CHALMERS | GÖTEBORGS UNIVERSITET

Matematiska Vetenskaper
Carl-Henrik Fant, Lennart Falk

16 december 2011

Matematik med Matlab för M, Td, E, V, Vt 2012.

LAB 2: Några geometriska uppgifter och plottning av figurer.

Inledning

Kommandon som kan vara till nytta i dessa uppgifter :

norm	beräknar längden av en vektor.
acos	arcus-cosinus.
dot	beräknar skalärprodukten av två vektorer (alternativt: x'*y om x och y är kolonnvektorer).
cross	beräknar vektorprodukten av två vektorer (aktuellt i nästa lab).

Vinkeln mellan två vektorer

Uppgift 1: Skriv en funktionsfil som beräknar vinkeln mellan två givna vektorer.
Matematisk formel :

$$\mathbf{x} \cdot \mathbf{y} = |\mathbf{x}|\,|\mathbf{y}| \cos v$$

Funktionsfil: *function* v=*vinkel*(x, y)
Input : x,y - två vektorer
Output : v - vinkeln mellan x och y.
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 $\mathbf{n} = (a, b, c)$ av längd 1. Tag en punkt med ortsvektorn $\mathbf{x}$, och låt $\mathbf{p}$ och $\mathbf{s}$ vara ortsvektorn för ortogonala projektionen respektive speglingen av $\mathbf{x}$ i planet π. Om nu $\mathbf{x}_0$ är ortsvektorn för en punkt i planet så ger projektionsformeln

$$\mathbf{p} - \mathbf{x} = ((\mathbf{x}_0 - \mathbf{x}) \cdot \mathbf{n})\mathbf{n} = (\mathbf{x}_0 \cdot \mathbf{n} - \mathbf{x} \cdot \mathbf{n})\mathbf{n} = (d - \mathbf{x} \cdot \mathbf{n})\mathbf{n}$$

(Observera att $\mathbf{x}_0 \cdot \mathbf{n} = d$ för en punkt på planet)
För spegelpunkten gäller $\mathbf{s} = \mathbf{x} + 2(\mathbf{p} - \mathbf{x})$, dvs

$$\mathbf{s} = \mathbf{x} + 2(d - \mathbf{x} \cdot \mathbf{n})\mathbf{n}$$

Uppgift 2: Skriv ett program som gör följande:

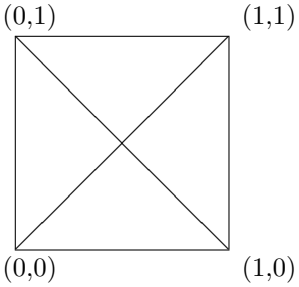
Givet planet $Ax + By + Cz = D$ och en punkt $\mathbf{x}$ i rummet, beräkna spegelbilden $\mathbf{s}$ av $\mathbf{x}$ i planet. Funktionsfil: *function* s=*spegel*(x, A, B, C, D)
Input : Punkten x och parametrarna i planets ekvation.
Output : Spegelbilden s.
För bästa anpassning till nästa uppgift, presentera alla förekommande 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

$$P = \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbf{p}_1 & \mathbf{p}_2 & \cdots & \mathbf{p}_n \end{bmatrix}$$

$\mathbf{x}$ och $\mathbf{y}$ är då radvektorer av längd n av x- resp y-koordinatern för punkterna, medan $\mathbf{p}_i$ är en kolonnvektor med kordinaterna för en punkt. Om vi vill rita en kurva på parameterform så väljer vi $\mathbf{p}_i = \begin{bmatrix} x(t_i) \\ y(t_i)) \end{bmatrix}$ för tillräckligt täta t-värden. En polygon får vi om vi låter $\mathbf{p}_i$ vara hörnen uppräknade i en lämplig ordning. Vi ritar nu figuren genom att plotta x-koordinaterna mot y-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 "spegel" i föregående avsnitt för att bestämma figurens spegelbild i linjen $x + 2y = 1$. Observera att spegling i plan ska bytas ut mot spegling i linje: välj $C = 0$ och mata in vektorn $\mathbf{x}$ med z-koordinaten noll. Plotta linjen, figuren och dess spegelbild i samma figur. Kommandot "axis equal" efter plot är nödvändigt för att speglingen ska se korrekt ut.

Rotationer av objekt i 3d

Först lite om basbyte och linjära transformationer.
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. Om rotationsaxeln är en annan, kan man byta till en annan bas $(\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3)$, där $\mathbf{f}_3$ är en vektor i rotationsaxelns riktning. Antag att koordinaterna för en vektor $\mathbf{u}$ i standardbasen $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$ heter (x, y, z) och i den nya basen (ξ, η, ζ), då är
$\mathbf{u} = x\mathbf{e}_1 + y\mathbf{e}_2 + z\mathbf{e}_3 = \xi\mathbf{f}_1 + \eta\mathbf{f}_2 + \zeta\mathbf{f}_3$, vilket också kan skrivas

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \mathbf{f}_1 & \mathbf{f}_2 & \mathbf{f}_3 \end{bmatrix} \begin{bmatrix} \xi \\ \eta \\ \zeta \end{bmatrix} = P \begin{bmatrix} \xi \\ \eta \\ \zeta \end{bmatrix} \qquad P\text{:s kolonner är de nya basvektorerna.}$$

Om nu T är en linjär transformation som i basen $(\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3)$ har matrisen M, och om
$T(\mathbf{u}) = x'\mathbf{e}_1 + y'\mathbf{e}_2 + z'\mathbf{e}_3 = \xi'\mathbf{f}_1 + \eta'\mathbf{f}_2 + \zeta'\mathbf{f}_3$, så gäller

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

Standardmatrisen för T är därför PMP^{-1}. (Denna kalkyl genomförs också på nästa sida).

Beräkning av rotationsmatriser

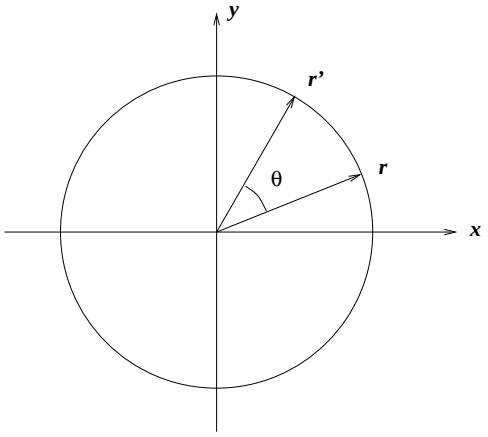
Matematisk bakgrund:

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

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

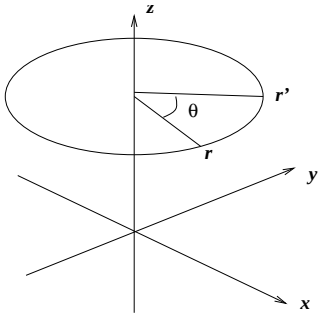
ger då en vridning av $\mathbf{r}$ en vinkel θ kring z-axeln (fig). Vi vill nu vrida en punkt kring en axel genom origo med riktningsvektor

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



Vi byter till en ny lämplig (högerorienterad) *ON-bas* $\mathcal{B} = \{\mathbf{f_1}, \mathbf{f_2}, \mathbf{f_3}\}$ som väljs så att $\mathbf{f_1} = \mathbf{u}/|\mathbf{u}|$, $\mathbf{f_2} = \mathbf{w}/|\mathbf{w}|$, och $\mathbf{f_3} = \mathbf{v}/|\mathbf{v}|$ där

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



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

$$\rho' = M\rho$$

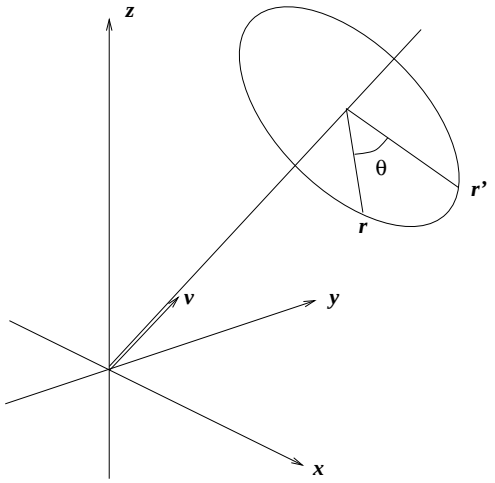
och

$$\mathbf{r}' = P\rho' = PM\rho = PMP^{-1}\mathbf{r}.$$

Notera att eftersom kolonnerna i P utgör en ON-bas så är $P^{-1} = P^T$. Multiplikation med matrisen $R = PMP^T$ ger således en vridning med vinkeln θ kring vektorn $\mathbf{v}$. En samtidig vridning av flera ortsvektorer $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_n$ åstadkommes genom matrismultiplikation

$$R[\mathbf{r}_1\ \mathbf{r}_2\ \ldots\ \mathbf{r}_n] = [\mathbf{r}'_1\ \mathbf{r}'_2\ \ldots\ \mathbf{r}'_n]$$

.



Uppgifter

a. **Rotationsmatrisen**
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 3x3 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!

b. **En roterande kub**

Skriv en funktionsfil *kub*(v,t) som åstadkommer en roterande kub i grafikfönstret till MATLAB där rotationsaxeln ges av vektorn v och kuben vrids vinkeln t 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 $k_1, \ldots k_8$, bilda sedan K genom att räkna upp vektorerna i lämplig ordning $K = [k1\ k2 \ldots]$ (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 x-koordinaterna mot y-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 1/100 varv, ritar (plotta kubmatrisens första rad mot den andra, precis som ovan) och ger kommandot *drawnow* eller *pause(*Δt*)* (se Matlab help!). 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 x-koordinater mot y-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 åstadkommas med sk homogena koordinater. Man arbetar med $4 \times 4-$matriser - se Lay, avsnitt 2.7!).