Bitte zur Übung kommen -> Evaluierung der Übungsgruppen !!

restart; *with*(*plots*);

[*animate*, *animate3d*, *animatecurve*, *arrow*, *changecoords*, *complexplot*, *complexplot3d*, *conformal*, *conformal3d*, *contourplot*, *contourplot3d*, *coordplot*, *coordplot3d*, *densityplot*, *display*, *dualaxisplot*, *fieldplot*, *fieldplot3d*, *gradplot*, *gradplot3d*, *graphplot3d*, *implicitplot*, *implicitplot3d*, *inequal*, *interactive*, *interactiveparams*, *intersectplot*, *listcontplot*, *listcontplot3d*, *listdensityplot*, *listplot*, *listplot3d*, *loglogplot*, *logplot*, *matrixplot*, *multiple*, *odeplot*, *pareto*, *plotcompare*, *pointplot*, *pointplot3d*, *polarplot*, *polygonplot*, *polygonplot3d*, *polyhedra_supported*, *polyhedraplot*, *rootlocus*, *semilogplot*, *setcolors*, *setoptions*, *setoptions3d*, *spacecurve*, *sparsematrixplot*, *surfdata*, *textplot*, *textplot3d*, *tubeplot*] **(1)**

VL := [8, 1], [8, 2], [8, 4], [8, 5], [7.5, 4.5], [6, 1.5], [6, 7], [6.5, 1.5], [6, 1], [6, 3], [6, 4], [6, 7], [5.5, 3.5], [5.5, 4.5], [5, 3], [5, 4], [5, 5], [4, 3.5], [4, 4], [4.1, 4.1], [4, 5], [3.5, 5], [3.5, 2], [3, 2], [3, 4.5], [3, 5], [3, 5.5], [3, 6], [2.5, 5.5], [2.5, 6], [2, 7.5], [2.5, 4], [3.5, 4], [3.6, 4], [5, 4], [5.5, 4], [6, 4], [7, 4], [5.5, 3.5], [4, 5], [4, 6], [3.5, 5], [7, 3], [7, 2];

[8, 1], [8, 2], [8, 4], [8, 5], [7.5, 4.5], [6, 1.5], [6, 7], [6.5, 1.5], [6, 1], [6, 3], [6, 4], [6, 7], [5.5, 3.5], [5.5, 4.5], [5, 3], [5, 4], [5, 5], [4, 3.5], [4, 4], [4.1, 4.1], [4, 5], [3.5, 5], [3.5, 2], [3, 2], [3, 4.5], [3, 5], [3, 5.5], [3, 6], [2.5, 5.5], [2.5, 6], [2, 7.5], [2.5, 4], [3.5, 4], [3.6, 4], [5, 4], [5.5, 4], [6, 4], [7, 4], [5.5, 3.5], [4, 5], [4, 6], [3.5, 5], [7, 3], [7, 2] **(2)**

PtCloud := *pointplot*([*VL*]);

PLOT(...) **(3)**

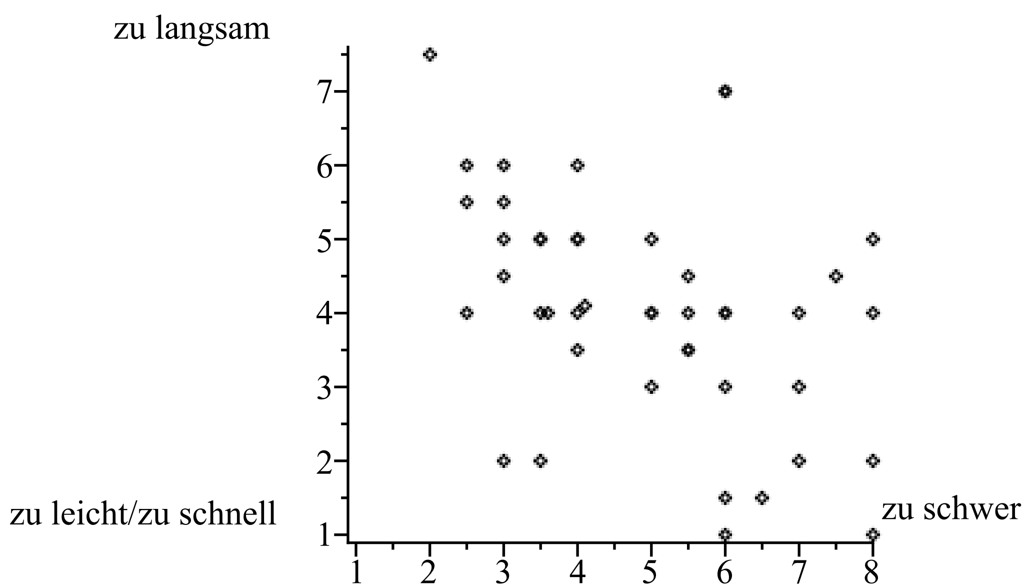
UE := [2.8, 4], [3.2, 4], [3.5, 4], [4, 4], [4.5, 4], [5, 4], [5.2, 4], [5.4, 4], [5.6, 4], [5.8, 4], [6, 4], [7, 4], [4, 4.1], [5, 5.1], [5, 5.5], [5.5, 5.5], [6, 5.5], [5, 5], [5.5, 5], [4.8, 3.5], [4.8, 3.7], [5.5, 3], [5.5, 3.5], [6, 3.5], [5.5, 2], [5.4, 2], [5.4, 2.2], [5, 4], [5, 4.5], [6, 2], [5.5, 4.5], [5.5, 4.8], [5.5, 4], [5, 4.5], [5, 4.9], [8, 4], [8, 2], [7.8, 2];

[2.8, 4], [3.2, 4], [3.5, 4], [4, 4], [4.5, 4], [5, 4], [5.2, 4], [5.4, 4], [5.6, 4], [5.8, 4], [6, 4], [7, 4], [4, 4.1], [5, 5.1], [5, 5.5], [5.5, 5.5], [6, 5.5], [5, 5], [5.5, 5], [4.8, 3.5], [4.8, 3.7], [5.5, 3], [5.5, 3.5], [6, 3.5], [5.5, 2], [5.4, 2], [5.4, 2.2], [5, 4], [5, 4.5], [6, 2], [5.5, 4.5], [5.5, 4.8], [5.5, 4], [5, 4.5], [5, 4.9], [8, 4], [8, 2], [7.8, 2] **(4)**

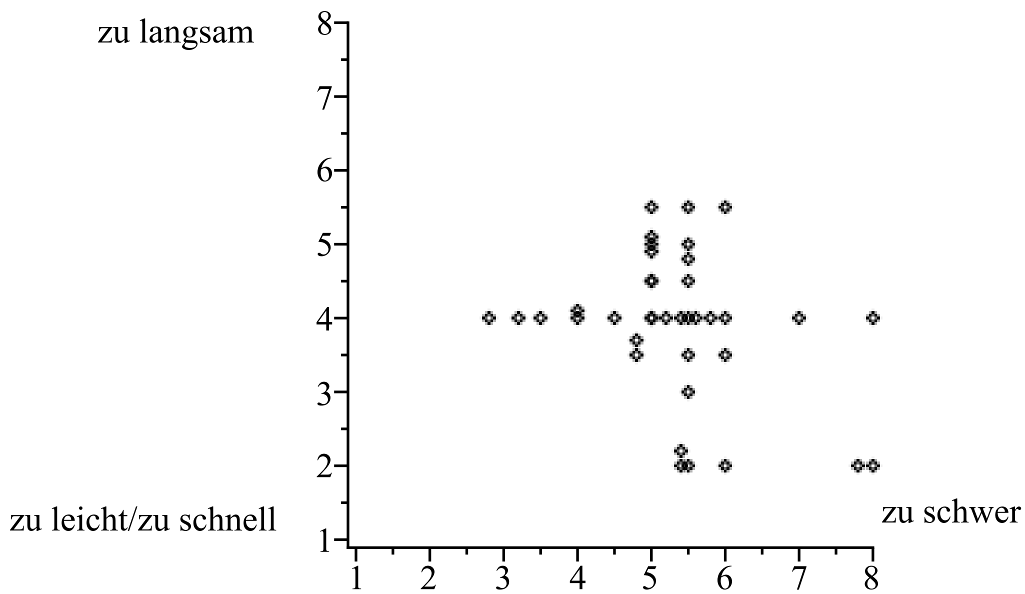
UEcloud := *pointplot*([*UE*]);

PLOT(...) **(5)**

display(*PtCloud*, *textplot*([8, 1.1, "zu schwer"], *align* = {*right*, *above*}), *textplot*([2, 7.6, "zu langsam   "], *align* = {*left*, *above*}), *textplot*([1, 1, "zu leicht/zu schnell   "], *align* = {*left*, *above*}), *scaling* = *constrained*);



display(*UEcloud*, *view* = [1 ..8, 1 ..8], *textplot*([8, 1.1, "zu schwer"], *align* = {*right*, *above*}),
textplot([2, 7.6, "zu langsam   "], *align* = {*left*, *above*}), *textplot*([1, 1,
"zu leicht/zu schnell   "], *align* = {*left*, *above*}), *scaling* = *constrained*);



Vectors and Matrices

> *restart*; *with*(*LinearAlgebra*);

[*&x*, *Add*, *Adjoint*, *BackwardSubstitute*, *BandMatrix*, *Basis*, *BezoutMatrix*, *BidiagonalForm*, *BilinearForm*, *CARE*, *CharacteristicMatrix*, *CharacteristicPolynomial*, *Column*, *ColumnDimension*, *ColumnOperation*, *ColumnSpace*, *CompanionMatrix*, *ConditionNumber*, *ConstantMatrix*, *ConstantVector*, *Copy*, *CreatePermutation*, *CrossProduct*, *DARE*, *DeleteColumn*, *DeleteRow*, *Determinant*, *Diagonal*, *DiagonalMatrix*, *Dimension*, *Dimensions*, *DotProduct*, *EigenConditionNumbers*, *Eigenvalues*, *Eigenvectors*, *Equal*, *ForwardSubstitute*, *FrobeniusForm*, *GaussianElimination*, *GenerateEquations*, *GenerateMatrix*, *Generic*, *GetResultDataType*, *GetResultShape*, *GivensRotationMatrix*, *GramSchmidt*, *HankelMatrix*, *HermiteForm*, *HermitianTranspose*, *HessenbergForm*, *HilbertMatrix*, *HouseholderMatrix*, *IdentityMatrix*, *IntersectionBasis*, *IsDefinite*, *IsOrthogonal*, *IsSimilar*, *IsUnitary*, *JordanBlockMatrix*, *JordanForm*, *KroneckerProduct*, *LA_Main*, *LUDecomposition*, *LeastSquares*, *LinearSolve*, *LyapunovSolve*, *Map*, *Map2*, *MatrixAdd*, *MatrixExponential*, *MatrixFunction*, *MatrixInverse*, *MatrixMatrixMultiply*, *MatrixNorm*, *MatrixPower*, *MatrixScalarMultiply*, *MatrixVectorMultiply*, **(6)** *MinimalPolynomial, Minor, Modular, Multiply, NoUserValue, Norm, Normalize, NullSpace, OuterProductMatrix, Permanent, Pivot, PopovForm, QRDecomposition, RandomMatrix, RandomVector, Rank, RationalCanonicalForm, ReducedRowEchelonForm, Row, RowDimension, RowOperation, RowSpace, ScalarMatrix, ScalarMultiply, ScalarVector, SchurForm, SingularValues, SmithForm, StronglyConnectedBlocks, SubMatrix, SubVector, SumBasis, SylvesterMatrix, SylvesterSolve, ToeplitzMatrix, Trace, Transpose, TridiagonalForm, UnitVector, VandermondeMatrix, VectorAdd, VectorAngle, VectorMatrixMultiply, VectorNorm, VectorScalarMultiply, ZeroMatrix, ZeroVector, Zip*]

> $f := x \to x^5 + x^2 - 3 \cdot x + 2; pp := plot(f(x), x = -2..3); ppp := plot(x^3 + x^2 - 3 \cdot x + 2, x = -1..2, color = black);$

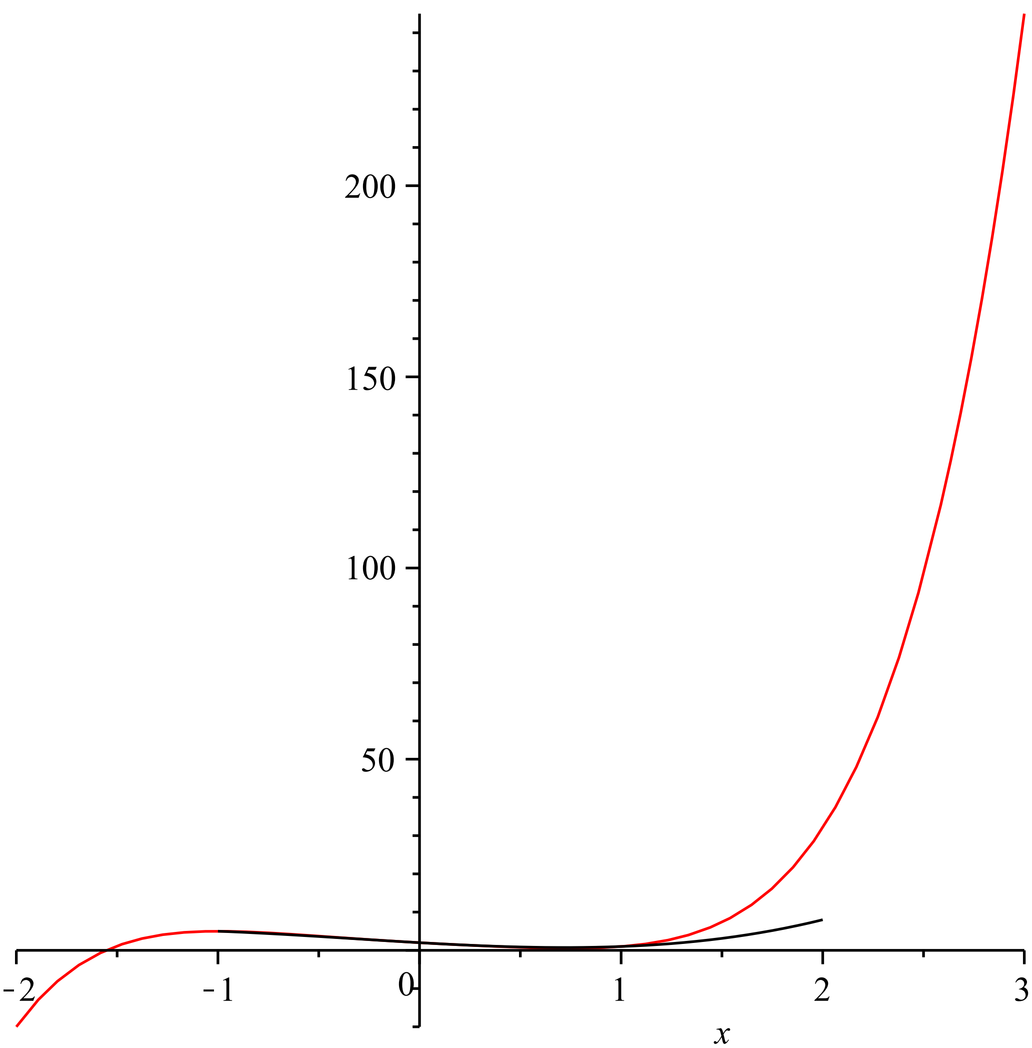
$$f := x \to x^5 + x^2 - 3\,x + 2$$
$$pp := PLOT(...)$$
$$ppp := PLOT(...) \tag{7}$$

> $display(pp);$

$$display(PLOT(...)) \tag{8}$$

> $with(plots):$

> $display(pp, ppp);$



Let us inspect (column) vectors.

> $p := \langle 0, 1 \rangle;\, r := \langle 1, 2 \rangle;$

$$p := \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$r := \begin{bmatrix} 1 \\ 2 \end{bmatrix} \tag{9}$$

> $p[1];$

$$0 \tag{10}$$

> $r[2];$

$$2 \tag{11}$$

> $p + r;$

$$\begin{bmatrix} 1 \\ 3 \end{bmatrix} \tag{12}$$

> $l := p + \lambda \cdot r;$

$$l := \begin{bmatrix} \lambda \\ 1 + 2\,\lambda \end{bmatrix} \tag{13}$$

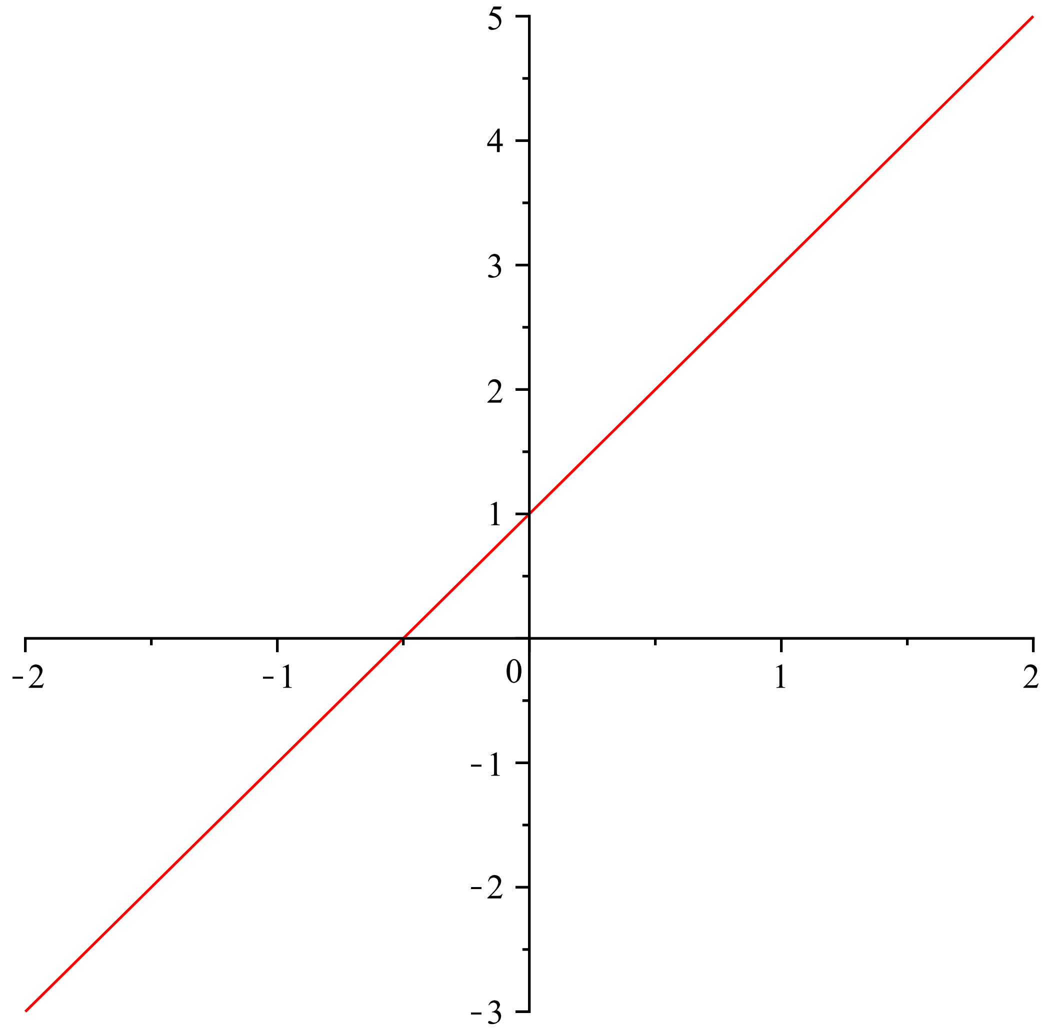
Now, we want to compute the shortest distance from point q := <2,1> to the line.

> $q := \langle 2, 1 \rangle;$

$$q := \begin{bmatrix} 2 \\ 1 \end{bmatrix} \tag{14}$$

> $lineplot := plot(\,[\,l[1],\, l[2],\, \lambda = -2\,..2\,]\,);\, display(lineplot);$

$$lineplot := PLOT(\ldots)$$



> $f := \lambda \to p + \lambda \cdot r;$

$$f := \lambda \to p + \lambda\, r \qquad \textbf{(15)}$$

> $s := seq([l[1], l[2]], \lambda = -2..2);$

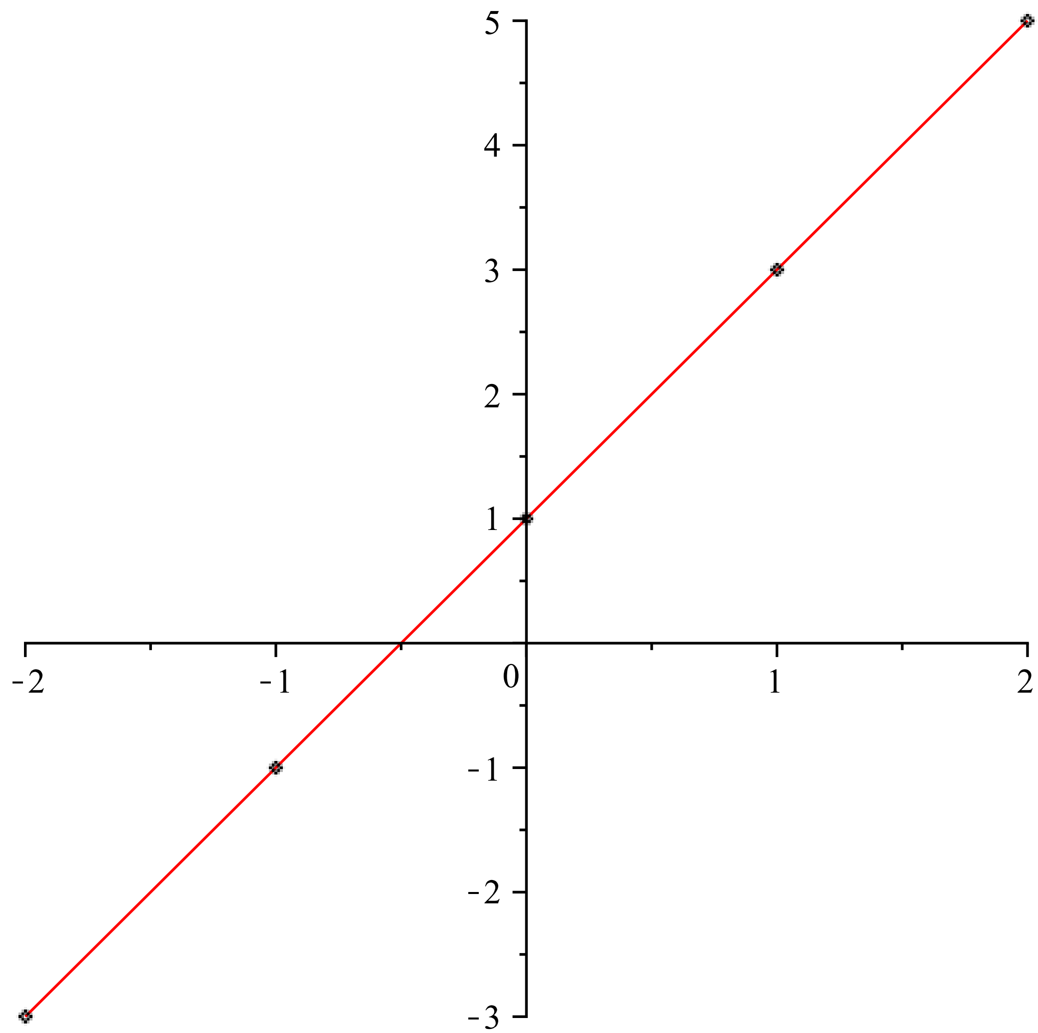
$$s := [-2, -3], [-1, -1], [0, 1], [1, 3], [2, 5] \qquad \textbf{(16)}$$

> $t := seq([f(x)[1], f(x)[2]], x = -2..2);$

$$t := [-2, -3], [-1, -1], [0, 1], [1, 3], [2, 5] \qquad \textbf{(17)}$$

> $pointline := pointplot([s]); display([lineplot, pointline]);$

$$pointline := PLOT(\ldots)$$



> $Qplot := pointplot(q);$

$$Qplot := PLOT(\ldots) \quad \textbf{(18)}$$

> $a1 := arrow([0, 0], p, width = [0.03, relative = true], head\_length = [0.2, relative = false], color = blue);$

$$a1 := PLOT(\ldots) \quad \textbf{(19)}$$

>

> $a2 := arrow(p, 0.25 \cdot r, width = [0.08, relative = true], head\_length = [0.2, relative = false], color = blue);$

$$a2 := PLOT(\ldots) \quad \textbf{(20)}$$

> $aHitQ := arrow\left(\left\langle \frac{2}{5}, \frac{9}{5} \right\rangle, q - \left\langle \frac{2}{5}, \frac{9}{5} \right\rangle, width = [0.0125, relative = true], head\_length = [0.1, relative = false], color = green\right);$

$$aHitQ := PLOT(\ldots) \quad \textbf{(21)}$$

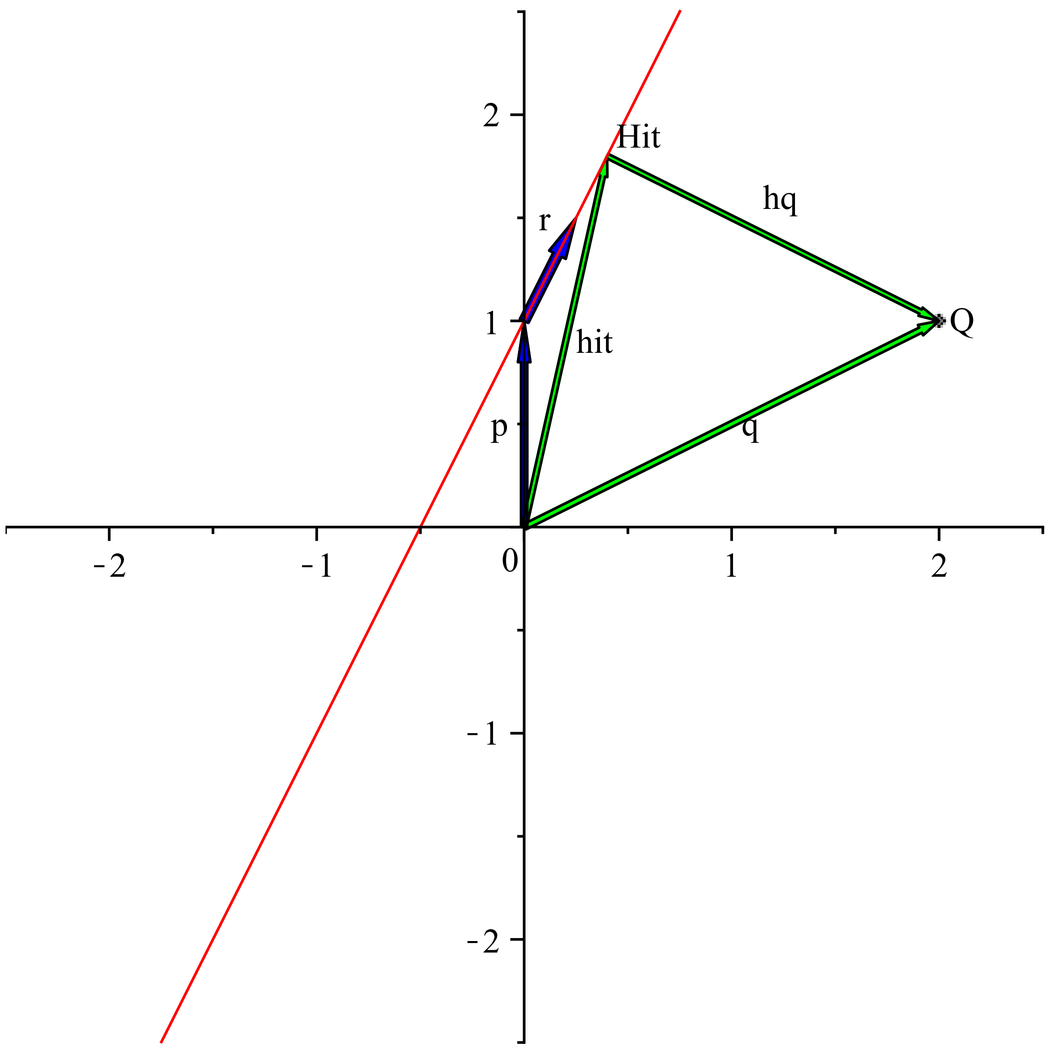
> $aQ := arrow(\langle 0, 0 \rangle, q, width = [0.0125, relative = true], head\_length = [0.1, relative = false], color = green);$

$$aQ := PLOT(\ldots) \quad \textbf{(22)}$$

> $aHit := arrow\left(\langle 0, 0\rangle, \left\langle \frac{2}{5}, \frac{9}{5} \right\rangle, width = [0.0125, relative = true], head\_length = [0.1, relative = false], color = green\right);$

$$aHit := PLOT(\ldots) \qquad \textbf{(23)}$$

> $display\left(a1, a2, aHit, aQ, aHitQ, Qplot, lineplot, view = [-2.5..2.5, -2.5..2.5], textplot([q[1], q[2], \text{"Q"}], align = \{right\}), textplot([1.1, 1.5, \text{"hq"}], align = \{right, above\}), textplot\left(\left[\frac{1}{5}, \frac{4}{5}, \text{"hit"}\right], align = \{right, above\}\right), textplot([1, 0.4, \text{"q"}], align = \{right, above\}), textplot\left(\left[\frac{2}{5}, \frac{9}{5}, \text{"Hit"}\right], align = \{right, above\}\right), textplot([-.02, 0.5, \text{"p"}], align = \{left\}), textplot([0.1, 1.4, \text{"r"}], align = \{above\})\right);$



>

Remarks:
q = hit+hq, thus hq=q-hit

moreover there is a lambda such that: hit = p + λr, thus hq = q - (p+λr)
now, we demand that hq ⊥ r, thus hq·r = 0, and therefore (q - (p+r)) · r = 0

Which λ does it? And what is the Point "Hit"?

> $\lambda := solve((q - (p + x \cdot r)).r = 0, x); p + r \cdot \lambda; f\left(\frac{2}{5}\right);$

$$\lambda := \frac{2}{5}$$

$$\begin{bmatrix} \frac{2}{5} \\ \frac{9}{5} \end{bmatrix}$$

$$\begin{bmatrix} \frac{2}{5} \\ \frac{9}{5} \end{bmatrix} \quad (24)$$

Remarks:
resorting (q - (p+λr)) · r = 0 leads to
(q - p - λ·r) . r = 0 and
q.r-p.r = λ·r.r and thus to $\lambda = \frac{(q - p).r}{r.r}$

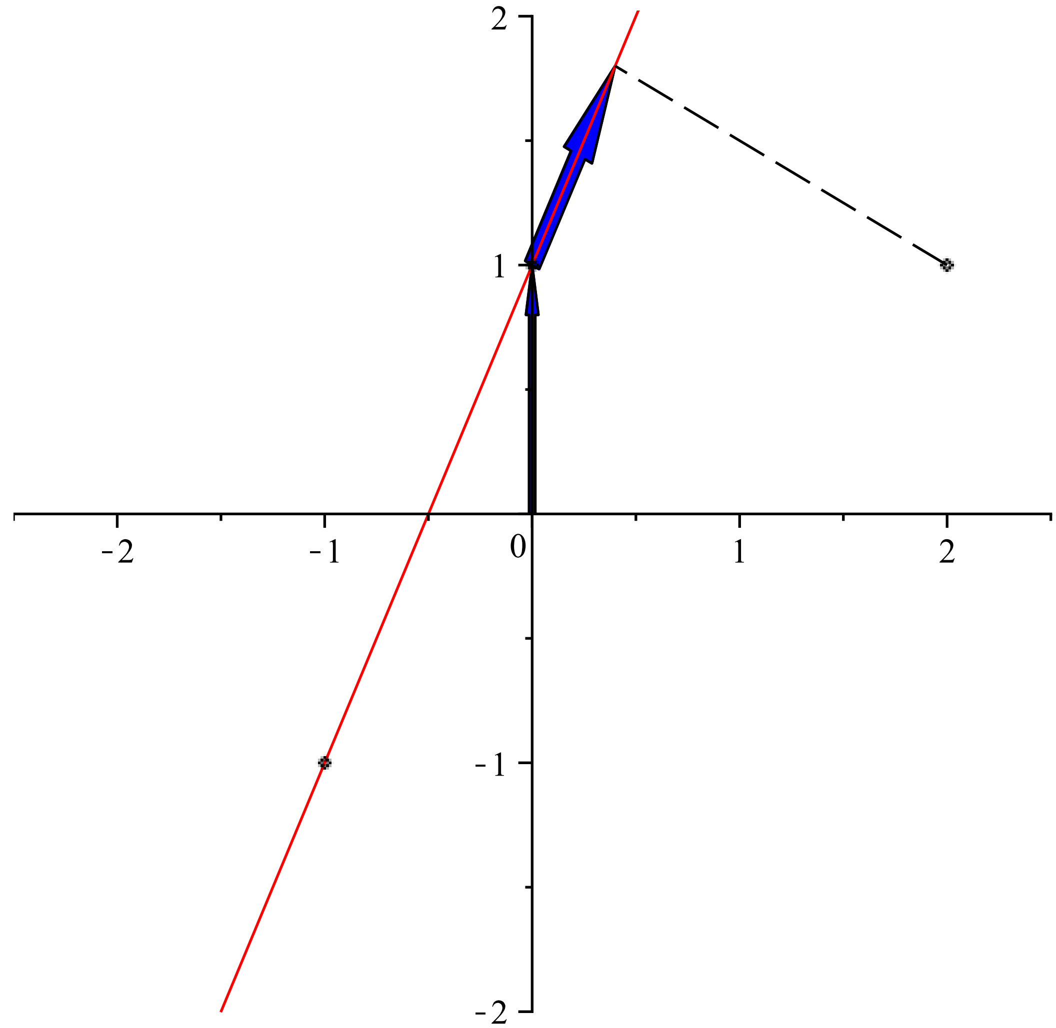
> $a22 := arrow\left(p, \frac{(q-p).r}{r.r} \cdot r, width = [0.075, relative = false], head\_length = [0.4, relative = false], color = blue\right);$

$$a22 := PLOT(\ldots) \quad (25)$$

> $HitPoint := subs\left(\lambda = \frac{DotProduct(q-p, r)}{DotProduct(r, r)}, l\right);$

$$HitPoint := \begin{bmatrix} \frac{2}{5} \\ \frac{9}{5} \end{bmatrix} \quad (26)$$

> $display([a1, a22, Qplot, lineplot, pointline, pointplot([q, HitPoint], connect = true, thickness = 1, linestyle = dash)], view = [-2.5 ..2.5, -2 ..2]);$



> *# at the end, we could plot the big arrow from (0,1), pointing exactly into the HitPoint*

>

> $\frac{DotProduct(q-p, r)}{DotProduct(r, r)}, \frac{(q-p).r}{r.r};$

$$\frac{2}{5}, \frac{2}{5} \tag{27}$$

$A := \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix}; B := \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix};$

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix}$$

$$\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \qquad \textbf{(28)}$$

$$B := \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix};$$

$$A := \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix} \qquad C := A.B$$

$$\begin{bmatrix} a_{11}\, b_{11} + a_{12}\, b_{21} & a_{11}\, b_{12} + a_{12}\, b_{22} \\ a_{21}\, b_{11} + a_{22}\, b_{21} & a_{21}\, b_{12} + a_{22}\, b_{22} \\ a_{31}\, b_{11} + a_{32}\, b_{21} & a_{31}\, b_{12} + a_{32}\, b_{22} \end{bmatrix} \qquad \textbf{(29)}$$

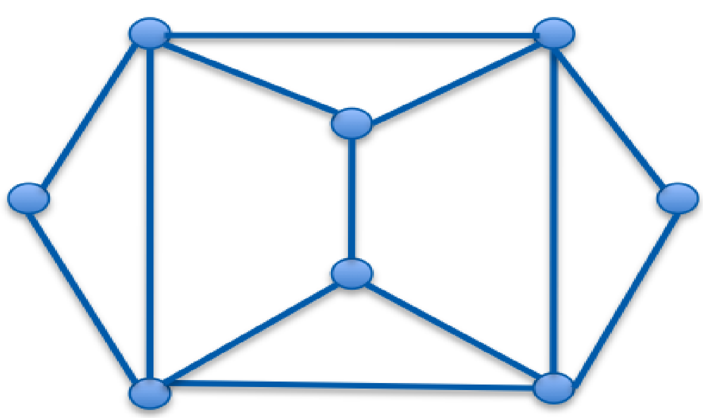
in general: $c_{ik} := a_{i1} \cdot b_{1\,k} + \ldots + a_{\mathbf{in}} \cdot b_{nk}$

cf. G. Fischer, Lineare Algebra, S. 71ff, Verknüpfungen von Matrizen

Graphs

What is a graph?

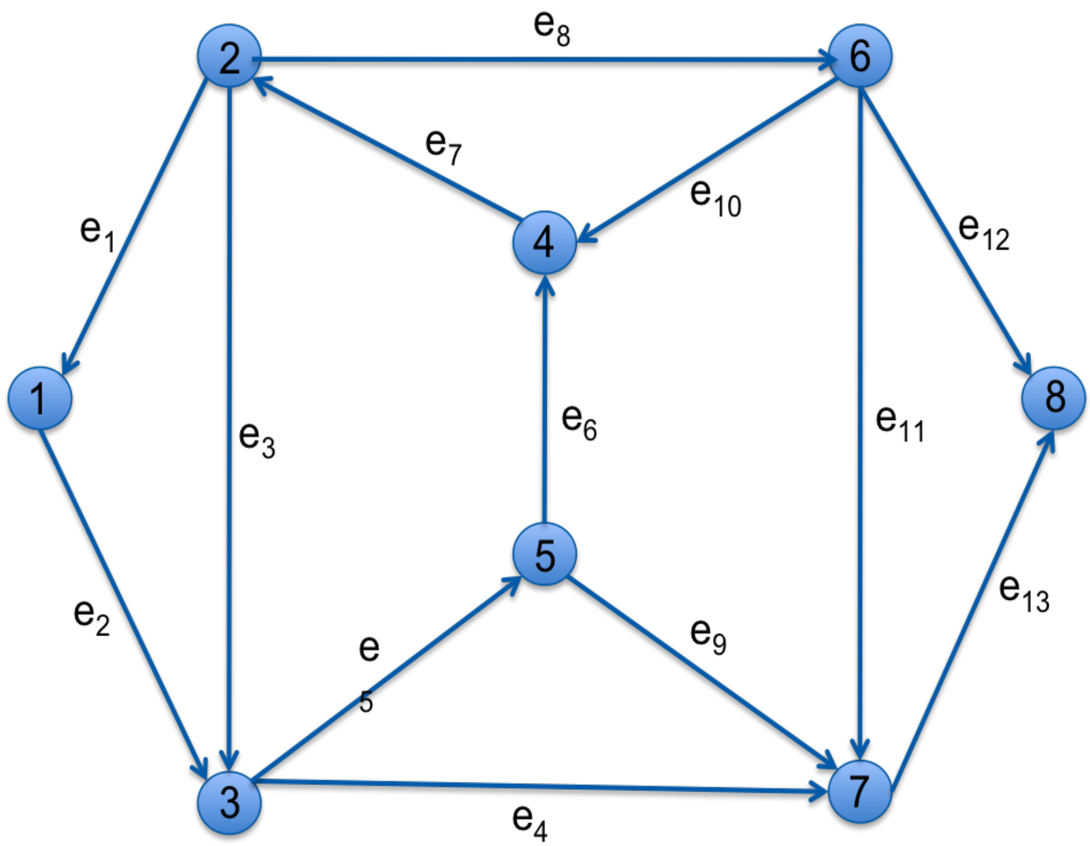
An undirected graph consists of a pair G=(V,E), where E ⊆{{u,v} | u,v∈ V}.
The elements of E are not ordered..



Elements from V are called nodes (or vertices; Knoten in dt.), elements from E are called edges (Kanten in dt.)

A directed graph (gerichteter Graph) is as well a pair G=(V,E). However, the elements of E are ordered pairs of elements from V. Thus, E ⊆ {(u,v) | u,v ∈ V}.
Elemente of V are called nodes, elements from E are called edges (im dt.: gerichtete Kanten oder Bögen)

.



Repetition: Flow Control (if, for, while, ...)

if <conditional expression> **then** <statement sequence>
| **elif** <conditional expression> **then** <statement sequence> |
| **else** <statement sequence> |
end if

(Note: Phrases located between | | are optional.)

> $a := 0;$

$$a := 0 \quad \textbf{(30)}$$

> **if** $(a > 0)$ **then** $f := x^2$ **fi**;

> **if** $(a = 0)$ **then** $f := x^2$ **fi**;

$$f := x^2 \quad \textbf{(31)}$$

> **if** $(a < 9)$ **then**
$f := x^2 + 1;$ # *";" is necessary, because: several statements without structure*
$g := x^2$ # *";" not necessary*

```
else
  g := x^2 + 1;
  f := x^2;
end if;
```

$$f := x^2 + 1$$
$$g := x^2 \quad (32)$$

The **for ...while ... do** loop

>

>

1) Print even numbers from 6 to 10.

```
> for i from 6 by 2 to 10 do print(i) end do;
```

6
8
10 (33)

2) Find the sum of all two-digit odd numbers from 11 to 99.

```
> mysum := 0;
  for i from 11 by 2 while i < 100 do
    mysum := mysum + i
  end do:
  mysum;
```

$mysum := 0$
2475 (34)

3) Multiply the entries of an expression sequence.

```
> restart;
  total := 1 :
  for z in 1, x, y, q^2, 3 do
    total := total·z
  end do:
  total;
  x := 2 :
  q := 3 :
  total;
```

$$3\, x\, y\, q^2$$
$$54\, y \quad (35)$$

3) Add together the contents of a list.

```
> ?cat
```

```
> restart;
  y := 3;
  myconstruction := "";
  for z in [1, "+", y, "·", "q^2", "·", 3] do
    myconstruction := cat(myconstruction, z) ;
  end do;
  myconstruction;
```

$$y := 3$$
$$myconstruction := ""$$
$$myconstruction := "1"$$
$$myconstruction := "1+"$$
$$myconstruction := "1+3"$$
$$myconstruction := "1+3*"$$
$$myconstruction := "1+3*q^2"$$
$$myconstruction := "1+3*q^2*"$$
$$myconstruction := "1+3*q^2*3"$$
$$"1+3*q^2*3" \quad (36)$$

```
> ?parse
> q := 4;
```

$$q := 4 \quad (37)$$

```
> qq := parse(myconstruction);
```

$$qq := 1 + 9\,q^2 \quad (38)$$

```
> qq;
```

$$145 \quad (39)$$

Similar-to-Fibonacci-Numbers are
sff[1] := 0; *sff*[2] := 1; *sff*[3] := 1; *sff*[*i*] := *sff*[*i* − 1] + *sff*[*i* − 2] + 1

```
> restart; sff := [seq(0, i = 1 ..100)]:
>
> sff[2] := 1; sff[3] := 1;
```

$$sff_2 := 1$$
$$sff_3 := 1 \quad (40)$$

```
> sff[4] := sff[2] + sff[3] + 1;
```

$$sff_4 := 3 \quad (41)$$

```
> for i from 4 to 100 do
    sff[i] := sff[i − 1] + sff[i − 2] + 1;
    if i = 97 then print(sff[i]) fi;
  end do:
```

$$10336141770971664 6143 \quad (42)$$

```
> sff1 := sff[1]; sff2 := sff[2]; sff3 := sff[3];
```

$$sff1 := 0$$
$$sff2 := 1$$
$$sff3 := 1 \quad (43)$$

```
> for i from 4 to 10000 do
    sff4 := sff3 + sff2 + 1;
    sff2 := sff3 : sff3 := sff4;
```

```
    if i = 97 then print(sff3) fi;
end do:
```