Geometriska transformationer

1 Inledning

Vi skall se på några geometriska transformationer; *rotation*, *skalning*, *translation*, *spegling* och *projektion*. Rotation och skalning är linjära avbildningar och kan beskrivas med en matriser. Translation däremot är *inte* en linjär avbildning, det är en affin avbildning.

När det gäller projektion och spegling får vi skilja på olika fall, t.ex. en ortogonal (vinkelrät) projektion i $\mathbb{R}^3$ på ett plan är en linjär avbildning om och endast om planet går genom origo.

2 Rotation, skalning och translation i $\mathbb{R}^2$

Betrakta en punkt $\mathbf{x} = (x_1, x_2)$ i $\mathbb{R}^2$. Rotation moturs med vinkeln ϕ runt origo ges av $\mathbf{T} : \mathbb{R}^2 \to \mathbb{R}^2$, där $\mathbf{T}(\mathbf{x}) = \mathbf{A}\mathbf{x}$ med avbildningsmatrisen

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

Motsvarande för skalning och translation blir

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

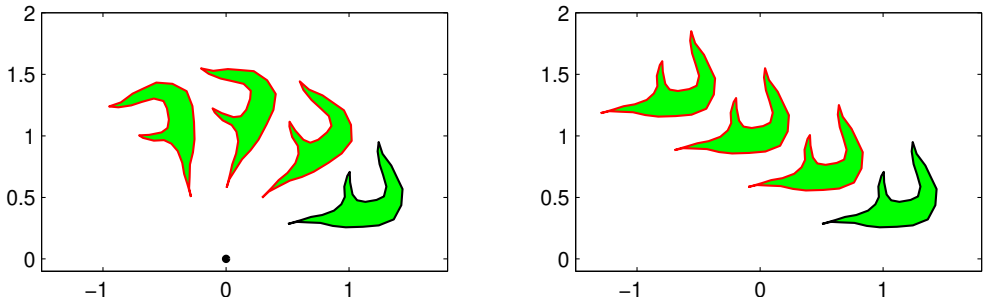
respektive

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

För translation finns ingen avbildningsmatris eftersom det inte är en linjär avbildning.

Vi illustrerar 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 med röd rand. Vi tänker oss att vi redan skapat koordinater i radvektorerna `X` och `Y` som beskriver det ursprungliga området.

```
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
```

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

3 Rotation, skalning och translation i $\mathbb{R}^3$

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

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

Rotationen sker medurs sett i axelriktningarna.

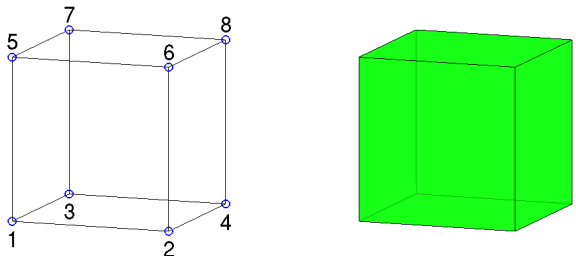
Motsvarande för skalning och translation blir

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

respektive

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

Nu skall ni rita en kub som vi skall transformera på lite olika sätt.



Vi ritar en enhetskub enligt

```
H=[0 1 0 1 0 1 0 1   % H(:,j), j:te kolonnen i H, ger koordinater för hörnpunkt j
   0 0 1 1 0 0 1 1   % size(H,2) ger antal hörnpunkter
   0 0 0 0 1 1 1 1];
S=[1 2 4 3           % S(i,:), i:te raden i S, ger nr på hörnpunkter på sida i
   1 2 6 5
   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
axis equal, axis tight, axis off, hold 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 så 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 2. Rita en kub och rotera den runt någon axel. Gör en translation av kuben bort från origo. Rotera den åter runt någon axel. Välj ett koordinatsystem så att kuben syns i alla lägen den hamnar i.

Uppgift 3. *Frivillig*! Nu skall vi göra om uppgift 2, fast som en *animation*. När ni roterade och translaterade kuben, så behöll ni i figuren resultatet efter varje rotation och translation. På så sätt fick ni en figur med många kuber och det var bra för att förstå vad som hände. Nu vill vi bara se en enda kub som roterar, blir translaterad för att sedan rotera igen. Rotationerna skall göras

genom att upprepade gånger rotera med en liten vinkel så att vi får en jämn rörelse. För varje liten rotation skall vi sudda figuren (`clf`) och rita upp kuben i den senaste positionen samt lägga in en kort paus (`pause`) så att vi hinner se resultatet. På motsvarande sätt med translationen. Som hjälp ger vi följande början till ett script där vi ser på den första rotationen

```
ax=[...]; % Gränser för koordinataxlarna, ax=[xmin xmax ymin ymax zmin zmax].
          % Dessa sätts så att hela scenen får plats
v=...;    % Liten vinkel för upprepad rotation, t.ex. v=pi/8
A=[...];  % Rotationsmatrisen
P=H;      % H är matrisen med kubens koordinater
clf
hold on
for i=1:size(S,1)  % Rita kuben
    Si=S(i,:); fill3(P(1,Si),P(2,Si),P(3,Si),'g')
end
hold off
axis equal, axis(ax)  % Undvik deformation och använd samma skalor på axlarna
box on, grid on       % Förstärker tredimensionella känslan något
pause(0.5) % Paus så vi hinner se kuben innan den roteras
kmax=...;  % Antal små rotationer, t.ex. kmax=16; (kmax*v=2*pi, ett varv)
for k=1:kmax  % Rotera kmax gånger
    P=A*P;
    clf
    hold on
    for i=1:size(S,1)  % Rita kuben i nya positionen
        Si=S(i,:); fill3(P(1,Si),P(2,Si),P(3,Si),'g')
    end
    hold off
    axis equal, axis(ax), box on, grid on
    pause(0.1)
end
```

Vi fortsätter med geometriska transformationer och ser på ortogonal (vinkelrät) projektion samt spegling.

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

4 Skalärprodukt och norm i MATLAB

Skalärprodukten mellan två vektorer $\mathbf{u}$ och $\mathbf{v}$ ges av

$$\mathbf{u} \cdot \mathbf{v} = \mathbf{u}^{\mathsf{T}}\mathbf{v} = u_1v_1 + u_2v_2 + u_3v_3$$

och normen, som motsvarar absolutbelopp, ges av

$$\|\mathbf{u}\| = \sqrt{u_1^2 + u_2^2 + u_3^2} \qquad (\|\mathbf{u}\|^2 = \mathbf{u} \cdot \mathbf{u})$$

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

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

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

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

och beräknas i MATLAB med

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

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

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

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

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

$$\| \mathbf{u} - \mathbf{v} \| = \sqrt{(u_1 - v_1)^2 + (u_2 - v_2)^2 + (u_3 - v_3)^2}$$

och beräknas med

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

Dessa beräkningar görs på samma sätt för vektorer i $\mathbb{R}^n$, oavsett om $n = 2$, 3, eller större.

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

$$\mathbf{u} \times \mathbf{v} = (u_2 v_3 - u_3 v_2,\ u_3 v_1 - u_1 v_3,\ u_1 v_2 - u_2 v_1)$$

vilket är en vektor i $\mathbb{R}^3$ och beräknas med funktionen `cross` enligt

```
>> cross(u,v)
```

5 Ortogonal projektion och spegling

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

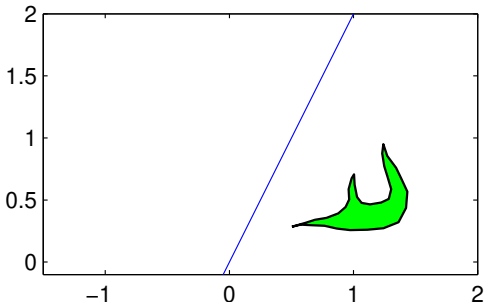
$$ax + by = d$$

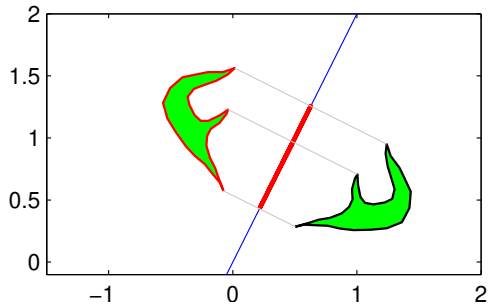
ges av $\widehat{\mathbf{x}} = \mathbf{x} + \alpha \mathbf{n}$, där $\mathbf{n} = (a, b)$ är normalen och

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

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

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

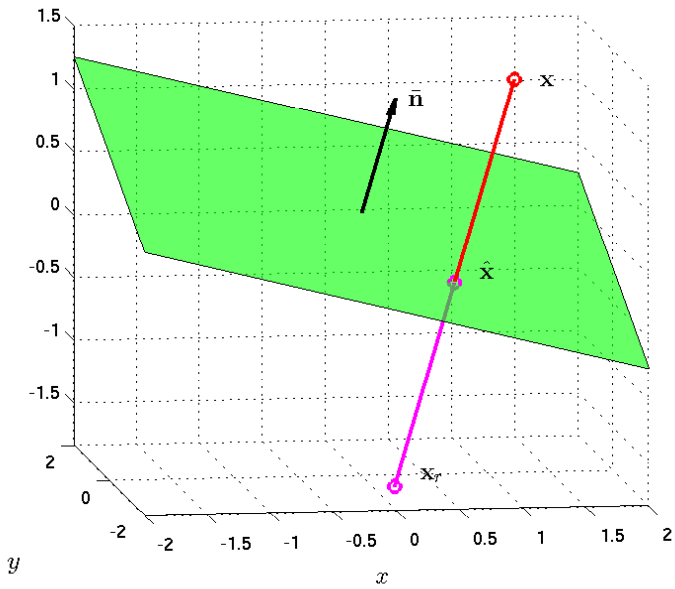




Vi skall bestämma ortogonala eller vinkelräta projektionen på planet

$$ax + by + cz = d$$

Planet har en normalvektor $\mathbf{n} = (a, b, c)$. I bilden har vi ritat en enhetsnormal $\bar{\mathbf{n}}$ och projektionen $\hat{\mathbf{x}}$ längs normalen av en punkt $\mathbf{x}$, dvs. den vinkelräta projektionen på planet.



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

Ekvationen för planet kan skrivas

$$\mathbf{n} \cdot \hat{\mathbf{x}} = d$$

och sätter vi in ansatsen får vi

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

och därmed

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

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

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

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;
fill3(X,Y,Z,'g','facealpha',0.7)
xlabel('x'), ylabel('y')
axis equal, grid on
```

Resultatet blir planet i figuren ovan. Egentligen är det ju bara en bit av planet, nämligen den del som ligger ovanför området $-2 \leq x \leq 2,\ -2 \leq y \leq 2$.

Uppgift 4. Rita planet vi just tittade på. Bestäm en normalvektor och rita ut den med en pil från en punkt på planet.

Vi kan rita en pil i rummet 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.

Uppgift 5. Rita samma plan i en ny bild. I samma bild skall ni rita en kub. Kuben skall vara så placerad att ingen av dess sidor skär planet. Rita därefter projektionen och speglingen av kuben i planet.