

Vector Geometry using Computers

(Book project in construction)

Walter Gander

`gander@inf.ethz.ch`

ETH Fachdidaktik Informatik

14. Oktober 2020

Motivation

- Vector geometry is **constructive**, many interesting problems, **nice algorithms**
- Classical books on vector geometry **don't use computers**
- Using computers is a **good training** for
 - **vector geometry** (full **understanding** of concepts necessary)
 - **programming exercises** (implement small nice algorithms)
- **New algorithms can be developed and applied** (computers are not restricted to only use algorithms which are suited for hand computations)

Rotations (Givensrotations), not suited for hand-computations!

$$\begin{array}{ccc}
 G_1 & G_2 & G_3 \\
 \begin{pmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} & \begin{pmatrix} \cos \alpha & 0 & -\sin \alpha \\ 0 & 1 & 0 \\ \sin \alpha & 0 & \cos \alpha \end{pmatrix} & \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha \\ 0 & \sin \alpha & \cos \alpha \end{pmatrix}
 \end{array}$$

rotation around x_3
in x_1x_2 -plane

rotation around x_2
in x_1x_3 -plane

rotation around x_1
in x_2x_3 -plane

- $$G_3G_2G_1 \begin{pmatrix} 2 & -2 & 0 \\ 4 & -6 & -1 \\ 8 & 4 & -6 \end{pmatrix} = \begin{pmatrix} -9.17 & -0.44 & 5.67 \\ 0 & -7.47 & 2.08 \\ 0 & 0 & 0.70 \end{pmatrix}$$

Rotate column vectors to upper triangular matrix

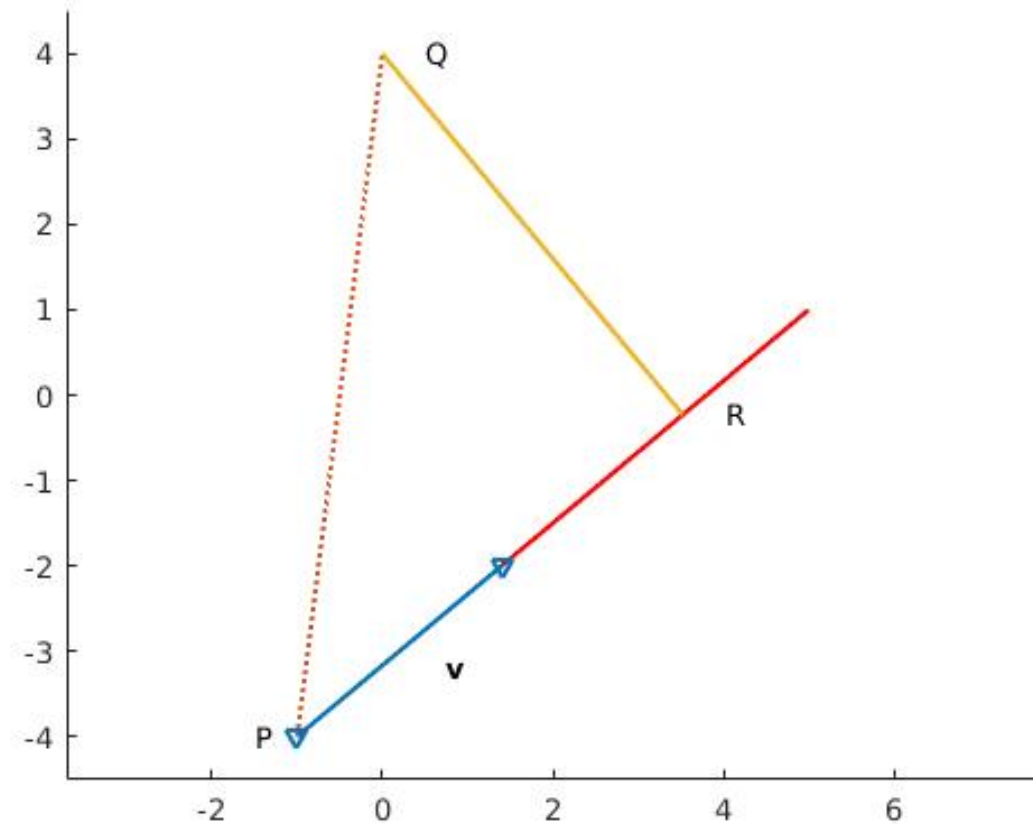
- Remarks about Descriptive Geometry!

Program for Givens-Reduction of a Linear System

```
function [R,c]=GivensReduction(A,b)
% GIVENSREDUCTION reduces the linear system A x= b to
% upper triangular form R x = c
[m,n]=size(A); [m,p]=size(b);
R=[A,b]; % append right hand sides
for i=1:n % for all columns
    for k=i+1:m % rotate R(k,i) to 0
        if R(k,i)~=0 % skip if already 0
            cot=-R(i,i)/R(k,i);
            si=1/sqrt(1+cot^2); co=si*cot;
            G=[co,-si;si,co]; % Givens rotation matrix
            R(i:k-i:k,i:n+p)=G*R(i:k-i:k,i:n+p);
        end % apply to rows i and k
    end;
end
c=R(:,n+1:n+p); R=R(:,1:n);
```

Point and Straight Line

- Given point Q and straight line $g: \mathbf{X} = \mathbf{P} + \lambda \mathbf{v}$
- Compute projected point R and distance
- Solve for 2D and 3D



Point and Straight Line in 2D, Traditional Solution

```
function [d,R]=PointLine2D(Q,P,v)
% POINTLINE2D computes the distance of point Q from the
%           line  $X=P+\text{lam}*v$  by projecting on the normal
%           and the projected point R
v=v/norm(v);           % unit vector
n=[v(2);-v(1)];       % construct normal
d=abs(n'*(Q-P));      % project on normal
A=[n,-v];             % R=intersect lines  $Q+nu\ n$  with  $P+ \text{lam}\ v$ 
h=A'*(P-Q);          % A is orthogonal so  $\text{inv}(A)=A^T$ 
R=P+h(2)*v;          %  $h=[nu;\text{lam}]$ 
```

Point and Straight Line in 3D, Traditional Solution

- Construct **orthogonal plane** to g through Q : $\mathbf{n} = \mathbf{v}$,
 $\implies n_1x_1 + n_2x_2 + n_3x_3 + c = 0$
- insert point Q to get $c = -\mathbf{Q}^T \mathbf{n}$,
- intersect plane with line $g \implies \mathbf{R}$ **projected point**
- **distance** = $\|\mathbf{Q} - \mathbf{R}\|$

```
function [d,R]=PointLineConv(Q,P,v)
% POINTLINECONV computes the distance d and the projected
% point R of a point Q from the straight line g: X=P+lambda v.
n=v; % normal of orth. plane to g
c=-Q'*n; % insert Q to get general normal form of plane
% n_1x_1+n_2x_2+n_3x_3+c=0
lam=-(n'*P+c)/(n'*v); % cut plane with g
R=P+lam*v; % intersection point R
d=norm(Q-R); % compute distance
```

Point and Straight Line, new Algorithm

- \mathbf{Q} on $g \iff \mathbf{Q} = \mathbf{P} + \lambda \mathbf{v} \iff \lambda \mathbf{v} = \mathbf{Q} - \mathbf{P}$

- $$\begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} \lambda = \begin{pmatrix} q_1 - p_1 \\ q_2 - p_2 \\ q_3 - p_3 \end{pmatrix} \text{Givensreduction} \implies \begin{pmatrix} b_1 \\ 0 \\ 0 \end{pmatrix} \lambda = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix}$$

- $\lambda = c_1/b_1$, projected point $\mathbf{R} = \mathbf{P} + \lambda \mathbf{v}$. Distance $d = \sqrt{c_2^2 + c_3^2}$

```
function [d,R]=PointLine(Q,P,v)
% POINTLINE computes the distance d and the projected
% point R of a point Q from the straight line g: X=P+lambda v.
[b,c]=GivensReduction(v,Q-P);
lambda=c(1)/b(1);           % solve for lambda
R=P+lambda*v;              % projected point
d=norm(c(2:end));          % distance
```

- The function `PointLine` works without changes also for 2D!

Straight Line: Conversion Parametric \rightarrow Normal Form

- $\mathbf{X} = \mathbf{P} + \lambda \mathbf{v} \quad \rightarrow \quad n_1 x_1 + n_2 x_2 + c = 0$
- $\mathbf{n} = \begin{pmatrix} v_2 \\ -v_1 \end{pmatrix} \implies n_1 x_1 + n_2 x_2 + c = 0$
- Insert \mathbf{P} to get $c = -\mathbf{P}^\top \mathbf{n} \implies$ general normal form
- **Normalize** to get normal form

$$\frac{n_1}{\sqrt{n_1^2 + n_2^2}} x_1 + \frac{n_2}{\sqrt{n_1^2 + n_2^2}} x_2 + \frac{c}{\sqrt{n_1^2 + n_2^2}} = 0$$

Parametric \rightarrow Normal Form, New Algorithm

- Rearrange $\mathbf{X} = \mathbf{P} + \lambda \mathbf{v}$ to
$$\begin{pmatrix} -v_1 & 1 & 0 \\ -v_2 & 0 & 1 \end{pmatrix} \begin{pmatrix} \lambda \\ x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} p_1 \\ p_2 \end{pmatrix}$$

- Use **Givensreduction** (one rotation) to eliminate λ

$$\begin{pmatrix} c & -s \\ s & c \end{pmatrix} \implies \begin{pmatrix} -cv_1 + sv_2 & c & -s \\ 0 & s & c \end{pmatrix} \begin{pmatrix} \lambda \\ x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} p'_1 \\ p'_2 \end{pmatrix}$$

- **Normal form** is $s x_1 + c x_2 - p'_2 = 0$

```
function g=Para2Normal2D(P,v)
% PARA2NORMAL2D converts a straight line in parameter form
% X=P+lambd v to normal form g=[n_1,n_2,c] with X'n+c=0.
B=[-v, eye(2)];
[B,b]=GivensReduction(B,P);
g=[B(2,2:3), -b(2)];
```

Plane: Parametric to Normal Form, Traditional Approach

- $\mathbf{X} = \mathbf{P} + \lambda \mathbf{r} + \mu \mathbf{s} \rightarrow g : n_1 x_1 + n_2 x_2 + n_3 x_3 + c = 0$
- $\mathbf{n} = \mathbf{r} \times \mathbf{s}$ **cross product**
- **insert point P** to get c

```
function g=Par2Normal3DConv(P,r,s)
% PAR2NORMALCONV converts the plane X=P+lambda r + mu s to
% normal form g=[n', c] where X'n+c=0 with ||n||=1.
```

```
n=kreuz(r,s); n=n(:);           % normal
c=-P'*n;                       % insert P
g=[n',c];
g=g/norm(n);                   % normal form
end
```

```
function n=kreuz(u,v);
% KREUZ computes the cross product of u and v.
% corresponds to Matlab function cross
n(1)=u(2)*v(3)-u(3)*v(2);
n(2)=u(3)*v(1)-u(1)*v(3);
n(3)=u(1)*v(2)-u(2)*v(1);
n=n(:);
end
```

Plane: Parametric to Normal Form, New Algorithm

- Rearrange plane equations $\mathbf{X} = \mathbf{A} + \lambda \mathbf{r} + \mu \mathbf{s}$ (3 equations, 5 unknowns)

$$\begin{pmatrix} -r_1 & -s_1 & 1 & 0 & 0 \\ -r_2 & -s_2 & 0 & 1 & 0 \\ -r_3 & -s_3 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \lambda \\ \mu \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix}$$

- Eliminate λ and μ using Givensreduction (3 rotations)
- The remaining third equation is the normal form (Hessenberg form).

```
function g=Par2Normal3D(A,r,s)
% PAR2NORMAL converts the plane X=A+lambda r + mu s to
% normal form g=[n', c] where X'n+c=0 with ||n||=1.
C=[-r, -s, eye(3)];           % form 3x5 matrix
[B,b]=GivensReduction(C,A);   % reduce system
g=[B(3,3:5), -b(3)];          % read the coefficients
```

Plane: from Normal to Parametric Form

$$E : n_1x_1 + n_2x_2 + n_3x_3 + c = 0 \quad \rightarrow \quad \mathbf{X} = \mathbf{P} + \lambda\mathbf{r} + \mu\mathbf{s}$$

- choose 3 points A , B and C on E in general position

$$\implies \mathbf{X} = \mathbf{A} + \lambda(\mathbf{B} - \mathbf{A}) + \mu(\mathbf{C} - \mathbf{A})$$

Good for computing by hand.

- For foolproof program make sure $\mathbf{r}, \mathbf{s} \perp \mathbf{n}$, and linear independent

Case 1: $n_k \neq 0, k = 1, 2, 3$

$$\mathbf{r} = \begin{pmatrix} 0 \\ n_3 \\ -n_2 \end{pmatrix} \quad \mathbf{s} = \begin{pmatrix} n_3 \\ 0 \\ -n_1 \end{pmatrix}.$$

Case 2: $n_1 = 0$

$$\mathbf{r} = \begin{pmatrix} 0 \\ n_3 \\ -n_2 \end{pmatrix} \quad \mathbf{s} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}.$$

Case 3: $n_2 = 0$

$$\mathbf{r} = \begin{pmatrix} -n_3 \\ 0 \\ n_1 \end{pmatrix} \quad \mathbf{s} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}.$$

Case 4: $n_3 = 0$

$$\mathbf{r} = \begin{pmatrix} -n_2 \\ n_1 \\ 0 \end{pmatrix} \quad \mathbf{s} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$

Conversion from Normal to Parametric Form(cont.)

```
function [P,r,s]=Norm2Para3D(g)
% NORM2PARA3D computes a parametric form of a plane from the normal form
n=g(1:3);
if abs(n(1))<1e-10           % construct linear independent
    r=[0;n(3);-n(2)]; s=[1;0;0]; % direction vectors orth. to normal
elseif abs(n(2))<1e-10
    r=[-n(3);0;n(1)]; s=[0;1;0];
elseif abs(n(3))<1e-10
    r=[-n(2);n(1);0]; s=[0;0;1];
else
    r=[0;n(3);-n(2)]; s=[n(3);0;-n(1)];
end

if abs(n(1))>1e-10           % choose point P on plane
    P=[-g(4)/n(1);0;0];
elseif abs(n(2))>1e-10
    P=[0;-g(4)/n(2);0];
else
    P=[0;0;-g(4)/n(3)];
end
```

Distance Point to Plane, Traditional Solution

- Given point \mathbf{Q} and plane $g : \mathbf{X} = \mathbf{P} + \lambda \mathbf{r} + \mu \mathbf{s}$
- Convert parametric to normal form
 - $\mathbf{n} = \mathbf{r} \times \mathbf{s}$ cross product
 - $\mathbf{n} = \mathbf{n} / \|\mathbf{n}\|$ normalize
 - insert $\mathbf{P} \implies c = -\mathbf{P}^\top \mathbf{n}$
 \implies normal form $n_1 x_1 + n_2 x_2 + n_3 x_3 + c = 0$
- Distance $d = |\mathbf{Q}^\top \mathbf{n} + c|$
- Intersect straight line $\mathbf{X} = \mathbf{Q} + \lambda \mathbf{n}$ with plane gives projected point \mathbf{Q}' .

```
function [d,Qp]=PointPlane3DConv(Q,P,r,s)
% POINTPLANE3DCONV computes the distance and the projected point Qp
% of the point Q from the plane X=P+lambda r + mu s.
% Traditional algorithm.
n=kreuz(r,s);           % normal vector
n=n(:); n=n/norm(n)    % normalize
c=-P'*n;               % insert P, get normal form of plane
d=abs(n'*Q+c);         % distance
lambda=-c-n'*Q;       % compute intersection point
Qp=Q+lambda*n;
end

function n=kreuz(u,v);
% KREUZ computes the cross product of u and v.
% One can also use the Matlab function cross
n(1)=u(2)*v(3)-u(3)*v(2);
n(2)=u(3)*v(1)-u(1)*v(3);
n(3)=u(1)*v(2)-u(2)*v(1);
end
```


Distance Point to Plane, New Algorithm

- Given point \mathbf{Q} and plane $g : \mathbf{X} = \mathbf{P} + \lambda \mathbf{r} + \mu \mathbf{s}$

- \mathbf{Q} on g ? $\iff \begin{pmatrix} r_1 & s_1 \\ r_2 & s_2 \\ r_3 & s_3 \end{pmatrix} \begin{pmatrix} \lambda \\ \mu \end{pmatrix} = \begin{pmatrix} q_1 - p_1 \\ q_2 - p_2 \\ q_3 - p_3 \end{pmatrix}.$

- **Givensreduction** $\implies \begin{pmatrix} a_{11} & a_{12} \\ 0 & a_{22} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \lambda \\ \mu \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix}.$

- $a_{22} \neq 0$ (direction vectors are not parallel)
- $c_3 = 0$: Q is on the plane.
- $c_3 \neq 0$: $d = |c_3|$ is distance, λ and μ give projected point Q'

```
function [d,Qp]=PointPlane3D(Q,P,r,s)
% POINTPLANE3D computes the distance and the projected point Qp
% of the point Q from the plane X=P+lambda r + mu s.
A=[r s]; c=(Q-P); % lin. system for lambda and mu
[B,b]=GivensReduction(A,c);
mue=b(2)/B(2,2); % backsubstitution
lambda=(b(1)-B(1,2)*mue)/B(1,1);
Qp=P+lambda*r+mue*s; % projected point
d=abs(b(3)); if d<1e-10, d=0;end
```

Plane and Straight Line

- Given plane $E : \mathbf{X} = \mathbf{P} + \lambda \mathbf{r} + \mu \mathbf{s}$ and straight line $g : \mathbf{X} = \mathbf{Q} + \rho \mathbf{u}$

- Intersection?** $\implies \mathbf{P} + \lambda \mathbf{r} + \mu \mathbf{s} = \mathbf{Q} + \rho \mathbf{u}$

$$\iff \begin{pmatrix} r_1 & s_1 & -u_1 \\ r_2 & s_2 & -u_2 \\ r_3 & s_2 & -u_2 \end{pmatrix} \begin{pmatrix} \lambda \\ \mu \\ \rho \end{pmatrix} = \begin{pmatrix} q_1 - p_1 \\ q_2 - p_2 \\ q_3 - p_3 \end{pmatrix}$$

- Givensreduction**

$$\implies \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22} & a_{23} \\ 0 & 0 & a_{33} \end{pmatrix} \begin{pmatrix} \lambda \\ \mu \\ \rho \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} \quad a_{11} \neq 0, a_{22} \neq 0$$

- $a_{33} \neq 0$: $\rho = c_3/a_{33}$, **intersection point** $R = \mathbf{Q} + \rho \mathbf{u}$
- $a_{33} = 0$: line g parallel to E , **distance** $d = |c_3|$
- Traditional Algorithm with Gaussian Elimination: **no distance!**

Two Planes $E_1 : \mathbf{X} = \mathbf{P} + \lambda \mathbf{r} + \mu \mathbf{s}$ $E_2 : \mathbf{X} = \mathbf{Q} + \rho \mathbf{u} + \nu \mathbf{v}$.

- Intersection: $\mathbf{P} + \lambda \mathbf{r} + \mu \mathbf{s} = \mathbf{Q} + \rho \mathbf{u} + \nu \mathbf{v}$
 $\iff \mathbf{r}\lambda + \mathbf{s}\mu - \mathbf{u}\rho - \mathbf{v}\nu = \mathbf{Q} - \mathbf{P}$

- Givens reduction

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \end{pmatrix} \begin{pmatrix} \lambda \\ \mu \\ \rho \\ \nu \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix}, \quad a_{11} \neq 0, a_{22} \neq 0$$

- $a_{33} \neq 0 \implies \rho = \frac{c_3}{a_{33}} - \frac{a_{34}}{a_{33}}\nu$, insert in E_2

intersection line $\mathbf{X} = \left(\mathbf{Q} + \frac{c_3}{a_{33}} \mathbf{u} \right) + \nu \left(\mathbf{v} - \frac{a_{34}}{a_{33}} \mathbf{u} \right)$

- $a_{33} = 0$: $\implies \mathbf{u}$ is linear dependent of \mathbf{r} and \mathbf{s}

If $a_{34} \neq 0 \implies \nu = \frac{c_3}{a_{34}}$, intersection line $\mathbf{X} = \left(\mathbf{Q} + \frac{c_3}{a_{34}} \mathbf{v} \right) + \rho \mathbf{u}$

If $a_{34} = 0 \implies E_1$ parallel E_2 , distance $d = |c_3|$

```
function [S,w,d]=TwoPlanesPara3D(P,r,s,Q,u,v)
% TWOPLANESPARA3D computes the intersection line X=S+sigma w
% of the two planes E1=P+lamba r + mue s and E2=Q+rho u + nu v.
% If the planes are parallel the distance d is computed
A=[r, s, -u, -v]; c=[Q-P];
[A,c]=GivensReduction(A,c);
if abs(A(3,3))>1e-10           % normal case
    d=[];
    S=Q+c(3)/A(3,3)*u;
    w=v-A(3,4)/A(3,3)*u ;
elseif abs(A(3,4))>1e-10     % solve for nu
    d=[];
    S=Q+c(3)/A(3,4)*v;
    w=u;
else                           % planes are parallel
    d=abs(c(3)); S=[];w=[];
end
```

Two Planes, Traditional Approach

$$E_1 : \mathbf{X} = \mathbf{P} + \lambda \mathbf{r} + \mu \mathbf{s} \quad E_2 : \mathbf{X} = \mathbf{Q} + \rho \mathbf{u} + \nu \mathbf{v}.$$

1. Convert the planes to normal form, normal vectors \mathbf{n}_{E_1} and \mathbf{n}_{E_2}
2. If \mathbf{n}_{E_1} and \mathbf{n}_{E_2} not parallel $\implies \exists$ intersection line.
 - Find general solution of normal equations of E_1 and E_2
 - Or find common point on E_1 and E_2 and $n = \mathbf{n}_{E_1} \times \mathbf{n}_{E_2}$
3. If \mathbf{n}_{E_1} and \mathbf{n}_{E_2} parallel, check if normal equations are multiples of each other:
 - (a) if multiple $\implies E_1$ and E_2 are coincident
 - (b) no multiple $\implies E_1$ and E_2 are parallel
 - (c) distance? compute e.g. distance of \mathbf{P} from E_2

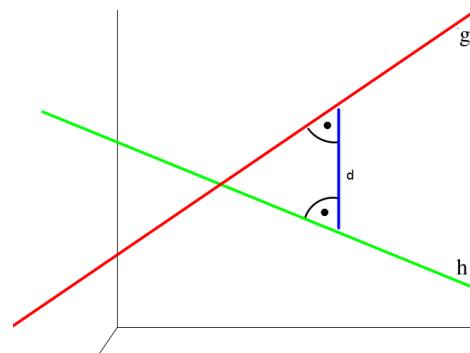
Source: abimatura | Mathe lernen

<https://www.abiturma.de/mathe-lernen/geometrie/lagebeziehungen-und-schnitt/lagebeziehung-ebene-ebene>

Two straight Lines in Space $g : \mathbf{X} = \mathbf{P} + \lambda \mathbf{r}$, $h : \mathbf{X} = \mathbf{Q} + \mu \mathbf{s}$.

- **Intersection?** $\mathbf{P} + \lambda \mathbf{r} = \mathbf{Q} + \mu \mathbf{s}$

$$\begin{pmatrix} r_1 & -s_1 \\ r_2 & -s_2 \\ r_3 & -s_3 \end{pmatrix} \begin{pmatrix} \lambda \\ \mu \end{pmatrix} = \begin{pmatrix} q_1 - p_1 \\ q_2 - p_2 \\ q_3 - p_3 \end{pmatrix}$$



- **Givensreduction** $\implies \begin{pmatrix} a_{11} & a_{12} \\ 0 & a_{22} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \lambda \\ \mu \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix}$.

- **Interpretation:**

- If $a_{22} \neq 0$ compute λ and μ . **Intersection point** if $c_3 = 0$.
If $c_3 \neq 0$, **lines skewed**, λ and μ give nearest points, distance $d = |c_3|$.
- If $a_{22} = 0 \implies$ lines parallel. Distance $d = \sqrt{c_2^2 + c_3^2}$.

```
function [d,R,S,Ac]= TwoLines3D(P,r,Q,s)
% TWOLINES3D computes the position of the straight lines
% X=P+lambda r and X=Q+nu s in space.
% R,S are the closest points on the lines
% d distance, d=||R-S||
% Ac is matrix of the reduced system together with right hand side c
%
A=[r,-s]; c=[Q-P]; % equations for lambda and mu
[A,c]=GivensReduction(A,c);
Ac = [A, c]; % reduced system
if abs(A(2,2))>1e-10 % can solve for lambda and mu
    mu=c(2)/A(2,2)
    lambda=(c(1)-A(1,2)*mu)/A(1,1)
    R=P+lambda*r; S=Q+mu*s;
    d=abs(c(3)); % d is equal to norm(R-S)
else % direction vectors are parallel
    d=norm(c(2:3)); % distance between parallel lines
    R=[]; S=[];
end
```


Two straight Lines, Traditional $g : \mathbf{X} = \mathbf{P} + \lambda \mathbf{r}$, $h : \mathbf{X} = \mathbf{Q} + \mu \mathbf{s}$.

Compare direction vectors \mathbf{r} and \mathbf{s}

- if **multiples** \implies g and h are parallel or coincident
 - point \mathbf{P} on h : coincident
 - point \mathbf{P} not on h : parallel
- \mathbf{r} and \mathbf{s} **not multiples** \implies intersection or warp
- solve $\mathbf{P} + \lambda \mathbf{r} = \mathbf{Q} + \mu \mathbf{s}$ and decide if intersect.

Handwritten notes on a video player screen showing a flowchart for comparing two lines. The flowchart starts with "RV anschauen" (check direction vectors). If they are multiples ("Vielfache"), the lines are either parallel ("// oder /") or coincident ("identisch"). If not multiples ("keine Vielfache"), the lines are either intersecting ("X oder /") or skew ("windschief"). The "X oder /" case leads to "Gleichsetzen" (equating) and "Schnittpunkt" (intersection point). The "windschief" case leads to "Punktprobe" (point test) and "parallel".

Video player interface showing the video title: "Lage von 2 Geraden, Vektorgeometrie, Parameterformen vergleichen, Ablauf | Mathe by Daniel Jung". The video has 417,846 views and was uploaded on 04.06.2012. The video player shows a progress bar at 0:00 / 3:38.

<https://www.youtube.com/watch?v=GYbf-kCRJJI>

Intersection Straight Line and Circle

- $g : n_1x_1 + n_2x_2 + c = 0, C : (x_1 - m_1)^2 + (x_2 - m_2)^2 = R^2$

- If $n_2 \neq 0$ $x_2 = \frac{-c - n_1x_1}{n_2},$

- $(x_1 - m_1)^2 + \left(\frac{-c - n_1x_1}{n_2} - m_2\right)^2 = R^2.$

⇒ quadratic equation $Ax_1^2 + Bx_1 + C = 0$

$$A = 1 + \left(\frac{n_1}{n_2}\right)^2 \quad B = -2m_1 + 2\left(\frac{c}{n_2} + m_2\right)\frac{n_1}{n_2}$$

$$C = m_1^2 + \left(\frac{c}{n_2} + m_2\right)^2 - R^2$$

- More complex for foolproof program

```
function [d,S]=LineNorCircle(g,M,R)
% LINENORCIRCLE examines the position of the straight line
%  $g=[g_1,g_2,g_3]=0$  in general normal form with the circle
%  $\|X-M\|^2=R^2$ .
%
g=g/norm(g(1:2));           % normal form
n=g(1:2)'; c=g(3);
if n(2)==0                 % special case
    discriminant=R^2-(c/n(1)+M(1))^2;
    if abs(discriminant)<1e-10, % tangent
        S=[-c/n(1);M(2)];
        d=0;
    elseif discriminant<0     % no intersection
        S=[];
        d=M'*n+c;           % distance g from center M
        d=d-R;              % distance g from circle
    else
        dis=sqrt(discriminant);
        S=[-c/n(1), -c/n(1) % intersection points
            M(2)+dis, M(2)-dis];
        d=[];
    end
end
```

```
else % n(2) ~= 0
    C=M(1)^2+(c/n(2)+M(2))^2-R^2;
    B=-2*M(1)+2*(c/n(2)+M(2))*n(1)/n(2);
    A=1+(n(1)/n(2))^2;
    discriminant=B^2-4*A*C;
    if abs(discriminant)<1e-10, % tangent
        x=-B/(2*A); y=(-c-n(1)*x)/n(2);
        S=[x;y]; d=0;
    elseif discriminant<0
        S=[]; % no intersection
        d=M'*n+c; % distance g from center M
        d=d-R; % distance g from circle
    else
        dis=sqrt(discriminant);
        x=[-B+dis, -B-dis]/(2*A); % intersection points
        y=(-c-n(1)*x)/n(2);
        S=[x; y];
        d=[];
    end
end
end
```

Alternative Algorithm: Reduce to Unit Circle

$$g : n_1x_1 + n_2x_2 + c = 0 \quad C : (x_1 - m_1)^2 + (x_2 - m_2)^2 = R^2$$

- **Variable transform** $x'_1 = \frac{x_1 - m_1}{R}, \quad x'_2 = \frac{x_2 - m_2}{R}$
- **Reduce to intersect with unit circle** $\mathbf{n}' = R\mathbf{n}, \quad c' = c + \mathbf{n}'^\top \mathbf{M}$
Solve $\mathbf{n}'^\top \mathbf{X}' + c' = 0, \quad \|\mathbf{X}'\|^2 = 1$
- **Back transformation:** $x_1 = m_1 + Rx'_1, \quad x_2 = m_2 + Rx'_2$
- Same as before but **somehow less complex**

```
function S=LineUnitCircle(g)
% LINEUNITCIRCLE computes the intersection points of the straight
% line g in general normal form  $g_1x_1+g_2x_2+g_3=0$  with the unit circle

nu=[g(1);g(2)]; nn=norm(nu);
g=g/nn; % normal form of g
n=[g(1);g(2)]; c=g(3);
if (abs(c)>1) % no intersection
    S=[]; % |c| distance from origin
else
    if n(2)==0 % special case  $x_1=const.$ 
        x1=-c/n(1); x2=sqrt(1-x1^2);
        S=[x1,x1; x2,-x2];
    else
        p=2*n(1)*c; q=c^2-n(2)^2;
        disk=(p/2)^2-q;
        if abs(disk)<1e-10
            x1=p/2;
            S=[x1; (-c-n(1)*x1)/n(2)]; % tangent point
        else
            disk=sqrt(disk);
            xm=abs(p/2)+disk; % compute larger solution
            if p>0,xm=-xm; end % in modulus
            x1=[xm;q/xm];
            x2=(-c-n(1)*x1)/n(2);
            S=[x1,x2]';
        end
    end
end
end
```

Different Algorithm: Goniometric Equation

- straight line g in normal, circle C in parametric form:

$$g : n_1 x_1 + n_2 x_2 + c = 0, \quad \|\mathbf{n}\| = 1, \quad C : \mathbf{X} = \mathbf{M} + R \begin{pmatrix} \cos \alpha \\ \sin \alpha \end{pmatrix}.$$

- Insert C in g : $n_1(m_1 + R \cos \alpha) + n_2(m_2 + R \sin \alpha) + c = 0$.

$$\implies n_1 \cos \alpha + n_2 \sin \alpha = D \quad \text{with } D = \frac{-c - \mathbf{n}^\top \mathbf{M}}{R}$$

- With $\mathbf{n} = \begin{pmatrix} \cos \varphi \\ \sin \varphi \end{pmatrix} \implies \cos(\varphi - \alpha) = D$
- Solutions if $|D| \leq 1 \implies \alpha = \varphi \mp \arccos D$
- Insert in circle C , \implies intersection points

```
function S=LineCircleGonio(g,M,R)
% LINECIRCLEGONIO computes the intersection of the straight line g
% with the circle (M,R) by solving a goniometric equation.
g=g/norm(g(1:2)); % compute normal form
n=[g(1);g(2)]; c=g(3);
D=(-c-n'*M)/R; % right hand side
if abs(D)>1
    S=[]; % no intersection
else
    phi=atan2(n(2),n(1));
    alpha=[phi-acos(D); phi+acos(D)];
    S=[M';M']+R*[cos(alpha),sin(alpha)];
    S=S';
end
```


To be continued ...