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

Plots

```
> PV := proc(v,col,wid)
  PlotVector(v, color=col, width=.03*wid, head_width=.1*wid, border=
  false, scaling=constrained);
  end proc:
  PA := proc(v1,v2)
  plot([(cos(t)*v1[1]+sin(t)*v2[1])/4,(cos(t)*v1[2]+sin(t)*v2[2])/4,
  t=0..Pi/2]);
  end proc:
  PL := proc(P,Q,col,wid)
  local t;
  plot([(1-t)*P[1]+t*Q[1],(1-t)*P[2]+t*Q[2],t=0..1],color=col,
  thickness=4);
  end proc:
  PT := proc(v,lab)
  local tf,n;
  tf := 0.1;
  n := sqrt(v[1]^2+v[2]^2):
  if n<>0 then
  textplot([v[1]*(1+tf/n),v[2]*(1+tf/n),lab])
  else
  textplot([tf,tf,lab])
  fi;
  end proc:
  e1 := Vector([1,0]):
  e2 := Vector([0,1]):
  E := Matrix([ [1,0], [0,1] ]):
  PC := proc(A,N,col,wid)
  local v1,v2,tf;
  v1 := MatrixVectorMultiply(A,e1);
  v2 := MatrixVectorMultiply(A,e2);
  tf := 1.1;
  PV(v1,col[1],wid), PV(v2,col[2],wid), PA(v1,v2), PT(v1,N*e[1]), PT
  (v2,N*e[2]);
  end proc:
```

Finde Fundamentallösung des linearen Gleichungssystems Bv=0 für eine reelle 2x2-Matrix B vom Rang 1:

```
> LGS := proc(B);
  local a,b,c,d,v;
  a := B[1,1];
  b := B[1,2];
  c := B[2,1];
  d := B[2,2];
  if evalf(a^2+b^2) > evalf(c^2+d^2)
  then v := Vector([b,-a])
  else v := Vector([d,-c])
  fi;
  #  Reskaliere zu Betrag 1
  v := v/sqrt(v[1]^2+v[2]^2);
  end proc:
```

Jordannormalform einer reellen 2x2-Matrix:
A=UJU^(-1) mit Jordannormalform J und Übergangsmatrix U

```
> JNF := proc(A,out)
  local CharPol,Spur,Det,Disc,Eigenwert,Eigenvektor,B,U,J,v1,v2,V;
  # Charakteristisches Polynom
  CharPol := collect(expand((X-A[1,1])*(X-A[2,2])-A[1,2]*A[2,1]),X);
  if out=verbose then
     print("Charakteristisches Polynom:",CharPol) fi;
  Spur := -coeff(CharPol,X,1);
  Det := coeff(CharPol,X,0);
  # Diskriminante
  Disc := evalf(Spur^2-4*Det);
  if Disc>0 then
  # Fall 1: verschiedene reelle Eigenwerte
  Eigenwert := convert([solve(CharPol,X)],list);
  if out=verbose then
     print("zwei verschiedene reelle Eigenwerte",Eigenwert[1],
  Eigenwert[2]) fi;
  Eigenvektor := [seq(
          LGS(Matrix([ [A[1,1]-Eigenwert[i],A[1,2]],
                       [A[2,1],A[2,2]-Eigenwert[i]] ]))
          ,i=1..2)];
  J := Matrix([ [Eigenwert[1],0],
                [0,Eigenwert[2]] ]);
  U := convert(Eigenvektor,Matrix);
  elif Disc=0 then
  # Fall 2: zweimal derselbe reelle Eigenwert
  if A[1,2]=0 and A[2,1]=0 then
  # Fall 2a: Skalarmatrix
  if out=verbose then
     print("Skalarmatrix mit reellem Eigenwert der Vielfachheit 2")
  fi;
  J := A;
  U := Matrix([ [1,0], [0,1] ]);
  else
  # Fall 2b: Jordanblock der Grösse 2x2:
  Eigenwert := solve(CharPol,X)[1];
  if out=verbose then
     print("Jordanblock der Grösse 2x2 zum reellen Eigenwert",
  Eigenwert) fi;
  v2 := Vector([1,0]);
  v1 := MatrixVectorMultiply(A,v2)-Eigenwert*v2;
  if v1[1]=0 and v1[2]=0 then
  v2 := Vector([0,1]);
  v1 := MatrixVectorMultiply(A,v2)-Eigenwert*v2;
  fi;
  J := Matrix([ [Eigenwert,1], [0,Eigenwert] ]);
  U := convert([v1,v2],Matrix);
  fi;
  else
  # Fall 3: nicht-reelle Eigenwerte
  if out=verbose then
     print("nicht trigonalisierbar") fi;
  v2 := Vector([1,0]);
  v1 := MatrixVectorMultiply(A,v2);
  J := Matrix([ [Spur,1], [-Det,0] ]);
  U := convert([v1,v2],Matrix);
  fi;
  # Output
```

```
U := simplify(U);
J := simplify(J);
if out=verbose then
   print("Jordannormalform",J,"Übergangsmatrix",U) fi;
# Teste Korrektheit:
(*
V := MatrixMatrixMultiply(A,U)-MatrixMatrixMultiply(U,J);
if Determinant(U)<>0 and
   expand(V[1,1])=0 and
   expand(V[1,2])=0 and
   expand(V[2,1])=0 and
   expand(V[2,2])=0
then
# Okay
else
print("Falsch:",A,Determinant(U),MatrixMatrixMultiply(A,U),
MatrixMatrixMultiply(U,J),V);
fi;
*)
[J,U];
end proc:
```

Visualisierung Jordannormalform:

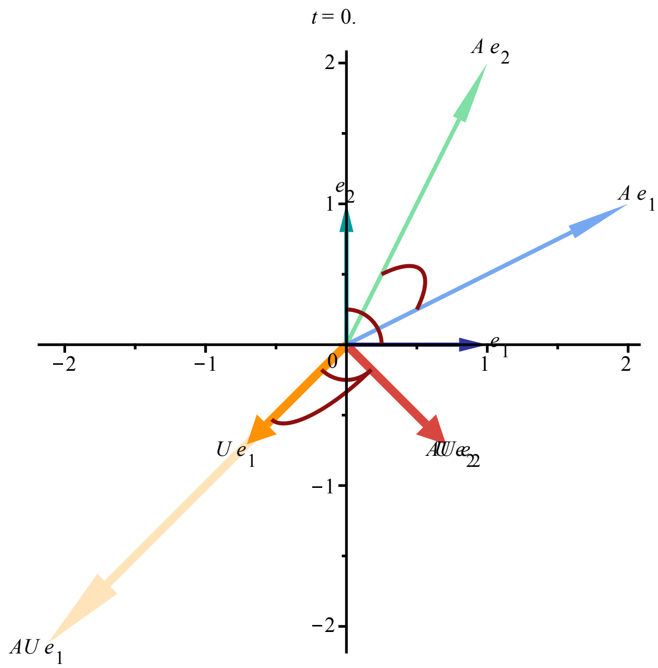
```
> JNFVis := proc(A1);
  local JU,J,U1,U,AU,AU1,A,F,P;
  JU := JNF(A1,verbose);
  J := JU[1];
  U1 := JU[2];
  AU1 := MatrixMatrixMultiply(A1,U1);
  P[1] := PC(E,1,[navy,"DarkCyan"],1);
  P[2] := PC(A1,'A',["CornflowerBlue",aquamarine],1);
  P[3] := PC(U1,'U',[coral,orange],2);
  P[4] := PC(AU1,'AU',["NavajoWhite","LightCoral"],2);
  P[0] := P[1],P[2],P[3],P[4];
  F := proc(t) display(P[t]) end proc:
  animate(F,[t],t=[0,1,2,3,4]);
  end proc:
> A := Matrix([ [2,1], [1,2] ]);
  JNFVis(A);
```

$$A := \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

"Charakteristisches Polynom:", $X^2 - 4X + 3$

"zwei verschiedene reelle Eigenwerte", 3, 1

"Jordannormalform", $\begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix}$, "Übergangsmatrix", $\begin{bmatrix} -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{bmatrix}$



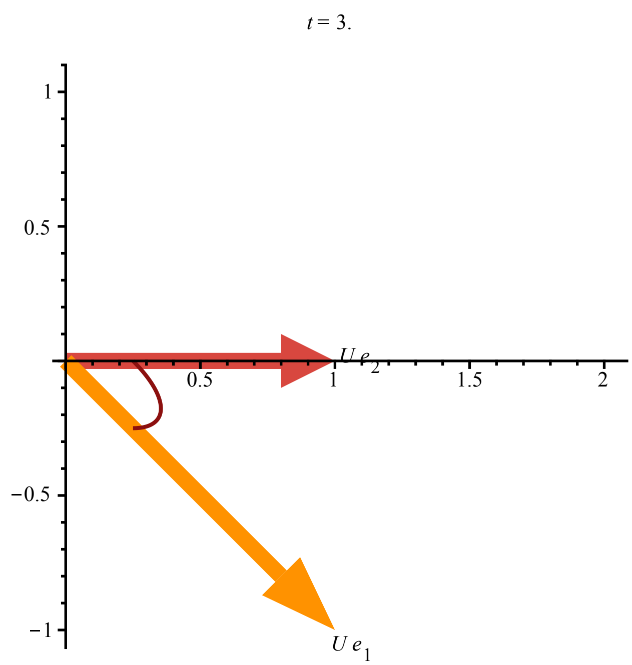

```
> A := Matrix([ [2,1], [-1,0] ]);
  JNFVis(A);
```

$$A := \begin{bmatrix} 2 & 1 \\ -1 & 0 \end{bmatrix}$$

"Charakteristisches Polynom:", $X^2 - 2\,X + 1$

"Jordanblock der Grösse 2x2 zum reellen Eigenwert", 1

"Jordannormalform", $\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$, "Übergangsmatrix", $\begin{bmatrix} 1 & 1 \\ -1 & 0 \end{bmatrix}$



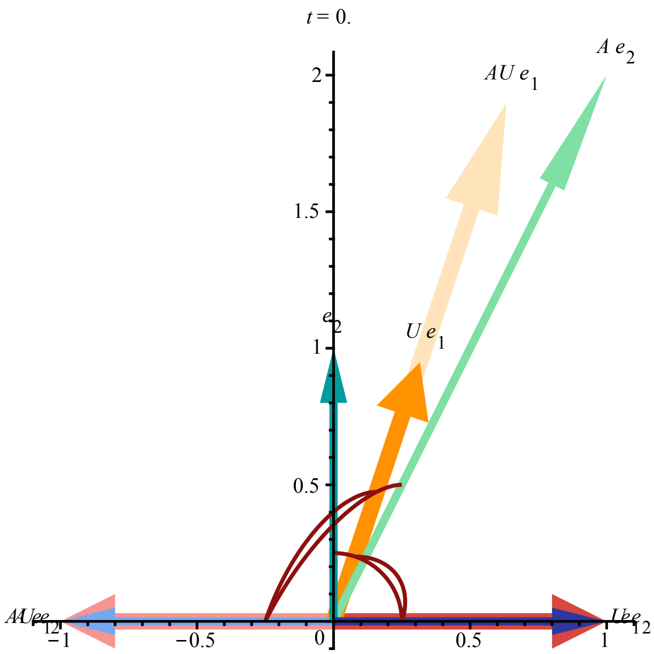

```
> A := Matrix([ [-1,1], [0,2] ]);
  JNFVis(A);
```

$$A := \begin{bmatrix} -1 & 1 \\ 0 & 2 \end{bmatrix}$$

"Charakteristisches Polynom:", $X^2 - X - 2$

"zwei verschiedene reelle Eigenwerte", 2, -1

"Jordannormalform", $\begin{bmatrix} 2 & 0 \\ 0 & -1 \end{bmatrix}$, "Übergangsmatrix", $\begin{bmatrix} \frac{\sqrt{10}}{10} & 1 \\ \frac{3\sqrt{10}}{10} & 0 \end{bmatrix}$



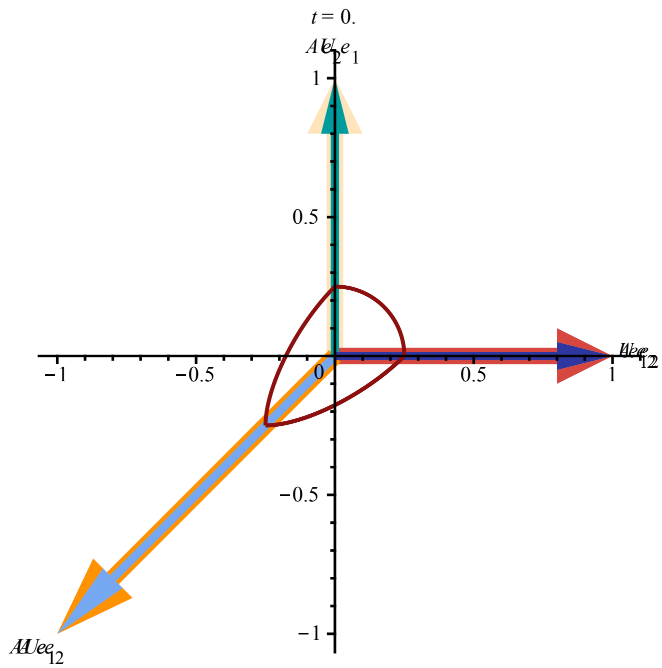

```
> A := Matrix([ [-1,1], [-1,0] ]);
  JNFVis(A);
```

$$A := \begin{bmatrix} -1 & 1 \\ -1 & 0 \end{bmatrix}$$

"Charakteristisches Polynom:", $X^2 + X + 1$

"nicht trigonalisierbar"

"Jordannormalform", $\begin{bmatrix} -1 & 1 \\ -1 & 0 \end{bmatrix}$, "Übergangsmatrix", $\begin{bmatrix} -1 & 1 \\ -1 & 0 \end{bmatrix}$



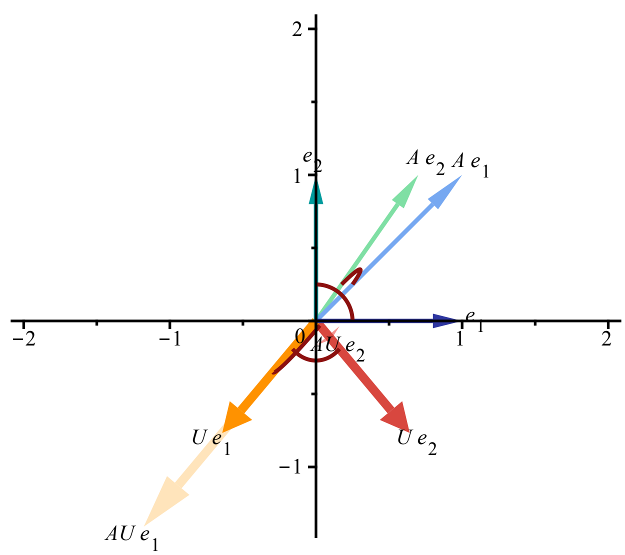
t = 0.
1
0.5
−0.5
−1
−1
−0.5
0
0.5
1

Visualisierung Jordannormalform einer Schar von Matrizen

```
> JNFVisAll := proc(A1);
  local JU,J,U1,U,AU,AU1,A,F,P;
  JU := JNF(A1,silent);
  J := JU[1];
  U1 := JU[2];
  AU1 := MatrixMatrixMultiply(A1,U1);
  P[1] := PC(E,1,[navy,"DarkCyan"],1);
  P[2] := PC(A1,'A',["CornflowerBlue",aquamarine],1);
  P[3] := PC(U1,'U',[coral,orange],2);
  P[4] := PC(AU1,'AU',["NavajoWhite","LightCoral"],2);
  display(P[1],P[2],P[3],P[4]);
  end proc:
> M := Matrix([ [1,t/10], [1,1] ]);
  animate(JNFVisAll,[Matrix([ [1,t/10], [1,1] ])],t=[seq(s,s=-20..20)
  ]);
```

$$M := \begin{bmatrix} 1 & \frac{t}{10} \\ 1 & 1 \end{bmatrix}$$

$t = 7.$



Spektralsatz: Diagonalisierung einer symmetrischen reellen 2x2-Matrix und Bestimmung der Hauptachsen:
A=UJU^(-1)=UJU^T mit Diagonalmatrix J und orthogonaler Matrix U

```
> Spektral := proc(A,out)
  local CharPol,Spur,Det,Disc,Eigenwert,U,J,v1,v2,n1,V;
  if A[1,2]<>A[2,1] then
  print("Matrix nicht symmetrisch")
  else
  # Charakteristisches Polynom
  CharPol := collect(expand((X-A[1,1])*(X-A[2,2])-A[1,2]*A[2,1]),X);
  if out=verbose then
     print("Charakteristisches Polynom:",CharPol) fi;
  Spur := -coeff(CharPol,X,1);
  Det := coeff(CharPol,X,0);
  # Diskriminante
  Disc := evalf(Spur^2-4*Det);
  if Disc>0 then
  # Fall 1: verschiedene reelle Eigenwerte
  Eigenwert := convert([solve(CharPol,X)],list);
  if out=verbose then
     print("Eindeutige Hauptachsen zu verschiedenen reellen
  Eigenwerten") fi;
  v1 := LGS(Matrix([ [A[1,1]-Eigenwert[1],A[1,2]],
                     [A[2,1],A[2,2]-Eigenwert[1]] ]));
  n1 := sqrt(v1[1]^2+v1[2]^2);
  v1 := v1/n1;
  v2 := Vector([-v1[2],v1[1]]);
  U := convert([v1,v2],Matrix);
  J := Matrix([ [Eigenwert[1],0],
                [0,Eigenwert[2]] ]);
  elif Disc=0 then
  # Fall 2: Skalarmatrix
  if out=verbose then
     print("Skalarmatrix mit reellem Eigenwert der Vielfachheit 2")
  fi;
  J := A;
  U := Matrix([ [1,0], [0,1] ]);
  fi;
  # Output
  U := simplify(U);
  J := simplify(J);
  if out=verbose then
     print("Jordannormalform",J,"Übergangsmatrix",U) fi;
  # Teste Korrektheit:
  (*
  V := MatrixMatrixMultiply(A,U)-MatrixMatrixMultiply(U,J);
  if Determinant(U)<>0 and
     expand(V[1,1])=0 and
     expand(V[1,2])=0 and
     expand(V[2,1])=0 and
     expand(V[2,2])=0
  then
  # Okay
  else
  print("Falsch:",Determinant(U),MatrixMatrixMultiply(A,U),
  MatrixMatrixMultiply(U,J));
  fi;
  *)
  [J,U];
```

```
  fi;
  end proc:
```

Visualisierung Spektralsatz:

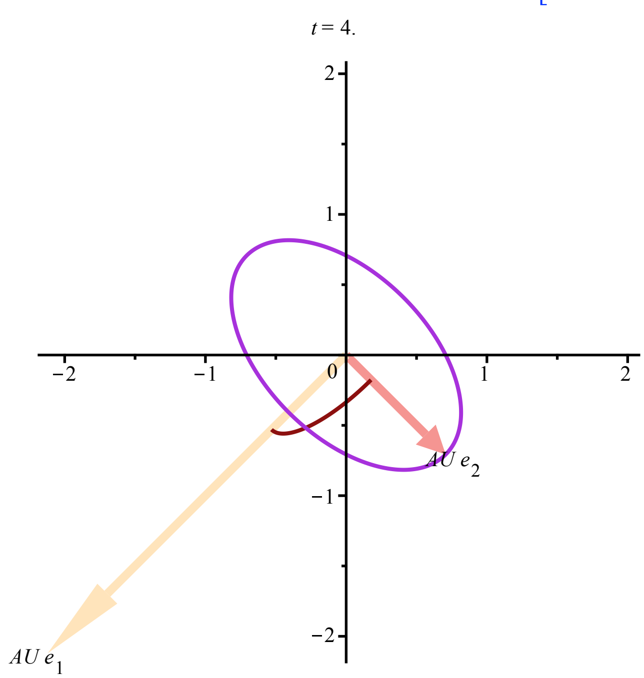
```
> SpektralVis := proc(A1);
  local JU,J,U1,U,V,AU,AU1,A,F,P,Quadrik;
  JU := Spektral(A1,verbose);
  J := JU[1];
  U1 := JU[2];
  AU1 := MatrixMatrixMultiply(A1,U1);
  P[1] := PC(E,1,[navy,"DarkCyan"],1);
  P[2] := PC(A1,'A',["CornflowerBlue",aquamarine],1);
  P[3] := PC(U1,'U',[coral,orange],2);
  P[4] := PC(AU1,'AU',["NavajoWhite","LightCoral"],2);
  P[0] := P[1],P[2],P[3],P[4];
  Quadrik := implicitplot(A1[1,1]*x^2+(A1[1,2]+A1[2,1])*x*y+A1[2,2]*
  y^2-1, x=-2..2, y=-2..2,color="DarkViolet");
  F := proc(t) display(P[t],Quadrik) end proc:
  animate(F,[t],t=[0,1,2,3,4]);
  end proc:
> A := Matrix([ [2,1], [1,2] ]);
  SpektralVis(A);
```

$$A := \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

"Charakteristisches Polynom:", $X^2 - 4X + 3$

"Eindeutige Hauptachsen zu verschiedenen reellen Eigenwerten"

"Jordannormalform", $\begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix}$, "Übergangsmatrix", $\begin{bmatrix} -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{bmatrix}$




```
> A := Matrix([ [-2,1], [1,2] ]);
  SpektralVis(A);
```

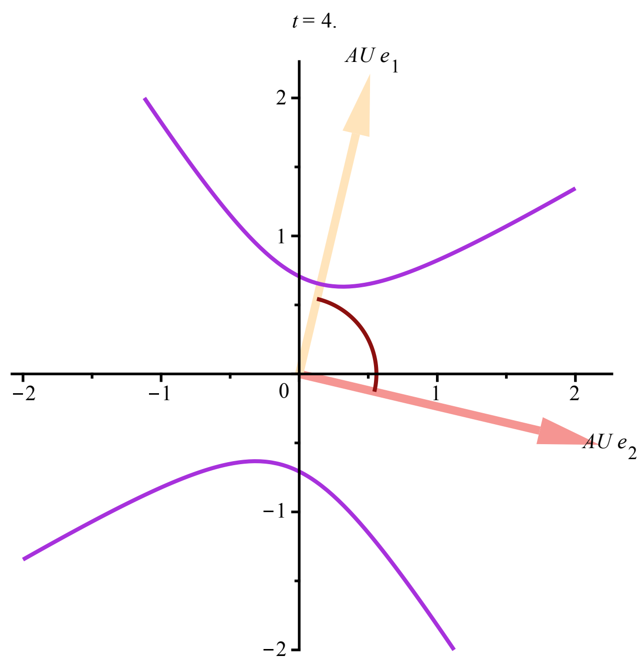
$$A := \begin{bmatrix} -2 & 1 \\ 1 & 2 \end{bmatrix}$$

"Charakteristisches Polynom:", $X^2 - 5$

"Eindeutige Hauptachsen zu verschiedenen reellen Eigenwerten"

"Jordannormalform", $\begin{bmatrix} \sqrt{5} & 0 \\ 0 & -\sqrt{5} \end{bmatrix}$, "Übergangsmatrix",

$$\begin{bmatrix} \frac{1}{\sqrt{10+4\sqrt{5}}} & -\frac{2+\sqrt{5}}{\sqrt{10+4\sqrt{5}}} \\ \frac{2+\sqrt{5}}{\sqrt{10+4\sqrt{5}}} & \frac{1}{\sqrt{10+4\sqrt{5}}} \end{bmatrix}$$




```
> A := Matrix([ [1,1], [1,-2] ]);
  SpektralVis(A);
```

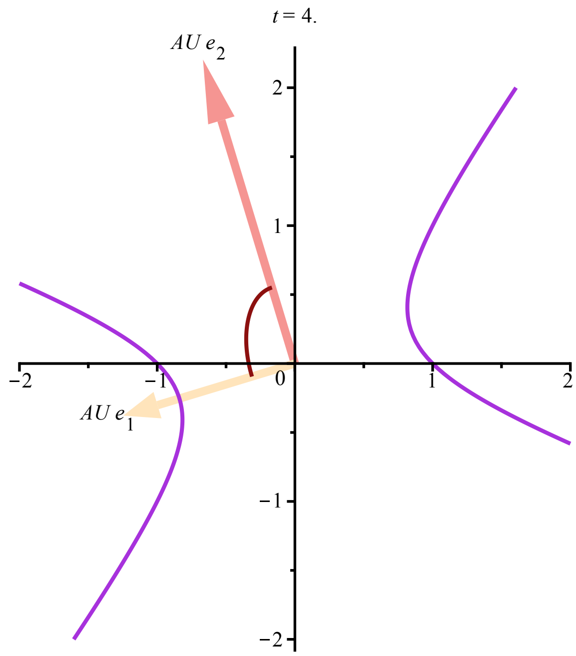
$$A := \begin{bmatrix} 1 & 1 \\ 1 & -2 \end{bmatrix}$$

"Charakteristisches Polynom:", $X^2 + X - 3$

"Eindeutige Hauptachsen zu verschiedenen reellen Eigenwerten"

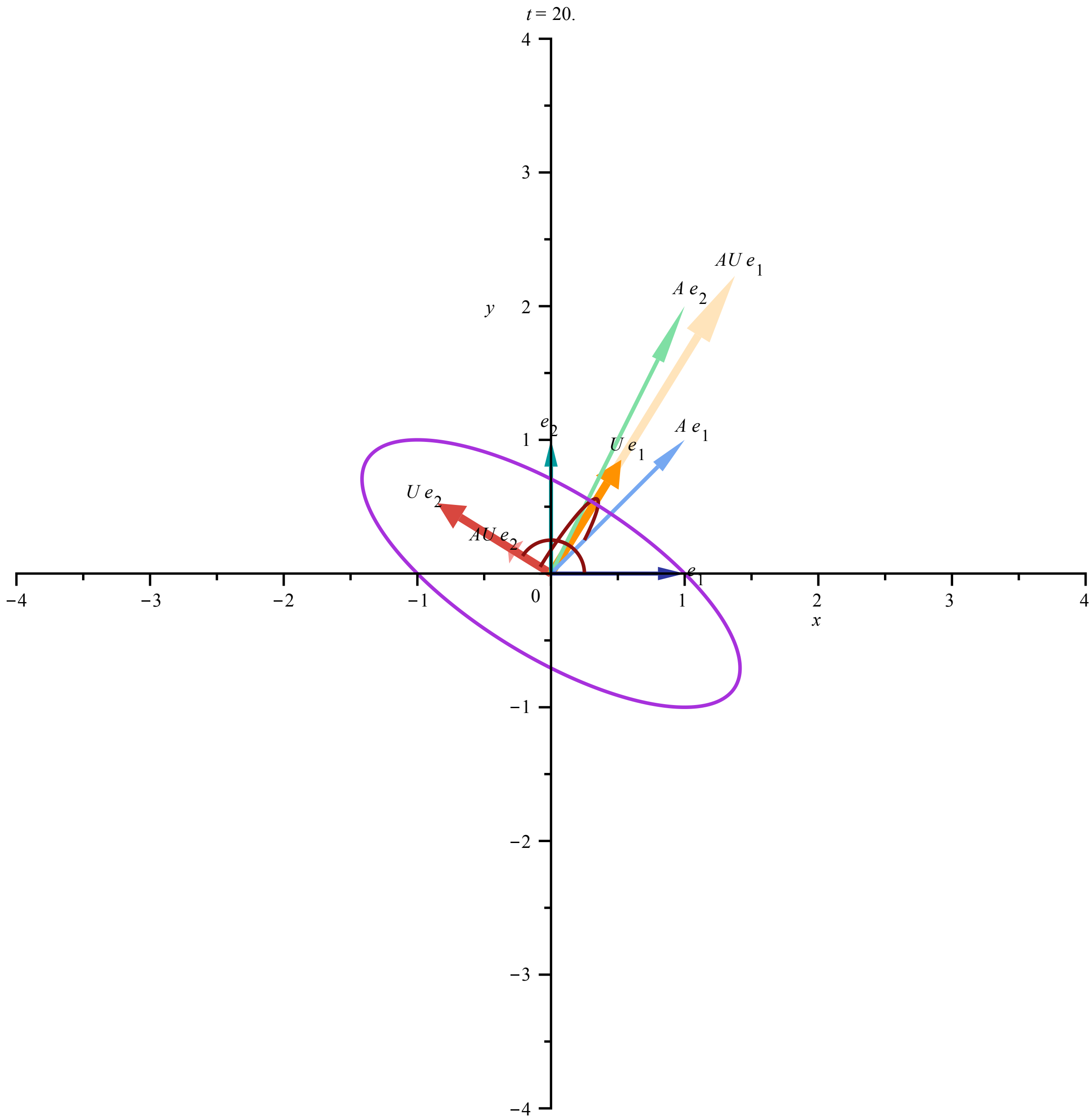
"Jordannormalform", $\begin{bmatrix} -\frac{1}{2} + \frac{\sqrt{13}}{2} & 0 \\ 0 & -\frac{1}{2} - \frac{\sqrt{13}}{2} \end{bmatrix}$, "Übergangsmatrix",

$$\begin{bmatrix} -\frac{3+\sqrt{13}}{\sqrt{26+6\sqrt{13}}} & \frac{2}{\sqrt{26+6\sqrt{13}}} \\ -\frac{2}{\sqrt{26+6\sqrt{13}}} & -\frac{3+\sqrt{13}}{\sqrt{26+6\sqrt{13}}} \end{bmatrix}$$

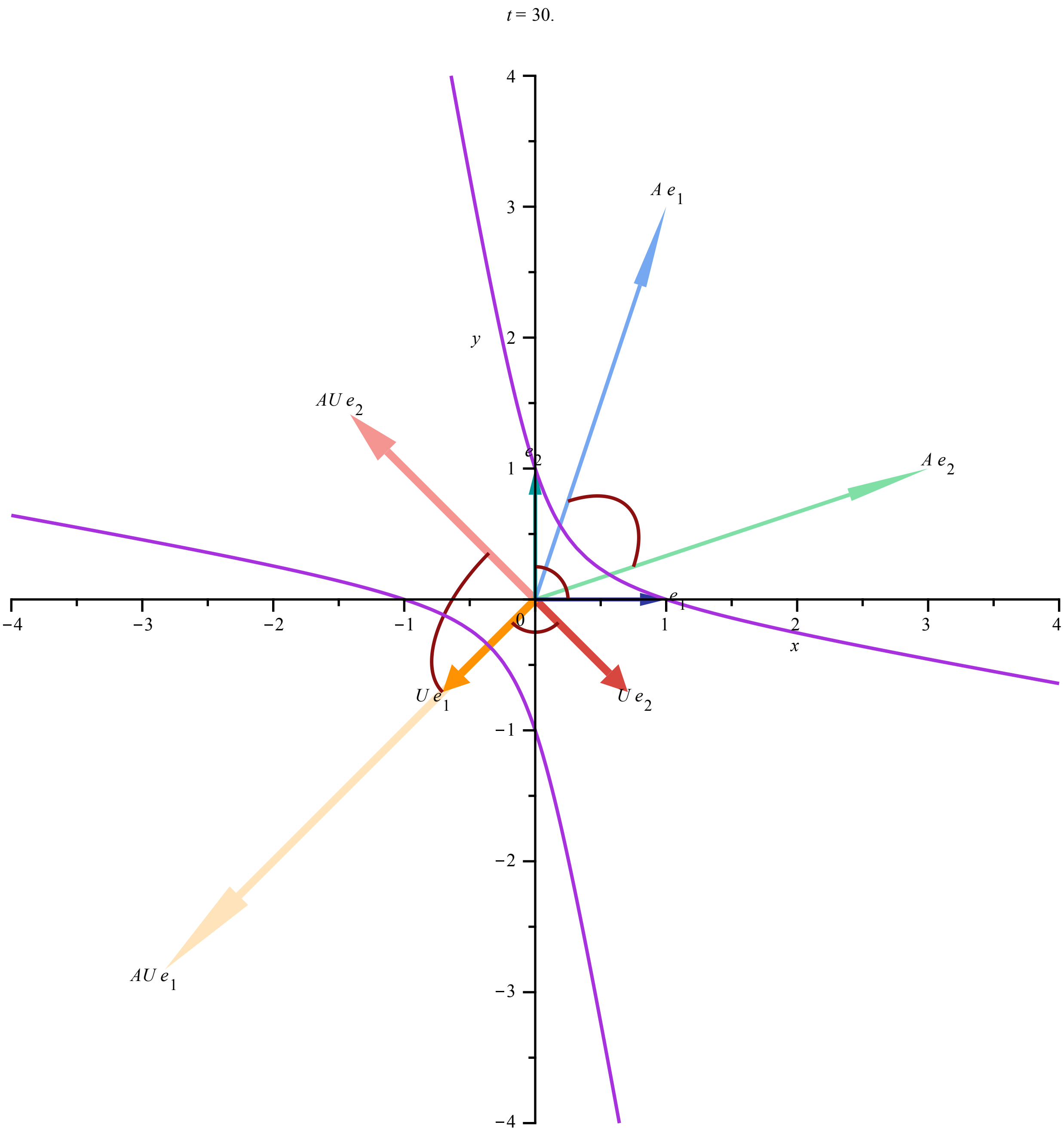


Visualisierung Spektralsatz für eine Schar von Matrizen

```
> SpektralVisAll := proc(A1);
  local JU,J,U1,U,V,AU,AU1,A,F,P,Quadrik,Pall;
  JU := Spektral(A1,silent);
  J := JU[1];
  U1 := JU[2];
  AU1 := MatrixMatrixMultiply(A1,U1);
  P[1] := PC(E,1,[navy,"DarkCyan"],1);
  P[2] := PC(A1,'A',["CornflowerBlue",aquamarine],1);
  P[3] := PC(U1,'U',[coral,orange],2);
  P[4] := PC(AU1,'AU',["NavajoWhite","LightCoral"],2);
  Quadrik := implicitplot(A1[1,1]*x^2+(A1[1,2]+A1[2,1])*x*y+A1[2,2]*
  y^2-1, x=-4..4, y=-4..4,color="DarkViolet");
  Pall[0] := display(P[1],P[2],P[3],P[4],Quadrik);
  end proc:
> animate(SpektralVisAll,[Matrix([ [1,1], [1,t/10] ])],t=[seq(s,s=
  -20..20)]);
```



```
> animate(SpektralVisAll,[Matrix([ [1,t/10], [t/10,1] ])],t=[seq(s,s=
  -30..30)]);
```



Argument eines Vektors

```
> argu := proc(x,y)
  if y=0 then
  if x>=0 then 0
  elif x<0 then Pi
  fi;
  else 2*arctan(y/(sqrt(x^2+y^2)+x))
  fi;
  end proc:
```

QR-Zerlegung einer reellen 2x2-Matrix:
A=QR mit orthogonaler Matrix Q und rechter oberer Dreiecksmatrix R

```
> QR := proc(A1)
  local a,b,c,d,n,Q,Q1,R,R1;
  a := A1[1,1];
  c := A1[2,1];
  if c<>0 then
   n := sqrt(a^2+c^2);
   Q1 := Matrix([[a/n,-c/n],[c/n,a/n]]);
   R1 := MatrixMatrixMultiply(Transpose(Q1),A1);
  else
   Q1 := IdentityMatrix(2);
   R1 := A1;
  fi;
  print(Q=Q1,R=R1);
  Q1,R1
  end proc:
> A := Matrix([ [2,1], [1,2] ]);
  QR(A):
```

$$A := \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

$$Q = \begin{bmatrix} \frac{2\sqrt{5}}{5} & -\frac{\sqrt{5}}{5} \\ \frac{\sqrt{5}}{5} & \frac{2\sqrt{5}}{5} \end{bmatrix}, R = \begin{bmatrix} \sqrt{5} & \frac{4\sqrt{5}}{5} \\ 0 & \frac{3\sqrt{5}}{5} \end{bmatrix} \qquad (1)$$

```
> A := Matrix([ [1,2], [-1,3] ]);
  QR(A):
```

$$A := \begin{bmatrix} 1 & 2 \\ -1 & 3 \end{bmatrix}$$

$$Q = \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}, R = \begin{bmatrix} \sqrt{2} & -\frac{\sqrt{2}}{2} \\ 0 & \frac{5\sqrt{2}}{2} \end{bmatrix} \qquad (2)$$

Visualisierung QR-Zerlegung:

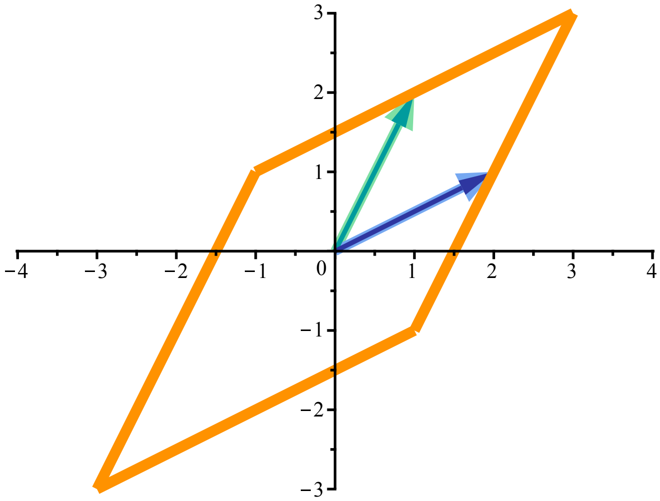
```
> QRVis := proc(A1);
  local Q,Q1,R,R1,e1,e2,EndAchsen,Zeich,Fig,theta,n0,n,i,A;
  Q1,R1 := QR(A1);
  # Zeichnung
  e1 := Vector([1,0]);
  e2 := Vector([0,1]);
  EndAchsen := {PV(MatrixVectorMultiply(A1,e1),"CornflowerBlue",4),
                PV(MatrixVectorMultiply(A1,e2),aquamarine,4)};
  Zeich := proc(T)
  local v1,v2;
  v1 := MatrixVectorMultiply(T,e1);
  v2 := MatrixVectorMultiply(T,e2);
  display({PV(v1,navy,2),
           PV(v2,"DarkCyan",2),
```

```
          PL( v1+v2, v1-v2,coral,4),
          PL( v1-v2,-v1-v2,coral,4),
          PL(-v1-v2,-v1+v2,coral,4),
          PL(-v1+v2, v1+v2,coral,4)}
         union EndAchsen);
end proc:
# Drehwinkel
theta := argu(Q1[1,1],Q1[2,1]);
# Fallunterscheidung
Fig := proc(n)
local E;
if n<n0 then
E := IdentityMatrix(2);
Zeich(E+n/n0*(R1-E));
else
Zeich(MatrixMatrixMultiply(
        Matrix([ [cos((n/n0-1)*theta),-sin((n/n0-1)*theta)],
                 [sin((n/n0-1)*theta), cos((n/n0-1)*theta)]]),
       R1));
fi;
end proc:
# Figur zeichnen
n0 := 20;
animate(Fig,[n],n=[seq(i,i=0..2*n0)]);
end proc:
> A := Matrix([ [2,1], [1,2] ]);
QRVis(A);
```

$$A := \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

$$Q = \begin{bmatrix} \frac{2\sqrt{5}}{5} & -\frac{\sqrt{5}}{5} \\ \frac{\sqrt{5}}{5} & \frac{2\sqrt{5}}{5} \end{bmatrix}, R = \begin{bmatrix} \sqrt{5} & \frac{4\sqrt{5}}{5} \\ 0 & \frac{3\sqrt{5}}{5} \end{bmatrix}$$

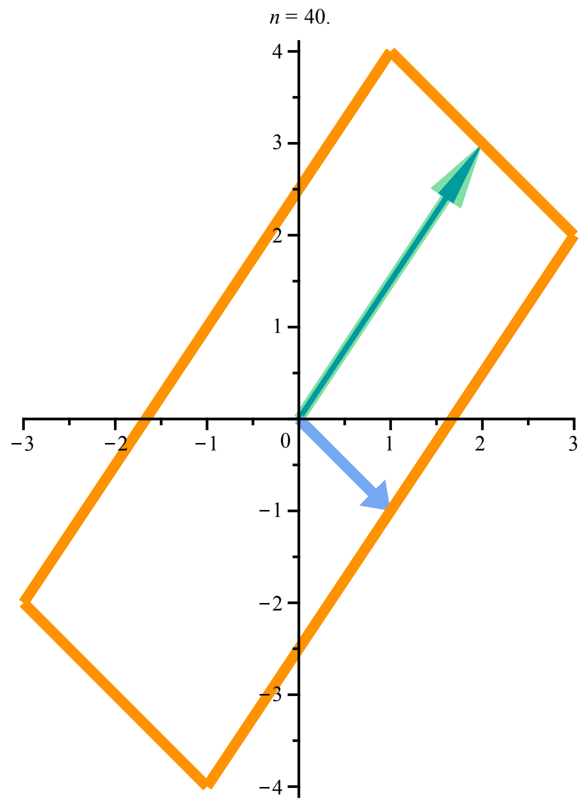
$n = 40.$




```
> A := Matrix([ [1,2], [-1,3] ]);
  QRVis(A);
```

$$A := \begin{bmatrix} 1 & 2 \\ -1 & 3 \end{bmatrix}$$

$$Q = \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}, R = \begin{bmatrix} \sqrt{2} & -\frac{\sqrt{2}}{2} \\ 0 & \frac{5\sqrt{2}}{2} \end{bmatrix}$$

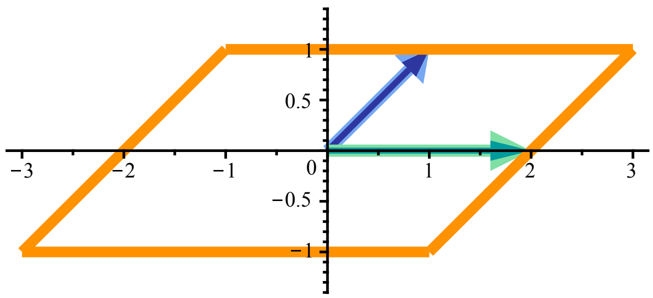



```
> A := Matrix([ [1,2], [1,0] ]);
  QRVis(A);
```

$$A := \begin{bmatrix} 1 & 2 \\ 1 & 0 \end{bmatrix}$$

$$Q = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}, R = \begin{bmatrix} \sqrt{2} & \sqrt{2} \\ 0 & -\sqrt{2} \end{bmatrix}$$

$n = 40.$

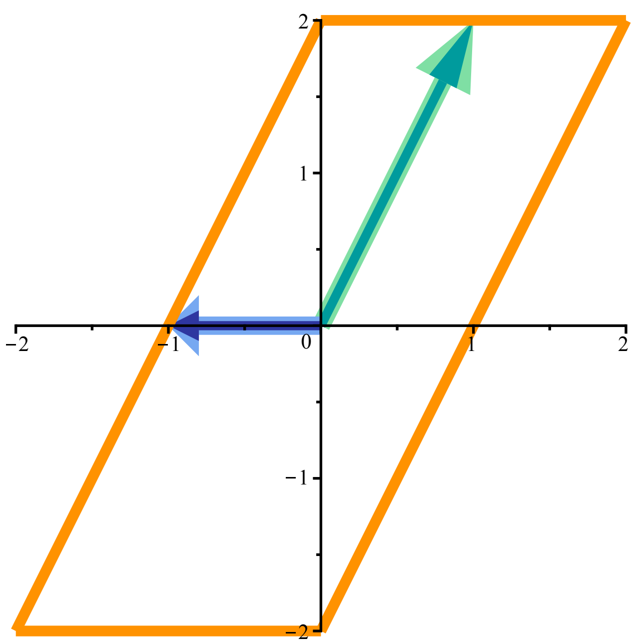



```
> A := Matrix([ [-1,1], [0,2] ]);
  QRVis(A);
```

$$A := \begin{bmatrix} -1 & 1 \\ 0 & 2 \end{bmatrix}$$

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, R = \begin{bmatrix} -1 & 1 \\ 0 & 2 \end{bmatrix}$$

$n = 40.$



Singulärwertzerlegung einer reellen 2x2-Matrix:
A=QDR mit orthogonalen Matrizen Q, R und Diagonalmatrix D

```
> Sing := proc(A,out)
  local B,JU,R,D,QD,Q,v1,v2,V;
  # Wende Spektralsatz auf A^TA an:
  B := MatrixMatrixMultiply(Transpose(A),A);
  JU := Spektral(B,silent);
  # rechte orthogonale Matrix:
  R := Transpose(JU[2]);
  # Diagonalmatrix
  D := Matrix([ [sqrt(JU[1][1,1]),0], [0,sqrt(JU[1][2,2])] ]);
  # linke orthogonale Matrix:
  QD := MatrixMatrixMultiply(A,JU[2]);
  if D[1,1]<>0 then
  if D[2,2]<>0 then
  Q := Matrix([ [QD[1,1]/D[1,1],QD[1,2]/D[2,2]], [QD[2,1]/D[1,1],QD
  [2,2]/D[2,2]] ]);
  else
  v1 := Vector([QD[1,1]/D[1,1],QD[2,1]/D[1,1]]);
  v2 := Vector([-v1[2],v1[1]]);
  Q := convert([v1,v2],Matrix);
  fi
  elif D[2,2]<>0 then
  v2 := Vector([QD[2,1]/D[2,2],QD[2,2]/D[2,2]]);
  v1 := Vector([v1[2],-v1[1]]);
  Q := convert([v1,v2],Matrix);
  else
  Q := Matrix([[1,0],[0,1]]);
  fi;
  Q := evalf(Q);
  D := evalf(D);
  R := evalf(R);
  if out=verbose then
     print(Q,D,R) fi;
  # Teste Korrektheit:
  (*
  V := MatrixMatrixMultiply(Q,MatrixMatrixMultiply(D,R))-A;
  if abs(V[1,1])+abs(V[1,2])+abs(V[2,1])+abs(V[2,2])>10^(-5) then
  print("Falsch:",A,V);
  fi;
  *)
  [Q,D,R];
  end proc:
```

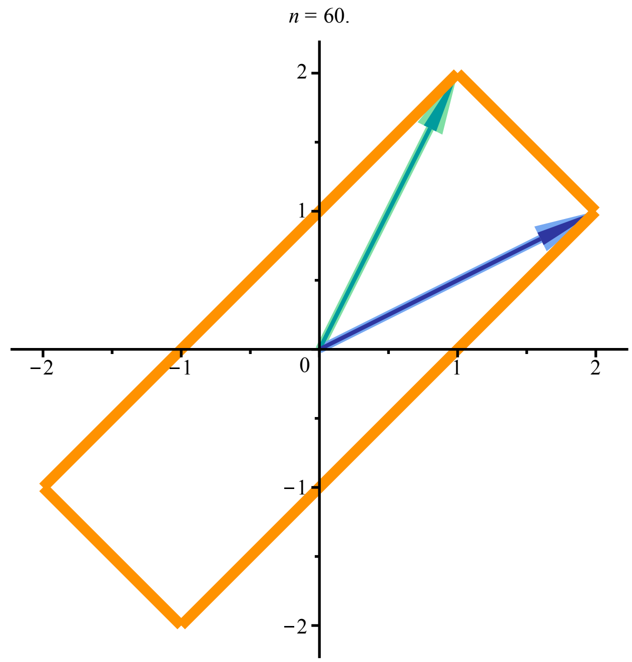
Visualisierung Singulärwertzerlegung:

```
> SingVis := proc(A1);
  local QDR,Q,Q1,D,D1,R,R1,E,theta,phi,d,C1,C2,P1,P2,AP1,AP2,n0,n,i,
  A,Fig;
  QDR := Sing(A1,silent);
  Q1 := QDR[1];
  D1 := QDR[2];
  R1 := QDR[3];
  # Integriere Spiegelung in Diagonalmatrix
  if Determinant(Q1)<0 then
  E := Matrix([[1,0],[0,-1]]);
  D1 := MatrixMatrixMultiply(E,D1);
  Q1 := MatrixMatrixMultiply(Q1,E);
  print("Spiegelung in Diagonalmatrix integriert");
  fi;
  print(Q=Q1,D=D1,R=R1);
```

```
# Drehwinkel
theta := argu(R1[1,1],R1[2,1]);
phi   := argu(Q1[1,1],Q1[2,1]);
# Figur vor und nach 1. Schritt
P1[0] := Vector([1,0]);
P2[0] := Vector([0,1]);
P1[1] := MatrixVectorMultiply(R1,P1[0]);
P2[1] := MatrixVectorMultiply(R1,P2[0]);
d := max(abs(R1[1,1]),abs(R1[1,2]));
C1[1] := Vector([d,d]);
C2[1] := Vector([-d,d]);
C1[0] := MatrixVectorMultiply(Transpose(R1),C1[1]);
C2[0] := MatrixVectorMultiply(Transpose(R1),C2[1]);
# Figur nach 2. Schritt
C1[2] := MatrixVectorMultiply(D1,C1[1]);
C2[2] := MatrixVectorMultiply(D1,C2[1]);
P1[2] := MatrixVectorMultiply(D1,P1[1]);
P2[2] := MatrixVectorMultiply(D1,P2[1]);
# Vektoren am Ende
AP1 := MatrixVectorMultiply(A1,P1[0]);
AP2 := MatrixVectorMultiply(A1,P2[0]);
# Figur zeichnen
n0 := 20;
Fig := proc(n)
local M,C1a,C2a,P1a,P2a;
if n<n0 then
M := Matrix([ [cos(n/n0*theta),-sin(n/n0*theta)],
              [sin(n/n0*theta), cos(n/n0*theta)]]);
C1a := MatrixVectorMultiply(M,C1[0]);
C2a := MatrixVectorMultiply(M,C2[0]);
P1a := MatrixVectorMultiply(M,P1[0]);
P2a := MatrixVectorMultiply(M,P2[0]);
elif n<2*n0 then
C1a := (2-n/n0)*C1[1]+(n/n0-1)*C1[2];
C2a := (2-n/n0)*C2[1]+(n/n0-1)*C2[2];
P1a := (2-n/n0)*P1[1]+(n/n0-1)*P1[2];
P2a := (2-n/n0)*P2[1]+(n/n0-1)*P2[2];
else
M := Matrix([ [cos((n/n0-2)*phi),-sin((n/n0-2)*phi)],
              [sin((n/n0-2)*phi), cos((n/n0-2)*phi)]]);
C1a := MatrixVectorMultiply(M,C1[2]);
C2a := MatrixVectorMultiply(M,C2[2]);
P1a := MatrixVectorMultiply(M,P1[2]);
P2a := MatrixVectorMultiply(M,P2[2]);
fi;
display(PV(P1a,navy,1),
        PV(P2a,"DarkCyan",1),
        PV(AP1,"CornflowerBlue",2),
        PV(AP2,"aquamarine",2),
        PL(C1a,C2a,coral,5),
        PL(C1a,-C2a,coral,5),
        PL(-C1a,C2a,coral,5),
        PL(-C1a,-C2a,coral,5))
end proc:
animate(Fig,[n],n=[seq(i,i=0..3*n0)]);
end proc:
> A := Matrix([ [2,1], [1,2] ]);
SingVis(A);
```

$$A := \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

$$Q = \begin{bmatrix} -0.7071067810 & 0.7071067810 \\ -0.7071067810 & -0.7071067810 \end{bmatrix}, \mathrm{D} = \begin{bmatrix} 3. & 0. \\ 0. & 1. \end{bmatrix}, R = \begin{bmatrix} -0.7071067810 & -0.7071067810 \\ 0.7071067810 & -0.7071067810 \end{bmatrix}$$

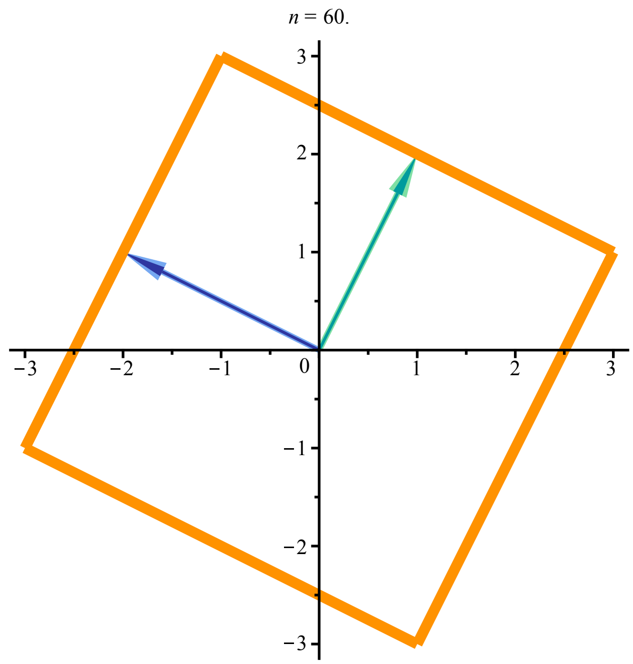



```
> A := Matrix([ [-2,1], [1,2] ]);
  SingVis(A);
```

$$A := \begin{bmatrix} -2 & 1 \\ 1 & 2 \end{bmatrix}$$

"Spiegelung in Diagonalmatrix integriert"

$$Q = \begin{bmatrix} -0.894427190800000 & -0.447213595400000 \\ 0.447213595400000 & -0.894427190800000 \end{bmatrix}, \mathrm{D} = \begin{bmatrix} 2.23606797700000 & 0. \\ 0. & -2.23606797700000 \end{bmatrix}, R = \begin{bmatrix} 1. & 0. \\ 0. & 1. \end{bmatrix}$$




```
> A := Matrix([ [1,2], [1,0] ]);
  SingVis(A);
```

$$A := \begin{bmatrix} 1 & 2 \\ 1 & 0 \end{bmatrix}$$

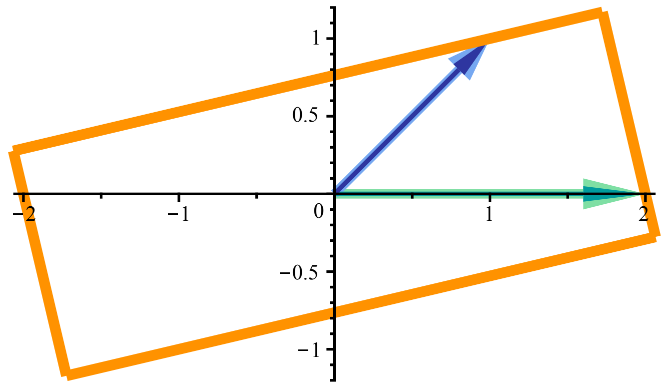
"Spiegelung in Diagonalmatrix integriert"

$$Q = \begin{bmatrix} 0.973248989700000 & -0.229752920200000 \\ 0.229752920600000 & 0.973248989400000 \end{bmatrix}, \mathrm{D}$$

$$= \begin{bmatrix} 2.28824561100000 & 0. \\ 0. & -0.874032049000000 \end{bmatrix}, R$$

$$= \begin{bmatrix} 0.5257311122 & 0.8506508084 \\ -0.8506508084 & 0.5257311122 \end{bmatrix}$$

$n = 60.$




```
> A := Matrix([ [-1,1], [0,2] ]);
  SingVis(A);
```

$$A := \begin{bmatrix} -1 & 1 \\ 0 & 2 \end{bmatrix}$$

"Spiegelung in Diagonalmatrix integriert"

$$Q = \begin{bmatrix} 0.525731112200000 & -0.850650808000000 \\ 0.850650808200000 & 0.525731112000000 \end{bmatrix}, \mathrm{D}$$

$$= \begin{bmatrix} 2.28824561100000 & 0. \\ 0. & -0.874032049000000 \end{bmatrix}, R$$

$$= \begin{bmatrix} -0.2297529205 & 0.9732489892 \\ -0.9732489892 & -0.2297529205 \end{bmatrix}$$

$n = 60.$