MATEMATISKA VETENSKAPER
Chalmers tekniska högskola
Examinator: Tony Stillfjord

TMV166 Linjär algebra för M

Datorlaboration 4: Geometriska transformationer och plottning av figurer

Allmänt

Vi diskuterar generellt $\mathbb{R}^n$ eller till och med oändligtdimensionella vektorrum, då egenskaperna för dessa är desamma som för $\mathbb{R}^3$. Befinner vi oss i $\mathbb{R}^3$ (vilket oftast är fallet) så kan vi dock visualisera vektorer på ett intuitivt sätt: de är riktade sträckor i rummet som utgår från origo. För att illustrera detta så kan vi rita dem i MATLAB. Strikt sett så projicerar vi då en vektor i $\mathbb{R}^3$ på ett plan i $\mathbb{R}^2$ (datorskärmen) och ritar denna. Vi kan också illustrera olika linjära transformationer genom att se hur de påverkar givna vektorer.

Mål

- Rita geometriska figurer i MATLAB
- Applicera linjära transformationer på dessa figurer

Användbara kommandon

Kom ihåg att den Euklidiska normen $\|v\|$ av en vektor v beräknas med `norm(v)` i MATLAB.

Skalärprodukten av x och y kan beräknas med `dot(x,y)`, men vanligtvis använder man `x'*y` ifall x och y båda är kolonnvektorer.

Ett ytterligare användbart kommando är `acos`, för *arcus cosinus*.

Vinkeln mellan två vektorer

Uppgift 1. Skriv en funktionsfil som beräknar vinkeln mellan två givna vektorer x och y. Om vinkeln betecknas θ så är den matematiska formeln

$$x \cdot y = \|x\| \, \|y\| \cos\theta.$$

Förslag på funktions-format:

Ursprunglig datorlaboration författad av Carl-Henrik Fant och Lennart Falk, uppdaterad av Tony Stillfjord

```
function theta = angle(x,y)
% Input: two vectors x, y
% Output: the angle between x and y

...

end
```

Programmet skall ge vinkeln mellan vektorerna mätt i radianer. Skriv gärna programmet så att det inte spelar någon roll om vektorerna matas in som kolonnvektorer eller som radvektorer. Testa funktionen på några olika par av vektorer. Det är klokt att bl.a. testa med vektorer där du vet svaret.

□

Spegling

Matematisk bakgrund

Låt planet π vara givet som $Ax+By+Cz = D$. Dividerar vi ekvationen med $\sqrt{A^2+B^2+C^2}$ så får vi en ny ekvation $ax+by+cz = d$, och en normalvektor $n = (a, b, c)$ av längd 1. (Kontrollera detta med `norm(n)`.) Tag en punkt med ortsvektorn x och låt x_0 vara en godtycklig punkt i planet. Då ges den *ortogonala projektionen* av ortsvektorn p där

$$p - x = \big((x_0 - x) \cdot n\big)n = (x_0 \cdot n - x \cdot n)n = (d - x \cdot n)n.$$

(Eftersom $x_0 \cdot n = d$ för en punkt i planet π.) Detta är *projektionsformeln*, vilket vi återkommer till senare i kursen. Tills dess räcker det att veta att en *ortogonal* projektion ger oss den närmaste punkten i planet till x. Denna är entydigt bestämd.

För spegelpunkten med ortsvektor s gäller att

$$s = x + 2(p - x) = x + 2(d - x \cdot n)n.$$

Här är $p - x$ sträckan från x till p, så faktorn 2 i ekvationen innebär att vi först går från x till planet och sen precis lika långt till i samma riktning.

Uppgift 2. Skriv ett program som givet planet $Ax + By + Cz = D$ och en punkt x i rummet beräknar spegelbilden s av x i planet.

Förslag på funktions-format:

```
function s = reflection(x, A, B, C, D)
% Input: a point x,
%        the parameters of the plane Ax + By + Cz = D
% Output: the reflection s of x in the plane

...

end
```

För bästa anpassning till nästa uppgift, representera alla vektorer som kolonnvektorer. Testa funktionerna på några lämpligt valda plan och punkter, för vilka man lätt kontrollerar att resultaten stämmer.

□

Ritning av godtyckliga figurer

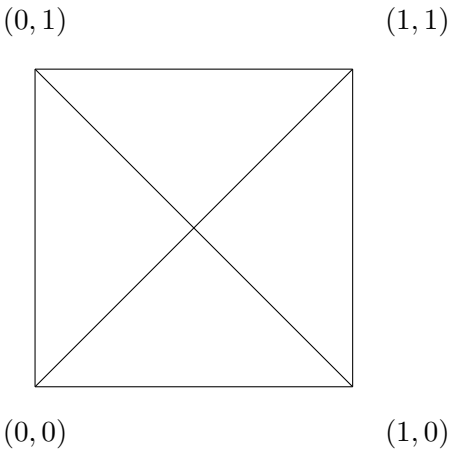
Vi vill kunna rita godtyckliga plana figurer med MATLAB, exempelvis en kurva $(x(t), y(t))$ på parameterform, eller en polygon. Detta kan göras genom att skapa en koordinatmatris för de punkter vi vill rita. Denna matris kan skrivas på något av följande sätt:

$$P = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} p_1 & p_2 & \cdots & p_n \end{bmatrix}.$$

Här är x och y radvektorer av längd n bestående av x- respektive y-koordinaterna för punkterna, medan p_i är en kolonnvektor med koordinaterna för punkt nummer i. Om man vill rita en kurva på parameterform så väljer vi $p_i = \begin{bmatrix} x(t_i) \\ y(t_i) \end{bmatrix}$ för tillräckligt täta t-värden. En polygon får vi om vi låter p_i vara hörnen uppräknade i en lämplig ordning. (Glöm inte att upprepa den första punken i slutet.) Vi ritar nu figuren genom att plotta y-koordinaterna mot x-koordinaterna med kommandot

```
>> plot(P(1,:), P(2,:))
```

Pröva genom att rita figuren nedan:



Uppgift 3. Rita en triangel i planet. Välj hörn så att triangeln inte skärs av linjen $x + 2y = 1$ (byt gärna ut någon koordinat om någon vinkel är nära 180°). Använd funktionen `reflection` från föregående uppgift för att bestämma figurens spegelbild i linjen $x + 2y = 1$. Observera att spegling i plan skall bytas ut mot spegling i linje: välj $C = 0$ och mata in vektorn x med z-koordinaten noll. Plotta linjen, figuren och dess spegelbild i samma figur. Kommandot `axis equal` efter `plot` är nödvändigt för att speglingen skall se korrekt ut. □

Rotationer av objekt i 3D

I nästa uppgift behöver vi rotera rummets vektorer kring en fritt vald axel. Om denna axel är z-axeln är det lätt att hitta standardmatrisen M för denna vridning. Ifall rotationsaxeln är en annan så kan man byta till en annan bas (f_1, f_2, f_3) där f_3 är en vektor i rotationsaxelns riktning. Om koordinaterna för en vektor u ges av (x, y, z) i standardbasen (e_1, e_2, e_3) och (ξ, η, ζ) i den nya basen så har vi

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = P \begin{bmatrix} \xi \\ \eta \\ \zeta \end{bmatrix}$$

där $P = \begin{bmatrix} f_1 & f_2 & f_3 \end{bmatrix}$ är basbytesmatrisen. Antag nu att T är en linjär transformation som i basen (f_1, f_2, f_3) har matrisen M, och att $T(u)$ har koordinaterna (x', y', z') i standardbasen och (ξ', η', ζ') i den nya basen. Men då gäller att

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = P \begin{bmatrix} \xi' \\ \eta' \\ \zeta' \end{bmatrix} = PM \begin{bmatrix} \xi \\ \eta \\ \zeta \end{bmatrix} = PMP^{-1} \begin{bmatrix} x \\ y \\ z \end{bmatrix},$$

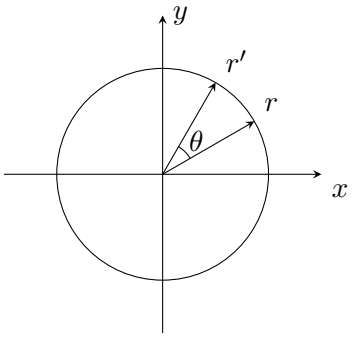
dvs. standardmatrisen för T ges av PMP^{-1}. Nu använder vi denna observation för att bestämma standardmatrisen för en rotation kring godtycklig axel i 3D:

Låt $r = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$ vara ortsvektorn för en punkt i rummet. Sambandet $r' = Mr$, där

$$M = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

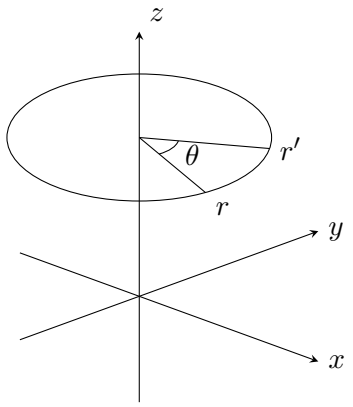
ger då en vridning av r med en vinkel θ kring z-axeln (fig.). Vi vill nu vrida en punkt kring en axel genom origo med riktningsvektor

$$v = \begin{bmatrix} a \\ b \\ c \end{bmatrix}.$$



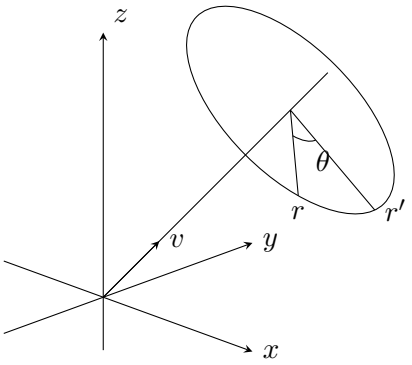
En lämplig ny (högerorienterad) bas $\mathbf{B} = \{f_1, f_2, f_3\}$ med f_3 i samma riktning som v får vi genom att ta $f_1 = u/\|u\|$, $f_2 = w/\|w\|$ och $f_3 = v/\|v\|$ där

$$u = \begin{bmatrix} b \\ -a \\ 0 \end{bmatrix}, \qquad w = \begin{bmatrix} ac \\ bc \\ -(a^2+b^2) \end{bmatrix}.$$



Med $P = \begin{bmatrix} f_1 & f_2 & f_3 \end{bmatrix}$ (basbytesmatrisen) så ges sambandet mellan nya och gamla koordinater av $r = P\rho$ där $\rho = [r]_{\mathbf{B}}$ är koordinaterna för r i basen $\mathbf{B}$. I denna bas så ges matrisen för rotationen av matrisen M ovan. Vi får

$$\rho' = M\rho \qquad \text{och} \qquad r' = PMP^{-1}r.$$



På grund av att vi valt en specifik bas, en så kallad *ON-bas*, så är kolonnerna i P ortonormerade (senare i kursen) och därmed är $P^{-1} = P^T$. Multiplikation med matrisen $R = PMP^T$ ger således en vridning med vinkeln θ kring vektorn v. En samtidig vridning av flera ortsvektorer $r_1, r_2, \ldots, r_n$ åstadkommes genom matrismultiplikationen

$$R \begin{bmatrix} r_1 & r_2 & \cdots & r_n \end{bmatrix} = \begin{bmatrix} r_1' & r_2' & \cdots & r_n' \end{bmatrix}.$$

Uppgift 4. Skriv en funktionsfil `R=rotation(v,t)` där
v är den vektor som ger rotationsaxeln,
t är den skalär som ger rotationsvinkeln och där
R är den beräknade 3×3 rotationsmatrisen.

Eventuell finputsning: Om v är parallell med z-axeln, blir vektorn u enligt proceduren ovan lika med nollvektorn. Tillfoga en villkorssats som gör att programmet klarar även detta fall. □

Uppgift 5. Skriv en funktionsfil `cube(v,dt)` som åstadkommer en roterande kub i grafikfönstret till MATLAB, där rotationsaxeln ges av vektorn v och kuben vrids vinkeln dt i varje litet rotationssteg. Någon utvariabel behövs inte, filen ska ju bara åstadkomma plottning.

Tillverka en matris K vars kolonner är koordinatvektorerna för kubens hörn, skrivna i en sådan ordning att vi kan plotta kubens alla kanter. Mata in dessa vektorer som $k_1, k_2, \ldots, k_8$ och bilda sedan K genom att räkna upp vektorerna i lämplig ordning: $K = \begin{bmatrix} k_1 & k_2 & \ldots \end{bmatrix}$ (rita figur som stöd). För att inte missa någon kant måste vi ta med varje hörn (minst) två gånger.

Vi ritar nu figuren genom att plotta y-koordinaterna mot x-koordinaterna med kommandot

```
>> plot(K(1,:),K(2,:))
```

Genom att utelämna z-koordinaterna får vi kubens projektion på xy-planet. Animationen blir snyggast om man låter kubens centrum ligga i origo. Programmet kan t.ex. bygga på en loop där man i varje steg vrider kuben exempelvis $dt = 1/100$ varv, ritar (plotta kubmatrisens första rad mot den andra, precis som ovan) och ger kommandot `drawnow` eller `pause(dt)` (använd `help` i MATLAB). Ange `axis square` efter plotkommandot.

□

Mera om plottning av 3D-objekt

Det handlar om att avbilda en projektion av objektet på ett plan. Ett enkelt sätt att göra detta är som rekommenderades i ovanstående uppgift att välja xy-planet som projektionsplan. Man plottar då punkternas y-koordinater mot x-koordinaterna (z-koordinaterna lämnas därhän). Eftersom man här kan välja rotationsaxel fritt, kan man ändå åstadkomma alla perspektiv på en roterande kub om man nöjer sig med att projicera på xy-planet.

Annars finns i MATLAB ett 3D-plot-kommando `plot3`. Kommandot i föregående uppgift blir då istället

```
>> plot3(K(1,:),K(2,:),K(:,3))
```

Den projektion som används här, innebär att linjer som är parallella också ser parallella ut i projektionen. I verkligheten ser man dock parallella linjer konvergera mot en punkt långt borta (t.ex. skenorna på ett järnvägsspår). Betraktar man kuben på lite håll, så påverkas parallelliteten ganska lite. Det ser alltså någorlunda realistiskt ut, om man tänker sig att kuben inte befinner sig alltför nära ögat. Däremot kan det vara svårt att se vad som är fram och bak på kuben, och en vanlig synvilla är att den plötsligt byter rotationsriktning. Detta kan avhjälpas genom att man märker en av kubens sidor, t ex genom att rita ut diagonalerna i den sidan. (Vill man åstadkomma nyss nämnda projektion med konvergerande linjer, kan detta göras med så kallade homogena koordinater. Man arbetar med 4×4-matriser — se Lay, avsnitt 2.7.