

```
    res = [res; n, t1, t2, t3];
end

figure;
plot(res(:,1),res(:,2),'b+',res(:,1),res(:,3),'r.',res(:,1),res(:,4)
xlabel('{\bf matrix size n}','fontsize',14);
ylabel('{\bf runtime [s]}','fontsize',14);
title('runtime measurements: different ways to implement matrix
  multiplication');
legend('mlmatprod','dotmatprod','loopmatprod','location','best');

print -depsc2 '../../LANMFigures/matprodtiming.eps';

figure;
semilogy(res(:,1),res(:,2),'b+',res(:,1),res(:,3),'r.',res(:,1),res(
xlabel('{\bf matrix size n}','fontsize',14);
ylabel('{\bf runtime [s]}','fontsize',14);
title('runtime measurements: different ways to implement matrix
  multiplication');
legend('mlmatprod','dotmatprod','loopmatprod','location','best');

print -depsc2 '../../LANMFigures/matprodtiminglog.eps';
```


R. Hiptmair
SAM, ETHZ

Rechenzeiten: Verschieden effiziente Implementierungen des Matrixprodukts (Code 2.3.1, Mac OS X 10.6, MATLAB R2012a, Intel Core i7, 2.66 GHz)





2.4 Dünnbesetzte Matrizen

Link zu den handgeschriebenen Präsentationsfolien

R. Hiptmair
SAM, ETHZ

Code 2.4.1: Initialisierung von dünnbesetzten Matrizen

```
function T = inittridiag(diagonal,superd,subd)
% MATLAB function for the initialization of a sparse tridiagonal matrix. The
% argument diagonal has to pass a vector of lenth N, which is used to
% initialize the diagonal of the matrix, whereas the N-1-vectors
% superd and subd contain the entries for super- and
% sub-diagonal of the matrix.
N = length(diagonal); % we aim for an N x N tridiagonal matrix
if ((length(superd) ~= N-1) || (length(subd) ~= N-1))
  error('length mismatch for offdiagonals');
end
I = [1:N,1:N-1,2:N]; % row indices for non-zero matrix entries
J = [1:N,2:N,1:N-1]; % column indices for non-zero matrix entries
a = [diagonal;superd;subd]; % Value vector for matrix entries
% Initialization of sparse N x N-matrix from tupels (i,j,a_{i,j}).
T = sparse(I,J,a,N,N);
```

Code 2.4.2: Zu `sparse` algebraisch äquivalente Funktion

```
function A = mysparse(I,J,a,n,m)
% Funktion constructing an n x m-matrix from two index vectors I,
% J and one entry vector a of the same length. The output
% matrix is the same as for the MATLAB built-in function sparse.
% However, it is stored as a dense matrix and not in sparse matrix
```

R. Hiptmair
SAM, ETHZ

```
% data format. Hence, this function must not be used as a
% substitute for MATLAB's sparse!
k = length(I); % number of index-entry tupels
if ((length(J) ~= k) || (length(a) ~= k)), error('Length
  mismatch'); end
% Set entries; if an index pair occurs twice, the corresponding entry
% values are added!
A = zeros(n,m); % Initialize a dense zero matrix.
for l=1:k
   A(I(l),J(l)) =  A(I(l),J(l)) + a(l);
end
```


Code 2.4.4: Diese Funktion liesse sich auch durch ein **einziges** MATLAB-Statement ersetzen. Welches?

```
function y = sparseact(I,J,a,n,x)
% A MATLAB function doing something interesting
% First check arguments for consistency
N = length(I);
if ((length(J) ~= N) || (length(a) ~= N))
  error('size mismatch');
end
m = length(x);
if (max(I) > n), error('size mismatch'); end
```

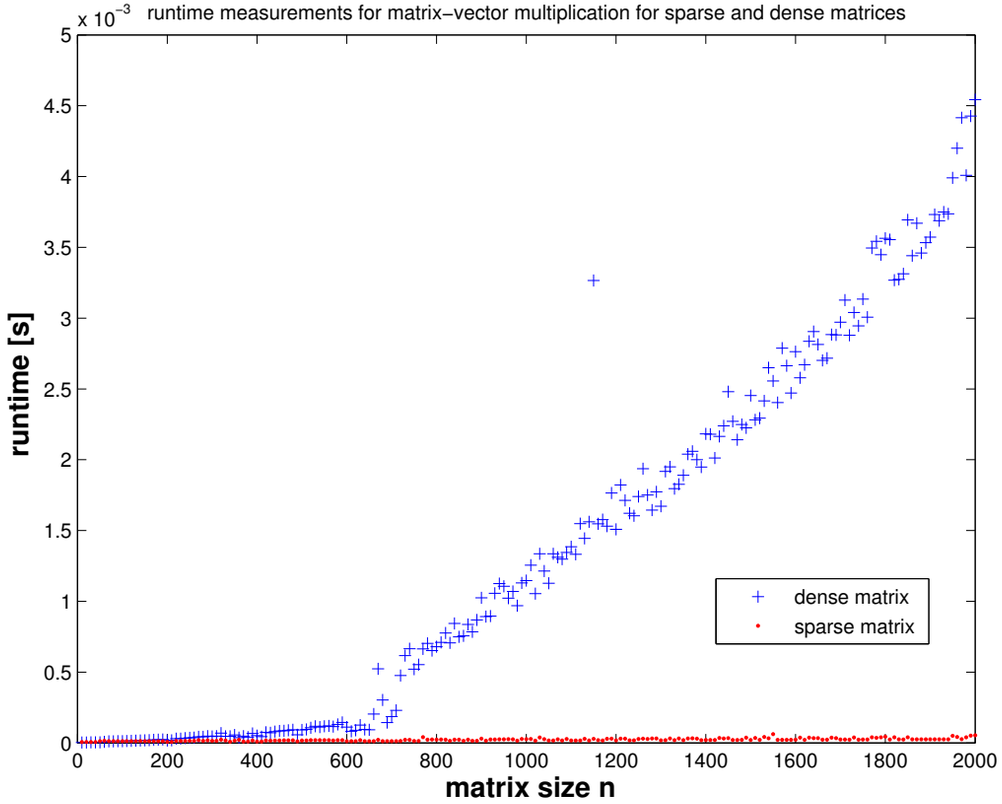
```
if (max(J) > m), error('size mismatch'); end

% The real thing
y = zeros(n,1);
for k=1:N, y(I(k)) = y(I(k)) + a(k)*x(J(k)); end
```

Code 2.4.5: Matrix×Vektor-Multiplikation mit dünnbesetzten Matrizen

```
function sparsemvtiming
% Measurement of runtimes for matrix×vector multiplication with
% a sparse matrix.
N = 2000;                                % Maximal size of matrix
% Initialize random column vectors of length N and N-1, respectively.
d = rand(N,1); dl = rand(N-1,1); du = rand(N-1,1); v = rand(N,1);
% Initialize dense triadiagonal matrix, see the documntation of the MATLAB
  command
% diag for details.
T_dense = diag(d)+diag(dl,-1)+diag(du,1);
% Initialize sparse triadiagonal matrix
T_sparse = inittridiag(d,du,dl);

res = []; % matrix for recording times
% conduct timings for vectors of different size n
for n=10:10:N
  % Extract sub-matrices, which will be sparse and dense matrices again
  Td = T_dense(1:n,1:n); Ts = T_sparse(1:n,1:n);
```


R. Hiptmair
SAM, ETHZ

```
  t1 = realmax; for j=1:3, tic; w1 = Td*v(1:n); t1 = min(toc,t1);
    end
  t2 = realmax; for j=1:3, tic; w2 = Ts*v(1:n); t2 = min(toc,t2);
    end
  norm(w1-w2), % Check for agreement of results
  res = [res; n, t1, t2];
end

% Create plots of runtimes.
figure;
plot(res(:,1),res(:,2),'b+',res(:,1),res(:,3),'r.');
xlabel('{\bf matrix size n}','fontsize',14);
ylabel('{\bf runtime [s]}','fontsize',14);
title(['runtime measurements for matrix-vector multiplication for
  sparse and ' ...
      'dense matrices']);
legend('dense matrix','sparse matrix','location','best');

print -depsc2 '../../LANMFigures/sparsemvtiming.eps';
end
```



◁ Rechenzeiten gemessen mit Code 2.4.4

Mac OS X 10.6, MATLAB R2012a,
Intel Core i7, 2.66 GHz,

R. Hiptmair
SAM, ETHZ

2.5 Lineare (Un-)Abhängigkeit

$(V, +, \cdot)$ Vektorraum.

Def. 2.5.A. Eine endliche Menge von Vektoren $\{\mathbf{v}^1, \ldots, \mathbf{v}^m\} \subset V$ heisst linear unabhängig (l.u.) wenn für Koeffizienten $c_1, \ldots, c_m \in \mathbb{R}$ aus

$$\sum_{j=1}^{m} c_j \mathbf{v}^j = 0 \quad \text{folgt} \quad c_1 = \cdots = c_m = 0 \, .$$

Andernfalls heisst sie linear abhängig (l.a.).

$\{\mathbf{v}^1, \ldots, \mathbf{v}^m\}$ linear unabhängig $\Leftrightarrow$ $\mathrm{Span}\left(\{\mathbf{v}^1, \ldots, \mathbf{v}^m\} \setminus \{\mathbf{v}^j\}\right) \neq \mathrm{Span}\{\mathbf{v}^1, \ldots, \mathbf{v}^m\}$
für jedes $j \in \{1, \ldots, m\}$.

$\Leftrightarrow$ $\mathbf{v}^j \notin \mathrm{Span}\left(\{\mathbf{v}^1, \ldots, \mathbf{v}^m\} \setminus \{\mathbf{v}^j\}\right)$
für jedes $j \in \{1, \ldots, m\}$.
(keiner der Vektoren $\mathbf{v}^j$ liegt in einer durch die übrigen aufgespannten Ebene)

$\{\mathbf{v}^1, \ldots, \mathbf{v}^m\}$ linear abhängig $\Leftrightarrow$ Es gibt mindestens ein $j \in \{1, \ldots, m\}$ so, dass
$\mathbf{v}^j \in \mathrm{Span}\left(\{\mathbf{v}^1, \ldots, \mathbf{v}^m\} \setminus \{\mathbf{v}^j\}\right)$.

Satz 2.5.B. Die Vektoren einer Basis von V sind linear unabhängig.

R. Hiptmair
SAM, ETHZ

Beispiel 2.5.C.: Die Polynome $\{1, x, x^2, \ldots, x^d\}$ sind linear unabhängig im Vektorraum $\mathbb{P}_d$ der Polynome vom Grad $\leq d$, $d \in \mathbb{N}$.

Beispiel 2.5.D.: Ist $\langle \cdot, \cdot \rangle$ ein Skalarprodukt auf V, und gilt für eine Menge von Vektoren $\{\mathbf{v}^1, \ldots, \mathbf{v}^m\} \subset V$

$$\mathbf{v}^j \neq 0 \quad \text{für alle } j \in \{1, \ldots, m\} ,$$
$$\left\langle \mathbf{v}^i, \mathbf{v}^j \right\rangle = 0 \quad \text{wenn } i \neq j \quad \text{(paarweise orthogonal)} ,$$

dann ist $\{\mathbf{v}^1, \ldots, \mathbf{v}^m\}$ linear unabhängig.

Satz 2.5.E. Ist $\{\mathbf{v}^1, \ldots, \mathbf{v}^m\} \subset V$ linear unabhängig, dann gilt dies auch fuer die Mengen

(i) $\{\mathbf{w}^1, \ldots, \mathbf{w}^m\}$ mit

$$\mathbf{w}^j := \begin{cases} \mathbf{v}^j & \text{für } j \in \{1, \ldots, m\} \setminus \{k\} , \\ \mathbf{v}^k + \mathbf{v}^l & \text{für ein } l \in \{1, \ldots, m\} \setminus \{k\} , \end{cases} \qquad k \in \{1, \ldots, m\} .$$

(ii) $\{\mathbf{w}^1, \ldots, \mathbf{w}^m\}$ mit

$$\mathbf{w}^j =: \begin{cases} \mathbf{v}^j & \text{für } j \in \{1, \ldots, m\} \setminus \{k\} , \\ \alpha \mathbf{v}^k & \text{für beliebiges } \alpha \in \mathbb{R}, \alpha \neq 0 . \end{cases}$$

2.6 Basiswechsel

R. Hiptmair
SAM, ETHZ

3 Lineare Gleichungssysteme

3.1 Grundlagen

Def. 3.1.A. Gegeben sei eine Koeffizientenmatrix $\mathbf{A} \in \mathbb{R}^{n,m}$ und ein Rechte-Seite-Vektor $\mathbf{b} \in \mathbb{R}^n$. Dann heisst die Gleichung $\mathbf{Ax} = \mathbf{b}$ eine lineares Gleichungssystem (LGS) mit Unbekannten x_j, $j = 1, \ldots, n$.

Lösungsmenge des LGS $\mathbf{Ax} = \mathbf{b}$: $\{\mathbf{x} \in \mathbb{R}^n : \mathbf{Ax} = \mathbf{b}\}$

Satz 3.1.B. Es sei $\mathbf{A} \in \mathbb{R}^{n,m}$, $\mathbf{b} \in \mathbb{R}^n$. Dann gilt:

(i) Die Lösungsmenge von $\mathbf{Ax} = 0$ is ein Unterraum von $\mathbb{R}^m$.

(ii) Die Lösungsmenge von $\mathbf{Ax} = \mathbf{b}$, falls nicht leer, ist ein affiner Raum in $\mathbb{R}^m$.

R. Hiptmair
SAM, ETHZ

3.2 Lineare Gleichungssysteme: Anwendungen

3.2.1 Schnitt von Hyperebenen

3.2.2 Umrechnung von Basiskoeffizienten

3.2.3 Testen linearer (Un)abhängigkeit im $\mathbb{R}^n$

3.3 Gausselimination

Link zu den handgeschriebenen Präsentationsfolien

Def. 3.3.A. Es sei $\mathbf{A} \in \mathbb{R}^{n,m}$ eine $n \times m$-Matrix. Die Matrix $\mathbf{B} = (b_{i,j}) \in \mathbb{R}^{n,m}$ geht aus $\mathbf{A} = (a_{i,j})$ durch Zeilenumformung hervor, wenn es
(i) $l, k \in \{1, n\}$, $l \neq k$, und $\beta \in \mathbb{R}$ so gibt, dass

$$b_{i,j} = \begin{cases} a_{i,j} & \text{, falls } i \neq k \text{ ,} \\ a_{k,j} - \beta a_{l,j} & \text{, falls } i = k \text{ ,} \end{cases} \quad i \in \{1, \ldots, n\}, j \in \{1, \ldots, m\} \text{ .}$$

oder (ii) ein $k \in \{1, \ldots, n\}$ und $\beta \neq 0$ so gibt, dass

$$b_{i,j} = \begin{cases} a_{i,j} & \text{, falls } i \neq k \text{ ,} \\ \beta a_{k,j} & \text{, falls } i = k \text{ ,} \end{cases} \quad i \in \{1, \ldots, n\}, j \in \{1, \ldots, m\} \text{ .}$$

In Worten: $\mathbf{B}$ geht aus $\mathbf{A}$ hervor indem das β-fache der l-ten Zeile zur k-ten Zeile addiert wird, oder die k-te Zeile mit $\beta \neq 0$ multipliziert wird.

MATLAB: `B = A; B(k,:) = B(k,:) + beta*B(l,:)`

Satz 3.3.B. Wendet man die gleiche Zeilenumformung auf die Koeffizientenmatrix und den Rechte-Seite-Vektor eines linearen Gleichungssystems an, so ändert sich die Lösungsmenge nicht.

Code 3.3.1: Gausselimination für ein allgemeines Gleichungssystem

```
function [A,b] = gaussianelimination(A,b)
% This function converts a linear system of equations with coefficient matrix
% A and right hand side vector b into canonical for
% ("Zeilenstufenform") by performing row transformations.
% A must pass an n×m-matrix and b a column vector of
% length n.
[n,m] = size(A); % Obtain number of rows n and number of columns m of A
% Remember that testing for equality is not a good idea, if floating point
% operations are involved, because roundoff will render this test meaningless.
% Therefore it is recommended that one tests relative equality up to a relative
% perturbation below a prescribed tolerance. This tolerance is specified in the
% next line.
tol = eps*n*m;

% First trick: add b as rightmost column to A in order to facilitate
% implementation of simultaneous row transformations of A and b
A = [A,b],

% j is the current column index, l the current row index.
```


R. Hiptmair
SAM, ETHZ

```
j = 1; l = 1;
while (j<=m)
  % Search from top of column j to find first non-zero entry.
  % In the absence of round-off, we could just test whether A(i,j) ==0,
  % otherwise use testzero(A(i,j),norm(A(:,j)),tol).
  i = l; while ((i<=n) && testzero(A(i,j),norm(A(:,j)),tol)), i =
    i+1; end
  if (i<=n)
    % Swap rows i and j in order to move a non-zero matrix entry to
    % position (l,j). Swapping rows is a valid row transformation!
    A([i,l],:) = A([l,i],:);
    % Rescale lth row to ensure A(l,j) = 1. This is a row
    % transformation of type (ii)
    A(l,:) = A(l,:)/A(l,j);
    % Now subtract multiples of lth row from the rows below to
    % annihilate the matrix entries in the jth column. This amounts to
    % applying row transformations of type (i).
    for k=[(1:l-1),(l+1:n)],  A(k,:) = A(k,:) - A(k,j)*A(l,:); end
    l = l+1;
  end
  fprintf('i = %d, l= %d, j = %d\n',i,l,j); A,
  j = j+1;
end
b = A(:,m+1); A = A(:,1:m);
```


R. Hiptmair
SAM, ETHZ

```
end

% Surrogate function for testing whether a matrix entry vanishes
function iszero = testzero(z,ref,tol)
  if (nargin == 1), iszero = (z == 0);
  else iszero = (abs(z) <= tol*ref); end
end
```

Def. 3.3.C. Eine Matrix $\mathbf{A} \in \mathbb{R}^{n,m} \setminus \{\mathbf{O}\}$ ist in Zeilenstufenform, falls es eine Zahl $r \in \{1,\ldots,\min\{n,m\}\}$ und Indices $1 \leq i_1 < i_2 < \cdots < i_r \leq m$ so gibt, dass ($\mathbf{e}^k \hat{=} k$. Einheitsvektor im $\mathbb{R}^n$)

$$A(:,i_k) = \mathbf{e}^k,\ k = 1,\ldots,r, \quad \text{und} \quad a_{ij} = 0 \quad \text{für alle } i > r\ .$$

r heisst der Rang von $\mathbf{A}$.

☞ Demonstration der Gausselimination für eine lineares Gleichunggsystem: `GS_example.m`

Satz 3.3.D. Durch den Gausseliminationsalgorithmus aus Code 3.3.0 lässt sich jede Matrix durch sukzessive Zeilenumformungen auf Zeilenstufenform bringen.

R. Hiptmair
SAM, ETHZ

Def. 3.3.E. Der Rang der Zeilenstufenform einer Matrix heisst dann Rang der Matrix.

3.4 Lösungsmengen linearer Gleichungssysteme

Link zu den handgeschriebenen Präsentationsfolien

3.5 Lösen linearer Gleichungssysteme in MATLAB

Lösen eines linearen Gleichungssystems mit quadratischer Koeffizientenmatrix in MATLAB: \

Code 3.5.1: (lsesolvedemo.m))Lösung eines linearen Gleichngssystems

```
1 A = [1 2 3;4 6 5;7 8 9]; b = [1;1;2;]; % Initialize matrix and vector
2 x = A\b, % solve linear system of equations A*x=b
3 x = linsolve(A,b); % An alternative way to write the backslash
4 r = A*x-b, % Test, whether a solution was found
```

```
A = [1 2 3;4 5 6;7 8 9] % 3×3-matrix with rank 2 (not regular!)
z = A\b, % In exact arithmetic this linear system has no solution
% However. MATLAB computes a solution, because the tiny perturbations
% introduced by roundoff make it believe that the linear system
% has a unique solution.
r = A*z-b, % test pseudo-solution: not quite a solution
```

Wichtig: Numerisch lassen sich nur lineare Gleichungssysteme mit quadratischer Koeffizientenmatrix, die eine eindeutige Lösung haben, zuverlässig lösen!

Für $\mathbf{A} \in \mathbb{R}^{n,n}$ vollbesetzt:

Rechenaufwand(`A\b`) = $O(n^3)$

(Drei geschachtelte Schleifen für Gausselimination)

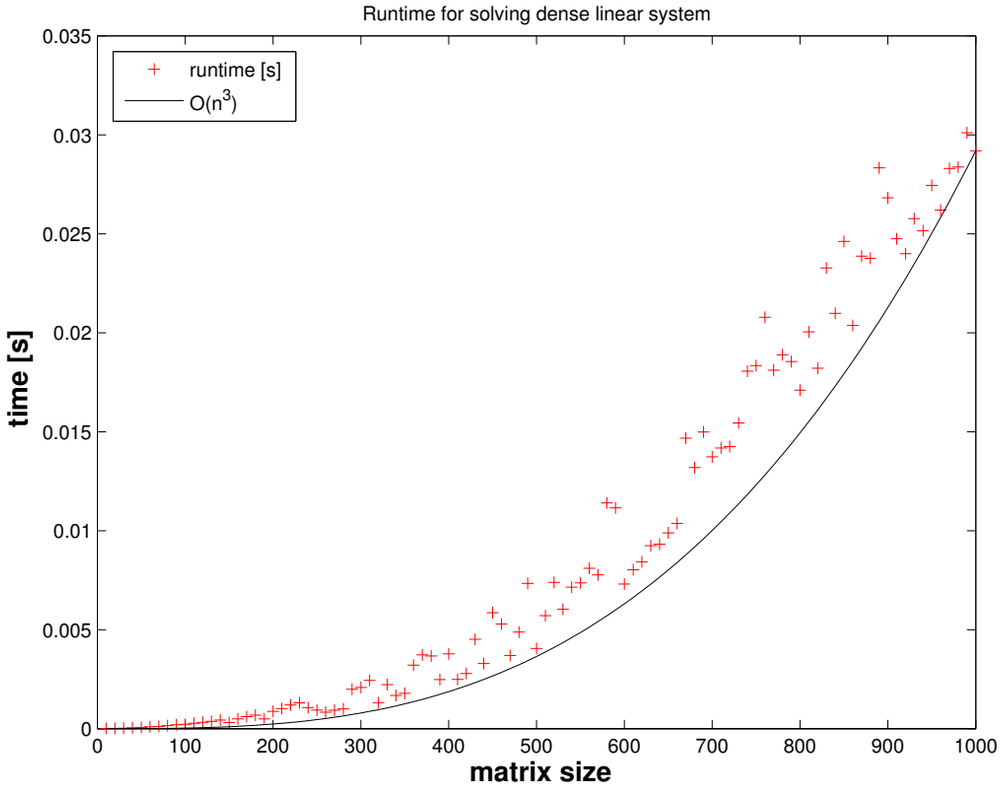
Code 3.5.3: (lsesolvetiming.m)) Rechenzeiten zum Lösen eines vollbesetzten linearen Gleichungssystems

```
% Runtime measurements for solution of dense linear system of
% equations in MATLAB
N = 2000; % maximum size of matrix
A = gallery('minij',N); % initialize a regular dense matrix: a_{i,j} = min{i,j}
v = rand(N,1); % initialize random right hand side vector
```


R. Hiptmair
SAM, ETHZ

```
6
7  res = []; % matrix for collecting the results
8  for i=10:10:N
9      B = A(1:i,1:i); % extract submatrix
10     w = v(1:i);     % extract subvector
11     t = realmax;
12     for j=1:3, tic; z = B\w; t = min(toc,t); end
13     res = [res; i, t],
14 end
15
16 figure; plot(res(:,1),res(:,2),'r+',...
17     res(:,1),res(:,1).^3/(res(end,1)^3)*res(end,2),'k-');
18 xlabel('{\bf matrix size}','fontsize',14);
19 ylabel('{\bf time [s]}','fontsize',14);
20 title('Runtime for solving dense linear system');
21 legend('runtime [s]','O(n^3)','location','best');
22
23 print -depsc2 '../../LANMFigures/lsesolvetiming.eps';
```


R. Hiptmair
SAM, ETHZ



◁ Rechenzeiten gemessen mit Code 3.5.1

Mac OS X 10.6, MATLAB R2012a,
Intel Core i7, 2.66 GHz,

Code 3.5.4: (sparselseslvtiming.m)) Rechenzeiten zum Lösen eines dünnbesetzten LGS

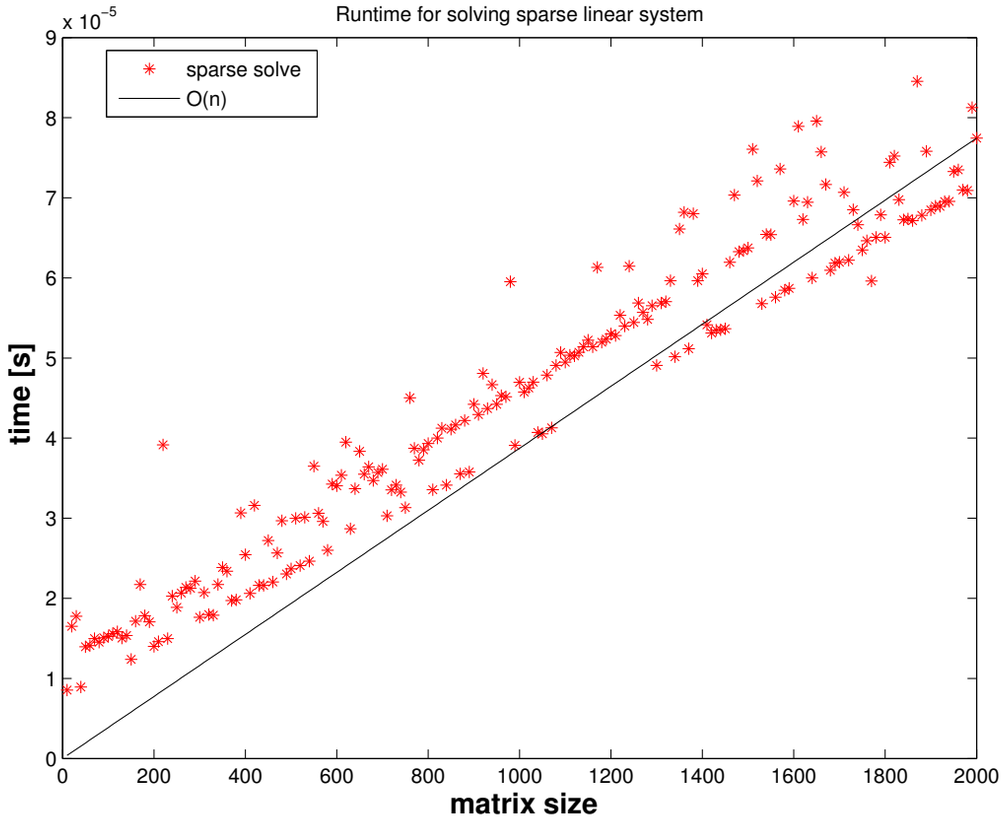
```
% Runtime measurements for solution of tridiagonal linear system of
% equations in MATLAB
N = 2000; % maximum size of matrix
% Initialize tridiagonal matrix using Code 2.4.0
A = inittridiag((1:N)',-ones(N-1,1),-ones(N-1,1));
v = rand(N,1); % initialize random right hand side vector

res = []; % matrix for collecting the results
for i=10:10:N
    B = A(1:i,1:i); % extract sparse submatrix
```

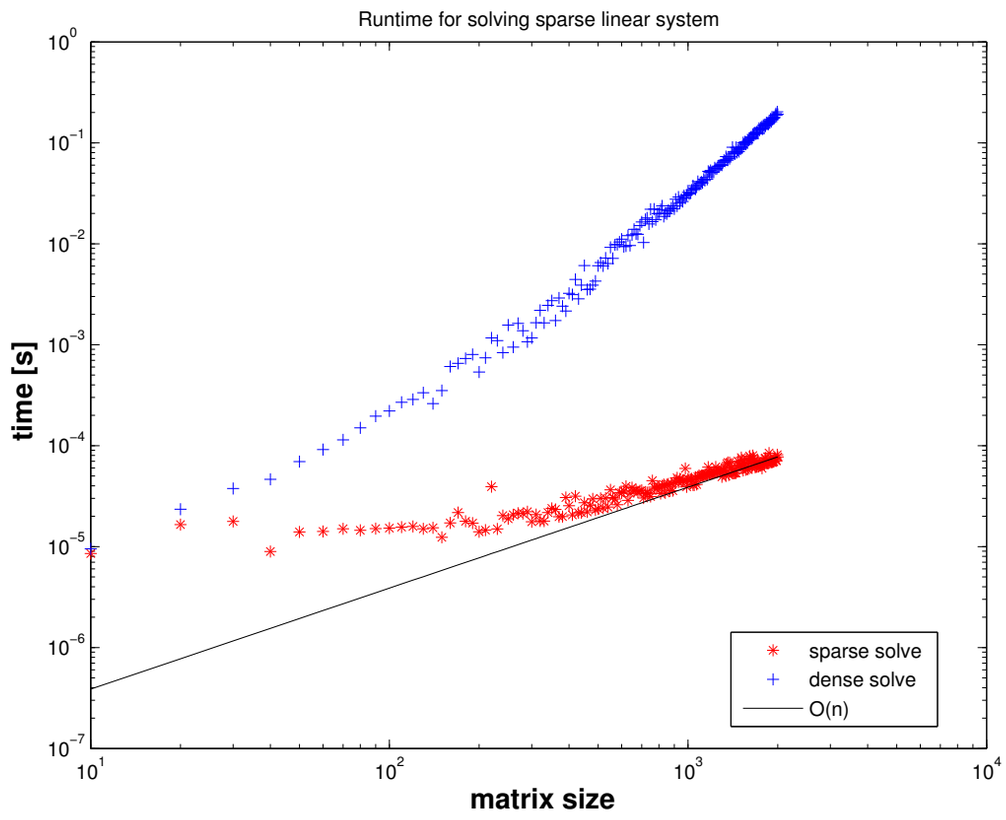
R. Hiptmair
SAM, ETHZ

```
    w = v(1:i);       % extract subvector
    C = full(B);      % Create dense matrix from B
    ts = realmax; td = realmax;
    for j=1:3, tic; z = B\w; ts = min(toc,ts); end
    for j=1:3, tic; z = C\w; td = min(toc,td); end
    res = [res; i, ts, td],
end

figure;
plot(res(:,1),res(:,2),'r*',...
     res(:,1),res(:,1)/(res(end,1))*res(end,2),'k-');
xlabel('{\bf matrix size}','fontsize',14);
ylabel('{\bf time [s]}','fontsize',14);
title('Runtime for solving sparse linear system');
legend('sparse solve','O(n)','location','best');

print -depsc2 '../../LANMFigures/sparselseslvtiming.eps';

figure;
loglog(res(:,1),res(:,2),'r*',...
       res(:,1),res(:,3),'b+',...
       res(:,1),res(:,1)/(res(end,1))*res(end,2),'k-');
xlabel('{\bf matrix size}','fontsize',14);
ylabel('{\bf time [s]}','fontsize',14);
```


R. Hiptmair
SAM, ETHZ

```
title('Runtime for solving sparse linear system');
legend('sparse solve','dense solve','O(n)','location','best');

print -depsc2 '../../LANMFigures/cmplseslvtiming.eps';
```





△ Rechenzeiten gemessen mit Code 3.5.3 (Mac OS X 10.6, MATLAB R2012a, Intel Core i7, 2.66 GHz)

Berücksichtige die Dünnbesetztheit einer Matrix immer bei der Implementierung numerischer Algorithmen (spezielle Datenstrukturen).

3.6 Inverse Matrix

Def. 3.6.A. Eine quadratische Matrix $\mathbf{A} \in \mathbb{R}^{n,n}$ heisst invertierbar oder regulär, wenn $\mathrm{Rang}(\mathbf{A}) = n$.

Satz 3.6.B. Zu jeder invertierbaren Matrix $\mathbf{A} \in \mathbb{R}^{n,n}$ gibt es eine eindeutige Matrix $\mathbf{B} \in \mathbb{R}^{n,n}$ so, dass

$$\mathbf{AB} = \mathbf{BA} = \mathbf{I} \quad \text{(Einheitsmatrix)} .$$

Diese Matrix $\mathbf{B}$ heisst die Inverse der Matrix $\mathbf{A}$, in Zeichen $\mathbf{B} = \mathbf{A}^{-1}$.

$\blacktriangleright$ $\mathbf{A}$ regulär: $\mathbf{Ax} = \mathbf{b} \quad \Leftrightarrow \quad \mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$

R. Hiptmair
SAM, ETHZ

Satz 3.6.C. Sind $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n,n}$ regulär, dann gilt dies auch für ihr Produkt $\mathbf{AB}$ und es gilt

$$(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1} .$$

3.7 Dimension von Unterräumen

$(V, +, \cdot) \hat{=}$ Vektorraum.

R. Hiptmair
SAM, ETHZ

Def. 3.7.A. Die Dimension eines Unter(vektor)raums $U \subset V$, in Zeichen $\dim(U)$ ist die (eindeutige) Anzahl der Vektoren (irgend)einer Basis von U.

Satz 3.7.B. Die Dimension eines Unterraums $U \subset V$ is die maximale Anzahl von linear unabhängigen Vektoren in jeder beliebigen Teilmenge von U.

Satz 3.7.C. Für $\{\mathbf{v}^1, \ldots, \mathbf{v}^m\} \subset \mathbb{R}^n$ gilt mit $\mathbf{V} := [\mathbf{v}^1, \ldots, \mathbf{v}^m]$

$$\dim \operatorname{Span}\{\mathbf{v}^1, \ldots, \mathbf{v}^m\} = \operatorname{Rang}(\mathbf{V}) = \operatorname{Rang}(\mathbf{V}^T) \; .$$

Def. 3.7.D. Für eine Matrix $\mathbf{A} \in \mathbb{R}^{n,m}$ ist

- der Spaltenraum oder das Bild von $\mathbf{A}$, in Zeichen $\operatorname{Bild}(\mathbf{A})$ des Span ihrer Spalten

$$\operatorname{Bild}(\mathbf{A}) = \operatorname{Span}\{\mathbf{A}(:,1), \ldots, \mathbf{A}(:,m)\} = \{\mathbf{Ax},\, \mathbf{x} \in \mathbb{R}^n\} \subset \mathbb{R}^n \; .$$

- der Zeilenraum von $\mathbf{A}$ der Span ihrer Zeilen

$$\operatorname{Span}\{\mathbf{A}(1,:), \ldots, \mathbf{A}(n,:)\} \subset \mathbb{R}^{1,m} \; .$$

Satz 3.7.C $\Rightarrow$ Gleiche Dimension von Spalten- und Zeilenraum einer Matrix!

$$\dim \operatorname{Bild}(\mathbf{A}) = \dim \operatorname{Span}\{\mathbf{A}(1,:), \ldots, \mathbf{A}(n,:)\} = \operatorname{Rang}(\mathbf{A})$$

Def. 3.7.E. Für eine Matrix $\mathbf{A} \in \mathbb{R}^{n,m}$ ist ihr Nullraum oder Kern gegeben durch

$$\mathrm{Kern}(\mathbf{A}) := \{\mathbf{x} \in \mathbb{R}^m : \ \mathbf{A}\mathbf{x} = \mathbf{0}\} \ .$$

Satz 3.7.F. Für eine Matrix $\mathbf{A} \in \mathbb{R}^{n,m}$ gilt

$$\dim \mathrm{Kern}(\mathbf{A}) = m - \mathrm{Rang}(\mathbf{A}) = m - \dim \mathrm{Bild}(\mathbf{A}) \ .$$

Satz 3.7.G. Eine quadratische Matrix $\mathbf{A} \in \mathbb{R}^{n,n}$ ist genau dann regulär, wenn $\mathrm{Kern}(\mathbf{A}) = \{\mathbf{0}\}$.

$\mathrm{Kern}(\mathbf{A}) = \{0\} \Leftrightarrow$ Das LGS $\mathbf{A}\mathbf{x} = \mathbf{0}$ hat nur die Lösung $\mathbf{x} = \mathbf{0}$.

R. Hiptmair
SAM, ETHZ

3.8 Anwendungsbeispiele linearer Gleichungssysteme

Link zu den handgeschriebenen Präsentationsfolien

3.8.1 Netzwerke

3.8.2 Ideale statische Fachwerke

➢ [2, Kapitel 11].

3.9 Kleinste-Quadrate-Lösungen

Link zu den handgeschriebenen Präsentationsfolien

R. Hiptmair
SAM, ETHZ

3.9.1 Überbestimmte lineare Gleichungssysteme

Satz 3.9.1.A Ein lineares Gleichungssystem $\mathbf{Ax} = \mathbf{b}$, $\mathbf{A} = [\mathbf{a}^1, \ldots, \mathbf{a}^m] \in \mathbb{R}^{n,m}$, hat genau dann eine Lösung, wenn gilt

$$\mathbf{b} \in \mathrm{Bild}(\mathbf{A}) = \mathrm{Span}\{\mathbf{a}^1, \ldots, \mathbf{a}^m\} \ .$$

3.9.2 Orthogonalprojektion

Satz 3.9.2.A Für $\{\mathbf{a}^1, \ldots, \mathbf{a}^m\} \subset \mathbb{R}^n$ ist die Orthogonalprojektion $\mathbf{b}^* \in \mathbb{R}^n$ von $\mathbf{b} \in \mathbb{R}^n$ auf $\mathrm{Span}\{\mathbf{a}^1, \ldots, \mathbf{a}^m\}$ gegeben durch

$$\mathbf{b}^* = \sum_{j=1}^{m} x_j \mathbf{a}^j \ ,$$

wobei $\mathbf{x} = (x_1, \ldots, x_m)^T \in \mathbb{R}^m$ eine Lösung der Normalengleichungen (NGL)

$$\mathbf{A}^T \mathbf{A}\mathbf{x} = \mathbf{A}^T \mathbf{b} \qquad \text{(3.9.2.B)}$$

mit $\mathbf{A} = [\mathbf{a}^1, \ldots, \mathbf{a}^m] \in \mathbb{R}^{n,m}$ ist.

Definition 3.9.2.C Eine Lösung der Normalengleichungen (3.9.2.B) heisst eine Kleinste-Quadrate-Lösung des linearen Gleichungssystems $\mathbf{A}\mathbf{x} = \mathbf{b}$.

Satz 3.9.2.D Ist das lineare Gleichungssystem $\mathbf{A}\mathbf{x} = \mathbf{b}$, $\mathbf{A} \in \mathbb{R}^{n,m}$, $\mathbf{b} \in \mathbb{R}^n$, lösbar, so ist jede Lösung auch eine kleinste-Quadrate-Lösung.

Beachte: Eine kleinste-Quadrate-Lösung ist nicht unbedingt eine Lösung!

Satz 3.9.2.E Für $\mathbf{A} \in \mathbb{R}^{n,m}$ gilt

$$\mathrm{Kern}(\mathbf{A}) = \mathrm{Kern}(\mathbf{A}^T\mathbf{A})$$

Ist insbesondere $\mathrm{Rang}(\mathbf{A}) = m$, dann ist $\mathbf{A}^T\mathbf{A}$ invertierbar.

In MATLAB: `x = A\b` berechnet Kleinste-Quadrate-Lösung, falls `A` eine $n \times m$-Matrix mit $m > n$.

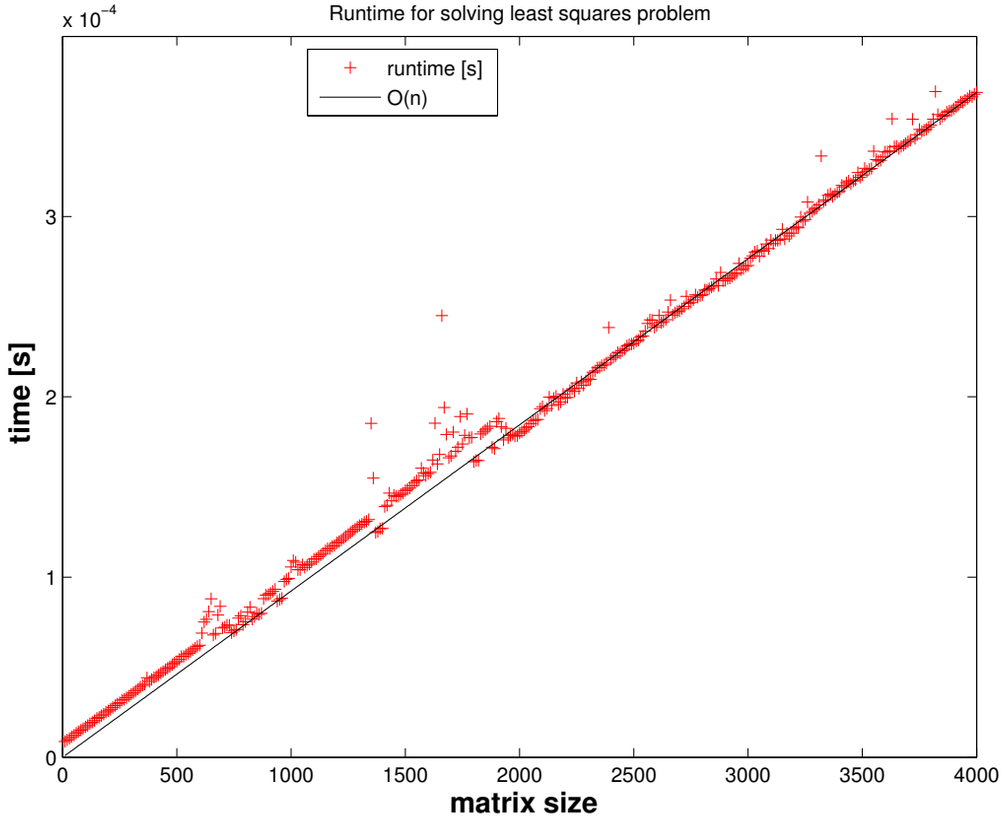
Code 3.9.1: Lineare Regression

```
function x = linearregression(t,y)
% Solution of linear regression problem (fitting of a line to data) for data
% points (t_i,y_i), i=1,...,n passed in the column vectors
% t and y.
% The return value is a 2-vector, containing the slope of the fitted line in
% x(1) and its offset in x(2)
n = length(t); if (length(y) ~= n), error('data size mismatch'); end
% Coefficient matrix of overdetermined linear system
A = [t,ones(n,1)];
% Determine least squares solution by using MATLAB's \-operator
x = A\y;
```

Code 3.9.3: (lsqtiming.m) Rechenzeiten zum Lösen eines linearen Ausgleichsproblems mit wenigen Unbekannten

```
% Runtime measurements for solution of dense linear least squares
% problem in MATLAB
m = 5; % Number of unknowns (width of matrix)
N = 4000; % maximum height of matrix
A = rand(N,m); % initialize a random matrix
v = rand(N,1); % initialize random right hand side vector

res = []; % matrix for collecting the results
for i=10:10:N
    B = A(1:i,:);    % extract submatrix
    w = v(1:i);      % extract subvector
    t = realmax;
    % measure time it takes to solve least squares problem. Note that the
    % \-operators triggers solution of linear least squares problem,
    % because we deal an underdetermined system
    for j=1:3, tic; z = B\w; t = min(toc,t); end
    res = [res; i, t];
end

figure; plot(res(:,1),res(:,2),'r+',...
    res(:,1),res(:,1).^1/(res(end,1)^1)*res(end,2),'k-');
xlabel('{\bf No. of equations}','fontsize',14);
```

```
ylabel('{\bf time [s]}','fontsize',14);
title('Runtime for solving least squares problem with 5 unknowns');
legend('runtime [s]','O(n)','location','best');

print -depsc2 '../../LANMFigures/lsqtiming.eps';
```



◁ Rechenzeiten gemessen mit Code 3.9.1

Mac OS X 10.6, MATLAB R2012a,
Intel Core i7, 2.66 GHz,

Rechenaufwand zur Lösung eines $n \times m$ linearen Ausgleichsproblems im Fall $n \gg m$: $O(n)$

(für festes m und grosses $n \to \infty$)

3.9.3 Residuenminimierung

Satz 3.9.3.B Jede kleinste-Quadrate-Lösung $\mathbf{x}^*$ von $\mathbf{Ax} = \mathbf{b}$, $\mathbf{a} \in \mathbb{R}^{n,m}$, $\mathbf{b} \in \mathbb{R}^n$ erfüllt

$$\|\mathbf{b} - \mathbf{Ax}^*\| = \min\{\|\mathbf{b} - \mathbf{Ax}\| : \mathbf{x} \in \mathbb{R}^m\}$$

$$\Updownarrow$$

$$\sum_{i=1}^{n} \left(b_i - \sum_{j=1}^{m} a_{i,j} x_j^* \right)^2 = \min \left\{ \sum_{i=1}^{n} \left(b_i - \sum_{j=1}^{m} a_{i,j} x_j \right)^2 : \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix} \in \mathbb{R}^m \right\} ,$$

d.h. die kleinste-Quadrate-Lösung realisiert ein Residuum $\mathbf{r} := \mathbf{b} - \mathbf{Ax}^*$ mit minimaler Norm.

4 Lineare Abbildungen

4.1 Grundlegende Konzepte und Eigenschaften

Seien $(V,+,\cdot)$, $(W,+,\cdot)$ und $(U,+,\cdot)$ Vektorräume.

Def. 4.1.A. Eine Abbildung (Funktion) $\mathsf{L} : V \to W$ ist linear, falls

$$\mathsf{L}(\mathbf{x}+\mathbf{y}) = \mathsf{L}(\mathbf{x}) + \mathsf{L}(\mathbf{y}) \quad \text{für alle} \quad \mathbf{x},\mathbf{y} \in V\ , \tag{L1}$$

$$\mathsf{L}(\alpha\mathbf{v}) = \alpha\mathsf{L}(\mathbf{v}) \quad \text{für alle} \quad \mathbf{v} \in V,\ \alpha \in \mathbb{R}\ . \tag{L2}$$

Lineare Abbildungen vertauschen mit der Bildung von Linearkombinationen:

$$\mathsf{L}(\sum_{j=1}^{m} c_j \mathbf{v}^j) = \sum_{j=1}^{m} c_j \mathsf{L}(\mathbf{v}^j) \quad \text{für } \mathbf{v}^j \in V, c_j \in \mathbb{R}\ .$$

R. Hiptmair
SAM, ETHZ

Notation: $\mathcal{L}(V,W) \hat{=}$ Menge der linearen Abbildungen $V \to W$.

Satz 4.1.B. Die Hintereinanderausführung (Komposition) zweier linearer Abbildungen ist wieder eine lineare Abbildung:

$$\mathsf{L} \in \mathcal{L}(V,W),\ \mathsf{S} \in \mathcal{L}(W,U) \quad \Rightarrow \quad \mathsf{S} \circ \mathsf{L} \in \mathcal{L}(V,U)\ .$$

Satz 4.1.C. Für eine lineare Abbildung $\mathsf{L} : V \to W$ is der Nullraum/Kern

$$\mathrm{Kern}(\mathsf{L}) := \{\mathbf{v} \in V :\ \mathsf{L}(\mathbf{v}) = 0\}$$

ein Untervektorraum von V, und das Bild

$$\mathrm{Bild}(\mathsf{L}) := \{\mathsf{L}(\mathbf{v}),\ \mathbf{v} \in V\}$$

ist ein Untervektorraum von W.

R. Hiptmair
SAM, ETHZ

Satz 4.1.D. Ist $U \subset V$ ein Unterraum/affiner Raum und $\mathsf{L} : V \to W$ eine lineare Abbildung, dann ist auch

$$\mathsf{L}(U) := \{\mathsf{L}(\mathbf{v}),\ \mathbf{v} \in U\} \subset W \tag{4.1.1}$$

ein Unterraum/affiner Raum.

Def. 4.1.E. Eine Teilmenge $M \subset V$ des Vektorraums $(V, +, \cdot)$ heisst konvex, falls

$$\mathbf{x}, \mathbf{y} \in M \quad \Rightarrow \quad \tau\mathbf{x} + (1-\tau)\mathbf{y} \in M \quad \text{für alle } 0 \le \tau \le 1\ ,$$

das heisst, falls mit je zwei Punkten auch ihre Verbindungsstrecke in M liegt.

Satz 4.1.F. Ist $\mathsf{L} : V \to W$ eine lineare Abbildung und $M \subset V$ konvex, dann ist auch $\mathsf{L}(M) \subset W$ eine konvexe Menge.

Visualisierung linearer Abbildungen $\mathbf{x} \mapsto \mathbf{A}\mathbf{x}$, $\mathbf{A} \in \mathbb{R}^{2,2}$, in 2D:

R. Hiptmair
SAM, ETHZ

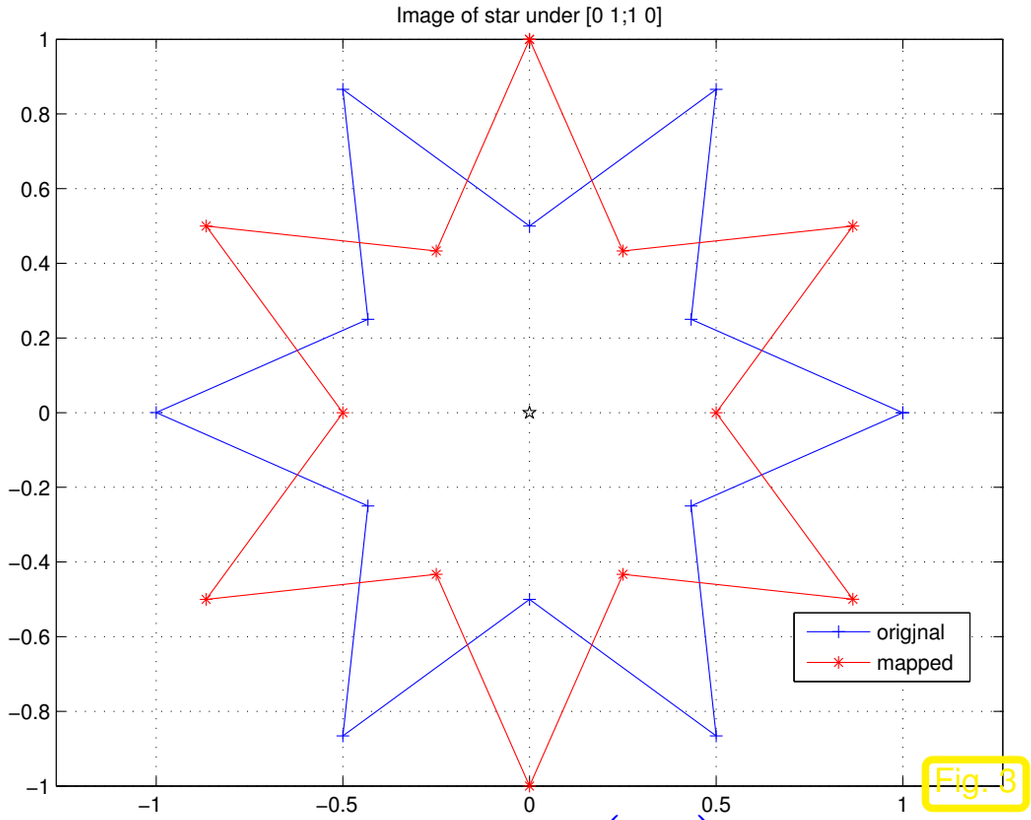


Fig. 3

Abb. 5: $\mathbf{A} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$

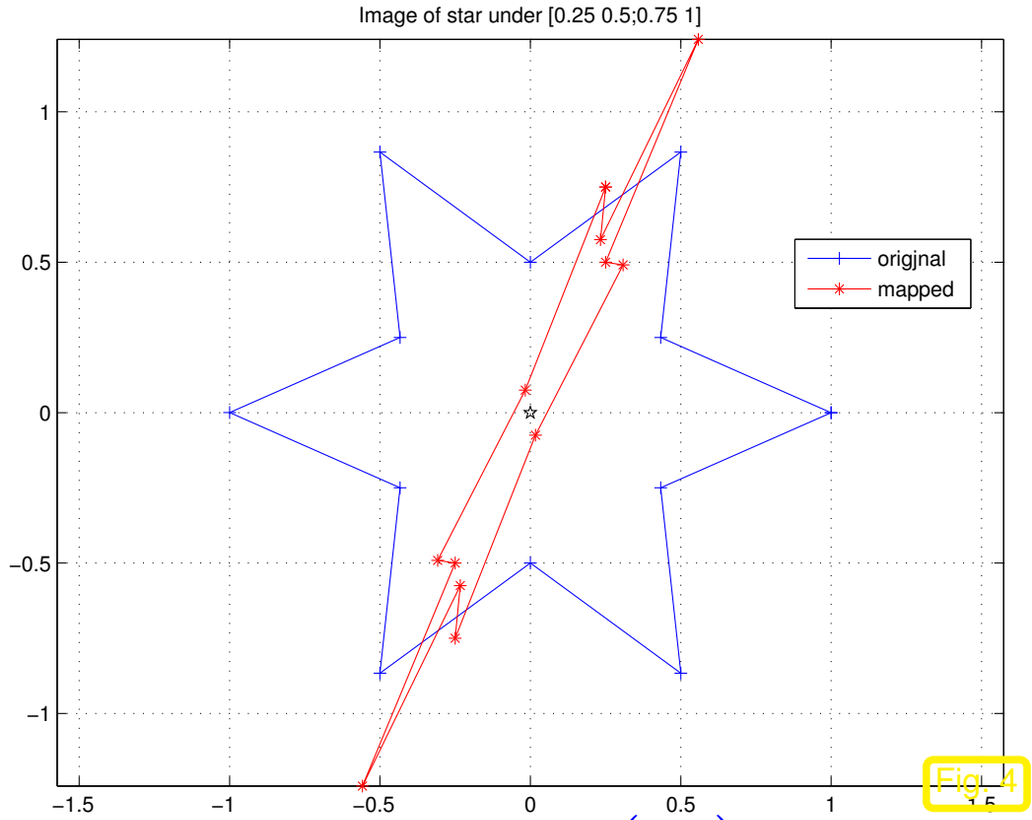


Fig. 4

Abb. 6: $\mathbf{A} = \frac{1}{4}\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$

R. Hiptmair
SAM, ETHZ

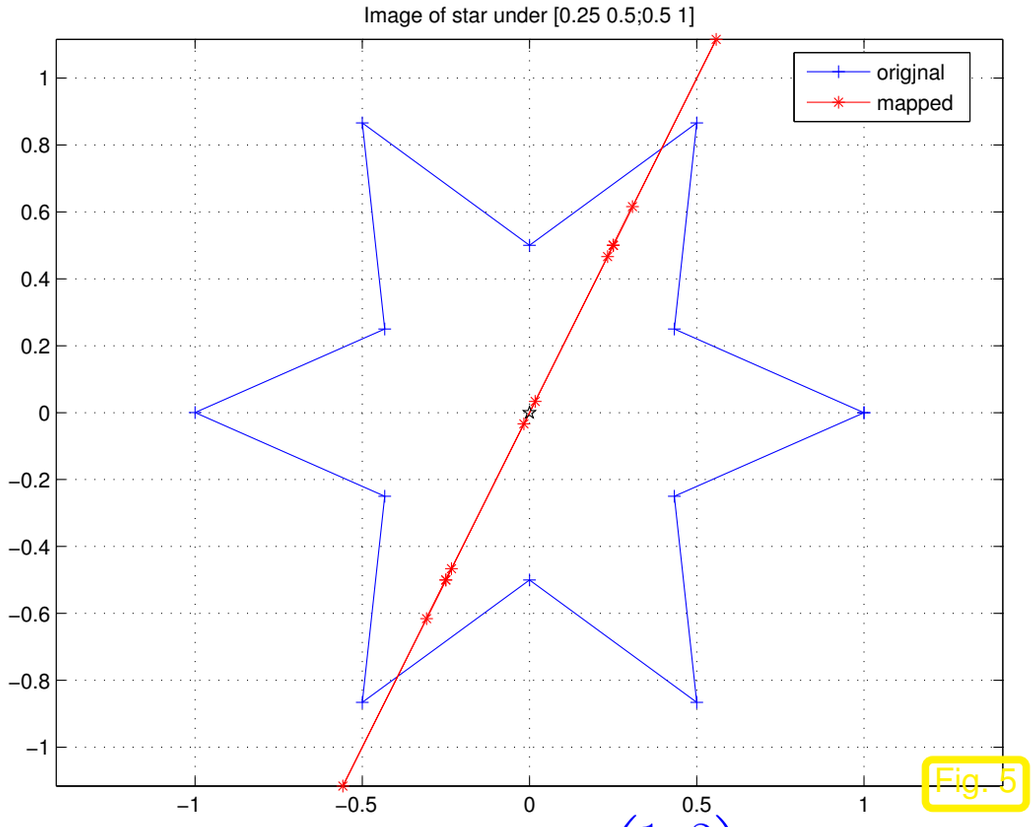


Fig. 5

Abb. 7: $\mathbf{A} = \frac{1}{4}\begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix}$

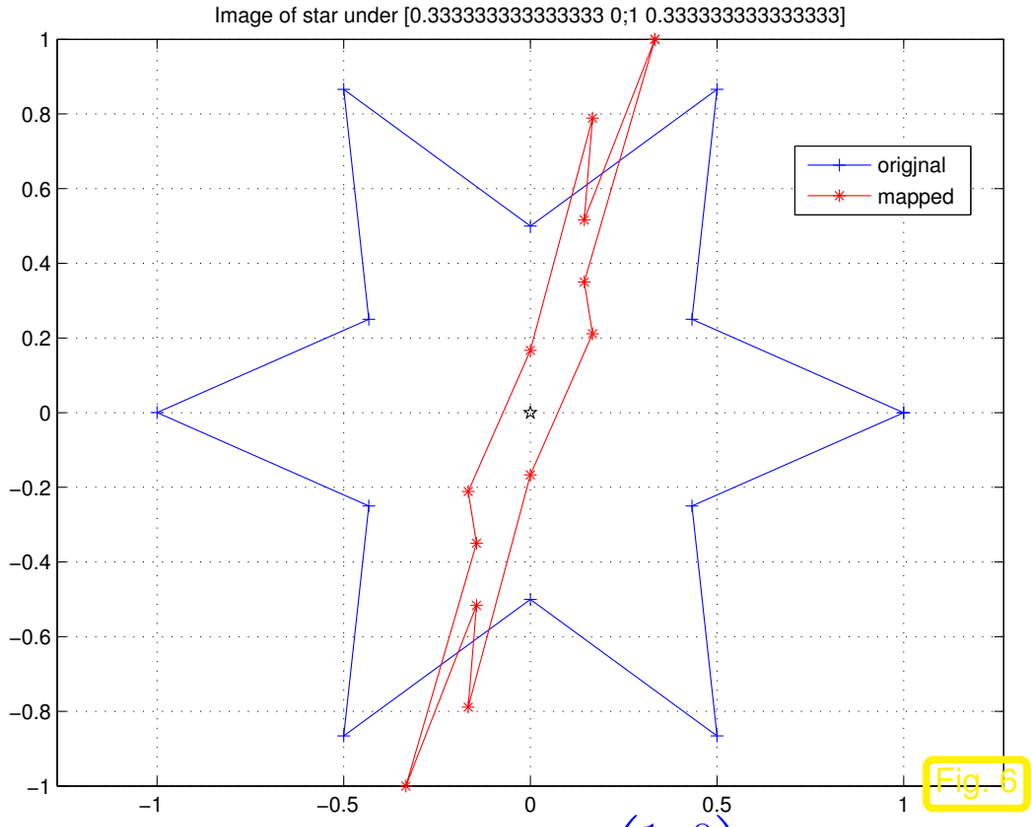


Fig. 6

Abb. 8: $\mathbf{A} = \frac{1}{3}\begin{pmatrix} 1 & 0 \\ 3 & 1 \end{pmatrix}$

Code 4.1.2: Visualisierung linearer Abbildungen in 2D

```
function linmapvis(A)
% Visualizes the action of the linear mapping described by matrix A by applyung
% it to a star.

% Create a 2×13-matrix x whose columns contain the coordinates of the
% vertices of the star
x_outer = [cos(2*pi*(0:6)/6);sin(2*pi*(0:6)/6)];
x_inner = 0.5*[cos(2*pi*(0:5)/6+pi/6);sin(2*pi*(0:5)/6+pi/6)];
```

R. Hiptmair
SAM, ETHZ

```
x = zeros(2,13); x(:,1:2:13) = x_outer; x(:,2:2:12) = x_inner;

% Map the star
y = A*x;

% Plot the original star and its image under A
figure('name','image of star');
plot(x(1,:),x(2,:),'b-+',y(1,:),y(2,:),'r-*',[0],[0],'kp'); hold
  on;
legend('origjnal','mapped','location','best'); grid on;
axis equal; title(['Image of star under ' mat2str(A)]);
```


R. Hiptmair
SAM, ETHZ

4.2 Matrixdarstellung linearer Abbildungen

Link zu den handgeschriebenen Präsentationsfolien

Satz 4.2.B. Eine lineare Abbildung $\mathsf{L} \in \mathcal{L}(V,W)$ ist eindeutig durch ihre Wirkung auf die Vektoren einer Basis von V bestimmt.

Ist $\mathcal{B}_V := \{\mathbf{b}^1, \ldots, \mathbf{b}^m\}$, $m := \dim V$, eine Basis von V und $\mathcal{B}_W := \{\mathbf{q}^1, \ldots, \mathbf{q}^n\}$, $n := \dim W$, eine Basis von W, so ist die

Matrixdarstellung ("Koordinatendarstellung") $\mathbf{A} := I_{\mathcal{B}_V}^{\mathcal{B}_W}(\mathsf{L}) \in \mathbb{R}^{n,m}$ von $\mathsf{L} \in \mathcal{L}(V, W)$

bzüglich der Basen $\mathcal{B}_V$ und $\mathcal{B}_W$ definiert durch

$$\mathbf{A} = (a_{l,j}): \quad \boxed{\mathsf{L}(\mathbf{b}^j) = \sum_{l=1}^{n} a_{l,j}\, \mathbf{q}^l} \,. \tag{4.2.C}$$

Satz 4.2.D. Für eine lineare Abbildung $\mathsf{L} \in \mathcal{L}(V, W)$ mit Matrixdarstellung $\mathbf{A} = I_{\mathcal{B}_V}^{\mathcal{B}_W}(\mathsf{L}) \in \mathbb{R}^{n,m}$ gilt

$$I_{\mathcal{B}_W}(\mathrm{Bild}(\mathsf{L})) = \mathrm{Bild}(\mathbf{A}) \,,$$
$$I_{\mathcal{B}_V}(\mathrm{Kern}(\mathsf{L})) = \mathrm{Kern}(\mathbf{A}) \,.$$

$$\dim \mathrm{Kern}(\mathsf{L}) = \dim V - \dim \mathrm{Bild}(\mathsf{L}) \,. \tag{4.2.E}$$

R. Hiptmair
SAM, ETHZ

Satz 4.2.F. (Matrixdarstellung der Komposition linearer Abbildungen)
Seien U, V, W Vektorräume der Dimensionen l, m, bzw. n mit Basen $\mathcal{B}_U$, $\mathcal{B}_V$ und $\mathcal{B}_W$.
Ferner, sei $\mathbf{A} \in \mathbb{R}^{n,m}$ die Matrixdarstellung von $\mathsf{L} \in \mathcal{L}(V, W)$ bzgl. der Basen $\mathcal{B}_V$ und $\mathcal{B}_W$, und $\mathbf{B} \in \mathbb{R}^{l,n}$ die Matrixdarstellung von $\mathsf{S} \in \mathcal{L}(W, U)$ bzgl. der Basen $\mathcal{B}_W$ und $\mathcal{B}_U$.
Dann hat $\mathsf{S} \circ \mathsf{L}$ die Matrixdarstellung $\mathbf{BA} \in \mathbb{R}^{l,m}$ bzgl. $\mathcal{B}_V$ und $\mathcal{B}_U$:

$$I_{\mathcal{B}_V}^{\mathcal{B}_U}(\mathsf{S} \circ \mathsf{L}) = I_{\mathcal{B}_W}^{\mathcal{B}_U}(\mathsf{S}) \cdot I_{\mathcal{B}_V}^{\mathcal{B}_W}(\mathsf{L}) \,.$$

R. Hiptmair
SAM, ETHZ

4.3 Matrixdarstellung bei Basiswechsel

Link zu den handgeschriebenen Präsentationsfolien

4.4 Lineare Selbstabbildungen

Satz 4.4.A. Seien $\mathcal{B}_V = \{\mathbf{b}^1, \ldots, \mathbf{b}^n\}$, $n := \dim V$, und $\widetilde{\mathcal{B}}_V = \{\widetilde{\mathbf{b}}^1, \ldots, \widetilde{\mathbf{b}}^n\}$ zwei Basen von V und die Basiswechselmatrix $\mathbf{S} = (s_{i,j})_{i,j=1}^n \in \mathbb{R}^{n,n}$ definiert durch ($\rightarrow$ Abschnitt 2.6)

$$\widetilde{\mathbf{b}}^j = \sum_{i=1}^n s_{i,j}\mathbf{b}^i \ .$$

Dann ist $\mathbf{S}$ invertierbar (regulär) und die Inverse $\mathbf{R} = (r_{i,l})_{i,l=1}^n := \mathbf{S}^{-1}$ erfüllt

$$\mathbf{b}^l = \sum_{j=1}^n r_{j,l}\widetilde{\mathbf{b}}^j \ .$$

Def. 4.4.C. Zwei Matrizen $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n,n}$ heissen ähnlich, wenn es eine invertierbare Matrix $\mathbf{S} \in \mathbb{R}^{n,n}$ so gibt dass

$$\mathbf{B} = \mathbf{S}^{-1}\mathbf{A}\mathbf{S} \ .$$

R. Hiptmair
SAM, ETHZ

4.5 Projektionen

Def. 4.5.A. Eine lineare Selbstabbildung $\mathsf{P} \in \mathcal{L}(V, V)$ heisst Projektion, wenn

$$\mathsf{P}^2 := \mathsf{P} \circ \mathsf{P} = \mathsf{P} \ .$$

Satz 4.5.B. Projektionen lassen jeden Vektor in ihrem Bild unverändert.

Def. 4.5.C. Seien U, W Unterräume von V. Dann ist V die direkte Summe von U und W, in Zeichen $V = U \oplus W$, wenn es zu jedem $\mathbf{v} \in V$ eindeutig bestimmte Vektoren $\mathbf{u} \in U$ und $\mathbf{w} \in W$ so gibt, dass $\mathbf{v} = \mathbf{u} + \mathbf{w}$.
W heisst dann ein Komplement von U in V, und U ein Komplement von W in V.

Korollar 4.5.C. Wenn (mit den Notationen von Satz 4.5.C) $V = U \oplus W$, dann gilt

- $\dim V = \dim U + \dim W$,
- $U \cap W = \{\mathbf{0}\}$.

R. Hiptmair
SAM, ETHZ

Satz 4.5.D. Für eine Projektion $\mathsf{P} \in \mathcal{L}(V,V)$ gilt

$$V = \mathrm{Kern}(\mathsf{P}) \oplus \mathrm{Bild}(\mathsf{P}) \ .$$

Man bezeichnet P dann als Projektion auf $\mathrm{Bild}(\mathsf{P})$ in Richtung von $\mathrm{Kern}(\mathsf{P})$.

Satz 4.5.E. Zu jeder Projektion $\mathsf{P} \in \mathcal{L}(V,V)$ gibt es eine Basis $\mathcal{B}_V$ von V, so dass P die Matrixdarstellung

$$\mathbf{P} = \begin{pmatrix} \mathbf{I} & \mathbf{O} \\ \mathbf{O} & \mathbf{O} \end{pmatrix} \in \mathbb{R}^{n,n}$$

besitzt, wobei $\mathbf{I}$ für die $r \times r$-Einheitsmatrix steht, $r = \mathrm{Rang}(\mathsf{P})$, und $\mathbf{O}$ für Nullmatrizen geeigneter Grösse.

R. Hiptmair
SAM, ETHZ

Für den Rest des Abschnitts sei $(V, +, \cdot, \langle\cdot,\cdot\rangle_V)$ ein Vektorraum mit Skalarprodukt $\langle\cdot,\cdot\rangle_V$.

Verallgemeinerung des Konzepts aus Abschnitt 3.9.2:

Def. 4.5.F. Eine Projektion $\mathsf{Q} \in \mathcal{L}(V,V)$ heisst Orthogonalprojektion, falls $\mathbf{v} - \mathsf{Q}(\mathbf{v})$ für jedes $\mathbf{v} \in V$ orthogonal zu $\mathrm{Bild}(\mathsf{Q})$ ist, d.h., falls

$$\langle \mathbf{x}, \mathbf{v} - \mathsf{Q}(\mathbf{v}) \rangle = 0 \quad \text{für alle } \mathbf{v} \in V \quad \text{und alle } \mathbf{x} \in \mathrm{Bild}(\mathsf{Q}) \ .$$

Satz 4.5.H. Zu jedem Unterraum $U \subset V$ gibt es eine eindeutig bestimmte Orthogonalprojektion $\mathsf{Q}_U \in \mathcal{L}(V,V)$ so, dass

$$\mathrm{Bild}(\mathsf{Q}_U) = U \ .$$

Sie heisst die orthogonale Projektion auf U.

R. Hiptmair
SAM, ETHZ

4.6 Isometrien

Link zu den handgeschriebenen Präsentationsfolien

Seien nun $(V, +, \cdot, \langle \cdot, \cdot \rangle_V)$, $(W, +, \cdot, \langle \cdot, \cdot \rangle_W)$ Vektorräume mit Skalarprodukten $\langle \cdot, \cdot \rangle_V$, $\langle \cdot, \cdot \rangle_W$. Sie induzieren Normen $\|\cdot\|_V$ und $\|\cdot\|_W$, siehe Abschnitt 1.5.

4.6.1 Lägenerhaltung

Def. 4.6.1.A. Eine lineare Abbildung $\mathsf{Q} \in \mathcal{L}(V, W)$ heisst Isometrie oder längenerhaltend, wenn

$$\|\mathsf{Q}(\mathbf{v})\|_W = \|\mathbf{v}\|_V \quad \text{für alle } \mathbf{v} \in V \ .$$

Satz 4.6.1.B. Für jede Isometrie $\mathsf{Q} \in \mathcal{L}(V, W)$ gilt, dass $\mathrm{Kern}(\mathsf{Q}) = \{0\}$.

Satz 4.6.1.C. Genau die längenerhaltenden linearen Abbildungen erhalten das Skalarprodukt, d.h. für jede Isometrie $\mathsf{Q} \in \mathcal{L}(V, W)$ gilt

$$\langle \mathsf{Q}\mathbf{x}, \mathsf{Q}\mathbf{y} \rangle_W = \langle \mathbf{x}, \mathbf{y} \rangle_V \quad \text{für alle } \mathbf{x}, \mathbf{y} \in V \ .$$

Längenerhaltende lineare Abbildungen sind auch winkelerhaltend (konform)!

4.6.2 Orthogonale Matrizen

Nun: $V = W = \mathbb{R}^n$ mit Euklidischem Skalarprodukt $\langle \cdot, \cdot \rangle$, siehe Def. 1.4.B.

Def. 4.6.2.A Eine Matrix $\mathbf{Q} \in \mathbb{R}^{n,n}$ heisst orthogonal, wenn $\mathbf{Q}^T = \mathbf{Q}^{-1}$

Satz 4.6.2.B. $\mathbf{Q} \in \mathbb{R}^{n,n}$ ist genau dann orthogonal, wenn die Abbildung $\mathbf{x} \to \mathbf{Q}\mathbf{x} \in \mathcal{L}(\mathbb{R}^n, \mathbb{R}^n)$ eine Isometrie ist.

R. Hiptmair
SAM, ETHZ

4.6.3 Spiegelungen

Def. 4.6.3.A. Sei $n := \dim V$, U ein Unterraum von V und P_U die orthogonale Projektion auf U. Die lineare Abbildung

$$\mathsf{R}_U := 2\mathsf{P}_U - \mathsf{Id} \in \mathcal{L}(V, V)$$

ist die Spiegelung an U.

Satz 4.6.3.B. Eine Spiegelung ist eine Isometrie.

4.6.4 Drehungen

Nun sei $V = \mathbb{R}^d$, $d = 2, 3$, versehen mit dem Euklidischen Skalarprodukt.

4.6.4.1 Drehungen im $\mathbb{R}^2$

R. Hiptmair
SAM, ETHZ

Def. 4.6.4.1.A. Eine Drehung im $\mathbb{R}^2$ ist jede lineare Abbildung, die bezüglich der Basis $\left\{ \binom{1}{0}, \binom{0}{1} \right\}$ durch eine Matrix der Form

$$\mathbf{D} := \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix} , \quad \varphi \in [0, 2\pi[.$$

beschrieben wird. Dabei heisst φ der Winkel der Drehung.

4.6.4.2 Drehungen im $\mathbb{R}^3$

Def. 4.6.4.2.B. Eine Drehung im $\mathbb{R}^3$ ist eine lineare Abbildung $\mathbb{R}^3 \mapsto \mathbb{R}^3$ zur der es eine Basis $\{\mathbf{n}^1, \mathbf{n}^2, \mathbf{n}^3\}$ des $\mathbb{R}^3$ aus paarweise orthogonalen und normierten Vektoren, d.h.

$$\left\langle \mathbf{n}^1, \mathbf{n}^2 \right\rangle = \left\langle \mathbf{n}^1, \mathbf{n}^3 \right\rangle = \left\langle \mathbf{n}^2, \mathbf{n}^3 \right\rangle = 0 \ ,$$
$$\left\| \mathbf{n}^1 \right\| = \left\| \mathbf{n}^2 \right\| = \left\| \mathbf{n}^3 \right\| = 1 \ ,$$

so gibt dass ihre Matrixdarstellung bzgl. dieser Basis die Gestalt

$$\mathbf{D} := \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \sin\varphi & \cos\varphi \end{pmatrix} \quad , \quad \text{mit} \quad \varphi \in [0, 2\pi[\ ,$$

hat. Dann heisst φ der Drehwinkel und die Gerade $\mathrm{Span}\{\mathbf{n}^1\}$ ist die Drehachse.

4.6.5 Orthonormalbasen

Sei nun $(V, +, \dot{})$ ein Vektorraum der Dimension n mit Skalarprodukt $\langle \cdot, \cdot \rangle$.

R. Hiptmair
SAM, ETHZ

Def. 4.6.5.A. Eine Basis $\{\mathbf{n}^1, \ldots, \mathbf{n}^n\}$ von V heisst Orthonormalbasis (ONB), falls

$$\left\langle \mathbf{n}^i, \mathbf{n}^j \right\rangle = \begin{cases} 0 & \text{, falls } i \neq j \text{ ,} \\ 1 & \text{, falls } i = j \text{ ,} \end{cases} \qquad i, j \in \{1, \ldots, n\} \text{ .}$$

Satz 4.6.5.B. Ist $\mathcal{B} := \{\mathbf{n}^1, \ldots, \mathbf{n}^n\}$ eine Orthonormalbasis von V, dann ist die Koordinatenabbildung $I_{\mathcal{B}} : V \to \mathbb{R}^n$,

$$I_{\mathcal{B}} : \begin{cases} V & \to \mathbb{R}^n \\ \mathbf{v} = \sum\limits_{j=1}^{n} c_j \mathbf{n}^j & \mapsto \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix} \end{cases}$$

eine Isometrie (, wenn im $\mathbb{R}^n$ das Euklidische Skalarprodukt verwendet wird).

Satz 4.6.5.E. Eine Matrix $\mathbf{Q} \in \mathbb{R}^{n,n}$ ist genau dann orthogonal, wenn ihre Spalten (oder transponierten Zeilen) eine Orthonormalbasis des $\mathbb{R}^n$ bilden.

Satz 4.6.5.C. Ist $\mathbf{Q} \in \mathcal{L}(V, V)$ eine Isometrie und $\{\mathbf{n}^1, \ldots, \mathbf{n}^n\}$ eine Orthonormalbasis von V, dann ist auch $\{\mathbf{Q}\mathbf{n}^1, \ldots, \mathbf{Q}\mathbf{n}^n\}$ eine Orthonormalbasis.

R. Hiptmair
SAM, ETHZ

Satz 4.6.5.D. Zu jeder Basis $\{\mathbf{b}^1, \ldots, \mathbf{b}^n\}$ von V gibt eine Orthonormalbasis $\{\mathbf{q}^1, \ldots, \mathbf{q}^n\}$ so, dass

$$\mathrm{Span}\{\mathbf{q}^1, \ldots, \mathbf{q}^k\} = \mathrm{Span}\{\mathbf{b}^1, \ldots, \mathbf{b}^k\} \quad \text{für alle} \quad k = 1, \ldots, n\,.$$

Code 4.6.1: Gram-Schmidt-Orthonormalisierung der Spaltenvektoren einer Matrix

```
function Q = gramschmidt(B)
% Gram-Schmidt orthonormalization of column vectors of matrix B.
% The columns of B are supposed to be linearly independent!
m = size(B,2);
% Build output matrix with orthonormal columns by successively adding columns
% to the matrix N from the right.

% First normalized vector is just the normalized first column of B
Q = B(:,1)/norm(B(:,1));
for l=2:m
  % This is the trick: use the orthonormal system that has already been built
    for
  % efficient implementation of orthogonal projection. First compute the
  % coefficient vector c of the orthogonal projection of the l-th
  % column of B onto the space spanned by the orthonormal vectors
  % constructed so far. This space agrees with the span of the first l-1
    columns
  % of B.
```

```
17    c = (B(:,l)'*Q)',
18    % The difference of the l-th column vector of B and its orthogonal
19    % projection given by N*c determines the direction n of the l-th
20    % orthonormal vector. Note that Q*c computes the orthogonal
21    % projection of the l-th column of B onto the space spanned
22    % by the first l-1 columns of Q.
23    n = B(:,l)-Q*c;
24    % It remains to normalize it.
25    Q = [Q, n/norm(n)];
26  end
```

Orthogonalisieren einer Basis des $\mathbb{R}^n$ in MATLAB mittels QR-Zerlegung: `[Q,R] = qr(B)`

Code 4.6.2: Gram-Schmidt-Orthonormalisierung der Spaltenvektoren mittels QR-Zerlegung

```
1  function Q = gramschmidtbyqr(B)
2  % Gram-Schmidt orthonormalization of column vectors of matrix B by
3  % using the QR-decompoisition provided by MATLAB.
4  % Note that the orthonormal vectors are unique only up to sign so that this
5  % function and the function gramschmidt may not produce the same
6  % output.
7  [Q,R] = qr(B);
```

Code 4.6.3: Berechnung der Matrixdarstellung einer Rotation

R. Hiptmair
SAM, ETHZ

```
function M = rotmatrix(a,phi)
% MATLAB function computing the matrix representation of a rotation about the
% axis in direction of the vector a ∈ ℝ^3 with angle φ.
% Note: Right hand rule should be applied to fix the sense of the rotation, but,
% for the sake of simplicity, this is not done in this code.
%
% We know that rotations have a canonical matrix representation with respect to
% an orthonormal basis of ℝ^3 with first vector in the direction of
% a. Thus the matrix representation of the rotation with respect to
% the standard Cartesian basis of unit vectors can be computed by using the
% formula (4.4.B) for the transformation of the matrix representation under a
% change of basis. Here the "old basis" is the special orthonormal basis, the
% "new basis" is the Cartesian basis. Hence the matrix S of
% formula (4.4.B) contains the coefficients of the Cartesian basis vectors with
% respect to the special orthonormal basis vectors. Since both bases are
% orthonormal, the transformation matrix will be orthogonal.

% First compute a special orthonormal basis with the normalized vector
% a as first element. This can be done by means of partial QR-decomposition
% of the 3×1-"matrix" a.
[Q,R] = qr(a);
% Note that the Q obtained thus is the inverse of the basis
% transformation matrix S introduced above. However, this is not a
% big deal, because S = Q^{-1} = Q^T for the orthogonal matrix
```


R. Hiptmair
SAM, ETHZ

```
% Q!

% This is the matrix representation of the rotation with respect to the
% specical orhtoonormal basis
c = cos(phi); s = sin(phi);
D = [1 , 0  , 0  ;...
     0 , c , -s  ;...
     0 , s , c  ];
% Finally compute the transformation of the matrix representation according to
% Formula (4.4.B). Note that a prime in MATLAB denotes transposition.
M = Q*D*Q';
```


R. Hiptmair
SAM, ETHZ

4.7 Affine Abbildungen

$(V, +, \cdot) \hat{=}$ n-dimensionaler Vektorraum.

Def. 4.7.A. Zu jedem Vektor $\mathbf{t} \in V$ heisst die Abbildung

$$\mathsf{T}_{\mathbf{t}} : \begin{cases} V & \to & V \\ \mathbf{x} & \mapsto & \mathbf{x} + \mathbf{t} \end{cases}$$

eine Translation (Verschiebung) um $\mathbf{t}$.

Def. 4.7.B. Jede Abbildung der Form $\mathsf{A} := \mathsf{T} \circ \mathsf{L}$, die sich als Hintereinanderausführung einer linearen Selbstabbildung $\mathsf{L} \in \mathcal{L}(V, V)$ von V und einer Translation T in V schreiben lässt, heisst eine affine (Selbst)-Abbildung von V.

Satz 4.7.C.[Hintereinanderausführung affiner Abbildungen]
Die Komposition affiner Selbstabbildungen von V ist wieder eine affine Selbstabbildung von V.

Satz 4.7.D. [Matrixdarstellung affiner Abbildungen]
Sei $\mathcal{B}_V$ eine Basis von V und A eine affine Selbstabbildung von V.
Dann gibt es eine Matrix $\mathbf{A} \in \mathbb{R}^{n,n}$ und einen Vektor $\mathbf{t} \in \mathbb{R}^n$ so, dass

$$I_{\mathcal{B}_V}(\mathsf{A}(\mathbf{v})) = \mathbf{A} I_{\mathcal{B}_V}(\mathbf{v}) + \mathbf{t} \quad \text{für alle } \mathbf{v} \in V \ .$$

R. Hiptmair
SAM, ETHZ

Satz 4.7.E. Sei $X \subset V$ ein affiner Teilraum von V und A eine affine Selbstabbildung von V. Dann ist auch $\mathsf{A}(X) \subset V$ ein affiner Teilraum von V.

Satz 4.7.F. Das Bild einer konvexen Teilmenge von V unter einer affinen Selbstabbildung ist wieder eine konvexe Teilmenge von V.

R. Hiptmair
SAM, ETHZ

5 Diagonalisierung

5.1 Matrixdiagonalisierung und Anwendungen

Link zu den handgeschriebenen Präsentationsfolien

Def. 5.1.A Eine Matrix $\mathbf{A} \in \mathbb{R}^{n,n}$, $n \in \mathbb{N}$, heisst diagonalisierbar, wenn sie zu einer Diagonalmatrix ähnlich ($\rightarrow$ Def. 4.4.C) ist, das heisst, wenn es eine invertierbare Matrix $\mathbf{S} \in \mathbb{R}^{n,n}$ so gibt, dass

$$\mathbf{S}^{-1}\mathbf{A}\mathbf{S} = \mathbf{D} \quad \text{mit} \quad \mathbf{D} = \begin{pmatrix} \lambda_1 & 0 & \dots & & \dots & 0 \\ 0 & \lambda_2 & 0 & \dots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & \ddots \\ \vdots & & & & \ddots & 0 \\ 0 & \dots & & \dots & 0 & \lambda_n \end{pmatrix}, \quad \lambda_i \in \mathbb{R}, \quad i = 1, \dots, n .$$

R. Hiptmair
SAM, ETHZ

✎ Notation: $\mathbf{D} = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$

Satz 5.1.C. Gilt für $\mathbf{A}, \mathbf{S} \in \mathbb{R}^{n,n}$, dass $\mathbf{S}^{-1}\mathbf{A}\mathbf{S} = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$, $\lambda_i \in \mathbb{R}$, dann hat das folgende Anfangswert für ein autonomes lineares Differentialgleichungssystem

$$\mathbf{y}'(x) := \begin{pmatrix} y_1'(x) \\ \vdots \\ \\ \vdots \\ y_n'(x) \end{pmatrix} = \mathbf{A} \begin{pmatrix} y_1(x) \\ \vdots \\ \\ \vdots \\ y_n(x) \end{pmatrix}, \quad \begin{pmatrix} y_1(0) \\ \vdots \\ \\ \vdots \\ y_n(0) \end{pmatrix} = \mathbf{y}_0 \in \mathbb{R}^n ,$$

die eindeutig Lösung

$$\mathbf{y}(x) = \mathbf{S} \operatorname{diag}(e^{\lambda_1 x}, \ldots, e^{\lambda_n x}) \mathbf{S}^{-1} \mathbf{y}_0 , \quad x \in \mathbb{R} .$$

Eine quadratische Matrix $\mathbf{A} \in \mathbb{R}^{n,n}$ kann in ein Polynom $p(t) = \alpha_k t^k + \alpha_{k-1} t^{k-1} + \cdots + \alpha_1 t + \alpha_0$ eingesetzt werden:

$$p(\mathbf{A}) := \alpha_k \mathbf{A}^k + \alpha_{k-1} \mathbf{A}^{k-1} + \cdots + \alpha_1 \mathbf{A} + \alpha_0 \mathbf{I} . \tag{5.1.D}$$

R. Hiptmair
SAM, ETHZ

Satz 5.1.E. Gilt für $\mathbf{A}, \mathbf{S} \in \mathbb{R}^{n,n}$, dass $\mathbf{S}^{-1}\mathbf{A}\mathbf{S} = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$, $\lambda_i \in \mathbb{R}$, dann gilt für jedes Polynom $p(t)$

$$p(\mathbf{A}) = \mathbf{S}\operatorname{diag}(p(\lambda_1), \ldots, p(\lambda_n))\mathbf{S}^{-1} \ .$$

5.2 Determinanten

Link zu den handgeschriebenen Präsentationsfolien

$(V, +, \cdot) \mathrel{\hat{=}}$ Vektorraum mit Dimension $n \in \mathbb{N}$.

Def. 5.2.A Eine Determinante ist eine lineare Abbildung

$$\det : \underbrace{V \times \cdots \times V}_{n \text{ mal}} \to \mathbb{R} \ ,$$

(d.h. $\det$ nimmt n verschiedene Vektoren als Argumente) für die gilt

1. $\det \neq 0$, das heisst es gibt Vektoren $\mathbf{v}^1, \ldots, \mathbf{v}^n \in V$ so dass $\det(\mathbf{v}^1, \ldots, \mathbf{v}^b) \neq 0$.
2. für jedes $k \in \{1, \ldots, n\}$ und beliebige Vektoren $\mathbf{w}^1, \ldots, \mathbf{w}^n \in V$ ist

$$\begin{cases} V & \to & \mathbb{R} \\ \mathbf{v} & \mapsto & \det(\mathbf{w}^1, \ldots, \mathbf{w}^{k-1}, \mathbf{v}, \mathbf{w}^{k+1}, \ldots, \mathbf{w}^n) \end{cases}$$

   eine lineare Abbildung.
3. Gilt für $\mathbf{v}^1, \ldots, \mathbf{v}^n \in V$, dass $\mathbf{v}^i = \mathbf{v}^j$ für irgendwelche zwei Indices $i, j \in \{1, \ldots, n\}$, $i \neq j$, dann folgt $\det(\mathbf{v}^1, \ldots, \mathbf{v}^n) = 0$.

Zu 1. sagt man “$\det$ ist nichttrivial”

Zu 2. sagt man “$\det$ ist linear in jedem Argument”

R. Hiptmair
SAM, ETHZ

Satz 5.2.B. Eine Determinante auf V erfüllt

$$\det(\mathbf{w}^1, \ldots, \mathbf{w}^{k-1}, \mathbf{w}^k + \alpha\mathbf{w}^i, \mathbf{w}^{k+1}, \ldots, \mathbf{w}^n) = \det(\mathbf{w}^1, \ldots, \mathbf{w}^{k-1}, \mathbf{w}^k, \mathbf{w}^{k+1}, \ldots, \mathbf{w}^n)$$

für beliebige $i, k \in \{1, \ldots, n\}$, $i \neq k$, beliebige Vektoren $\mathbf{w}^1, \ldots, \mathbf{w}^n \in V$ und alle $\alpha \in \mathbb{R}$.

Satz 5.2.C. Beim Vertauschen zweier Argumente einer Determinante wechselt deren Vorzeichen.

Für uns das wichtigste Resultat: Determinanten als “Sonden für lineare Abh|’angigkeit”

Satz 5.2.D. Für beliebige Vektoren $\mathbf{v}^1, \ldots, \mathbf{v}^n \in V$ gilt

$$\{\mathbf{v}^1, \ldots, \mathbf{v}^n\} \quad \text{linear unabhängig} \quad \Leftrightarrow \quad \det(\mathbf{v}^1, \ldots, \mathbf{v}^n) \neq 0 \, .$$

Satz 5.2.E. Eine Determinante auf V ist eindeutig durch ihre Wirkung auf eine (einzige) Basis von V festgelegt.

Def. 5.2.F Die (Standard-)Determinante auf $\mathbb{R}^n$ ist die Determinanten auf dem Vektorraum $\mathbb{R}^n$, für die

$$\det\left(\begin{pmatrix}1\\0\\\vdots\\0\end{pmatrix},\begin{pmatrix}0\\1\\\vdots\\0\end{pmatrix},\ldots,\begin{pmatrix}0\\\vdots\\1\\0\end{pmatrix},\begin{pmatrix}0\\\vdots\\0\\1\end{pmatrix}\right)=1\,.$$

Def. 5.2.G Die Determinante einer Matrix $\mathbf{A}\in\mathbb{R}^{n,n}$, in Zeichen $\det(\mathbf{A})$, ist die Determinante ihrer Spalten, aufgefasst als Vektoren im $\mathbb{R}^n$.

Satz 5.2.H. Für alle $\mathbf{A}\in\mathbb{R}^{n,n}$ gilt: $\det(\mathbf{A})=\det(\mathbf{A}^T)$

Numerische Determinantenberechnung in MATLAB: Funkion `det(A)`

Satz 5.2.I. Eine Matrix $\mathbf{A}\in\mathbb{R}^n$ ist genau dann invertierbar, wenn $\det(\mathbf{A})\neq 0$.

Satz 5.2.J. [Determinantenmultiplikationssatz]
Für beliebige Matrizen $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n,n}$ gilt

$$\det(\mathbf{AB}) = \det(\mathbf{A}) \det(\mathbf{B}) \ .$$

Satz 5.2.K. Ist $\mathbf{Q} \in \mathbb{R}^{n,n}$ orthogonal, so gilt $\det(\mathbf{Q}) \in \{1, -1\}$.

Satz 5.2.M.
Ähnliche Matrizen ($\rightarrow$ Def. 4.4.C) haben gleiche Determinante.

R. Hiptmair
SAM, ETHZ

"Determinante einer linearen Selbstabbildung eines Vektorraums" ist ein sinnvolles Konzept.

Def. 5.2.O Die Determinante $\det(\mathsf{L})$ einer linearen Selbstabbildung $\mathsf{L} \in \mathcal{L}(V, V)$ eines Vektorraums V ist die Determinante einer beliebigen (!) Matrixdarstellung.

✎ Notation: $\mathrm{vol}(M) \hat{=}$ Fläche/Volumen einer "Figur" $M \subset \mathbb{R}^n$

Satz 5.2.P. Für $M \subset \mathbb{R}^n$ und eine lineare Selbstabbildung $\mathsf{L} \in \mathcal{L}(\mathbb{R}^n, \mathbb{R}^n)$ gilt

$$\mathrm{vol}(\mathsf{L}(M)) = |\det(\mathsf{L})| \cdot \mathrm{vol}(M) \ ,$$

das heisst, die Determinante gibt die durch eine lineare Selbstabbildung des $\mathbb{R}^n$ bewirkte relative Volumenänderung an.

5.3 Rechnen in $\mathbb{C}^n$

Link zu den handgeschriebenen Präsentationsfolien

Man kann Vektorräume über den komplexen Zahlen $\mathbb{C}$ ("komplexe Vektorräume") ganz analog zu den reellen Vektorräumen definieren, die in Abschnitt 1.2 eingeführt wurden: die Skalarmultiplikation wird einfach für komplexe Zahlen definiert, wobei alle Rechenregeln (VR1)–(VR8) völlig unverändert bleiben.

$\mathbb{C}^n$ mit komponentenweiser Addition $+$ und komponentenweise Multiplikation $\cdot$ mit komplexen Skalaren ist ein Vektorraum über $\mathbb{C}$. Die komplexen Matrizen $\in \mathbb{C}^{n,n}$ beschreiben die linearen Selbstabbildungen des $\mathbb{C}^n$. Dabei überträgt sich die Definition einer linearen Abbildung unverändert, wenn man nur in (L2) komplexe Skalare $\alpha \in \mathbb{C}$ zulässt.

Im $\mathbb{C}^n$ und mit komplexen Matrizen kann man (fast immer) so rechnen wie im $\mathbb{R}^n$, wenn man nur die Addition und Multiplikation in $\mathbb{C}$ benutzt.

Ausnahmen:

- Das Euklidische Skalarprodukt in $\mathbb{C}^n$ ist gegeben durch

$$\langle \mathbf{v}, \mathbf{w} \rangle = \sum_{j=1}^{n} v_j \overline{w}_j \;, \quad \mathbf{v}, \mathbf{w} \in \mathbb{C}^n \;. \tag{5.3.A}$$

Dabei bezeichnet $\overline{}$ die komplexe Konjugation.

Konjugation bei Vertauschen der Vektoren! $\langle \mathbf{v}, \mathbf{w} \rangle = \overline{\langle \mathbf{w}, \mathbf{v} \rangle}$.

Zwei Vektoren $\mathbf{v}, \mathbf{w} \in \mathbb{C}^n$ mit $\langle \mathbf{v}, \mathbf{w} \rangle = 0$ heissen weiterhin orthogonal.

Definition der Norm eines Vektors $\mathbf{v} = (v_1, \ldots, v_n)^T \in \mathbb{C}^n$ wie in Def. 1.5.A:

$$\|\mathbf{v}\| := \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle} = \sqrt{\sum_{j=1}^{n} |v_j|^2} \qquad \text{(Beachte: } \langle \mathbf{v}, \mathbf{v} \rangle \in \mathbb{R}_0^+ \text{)} \;.$$

$$\Rightarrow \quad \|\lambda \mathbf{v}\| = |\lambda| \, \|\mathbf{v}\| \quad \text{für alle} \quad \mathbf{v} \in \mathbb{C}^n, \, \lambda \in \mathbb{C} \;.$$

R. Hiptmair
SAM, ETHZ

- Anstelle des Transponierten einer Matrix verwende das konjugiert Transponierte, kenntlich gemacht durch Superskript ‘H‘:

$$\mathbf{A} = \begin{pmatrix} a_{11} & \dots & a_{1m} \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nm} \end{pmatrix} \in \mathbb{C}^{n,m} \quad \Rightarrow \quad \mathbf{A}^H := \begin{pmatrix} \bar{a}_{11} & \dots & \bar{a}_{n1} \\ \vdots & & \vdots \\ \bar{a}_{1m} & \dots & \bar{a}_{mn} \end{pmatrix} \in \mathbb{C}^{m,n} .$$

Diese Definition ist motiviert durch die Formel:

$$\langle \mathbf{Av}, \mathbf{w} \rangle = \left\langle \mathbf{v}, \mathbf{A}^H \mathbf{w} \right\rangle \quad \text{für alle} \quad \mathbf{A} \in \mathbb{C}^{n,m} , \quad \mathbf{v} \in \mathbb{C}^m, \mathbf{w} \in \mathbb{C}^m . \tag{5.3.B}$$

- Der Begriff der orthogonalen Matrix ($\rightarrow$ Def. 4.6.2.A) wird ersetzt durch den Begriff der unitären Matrix.

Def. 5.3.C Eine Matrix $\mathbf{U} \in \mathbb{C}^{n,n}$ heisst unitär, wenn $\mathbf{U}^{-1} = \mathbf{U}^H$.

Wegen (5.3.B) beschreiben unitäre Matrizen Isometrien im $\mathbb{C}^n$ versehen mit dem Euklidischen Skalarprodukt (5.3.A).

5.4 Eigenvektoren und Eigenwerte

Def. 5.4.A Es sei $\mathbf{A} \in \mathbb{C}^{n,n}$, $n \in \mathbb{N}$.

- Eine Zahl $\lambda \in \mathbb{C}$ heisst Eigenwert (EW) der Matrix $\mathbf{A}$, wenn $\mathrm{Kern}(\mathbf{A} - \lambda\mathbf{I}) \neq \{\mathbf{0}\}$.
- Ein $\mathbf{v} \in \mathbb{C}^n$, $\mathbf{v} \neq 0$, heisst Eigenvektor (EV) der Matrix $\mathbf{A}$ zum Eigenwert $\lambda \in \mathbb{C}$, wenn

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v} \quad \Leftrightarrow \quad \mathbf{v} \in \mathrm{Kern}(\mathbf{A} - \lambda\mathbf{I}) \ .$$

✎ Notation: $\mathrm{EW}(\mathbf{A}) := \{\lambda \in \mathbb{C}\colon \lambda \text{ ist Eigenwert von } \mathbf{A}\}$

R. Hiptmair
SAM, ETHZ

Satz 5.4.B. Eine Matrix $\mathbf{A} \in \mathbb{C}^n$ ist genau dann diagonalisierbar, wenn es eine Basis des $\mathbb{C}^n$ aus Eigenvektoren von $\mathbf{A}$ gibt.

Satz 5.4.C. [Diagonalisierbarkeit und Eigenwerte von Projektionen]
Erfüllt $\mathbf{P} \in \mathbb{C}^{n,n}$, dass $\mathbf{P}^2 = \mathbf{P}$, dann ist $\mathbf{P}$ diagonalisierbar und $\mathrm{EW}(\mathbf{P}) \subset \{0, 1\}$.

Satz 5.4.D. [Eigenwerte sind Nullstellen des charakteristischen Polynoms]
Mit dem charakeristischen Polynom $\chi_{\mathbf{A}}(\lambda) := \det(\mathbf{A} - \lambda\mathbf{I})$ gilt

$$\mathrm{EW}(\mathbf{A}) = \{\lambda \in \mathbb{C}\colon \chi_{\mathbf{A}}(\lambda) = 0\} \ .$$

Link zu den handgeschriebenen Präsentationsfolien mit Beispielen

Satz 5.4.E. Die charakteristischen Polynome ähnlicher Matrizen ($\rightarrow$ Def. 4.4.C) sind gleich: Für beliebige $\mathbf{A} \in \mathbb{C}^{n,n}$ und invertierbare $\mathbf{S} \in \mathbb{C}^{n,n}$ gilt

$$\chi_{\mathbf{S}^{-1}\mathbf{A}\mathbf{S}} = \chi_{\mathbf{A}} \ .$$

Das “charakteristische Polynom einer linearen Abbildung” ist ein sinnvolles Konzept.

$$\mathrm{EW}\left(\begin{array}{c|c} \mathbf{B} & * \\ \hline \mathbf{0} & \mathbf{C} \end{array}\right) = \mathrm{EW}(\mathbf{B}) \cup \mathrm{EW}(\mathbf{C}) \quad \text{für } \mathbf{B} \in \mathbb{C}^{k,k},\ \mathbf{C} \in \mathbb{C}^{n-k,n-k} \ . \qquad \text{(5.4.F)}$$

R. Hiptmair
SAM, ETHZ

Satz 5.4.G. $\overline{\mathrm{EW}(\mathbf{A})} = \mathrm{EW}(\mathbf{A}^H)$

Satz 5.4.H. Für eine Matrix $\mathbf{A} \in \mathbb{C}^{n,n}$ seien $\mathbf{v}^1, \ldots, \mathbf{v}^k$, $k \in \{1, \ldots, n\}$, Eigenvektoren zu paarweise verschiedenen Eigenwerten. Dann ist $\{\mathbf{v}^1, \ldots, \mathbf{v}^k\}$ linear unabhängig.

Satz 5.4.I. Hat $\mathbf{A} \in \mathbb{C}^{n,n}$ n verschiedene Eigenwerte, dann ist $\mathbf{A}$ diagonalisierbar.

Satz 5.4.J. Gilt für eine Matrix $\mathbf{A} \in \mathbb{C}^{n,n}$ mit Eigenwerten $\mathrm{EW}(\mathbf{A}) = \{\lambda_1, \ldots, \lambda_k\} \subset \mathbb{C}$, $1 \leq k \leq n$, dass

$$\sum_{l=1}^{k} \dim \mathrm{Kern}(\mathbf{A} - \lambda_l \mathbf{I}) = n \ ,$$

dann ist $\mathbf{A}$ diagonalisierbar.

R. Hiptmair
SAM, ETHZ

5.5 Diagonalisierbarkeit

Eines der wichtigsten Resultate der linearen Algebra:

Satz 5.5.A. [Lemma von Schur]
Zu jeder Matrix $\mathbf{A} \in \mathbb{C}^{n,n}$ gibt es eine unitäre Matrix $\mathbf{U} \in \mathbb{C}^{n,n}$ so, dass $\mathbf{U}^H \mathbf{A}\mathbf{U}$ eine (obere) Dreiecksmatrix ist. Deren Diagonaleinträge sind die Eigenwerte von $\mathbf{A}$.

Hilfsmittel für den Beweis:

Satz 5.4.B. [Fundamentalsatz der Algebra]
Jedes nichtkonstante Polynom $p(z) := \alpha_k z^k + \alpha_{k-1} z^{k-1} + \cdots + \alpha_1 z + \alpha_0$, $\alpha_k \in \mathbb{C}$, hat mindestens eine Nullstelle $z_0 \in \mathbb{C}$ mit $p(z_0) = 0$.

R. Hiptmair
SAM, ETHZ

Satz 5.4.C. [Diagonalisierbarkeit von Matrizen, die mit ihrer konjugiert Transponierten vertauschbar sind]
Gilt für $\mathbf{A} \in \mathbb{C}^{n,n}$, dass $\mathbf{A}\mathbf{A}^H = \mathbf{A}^H\mathbf{A}$, dann gibt es eine unitäre Matrix $\mathbf{U} \in \mathbb{C}^{n,n}$ so, dass

$$\mathbf{U}^H\mathbf{A}\mathbf{U} = \mathbf{D} := \operatorname{diag}(\lambda_1, \ldots, \lambda_n) \in \mathbb{C}^{n,n} .$$

Die $\lambda_k \in \mathbb{C}$ sind die Eigenwerte von $\mathbf{A}$, die Spalten von $\mathbf{U}$ die zugehörigen paarweise orthogonalen Eigenvektoren.

Spezielle Matrizen, die mit ihrer konjugiert Transponierten vertauschbar sind:

- Symmetrische Matrizen (Hermitesche Matrizen): $\mathbf{A} \in \mathbb{C}^{n,n}$ mit $\mathbf{A}^H = \mathbf{A}$
- Schiefsymmetrische Matrizen: $\mathbf{A} \in \mathbb{C}^{n,n}$ mit $\mathbf{A}^T = -\mathbf{A}$
- Orthogonale Matrizen: $\mathbf{Q} \in \mathbb{R}^{n,n}$ mit $\mathbf{Q}^T = \mathbf{Q}^{-1}$
- Unitäre Matrizen: $\mathbf{U} \in \mathbb{C}^{n,n}$ mit $\mathbf{U}^H = \mathbf{U}^{-1}$

Korrektur einer Umformung aus der Vorlesung: Zu zeigen ist, dass alle Eigenwerte einer symmetrischen Matrix reell sind:

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v} \quad \text{mit Eigenvktor } \mathbf{v} \neq 0\,, \text{ Eigenwert } \lambda \in \mathbb{C}$$

$$\blacktriangleright \quad \langle \mathbf{Av}, \mathbf{v} \rangle = \langle \lambda \mathbf{v}, \mathbf{v} \rangle = \lambda \langle \mathbf{v}, \mathbf{v} \rangle \quad \Rightarrow \quad \lambda = \frac{\langle \mathbf{Av}, \mathbf{v} \rangle}{\|\mathbf{v}\|^2}$$

$$\blacktriangleright \quad \lambda = \frac{\left\langle \mathbf{v}, \mathbf{A}^H \mathbf{v} \right\rangle}{\|\mathbf{v}\|^2} = \frac{\langle \mathbf{v}, \mathbf{Av} \rangle}{\|\mathbf{v}\|^2} = \frac{\overline{\langle \mathbf{Av}, \mathbf{v} \rangle}}{\|\mathbf{v}\|^2} = \overline{\lambda} \quad \Rightarrow \quad \lambda \in \mathbb{R} \, .$$

Satz 5.4.D. Zu jeder symmetrischen Matrix $\mathbf{A} \in \mathbb{R}^{n,n}$ gibt es eine orthogonale Matrix $\mathbf{Q} \in \mathbb{R}^{n,n}$ so, dass

$$\mathbf{Q}^T \mathbf{AQ} = \mathbf{D} = \operatorname{diag}(\lambda_1, \ldots, \lambda_n) \in \mathbb{R}^{n,n} \, .$$

Die Spalten von $\mathbf{Q}$ bilden eine Orthonormalbasis des $\mathbb{R}^n$ aus Eigenvektoren von $\mathbf{A}$

Satz 5.4.E. Für die Eigenwerte einer orthogonalen Matrix $\mathbf{Q} \in \mathbb{R}^{n,n}$ gilt

$$\mathrm{EW}(\mathbf{Q}) \subset \{\lambda \in \mathbb{C} \colon |\lambda| = 1\} \, .$$

Eigenwerte und Eigenvektoren von $\mathbf{A} \in \mathbb{C}^{n,n}$ in MATLAB:

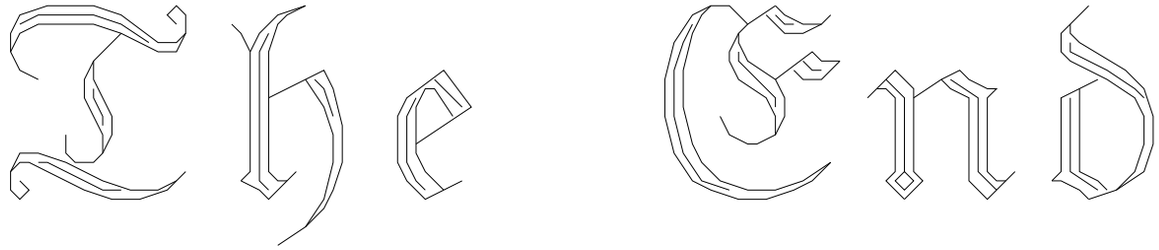
```
lambda = eig(A),
```

```
[S,D] = eig(A)
```


- `lambda`: Spaltenvektor $\in \mathbb{C}^n$, der die Eigenwerte von $\mathbf{A}$ enthält

  `S`: $n \times n$-Matrix mit (normierten) Eigenvektoren

  `D`: $n \times n$-Diagonalmatrix mit Eigenwerten auf der Diagonalen



R. Hiptmair
SAM, ETHZ

Literaturverzeichnis

[1] K. Nipp and D. Stoffer. Lineare Algebra. vdf Hochschulverlag, Zürich, 5 edition, 2002.

[2] M.B. Sayir, J. Dual, and S. Kaufmann. Ingenieurmathematik. Vieweg Studium. Vieweg+Teubner, 2nd edition, 2008.

[3] G. Strang. Lineare Algebra. Springer, 2003.