# Blatt 8

## Aufgabe 30

```
> restart:
> with(LinearAlgebra):

> A := <<1|1|4>,<1|-3|2>,<1|2|-1>>;
> B := <<7|8|9>, <4|5|6>, <1|2|3>>;
> b := <-1, 0, 1>;
> Rank(A);
> # Die Matrix A hat Rang 3. Daher ist das Gleichungssystem Ax=b
  eindeutig loesbar, denn es gibt genau eine Loesung.

> Rank(B);
> # Die Matrix B hat Rang 2. Daher besitzt das Gleichungssystem
  Bx=b mehrere oder keine Loesungen.

> # Zeilenstufenformen der Matrizen <A|b>,<B|b>
> G1 := <A|b>;
> G2 := <B|b>;
> ZA := ReducedRowEchelonForm(G1);
> ZB := ReducedRowEchelonForm(G2);

> # Loesen der Gleichungssysteme
> x_a := LinearSolve(A, b);
> x_b := LinearSolve(B, b);

> # Ueberpruefen der Ergebnisse durch Einsetzen
> A . x_a;
> B . x_b;
```

### Aufgabe 31

```
> restart:
> with(plots):
> with(LinearAlgebra):

> dreh := <<3, 1> | <1, 4>>;

> # Sei y in Bild(f). Dann gilt y = A*x fuer ein x aus K. Da A
  invertierbar, gilt x = A^(-1)*y.
> # Somit folgt y = A*( A^(-1)*y ), wobei A^(-1)*y in K, also
```

Norm 1 hat.

```
> G := dreh^(-1) . <x,y>;
> implicitplot(VectorNorm(G,2)=1,x=-5..5,y=-5..5,numpoints=
  100000);

> # mit Parametrisierung
> K := plot([ cos(t), sin(t), t=0..2*Pi ],color=red,numpoints=
  2000):
> par := < cos(t), sin(t) >;
> E1 := dreh.par;
> E_1 := plot([E1[1],E1[2],t=0..2*Pi],color=green,numpoints=3000)
  :
> display({ K, E_1 }, numpoints = 2000, scaling = constrained);
```

## Aufgabe 32

```
> restart:
> with(plots):
> with(LinearAlgebra):
```

(a)

```
> A := << 4, 2, 1 > | < 2, 3, 1 > | < 1, 1, 2 >>;
  #A := << 4, 1, 0 > | < 1, 3, 0 > | < 0, 0, 1 >>;
> # Gleichung
> G := expand(Transpose(A^(-1) . < x, y, z >).(A^(-1) . < x, y, z
  >));
> # Parametrisierung
> B := A . < cos(t) * sin(s), sin(t) * sin(s), cos(s) >;

> plot3d([ B[1], B[2], B[3] ], t = 0..2*Pi, s = 0..Pi, numpoints
  = 3000, style = wireframe, shading = zgrayscale, axes = boxed,
  scaling=constrained);
> implicitplot3d(G = 1, x = -5..5, y = -5..5, z = -3..3,numpoints
  = 5000, style = wireframe, shading = zgrayscale, axes = boxed,
  scaling = constrained);
```

(b)

```
> E_W, E_V := Eigenvectors(A):
> # Test
> for kk from 1 to 3 do
    simplify(E_W[kk] * E_V[1..-1, kk] - A . E_V[1..-1, kk]);
  end do;
> # Etwas kompakter vielleicht?
> EW := map(simplify@evalc, E_W);
> EV := map(simplify@evalc, E_V):
```

```
> # Die aufgespannten Eigenräume
> V1 := t * (EV[1..-1, 1] / Norm(EV[1..-1, 1], 2)):
> V2 := t * (EV[1..-1, 2] / Norm(EV[1..-1, 2], 2)):
> V3 := t * (EV[1..-1, 3] / Norm(EV[1..-1, 3], 2)):

> P1 := plot3d([ B[1], B[2], B[3] ], t = 0..2*Pi, s = 0..Pi,
  numpoints = 3000, style = wireframe, axes = boxed, scaling=
  constrained);
> # Eigenräume einzeichnen
> P2 := spacecurve(Matrix(V1), t = 0 .. EW[1], thickness = 2,
  colour = red);
> P3 := spacecurve(Matrix(V2), t = 0 .. EW[2], thickness = 2,
  colour = blue);
> P4 := spacecurve(Matrix(V3), t = 0 .. EW[3], thickness = 2,
  colour = green);
> Matrix(V1):
> display(P1, P2, P3, P4);
```

## Aufgabe 33

```
> restart:

> f := cos(x) / x;
> G := Int(f, x);
> G = value(G);
> F := simplify(value(G));

> Tf:= series(f, x = 0, 12);
> Pf := convert(Tf, polynom);

> TF := series(F, x = 0, 12);
> PF := convert(TF, polynom);

> Abl_Taylor := diff(PF, x);
> vergleich := Abl_Taylor - Pf;
```