Inledning

I denna laboration skall vi se på några geometriska transformationer i $\mathbb{R}^2$ och $\mathbb{R}^3$ som ges av linjära eller affina avbildningar.

En avbildning $\boldsymbol{F}\colon \mathbb{R}^n \to \mathbb{R}^m$ kallas *linjär* om det för alla $\boldsymbol{u},\ \boldsymbol{v} \in \mathbb{R}^n$ och alla $\alpha,\ \beta \in \mathbb{R}$ gäller att

$$\boldsymbol{F}(\alpha\boldsymbol{u} + \beta\boldsymbol{v}) = \alpha\boldsymbol{F}(\boldsymbol{u}) + \beta\boldsymbol{F}(\boldsymbol{v})$$

Till en linjär avbildning $\boldsymbol{F}\colon \mathbb{R}^n \to \mathbb{R}^m$ finns det alltid en *entydigt bestämd* matris $\boldsymbol{A}$ sådan att $\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x}$. Matrisen $\boldsymbol{A}$ kallas *standardmatrisen* till $\boldsymbol{F}$ och det gäller att $\boldsymbol{a}_j = \boldsymbol{F}(\boldsymbol{e}_j)$.

Omvänt så definierar varje $m \times n$-matris $\boldsymbol{A}$ en linjär avbildning $\boldsymbol{F}\colon \mathbb{R}^n \to \mathbb{R}^m$ med $\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x}$.

En avbildning $\boldsymbol{F} : \mathbb{R}^n \to \mathbb{R}^m$ med $\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$ kallas för *affin*. Speciellt i samband med datorgrafik inför man s.k. *homogena koordinater*, genom att lägga till en artificiell koordinat, så att även affina avbildningar kan beskrivas med matriser.

Men allra först ser vi kort hur man beräknar skalärprodukt, norm och liknande i MATLAB.

Skalärprodukten mellan två vektorer $\boldsymbol{u}$ och $\boldsymbol{v}$ i $\mathbb{R}^n$ ges av

$$\boldsymbol{u} \cdot \boldsymbol{v} = \boldsymbol{u}^{\mathsf{T}}\boldsymbol{v} = u_1v_1 + u_2v_2 + \cdots + u_nv_n$$

och normen, som motsvarar absolutbelopp, ges av

$$\|\,\boldsymbol{u}\,\| = \sqrt{u_1^2 + u_2^2 + \cdots + u_n^2} \qquad (\|\,\boldsymbol{u}\,\|^2 = \boldsymbol{u} \cdot \boldsymbol{u})$$

Med MATLAB beräknar vi skalärprodukt och norm med funktionerna `dot` och `norm` enligt

```
>> dot(u,v)
>> norm(u)
```

Vinkeln ϕ mellan två vektorer $\boldsymbol{u}$ och $\boldsymbol{v}$ ges av

$$\phi = \arccos \frac{\boldsymbol{u} \cdot \boldsymbol{v}}{\|\,\boldsymbol{u}\,\|\,\|\,\boldsymbol{v}\,\|}$$

och beräknas i MATLAB med

```
>> phi=acos(dot(u,v)/(norm(u)*norm(v)))
```

Vi påminner oss om att vektorerna $\boldsymbol{u}$ och $\boldsymbol{v}$ är ortogonala eller vinkelräta mot varandra om $\boldsymbol{u} \cdot \boldsymbol{v} = 0$, vilket ofta betecknas $\boldsymbol{u} \perp \boldsymbol{v}$.

En enhetsvektor är en vektor $\boldsymbol{u}$ med $\|\,\boldsymbol{u}\,\| = 1$. Vill vi bestämma en enhetsvektor $\boldsymbol{u}$ med samma riktning som vektorn $\boldsymbol{v}$ så ges den av $\boldsymbol{u} = \frac{1}{\|\boldsymbol{v}\|}\boldsymbol{v}$ och i MATLAB skulle vi skriva

```
>> u=v/norm(v)
```

Avståndet mellan två vektorer $\boldsymbol{u}$ och $\boldsymbol{v}$ ges av

$$\| \boldsymbol{u} - \boldsymbol{v} \| = \sqrt{(u_1 - v_1)^2 + (u_2 - v_2)^2 + \cdots + (u_n - v_n)^2}$$

och beräknas med

```
>> norm(u-v)
```

Även om vi inte använder den just nu, så nämner vi kryssprodukten i $\mathbb{R}^3$ som ges av

$$\boldsymbol{u} \times \boldsymbol{v} = (u_2v_3 - u_3v_2,\ u_3v_1 - u_1v_3,\ u_1v_2 - u_2v_1)$$

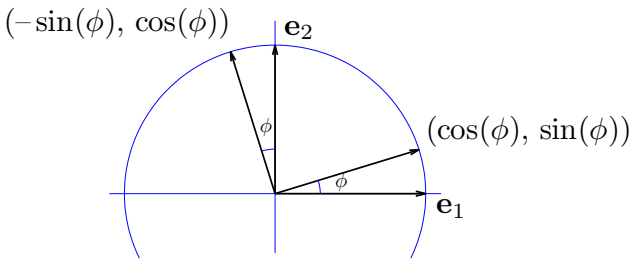
vilket är en vektor i $\mathbb{R}^3$ och beräknas med `cross(u,v)`.

Geometriska transformationer

Rotation och skalning är linjära avbildningar och kan beskrivas med standardmatriser, däremot är translation en affin avbildning. När det gäller projektion får vi skilja på olika fall, t.ex. en ortogonal projektion i $\mathbb{R}^3$ på ett plan är en linjär avbildning om och endast om planet går genom origo.

Rotation, skalning och translation

Som exempel på rotation tar vi: Låt $\boldsymbol{T} : \mathbb{R}^2 \to \mathbb{R}^2$ vara en rotation motsols med vinkeln ϕ runt origo.



Vi får

$$\boldsymbol{T}(\boldsymbol{e}_1) = \begin{bmatrix} \cos(\phi) \\ \sin(\phi) \end{bmatrix},\quad \boldsymbol{T}(\boldsymbol{e}_2) = \begin{bmatrix} -\sin(\phi) \\ \cos(\phi) \end{bmatrix}$$

med standardmatrisen

$$\boldsymbol{A} = \begin{bmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{bmatrix}$$

Betraktar vi en punkt $\boldsymbol{x} = (x_1, x_2)$ i $\mathbb{R}^2$ som vi vill rotera runt origo så ges dess nya position av $\boldsymbol{T}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x}$.

Låt $\boldsymbol{S} : \mathbb{R}^2 \to \mathbb{R}^2$ vara en skalning med faktorerna s_1 och s_2 i respektive axelriktning. Vi får

$$\boldsymbol{S}(\boldsymbol{e}_1) = \begin{bmatrix} s_1 \\ 0 \end{bmatrix},\quad \boldsymbol{S}(\boldsymbol{e}_2) = \begin{bmatrix} 0 \\ s_2 \end{bmatrix},\quad \boldsymbol{B} = \begin{bmatrix} s_1 & 0 \\ 0 & s_2 \end{bmatrix}$$

och skalningen ges av

$$\boldsymbol{S}(\boldsymbol{x}) = \boldsymbol{B}\boldsymbol{x} = \begin{bmatrix} s_1 & 0 \\ 0 & s_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

En translation med vektorn $\boldsymbol{t} = \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$ ges av avbildningen $\boldsymbol{F} : \mathbb{R}^2 \to \mathbb{R}^2$,

$$\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{t} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

men här finns ingen standardmatris.

Betrakta en punkt $\boldsymbol{x} = (x_1, x_2, x_3)$ i $\mathbb{R}^3$. Rotation kring x_1-, x_2- och x_3-axlarna ges av $\boldsymbol{T} : \mathbb{R}^3 \to \mathbb{R}^3$, där $\boldsymbol{T}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x}$ med följande respektive standardmatriser

$$\boldsymbol{A} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix}, \quad \boldsymbol{A} = \begin{bmatrix} \cos(\phi) & 0 & \sin(\phi) \\ 0 & 1 & 0 \\ -\sin(\phi) & 0 & \cos(\phi) \end{bmatrix}, \quad \boldsymbol{A} = \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotationen sker medurs sett i respektive axelriktning. Att rotera runt en sned axel klarar vi av med ett basbyte, mer om det senare.

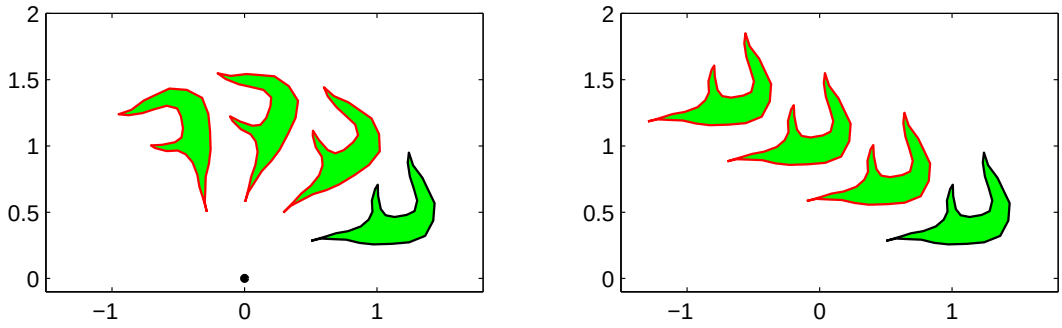
Skalning och translation i $\mathbb{R}^3$ ges av

$$\boldsymbol{S}(\boldsymbol{x}) = \boldsymbol{B}\boldsymbol{x} = \begin{bmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & s_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

respektive

$$\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{t} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}$$

Vi illustrerar rotation och translation i $\mathbb{R}^2$ med MATLAB. I figuren nedan till vänster ser vi ett polygonområde med svart rand som vi roterar några gånger med vinkeln $\frac{\pi}{6}$ och ritar de roterade områdena med röd rand.



Vi tänker oss att vi redan skapat koordinater i radvektorerna `X` och `Y` som beskriver det ursprungliga området. Så här ritar vi upp ursprungliga området och de successiva rotationerna.

```
>> fill(X,Y,'g','edgecolor','k','linewidth',1), hold on
>> axis equal, axis([-1.5 2 -0.1 2]), pause(1)
>> v=pi/6; A=[cos(v) -sin(v); sin(v) cos(v)];
>> P=[X;Y];
>> for i=1:3
       P=A*P;   % Varje koordinatpar roteras med vinkeln pi/6
       fill(P(1,:),P(2,:),'g','edgecolor','r','linewidth',1), pause(1)
```

```
   end
>> plot(0,0,'ko','linewidth',2,'markersize',2)  % origo
>> hold off
```

Till höger ser vi samma område i ursprungsläget (svart kant) samt några upprepade translationer (röd kant) med vektorn t.

```
>> fill(X,Y,'g','edgecolor','k','linewidth',1), hold on
>> axis equal, axis([-1.5 2 -0.1 2]), pause(1)
>> t=[-0.6;0.3];
>> P=[X;Y];
>> for i=1:3
       P=P+t*ones(size(X));
       fill(P(1,:),P(2,:),'g','edgecolor','r','linewidth',1), pause(1)
   end
>> hold off
```

Vi placerar de två vektorerna med koordinater i en matris med `P=[X;Y]` så att koordinaterna för punkterna ligger som kolonner.

$$\boldsymbol{P} = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \\ y_1 & y_2 & \cdots & y_n \end{bmatrix} = [\boldsymbol{P}_1 \;\; \boldsymbol{P}_2 \;\cdots\; \boldsymbol{P}_n], \text{ med } \boldsymbol{P}_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}.$$

Vi roterar i MATLAB med `P=A*P` så att alla punkter roteras samtidigt, $\widehat{\boldsymbol{P}}_i = \boldsymbol{A}\boldsymbol{P}_i$, $i = 1, \cdots, n$,

$$\widehat{\boldsymbol{P}} = \left[\widehat{\boldsymbol{P}}_1 \;\; \widehat{\boldsymbol{P}}_2 \;\cdots\; \widehat{\boldsymbol{P}}_n\right] = [\boldsymbol{A}\boldsymbol{P}_1 \;\; \boldsymbol{A}\boldsymbol{P}_2 \;\cdots\; \boldsymbol{A}\boldsymbol{P}_n] = \boldsymbol{A}\,[\boldsymbol{P}_1 \;\; \boldsymbol{P}_2 \;\cdots\; \boldsymbol{P}_n] = \boldsymbol{A}\boldsymbol{P}$$

Med `P=P+t*ones(size(X))` translaterar vi alla punkter samtidigt, $\widehat{\boldsymbol{P}}_i = \boldsymbol{P}_i + \boldsymbol{t}$, $i = 1, \cdots, n$,

$$\widehat{\boldsymbol{P}} = \left[\widehat{\boldsymbol{P}}_1 \;\; \widehat{\boldsymbol{P}}_2 \;\cdots\; \widehat{\boldsymbol{P}}_n\right] = [\boldsymbol{P}_1 + \boldsymbol{t} \;\; \boldsymbol{P}_2 + \boldsymbol{t} \;\cdots\; \boldsymbol{P}_n + \boldsymbol{t}] =$$

$$= \boldsymbol{P} + [\boldsymbol{t} \;\; \boldsymbol{t} \;\cdots\; \boldsymbol{t}] = \boldsymbol{P} + \begin{bmatrix} t_1 & t_1 & \cdots & t_1 \\ t_2 & t_2 & \cdots & t_2 \end{bmatrix} = \boldsymbol{P} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}$$

Uppgift 1. Rotera och translatera ett polygonområde ni genererar själva, t.ex. en triangel.

Ortogonal projektion och spegling

Ortogonal eller vinkelrät projektion av en punkt $\boldsymbol{x}$ på linjen

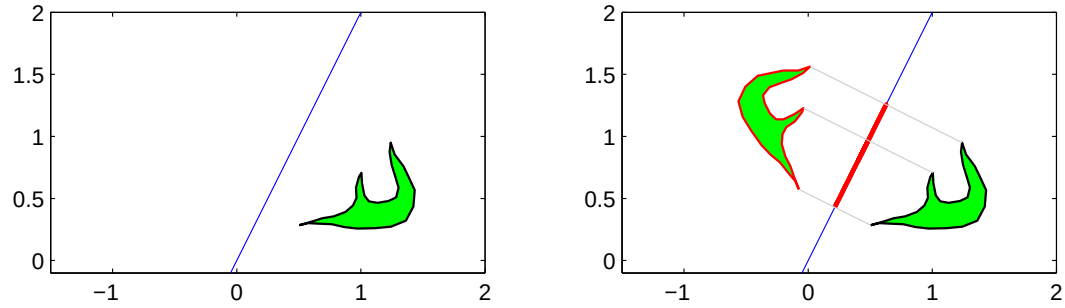
$$ax + by = d$$

ges av $\hat{\boldsymbol{x}} = \boldsymbol{x} + \alpha\boldsymbol{n}$, där $\boldsymbol{n} = (a, b)$ är normalen och

$$\alpha = \frac{d - \boldsymbol{n} \cdot \boldsymbol{x}}{\boldsymbol{n} \cdot \boldsymbol{n}}$$

Speglingen i linjen av $\boldsymbol{x}$ ges av $\boldsymbol{x}_r = \boldsymbol{x} + 2\alpha\,\boldsymbol{n}$.

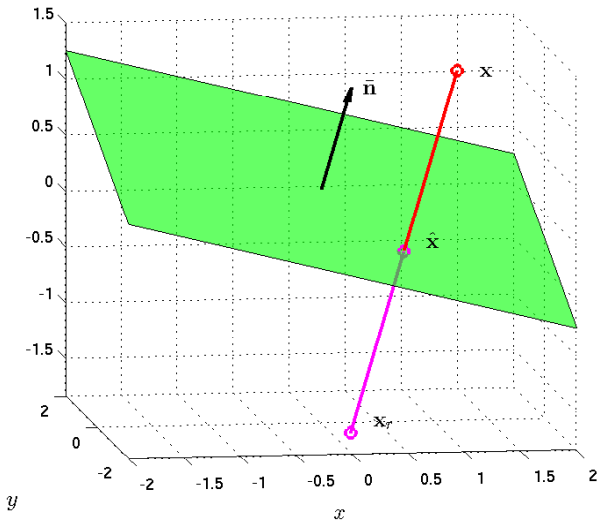
Om vi projicerar och speglar vårt område i en rät linje skulle det kunna se ut så här.



Nu skall vi se hur det blir i rummet: Bestäm ortogonala projektionen på planet

$$ax + by + cz = d$$

Planet har normalvektorn $\boldsymbol{n} = (a, b, c)$. I bilden har vi ritat enhetsnormal $\bar{\boldsymbol{n}}$ och ortogonala projektionen $\hat{\boldsymbol{x}}$ längs enhetsnormalen av en punkt $\boldsymbol{x}$.



Vi gör ansatsen $\hat{\boldsymbol{x}} = \boldsymbol{x} + \alpha\,\boldsymbol{n}$, där α skall bestämmas så att $\hat{\boldsymbol{x}}$ ligger på planet. Ekvationen för planet kan skrivas

$$\boldsymbol{n} \cdot \boldsymbol{x} = d$$

och sätter vi in ansatsen får vi

$$\boldsymbol{n} \cdot \hat{\boldsymbol{x}} = \boldsymbol{n} \cdot (\boldsymbol{x} + \alpha \boldsymbol{n}) = \boldsymbol{n} \cdot \boldsymbol{x} + \alpha\, \boldsymbol{n} \cdot \boldsymbol{n} = d$$

och därmed

$$\alpha = \frac{d - \boldsymbol{n} \cdot \boldsymbol{x}}{\boldsymbol{n} \cdot \boldsymbol{n}}$$

För speglingen av $\boldsymbol{x}_r$ av punkten $\boldsymbol{x}$ i planet gäller

$$\boldsymbol{x}_r = \boldsymbol{x} + 2\alpha\, \boldsymbol{n}$$

med samma val av α.

Nu skall vi i MATLAB rita planet $ax + by + cz = d$, för $a = 1, b = -1, c = 4$ och $d = 1$.

Eftersom $c \neq 0$ så kan vi lösa ut z, i annat fall får vi modifiera koden

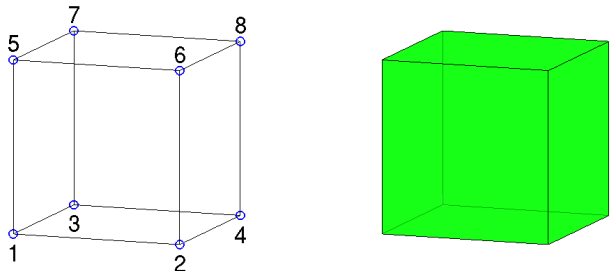
```
>> xmin=-2; xmax=2; ymin=-2; ymax=2;
>> a=1; b=-1; c=4; d=1;
>> X=[xmin xmax xmax xmin]; Y=[ymin ymin ymax ymax];
>> Z=(d-a*X-b*Y)/c;
>> figure(1), clf
>> fill3(X,Y,Z,'g','facealpha',0.7)
>> xlabel('x'), ylabel('y')
>> grid on
```

Resultatet blir planet i figuren ovan.

Uppgift 2. Rita planet vi just tittade på. Bestäm en normalvektor och rita ut den med en pil från en punkt på planet. En pil i rummet kan vi rita med `quiver3(x,y,z,a,b,c,s)`, där x, y, z ger koordinaterna för den punkt som pilen skall ritas från, a, b, c ger pilens utsträckning och s är en skalfaktor (normalt tar vi $s = 0$ vilket ritar utan skalning, medan t.ex. $s = 2$ gör pilarna dubbelt så långa). Välj en punkt $\mathbf{x}$, rita ut den, bestäm dess vinkelräta projektion $\hat{\mathbf{x}}$ på planet och rita ut även den. Slutligen rita också ut speglingen $\mathbf{x}_r$ av $\mathbf{x}$. Markera normalvektorn och de olika punkterna med texter. T.ex. normalvektorn kan markeras med `text(u,v,w,'n')`, där u, v, w är koordinaterna för positionen av vänster sida av texten och `'n'` är texten som skall skrivas, dvs. ett n. Använd `axis equal` så att vi ser om vinklar är räta.

Vi skall rotera och projicera en kub. Kuben ritar vi med följande skriptfil

```
c=1/sqrt(3);
H=[-c  c -c  c -c  c -c  c    % H(:,j), j:te kolonnen i H, ger koordinater
   -c -c  c  c -c -c  c  c    % för punkt j
   -c -c -c -c  c  c  c  c];
S=[1 2 4 3                    % S(i,:), i:te raden i S, ger nr på hörnpunkter
   1 2 6 5                    % på sida i
   1 3 7 5
   3 4 8 7
   2 4 8 6
   5 6 8 7];                  % size(S,1) ger antal sidor
figure(1), clf, hold on
for i=1:size(S,1)
    Si=S(i,:); fill3(H(1,Si),H(2,Si),H(3,Si),'g','facealpha',0.7)
end
hold off, axis equal tight off, view(20,10)
```



Lägg lite tid på att tänka igenom det vi gjort. Hörnpunkternas koordinater ligger som kolonner i matrisen `H`. På raderna i matrisen `S` har vi numren på hörnpunkterna på sidorna, t.ex. första raden (`1 2 4 3`) ger numren på hörnpunkterna som ger botten på kuben.

Antal kolonner i `H`, som ges av `size(H,2)`, är antalet hörnpunkter. Antalet rader i `S`, som ges av `size(S,1)`, är antalet sidor.

Med `for`-satsen ritar vi upp alla sidor. Vi plockar ut en sidas hörnpunkter med `Si=S(i,:)` och motsvarande kolonner i `H`, dvs. `H(:,Si)` ger koordinaterna för hörnpunkterna.

Vi ritar sidan med `fill3`, som fungerar som `fill` fast i rummet. Här måste vi separera x_1-, x_2- och x_3-koordinaterna. Med `H(1,Si)` får vi x_1-koordinaterna för hörnpunkterna på sidan, osv.

Vi får kuben lite lätt genomskinlig med `'facealpha',0.7`. För solid kub sätt `'facealpha',1` eller utelämna. Med `axis equal` får vi skalor på axlarna sådana att en kub ser ut som en kub och inte blir tillplattad. Vidare ger `axis tight` ett koordinatsystem *utan* luft runt kuben som vi ritat och `axis off` gör att axlarna och skalmarkeringar inte syns. Med `view(20,10)` ges betraktelsevinklar, se gärna hjälptexterna.

Uppgift 3. Rotera en kub runt någon axel (rotationsmatrisen gav vi tidigare i texten). Gör en translation av kuben bort från origo. Rotera den åter runt någon axel.

Uppgift 4. Rita planet från uppgift 2. I samma bild skall ni sedan rita en translaterad kub. Translationen skall vara sådan att kuben inte skär planet. Rita därefter, i samma bild, även projektionen och speglingen av kuben i planet.

Homogena koordinater

Rotation och skalning är linjära avbildningar och kan beskrivas med standardmatriser $\boldsymbol{T}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x}$, däremot är translation *inte* en linjär avbildning utan en *affin* avbildning $\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$.

En affin avbildning $\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x}+\boldsymbol{b}$ kan beskrivas med matrismultiplikation om vi inför *homogena koordinater*, vilket är en extra koordinat w enligt

$$\begin{bmatrix} \hat{\boldsymbol{x}} \\ \hat{w} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{b} \\ \boldsymbol{0}^T & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ w \end{bmatrix}$$

där $\boldsymbol{0}^T$ är en rad med nollor.

Vi kommer då ha

$$\begin{aligned} \hat{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}w \\ \hat{w} &= w \end{aligned}$$

Så om låter vi $w = 1$ så får vi $\hat{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$ och $\hat{w} = 1$.

Translation med en vektor $\boldsymbol{t}$ ges av

$$\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{t}$$

Vi har $\boldsymbol{A} = \boldsymbol{I}$, dvs. enhetsmatrisen, och $\boldsymbol{b} = \boldsymbol{t}$, dvs. translationsvektorn, och vi kan beskriva translationen med matrismultiplikation

$$\begin{bmatrix} \hat{\boldsymbol{x}} \\ \hat{w} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{t} \\ \boldsymbol{0}^T & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ w \end{bmatrix}$$

som vi också kan skriva

$$\hat{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{t}w$$
$$\hat{w} = w$$

Tar vi $w = 1$ så får vi $\hat{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{t}$.

Vill vi använda oss av homogena koordinater är det praktiskt att använda lika stora matriser även för linjära avbildningar.

En linjär avbildning $\boldsymbol{T}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x}$ kan bäddas in med homogena koordinater enligt

$$\begin{bmatrix} \hat{\boldsymbol{x}} \\ \hat{w} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{0} \\ \boldsymbol{0}^T & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ w \end{bmatrix}$$

och vi ser att $\hat{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x}$.

Vi skall se lite mer på den ortogonala projektionen på ett plan $ax + by + cz = d$ där a, b, c och d är konstanter.

Om $\boldsymbol{x}$ är den punkt vi projicerar och $\hat{\boldsymbol{x}}$ är projektionen så gäller

$$\hat{\boldsymbol{x}} = \boldsymbol{x} + \alpha\,\boldsymbol{n}, \qquad \alpha = \frac{d - \boldsymbol{n} \cdot \boldsymbol{x}}{\boldsymbol{n} \cdot \boldsymbol{n}}$$

där $\boldsymbol{n} = (a, b, c)$ är en normalvektor till planet.

Om $d = 0$, dvs. planet går genom origo, har vi

$$\hat{\boldsymbol{x}} = \boldsymbol{x} + \alpha\,\boldsymbol{n} = \boldsymbol{x} - \frac{\boldsymbol{n} \cdot \boldsymbol{x}}{\boldsymbol{n} \cdot \boldsymbol{n}}\,\boldsymbol{n}$$

Eftersom $\boldsymbol{n} \cdot \boldsymbol{x}$ är en skalär så gäller

$$\hat{\boldsymbol{x}} = \boldsymbol{x} - \frac{(\boldsymbol{n} \cdot \boldsymbol{x})}{\boldsymbol{n} \cdot \boldsymbol{n}}\,\boldsymbol{n} = \boldsymbol{x} - \frac{1}{\boldsymbol{n} \cdot \boldsymbol{n}}\,\boldsymbol{n}\,(\boldsymbol{n} \cdot \boldsymbol{x})$$

Vidare gäller att $\boldsymbol{n} \cdot \boldsymbol{x} = \boldsymbol{n}^{\mathsf{T}}\boldsymbol{x}$, dvs. skalärprodukten kan beräknas genom att $\boldsymbol{n}^{\mathsf{T}}$ som är en radvektor matrismultipliceras med kolonnvektorn $\boldsymbol{x}$ och vi får

$$\hat{\boldsymbol{x}} = \boldsymbol{x} - \frac{1}{\boldsymbol{n} \cdot \boldsymbol{n}}\,\boldsymbol{n}\,(\boldsymbol{n}^{\mathsf{T}}\boldsymbol{x}) = \boldsymbol{x} - \frac{1}{\boldsymbol{n} \cdot \boldsymbol{n}}\,(\boldsymbol{n}\,\boldsymbol{n}^{\mathsf{T}})\,\boldsymbol{x} = \left(\boldsymbol{I} - \frac{1}{\boldsymbol{n} \cdot \boldsymbol{n}}\,\boldsymbol{n}\,\boldsymbol{n}^{\mathsf{T}}\right)\boldsymbol{x}$$

där vi utnyttjade att $\boldsymbol{n}\,(\boldsymbol{n}^{\mathsf{T}}\boldsymbol{x}) = (\boldsymbol{n}\,\boldsymbol{n}^{\mathsf{T}})\,\boldsymbol{x}$.

Här är $\boldsymbol{n}\,\boldsymbol{n}^{\mathsf{T}}$ en matris (en s.k. ytterprodukt), medan $\boldsymbol{n} \cdot \boldsymbol{n} = \boldsymbol{n}^{\mathsf{T}}\boldsymbol{n}$ är ett tal (skalärprodukt eller innerprodukt).

Vi har kommit fram till

$$\hat{\boldsymbol{x}} = \left(\boldsymbol{I} - \frac{1}{\boldsymbol{n} \cdot \boldsymbol{n}}\,\boldsymbol{n}\,\boldsymbol{n}^{\mathsf{T}}\right)\boldsymbol{x} = \boldsymbol{P}\boldsymbol{x}$$

Alltså en linjär avbildning med standardmatrisen $\boldsymbol{P}$.

För spegling gäller

$$\boldsymbol{x}_r = \left(\boldsymbol{I} - \frac{2}{\boldsymbol{n} \cdot \boldsymbol{n}}\,\boldsymbol{n}\,\boldsymbol{n}^{\mathsf{T}}\right)\boldsymbol{x} = \boldsymbol{R}\boldsymbol{x}$$

Alltså en linjär avbildning med standardmatrisen $\boldsymbol{R}$.

Om $d \neq 0$, dvs. planet går *inte* genom origo, har vi

$$\hat{\boldsymbol{x}} = \boldsymbol{x} + \alpha\,\boldsymbol{n} = \boldsymbol{x} + \frac{d - \boldsymbol{n}\cdot\boldsymbol{x}}{\boldsymbol{n}\cdot\boldsymbol{n}}\,\boldsymbol{n} = \left(\boldsymbol{I} - \frac{1}{\boldsymbol{n}\cdot\boldsymbol{n}}\,\boldsymbol{n}\,\boldsymbol{n}^{\mathsf{T}}\right)\boldsymbol{x} + \frac{d}{\boldsymbol{n}\cdot\boldsymbol{n}}\,\boldsymbol{n} = \boldsymbol{P}\boldsymbol{x} + \beta\boldsymbol{n}$$

Detta är en affin avbildning som vi beskriver med (homogena koordinater)

$$\begin{bmatrix} \hat{\boldsymbol{x}} \\ \hat{w} \end{bmatrix} = \begin{bmatrix} \boldsymbol{P} & \beta\boldsymbol{n} \\ \boldsymbol{0}^{\mathsf{T}} & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ w \end{bmatrix}$$

dvs. med hjälp av matrismultiplikation.

Motsvarande gäller för spegling, då $d \neq 0$.

Rotation runt sned axel i $\mathbb{R}^3$

Betrakta en punkt $\mathbf{x} = (x_1,\, x_2,\, x_3)$ i $\mathbb{R}^3$. Antag vi vill rotera pukten runt en axel, vars riktning ges av vektorn $\mathbf{v}$, med en vinkel ϕ. Rotationen skall göras medurs relativt riktningen på vektorn.

Vi kan då via ett basbyte återföra denna rotation till en rotation runt x_1-, x_2- eller x_3-axeln. För dessa har vi redan sett på standardmatriser.

Säg att vi vill återföra till en rotation runt x_1-axeln. Vi normaliserar $\mathbf{v}$, dvs. vi bildar $\mathbf{v}_1 = \alpha\mathbf{v}$ där skalfaktorn α väljs så att $\mathbf{v}_1$ får enhetslängd. Därefter väljer vi två vektorer $\mathbf{v}_2$ och $\mathbf{v}_3$, båda av enhetslängd, så att $\mathbf{v}_2$ och $\mathbf{v}_3$ är vinkelräta mot $\mathbf{v}_1$ och vinkelräta mot varandra.

Vi skall helt enkelt se till att $\{\mathbf{v}_2, \mathbf{v}_3\}$ blir en ortogonalbas för nollrummet $\mathrm{Nul}(\mathbf{v}_1^{\mathsf{T}})$. Denna bas kan vi lätt beräkna i MATLAB med funktionen `null`.

Vi undersöker om $\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}$ är en höger orienterad bas genom att beräkna determinanten

$$D = \det\left([\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3]\right).$$

Om $D > 0$ så väljer vi $\mathbf{b}_1 = \mathbf{v}_1$, $\mathbf{b}_2 = \mathbf{v}_2$, $\mathbf{b}_3 = \mathbf{v}_3$ annars tar vi $\mathbf{b}_1 = \mathbf{v}_1$, $\mathbf{b}_2 = \mathbf{v}_3$, $\mathbf{b}_3 = \mathbf{v}_2$.

Nu bildar vi basbytesmatrisen $\mathbf{P} = [\,\mathbf{b}_1\ \mathbf{b}_2\ \mathbf{b}_3\,]$ och eftersom kolonnerna ortogonala och normerade så gäller $\mathbf{P}^{-1} = \mathbf{P}^{\mathsf{T}}$.

Standardmatrisen för rotation runt axeln som ges av $\mathbf{v}$ blir

$$\mathbf{A}_{\mathbf{v}} = \mathbf{P}\mathbf{A}\mathbf{P}^{-1} = \mathbf{P}\mathbf{A}\mathbf{P}^{\mathsf{T}}$$

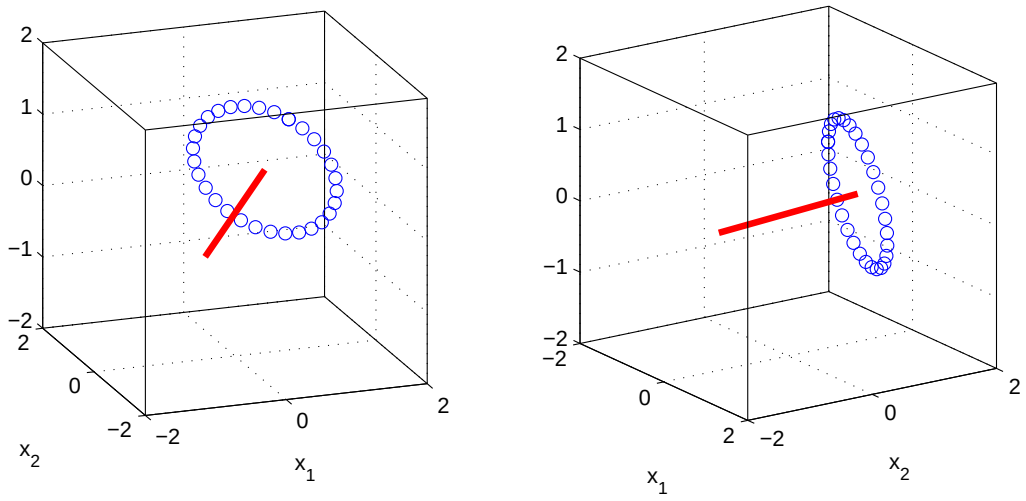
där

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix}$$

är standardmatrisen för rotationen runt x_1-axeln.

Detaljerna ovan tar nog lite tid att förstå, vi utnyttjar både vektorrum och basbyte. Det hindrar inte att vi kan rotera i MATLAB redan nu.

Nedan ser vi en rotation av en punkt runt en viss axel, upprepad några gånger, sett från två olika betraktelsevinklar.



Så här gjorde vi en skriptfil i MATLAB

```
phi=pi/15;
A=[1 0 0; 0 cos(phi) -sin(phi); 0 sin(phi) cos(phi)];
v=[2;2;1]; v=v/norm(v); Z=null(v'); P=[v,Z];
if det(P)<0, P(:,[2 3])=P(:,[3 2]); end
Av=P*A*P';
x=[0.8; 0.1; 1.2];
plot3(x(1),x(2),x(3),'o'), hold on
for i=1:30
    x=Av*x;
    plot3(x(1),x(2),x(3),'o')
end
plot3([-v(1) v(1)],[-v(2) v(2)],[-v(3) v(3)],'r','linewidth',3)
box on, grid on, hold off
axis equal, axis([-2 2 -2 2 -2 2]), axis vis3d
```

Uppgift 5. Rotera, med en liten vinkel upprepade gånger, en kub runt någon sned axel som ni själva väljer (dock inte samma punkt och axel som i exemplet). Se till att kuben är placerad i förhållande till axeln så att rotationen syns tydligt. Kubem får givetvis inte deformeras under rotationen.