

Instruktioner och lösningsförslag till uppgift 1, del i
momentet
T1, Matlabprogrammering, modellering av tvådimensionell
strålgång,
ingående i kursen
ET1522 SOM17 Beräknings- och simuleringsteknik (distans)
påbörjat 2013, senaste uppdateringen 12 juni 2017

Frida Gleisner, Anders Hultgren och Matz Lenells

12 juni 2017

1 Inledning

I det här dokumentet hittar du instruktioner för att lösa uppgift 1. Programmet i Matlab, som är lösningen till uppgiften, kommer här byggas upp stegvis med syftet att du ska förstå varje delmoment. Den stegvisa uppbyggnaden innebär att delar av programkoden kommer att skrivas om efter hand, allt för att tydliggöra betydelsen av de olika kommandon som används.

Du ska följa instruktionerna, alternativt skriva ett eget program, samt lägga till dina egna kommentarer i programmet. Din m-fil ska sedan läggas in på it's learning.

1.1 Att tänka på vid inlämning av filer

Anmärkning 1. När ni lämnar in filer så tillfoga era namn i filnamnet. Nedan finns exempel på hur man kan skriva.

```
raypolygonAsaOrn.m
```

om man heter Åsa Örn och

```
raypolygonEHakansson.m
```

om man heter Esmaralda Håkansson.

Matlab är känslig för valet av filnamn, därför bör man undvika tecken såsom å, ä och ö. Vidare ska man inte ha blanktecken i filnamn.

□

Anmärkning 2. Vi har fått in någon fil som inte går att köra. Detta kan bero på att ni har ändrat filnamnet och exempelvis använt ett "ö" i filnamnet. För att minska risken att vi inte kan köra de filer som ni lämnar in så ber vi er att provköra just de filer som ni lämnar in. Vidare, använd en ny version av Matlab, gärna från 2017 och inte senare än från 2015.

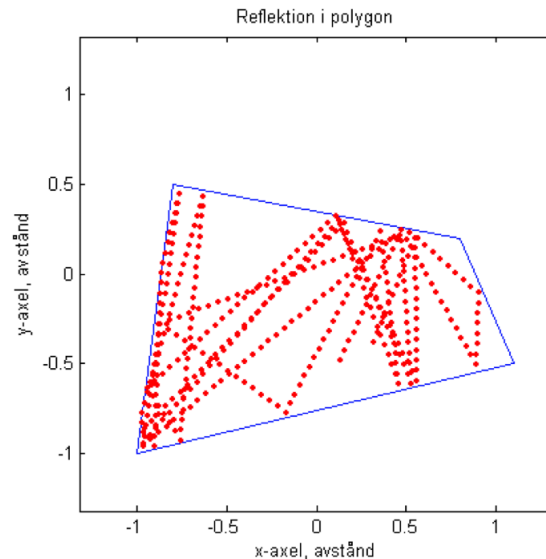
□

1.2 Instruktioner för att genomföra uppgiften

Instruktionerna består av delarna

1. Rita en polygon

Reflektion av stråle



Figur 1: Så här kan polygonen och strålgången se ut som genereras av det program som ska göras i denna uppgift.

2. Rita en stråle
3. Anpassning av stråle och diagram
4. Reflektion av stråle
 - 4.1 Detektion av kant
 - 4.2 Reflektion
 - 4.3 Justering av reflektionspunkt
5. Fel som kan uppstå (välj annan rubrik)

Avdelningen Reflektion av stråle"kompletteras med en power point-presentation.

2 Rita en polygon

Vi ska nu skapa ett nytt program som ska generera en reflekterande stråle i en polygon, se figur 1. Programmet kommer att likna rayrectangle.m som du öppnade när du läste dokumentet *A very short introduction to Matlab*.

För att skapa ett nytt program skriver du

```
edit raypolygonAsaOrn
```

i kommandofönstret. Delen AsaOrn i filnamnet raypolygonAsaOrn ska du byta ut mot ditt eget namn, se anmärkning 1 i avsnitt 1.1. Glöm inte att trycka på enter. Svara ja på frågan som kommer upp i popup-fönstret. Nu ska ett nytt fönster öppnas i editorn där det nya programmet ska skrivas. Skriv

```

%% Ray in a polygon

%Definition of the four vertices
v1=[3;0];
v2=[0;3];
v3=[-2;0];
v4=[-1;-2];
%Definition of the polygon
polygon=[v1 v2 v3 v4 v1];
%Plot the polygon
plot(polygon(1,:),polygon(2,:))

```

Du har här fått förslag på kommentarer (efter %-tecknen), i fortsättningen får du skriva egna kommentarer. Kör filen genom att skriva

```
raypolygonAsaOrn
```

i kommandofönstret och tryck return, kom ihåg att det är den senast sparade versionen av programmet som anropas.

Anmärkning 3. Du kan beordra Matlab att köra kommandofilen raypolygonAsaOrn på flera olika sätt.

1. Som nämnts ovan.
2. Om du har kommandofönstret aktivt kan du bläddra igenom tidigare kommandon genom att använda piltangenterna på tangentbordet. På så sätt kan du undvika att skriva samma kommandon flera gånger.
3. Du editerar kommandofilen i ett fönster som i sin överkant har en list med olika knappar. En av knapparna är pilformad och grön och under knappen står det "Run". När du trycker på knappen sparas filen och programmet körs.
4. Tryck på funktionsknappen F5, då fås samma effekt som om Run-knappen trycks, se föregående punkt.

□

Matlab ska nu rita en polygon i fönstret Figur 1 (eftersom inget annat nummer på fönster angavs). Figur 2 i detta dokument visar hur Matlabs figur kan se ut.

Programmen vi skriver i Matlab saknar kommandona `begin` i början och `end` i slutet. När ett program anropas motsvaras det av att skriva alla kommandon i en följd i kommandofönstret. Detta gör att variabler kan behöva rensas i början av ett program för att inte felaktiga värden ska användas. Detta kan göras med kommandot `clear`. Vidare kan även en figur behöva rensas för att inte figurer ska ritas ovanpå varandra. Vi ska nu lägga till några rader som rensar figuren och anger diagrammets storlek. Skriv följande rader i editorn under definitionen av polygonen

```

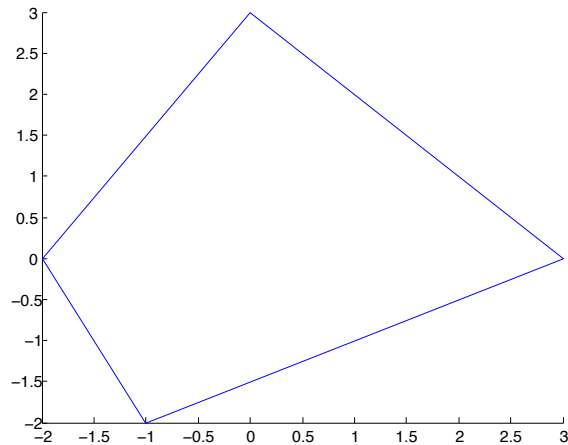
figure(gcf)
clf
axis([-5 5 -5 5]), axis('equal')
hold on

```

Det första kommandot flyttar fram figurfönstret så att det ligger överst på skärmen, akronymen `gcf` står för *get current figure*.

Kommandot `clf` säger åt Matlab att rensa figuren (*clear figure*).

Kommandot `axis([-5 5 -5 5])`, om det står för sig själv, ger ett rektangulärt diagram där stegen på x-axeln är längre än stegen på y-axeln. Läger vi till `axis('equal')`



Figur 2: Polygon som ritas när koden som börjar på sidan 3 körs.

kommer diagrammet fortsätta att vara rektangulärt men avstånden mellan stegen på axlarna blir lika långa, värdena på axlarna bestäms delvis av hur polygonen ser ut. Om man inte har samma skala i höjd- och längsled blir strålgången konstig, se figur 17. Vill man ha ett kvadratisk diagram kan `equal` bytas ut mot `square`.

Kommandot `hold on` berättar för Matlab att vad som angetts för axlarna ska fortsätta gälla när polygonen ritas. Kommandot åtföljs av `hold off` som läggs till längst ner i programmet. Spara och kör programmet.

Testa gärna vad som händer om du tar bort vissa kommandon (kan göras genom att skriva ett `%`-tecken först i raden). Ta exempelvis bort `clf` och ändra några gånger på koordinaterna för polygonen.

Vi ska nu titta på den rad i programmet som ritar polygonen för att förstå hur den är uppbyggd.

```
plot (polygon(1, :), polygon(2, :))
```

Kommandot `plot(3, 2)` skulle gett en punkt med koordinaterna (3,2), men här använder vi oss av matrisen `polygon`. Skriv i kommandofönstret

```
polygon
```

vilket bör ge dig en matris i kommandofönstret med polygonens fyra hörn som fyra kolonnvektorer samt vektorn `v1` som sista avslutande kolonn. Testa att skriva

```
polygon(2, 4)
```

du bör nu få matrisens värde för andra raden, fjärde kolonnen. Skriv istället

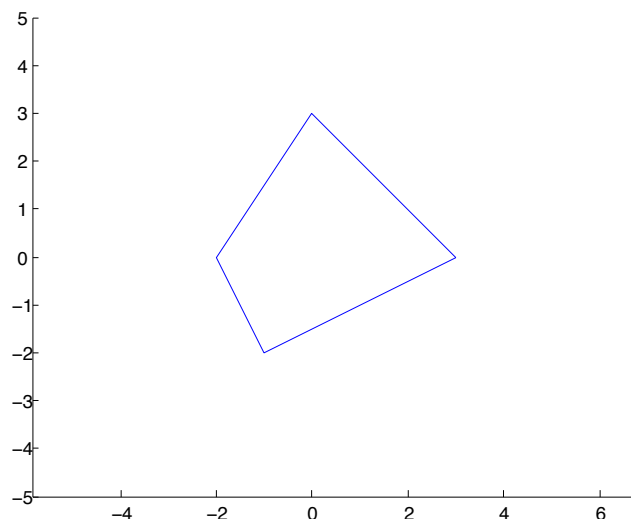
```
polygon(2, :)
```

och hela andra raden i matrisen visas.

Kommandot `plot` i programmet parar samman första raden med andra raden för matrisen `polygon`.

Programmet är nu tänkt att generera ett diagram som det i figur 3 och kan se ut så här (för vissa kommandon är den inbördes ordningen inte av betydelse för resultatet).

```
% Ray in a polygon
```



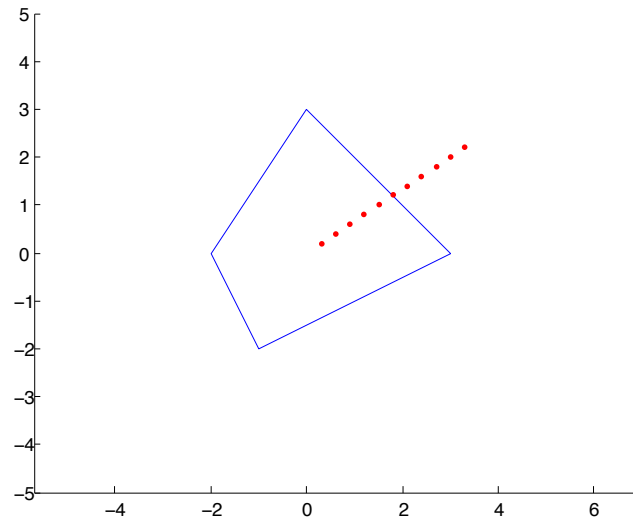
Figur 3: Polygon som ritas när koden som börjar på sidan 4 körs.

```
%Definition of the four vertices
v1=[3;0];
v2=[0;3];
v3=[-2;0];
v4=[-1;-2];
%Definition of the polygon
polygon=[v1 v2 v3 v4 v1];
clf
figure(gcf)
axis([-5 5 -5 5]), axis('equal')
hold on
%Plot the polygon
plot(polygon(1,:),polygon(2,:))
hold off
```

3 Rita en stråle

Med början i origo ska vi nu rita en stråle som en röd prickad linje, till en början kommer strålen inte att reflekteras i polygonens kanter. Strålen har hastighet och riktning som vi behöver ange, vi behöver också bestämma stegantal. Skriv på raden efter kommandot `plot`

```
r0=[0;0];
r=r0;
c=[.3;.2];
dt=1;
kmax=10;
for t=0:dt:dt*kmax;
    r=r+c*dt;
    pause(0.1)
    plot(r(1),r(2),'.r')
```



Figur 4: Polygon och stråle som ritas när koden som börjar på sidan 6 körs.

end

Vektorn som visar hur strålen rör sig kallar vi för r , $r0$ anger utgångsläget för strålen.

Hastighetsvektorn är c , här satt till $(0.3, 0.2)$.

Genom for-satsen kommer Matlab tilldela värden för r från 0 till $dt * kmax$. Eftersom $kmax$ är satt till 10 blir antalet värden 11.

För att få strålen att se ut att röra på sig lägger vi till en paus. (Pausen kan göras mindre om stora värden på $kmax$ används.

Tillägget `' .r'` vid kommandot `plot` berättar för Matlab att linjen ska bestå av punkter och vara röd. För fler alternativ skriv `help plot` i kommandofönstret.

När programmet körs ska en röd prickad stråle ritas in i samma diagram som polygonen.

Programmet är nu tänkt att ge ett diagram som det i figur 4 och kan se ut så här:

```
%% Ray in a polygon

%Definition of the four vertices
v1=[3;0];
v2=[0;3];
v3=[-2;0];
v4=[-1;-2];
%Definition of the polygon
polygon=[v1 v2 v3 v4 v1];
figure(gcf)
clf
axis([-5 5 -5 5]), axis ('equal')
hold on
%Plot the polygon
plot(polygon(1,:),polygon(2,:))
r0=[0;0];
r=r0;
c=[.3;.2];
dt=1;
```

```

kmax=10;
for t=0:dt:dt*kmax;
    r=r+c*dt;
    pause(0.1)
    plot(r(1),r(2),'.r')
end
hold off

```

4 Anpassning av stråle och diagram

De värden som angetts för polygonens hörn, diagrammets storlek och strålens steglängd ger en väl anpassad bild. Ändras polygonen till att bli avsevärt mindre eller större blir resultatet inte lika bra. För att anpassa diagram och steglängd till polygonen lägger vi till en variabel `scale` som beror på den koordinat i polygonen som har högst absolutvärde. För de värden som angavs ovan är det 3. För att hitta detta värde använder vi kommandona `abs` och `max`. Vi börjar med att pröva dessa kommandon i kommandofönstret. Skriv

```

polygon
abs(polygon)

```

Detta ska ge dig matrisen för polygonen och sedan absolutvärdena för samma matris. Skriv sedan

```

max(abs(polygon))
max(max(abs(polygon)))

```

Du bör nu ha hittat det största absolutvärdet i matrisen. Vi sätter variabeln `scale` till 1.2 gånger matrisens absolutvärde

```

scale=1.2*max(max(abs(polygon)));

```

Raden kan skrivas in i programmet efter definitionen av polygonen. Vår nya variabel hjälper oss att anpassa längden av axlarna i figuren och steglängden för strålen.

Vi byter ut värdena för längden av axlarna (tidigare satta till -5 och 5) mot `scale`

```

axis([-scale scale -scale scale]), axis('equal')

```

Figurfönstret kommer nu vara anpassat till figuren även om polygonen ändras.

För att anpassa strålen till storleken på polygonen byter vi ut värdet för `dt`

```

dt=.1*scale;

```

För att namnge figurfönstret och sätta ut enheter på axlarna skriver vi längst ner i programmet

```

title('Ray reflection in a polygone')
xlabel('x-axis, distance [m]')
ylabel('y-axis, distance [m]')

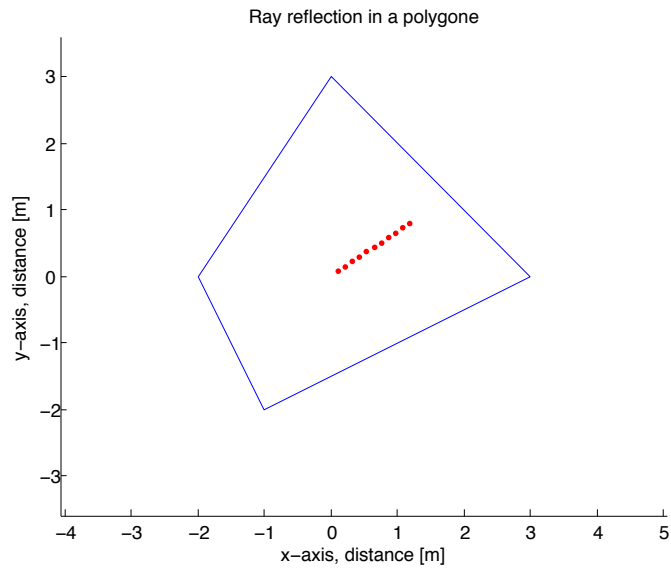
```

Programmet är nu tänkt att ge ett diagram som det i figur 5 och kan se ut så här:

```

%% Ray in a polygon

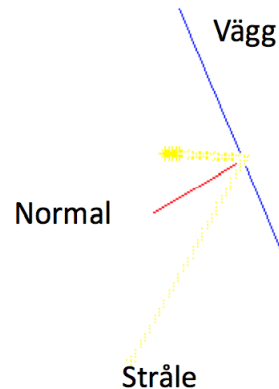
```



Figur 5: Polygon och stråle som ritas när programmet som börjar på sidan 7 körs.

```
%Definition of the four vertices
v1=[3;0];
v2=[0;3];
v3=[-2;0];
v4=[-1;-2];
%Definition of the polygon
polygon=[v1 v2 v3 v4 v1];
scale=1.2*max(max(abs(polygon)));
figure(gcf)
clf
axis([-scale scale -scale scale]), axis ('equal')
hold on
%Plot the polygon
plot(polygon(1,:),polygon(2,:))
r0=[0;0];
r=r0;
c=[.3;.2];
dt=.1*scale;
kmax=10;
for t=0:dt:dt*kmax;
    r=r+c*dt;
    pause(0.1)
    plot(r(1),r(2),'.r')
end
hold off
title('Ray reflection in a polygone')
xlabel('x-axis, distance [m]')
ylabel('y-axis, distance [m]')
```


Frågeställningar vid reflektion



Figur 6

- Hur detekteras vilken kant som träffas av strålen?
- Hur ändras riktningen på strålen?

5 Reflektion av stråle

För att få strålen att reflekteras i polygonens sidor behöver vi veta när strålen når en sida och riktningen för strålen ska sedan räknas om. Nu följer en kortfattad genomgång av matematiken som behövs för att göra dessa beräkningar. Genomgången finns även som power point, filen heter Reflektion och finns på itslearning.

5.1 Hörn- och kantvektorer samt normaler

5.1.1 Beteckning av hörn- och kantvektorer

Beteckna ett av polygonens hörn med v_1 , se figur 7. Gå moturs i polygonen och beteckna nästa hörn med v_2 . Fortsätt och beteckna hörnen med v_3 tom v_4 . Låt kantvektorn u_1 ges av $u_1 = v_2 - v_1$. Kantvektorn u_2 ges av $u_2 = v_3 - v_2$ och fortsätt på det viset så att slutligen kantvektorn u_4 ges av $u_4 = v_1 - v_4$.

Om man tänker sig att man går längs med kantvektorerna i positiv riktning så kommer man ha polygonens inre del till vänster om sig.

Här används begreppen sida och kant synonymt.

5.1.2 Inåtgående normaler

Till varje kantvektor ritar vi en normal och normalerna betecknas med n_1 tom n_4 , se figur 8. Om vi tänker oss att normalen utgår från kantens mitt så ska normalen riktas in mot polygonen. En sådan normal till en kant kan fås genom att vrida kanten 90 grader i positiv riktning.

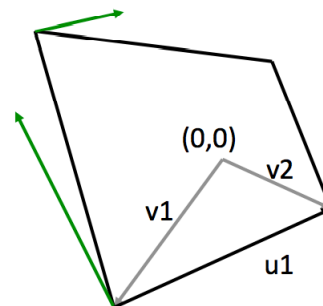
Vi väljer att normaler n_1 börjar i hörnet v_1 och n_2 i hörnet v_2 etc. I detta fall finns normaler som inte pekar in i polygonen. Detta beror på att vi har valt att dra normalen från ett hörn. Om normalen börjar inne på sin sida så kommer normalen att åtminstone till en början att gå in i polygonen.

För varje normal gäller att den är vriden 90° moturs jämfört med sidan den tillhör.

Beteckning av hörn- och kantvektorer

Beteckna ett av polygonens hörn med v_1 . Gå moturs i polygonen beteckna nästa hörn med v_2 . Fortsätt och beteckna hörnen med v_3 tom v_4 . Låt kantvektorn u_1 ges av $u_1 = v_2 - v_1$, Kantvektorn u_2 ges av $u_2 = v_3 - v_2$ och fortsätt på det viset så att slutligen kantvektorn u_4 ges av $u_4 = v_1 - v_4$. Om man tänker sig att man går längs med kantvektorerna i positiv riktning så kommer man ha polygonens inre del till vänster om sig. Här används begreppen sida och kant synonymt.

Linjen som innehåller en kant kallas för en kantlinje.



Figur 7

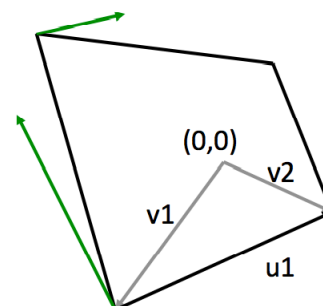
Inåtgående normaler till kanterna

Till varje kantvektor ritas vi en normal och normalerna betecknas med n_1 tom n_4 . Om vi tänker oss att normalen utgår från kantens mitt så ska normalen riktas in mot polygonen. En sådan normal till en kant kan fås genom att vrida kanten 90 grader i positiv riktning.

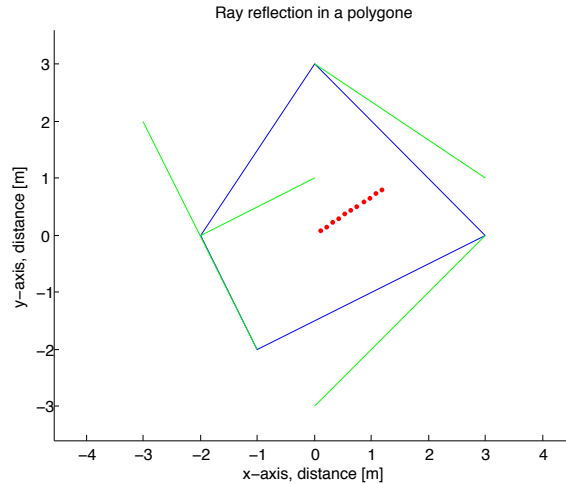
Vi väljer att normaler n_1 börjar i hörnet v_1 och n_2 i hörnet v_2 etc.

I detta fall finns normaler som inte pekar in i polygonen. Detta beror på att vi har valt att dra normalen från ett hörn. Om normalen börjar inne på sin sida så kommer normalen att åtminstone till en början att gå in i polygonen.

För varje normal gäller att den är vriden 90° moturs jämfört med sidan den tillhör.



Figur 8



Figur 9: Polygonen med normaler ritade.

5.1.3 Beräkning av normal

På föregående sida sägs att en normal till en kant fås genom att vrida motsvarande kantvektor 90 grader i positiv riktning, dvs 90 grader moturs.

Normalen kan beräknas på olika sätt.

En vridning 90° moturs av vektorn $u_1 = (x_1, y_1)$ ger vektorn $(-y_1, x_1)$. En rotationsmatris kan användas, mer om detta i instruktionerna för lösningen av nästa problem.

$$\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -y \\ x \end{pmatrix} \quad (5.1)$$

Normalen är här beräknad och ritad så att den har samma längd som kanten den är normal till. För den här delen av uppgiften är det endast normalens riktning som är av betydelse.

5.1.4 Kontroll av normalerna

För att kontrollera om normalerna är rätt beräknade kan vi rita ut dem i polygonen. (Dessa kommandon kan sedan tas bort eller göras överksammas med hjälp av %-tecken). Normalen n1 kan ritas ut genom att vi skriver

```
plot([0;n1(1)], [0;n1(2)])
```

Nu kommer vektorn som ritas ut börja i (0,0) och sluta i (n1(1),n1(2)). Liknande kommandon kan skrivas för de andra normalerna. Vill vi istället att normalen n1 ska börja i v1, n2 ska börja i v2 osv. kan vi flytta normalerna. För att n1 ska börja i v1 sätter vi (v1(1),v1(2)) som startpunkt. Ändpunkt för n1 blir v1+n1 dvs. (n1(1)+v1(1), n1(2)+v1(2))

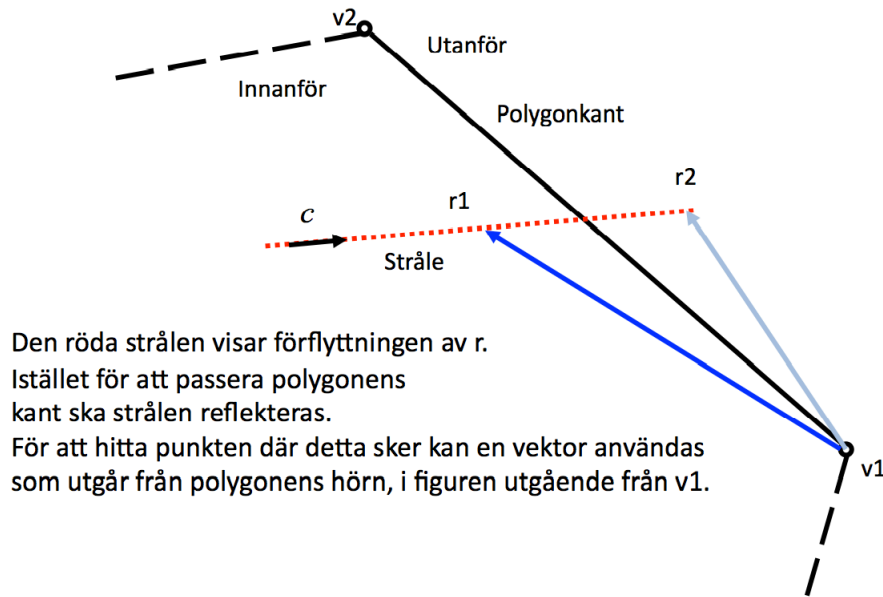
```
plot([v1(1);n1(1)+v1(1)], [v1(2);n1(2)+v1(2)], 'g')
```

```
plot([v2(1);n2(1)+v2(1)], [v2(2);n2(2)+v2(2)], 'g')
```

```
plot([v3(1);n3(1)+v3(1)], [v3(2);n3(2)+v3(2)], 'g')
```

```
plot([v4(1);n4(1)+v4(1)], [v4(2);n4(2)+v4(2)], 'g')
```

Detektion av kant



Figur 10

Ritas vektorerna ut bör diagrammet se ut som visas i figur 9.

5.2 Detektion av kant

För att veta hur reflektionen av strålen ska beräknas behöver vi veta när en kant träffas och vilken kant som träffas. Genom att beräkna en normal till varje kant kan vi genom projektion ta reda på om strålen är utanför polygonen.

Figurerna 10 - 12 visar idén bakom den metod vi här ska använda för att bestämma om en punkt ligger inne i polygonen. I figur 13 visas det kriterium som vi använder oss av.

Vi behöver först beräkna vektorerna för polygonens fyra sidor. Skriv efter definitionen av polygonen

```
u1=v2-v1;
u2=v3-v2;
u3=v4-v3;
u4=v1-v4;
```

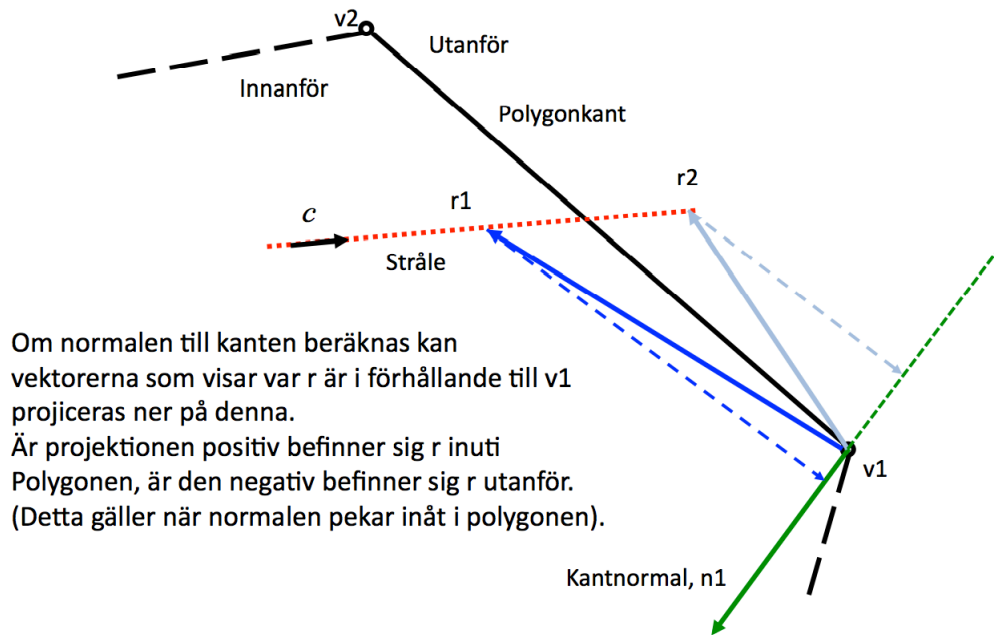
Vektorn $u1$ är nu den vektor som går från $v1$ till $v2$ osv. För att beräkna normalerna till dessa vektorer vrider vi dem 90° moturs. Skriv efter definitionen av kanterna

```
n1=[-u1(2); u1(1)];
n2=[-u2(2); u2(1)];
n3=[-u3(2); u3(1)];
n4=[-u4(2); u4(1)];
```

Vi ska nu projicera placeringen av r jämfört med vart och ett av de fyra hörnen, med normalen som tillhör hörnet, för att se om den är negativ. För detta använder vi skalärprodukt (vänta med att skriva in):

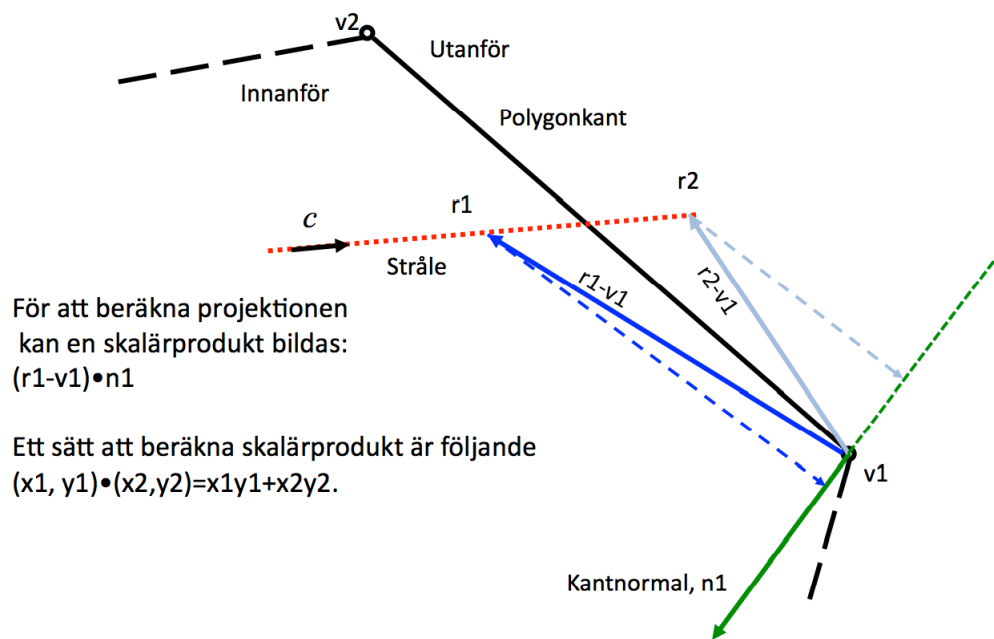
```
dot(n1, r-v1) <= 0
dot(n2, r-v2) <= 0
```

Detektion av kant



Figur 11

Detektion av kant



Figur 12

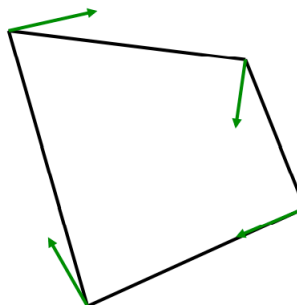
Detektion av kant, kriterium

För varje kantlinje gäller att polygonen helt ligger på kantlinjens ena sida. För att en punkt p ska kunna ligga i polygonen så måste den ligga på samma sida om kantlinje 1 som normalen n_1 ligger.

Det betyder att skalärprodukten

$$(p-v_1) \cdot n_1 \geq 0$$

Om denna olikhet gäller för varje kantlinje så ligger punkten p i polygonen.



Figur 13

Test av kriterium

I föregående bild anger vi ett kriterium för att en punkt p ligger i polygonen. Vi testar detta kriterium genom tänka oss en punkt p inne i polygonen. Kontrollera att för en sådan punkt är alla fyra skalärprodukterna positiva.

Låt sedan punkten p ligga utanför polygonen.

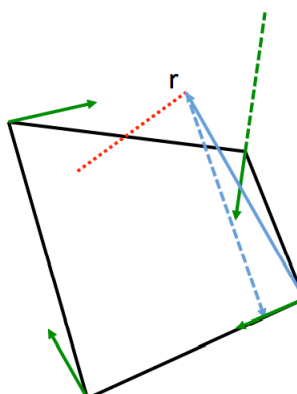
Blir kriteriets skalärprodukt mindre än noll för alla normaler?

Blir kriteriets skalärprodukter mindre än noll för någon normal?

Kan kriteriets skalärprodukter bli mindre än noll för två olika normaler?

Kan kriteriets skalärprodukter bli mindre än noll för tre olika normaler?

positiv
skalärprodukt



Figur 14

```
dot (n3,r-v3)<=0
dot (n4,r-v4)<=0
```

Vi vill att Matlab, för varje punkt på strålen, undersöker om någon av polygonens kanter har passerats. För detta skriver vi en in en if-sats som placeras inuti for-satsen under kommandot paus.

```
if dot (n1,r-v1)<=0
    %
elseif dot (n2,r-v2)<=0
    %
elseif dot (n3,r-v3)<=0
    %
elseif dot (n4,r-v4)<=0
    %
end
```

Matlab kommer nu att kunna detektera när stråler passerar en kant. När detta sker ska strålen reflekteras. Kommandona som genererar reflektionen kommer vi sedan skriva in efter var och en av de fyra skalärprodukterna.

Programmet är nu tänkt att se ut så här:

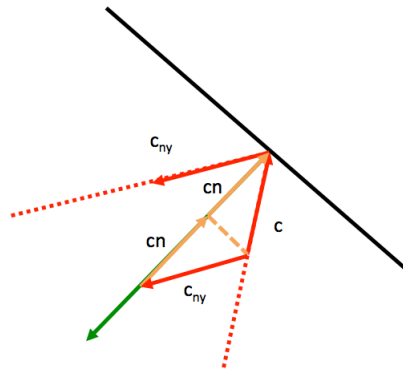
```
% Ray in a polygon
%Definition of the four vertices
v1=[3;0];
v2=[0;3];
v3=[-2;0];
v4=[-1;-2];
%Definition of the polygon
polygon=[v1 v2 v3 v4 v1];
u1=v2-v1;
u2=v3-v2;
u3=v4-v3;
u4=v1-v4;
n1=[-u1(2);u1(1)];
n2=[-u2(2);u2(1)];
n3=[-u3(2);u3(1)];
n4=[-u4(2);u4(1)];
scale=1.2*max(max(abs(polygon)));
figure(gcf)
clf
axis([-scale scale -scale scale]), axis ('equal')
hold on
%% Plot the polygon
plot(polygon(1,:),polygon(2,:))
r0=[0;0];
r=r0;
c=[.3;.2];
dt=.1*scale;
kmax=10;
for t=0:dt:dt*kmax;
    r=r+c*dt;
    pause(0.1)
    if dot (n1,r-v1)<=0
```

Beräkna ny riktning för strålen, 3

För att beräkna hur strålen reflekteras i kanten, beräknar vi en ny hastighetsvektor.

Den nya vektorn kan beräknas genom att hastighetsvektorn, c , projekteras på normalen.

$$c_{ny} = c - 2cn$$



Figur 15

```
%
elseif dot(n2,r-v2)<=0
%
elseif dot(n3,r-v3)<=0
%
elseif dot(n4,r-v4)<=0
%
end
plot(r(1),r(2),'r')
end
hold off
title('Ray reflection in a polygone')
xlabel('x-axis, distance [m]')
ylabel('y-axis, distance [m]')
```

5.3 Reflektion

Reflektionen av strålen i polygonens kant beräknas genom att hastighetsvektorn c projiceras på en normaliserad normal, alltså en normal med längden ett. Se figur 15. Vid detektionen av kanterna spelar längden på normalen ingen roll och alltså går det bra att ändra i beräkningarna för hur normalerna beräknas. Vi gör ett tillägg på varje rad så att de ser ut så här

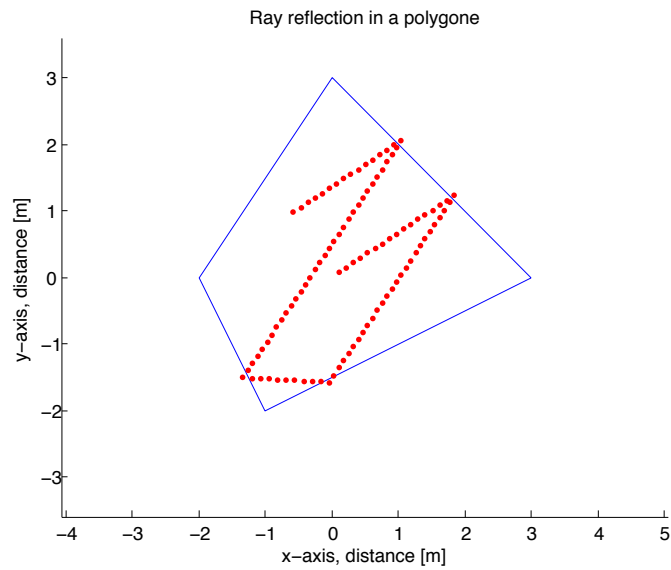
```
n1=[-u1(2);u1(1)]/sqrt(u1(1)^2+u1(2)^2);
n2=[-u2(2);u2(1)]/sqrt(u2(1)^2+u2(2)^2);
n3=[-u3(2);u3(1)]/sqrt(u3(1)^2+u3(2)^2);
n4=[-u4(2);u4(1)]/sqrt(u4(1)^2+u4(2)^2);
```

För varje möjlig detektion av kant ska vi nu beräkna en eventuell förändring av hastighetsvektorn, en förändring som bara sker om den aktuella delen i if-satsen är uppfylld. Den första förändringen blir

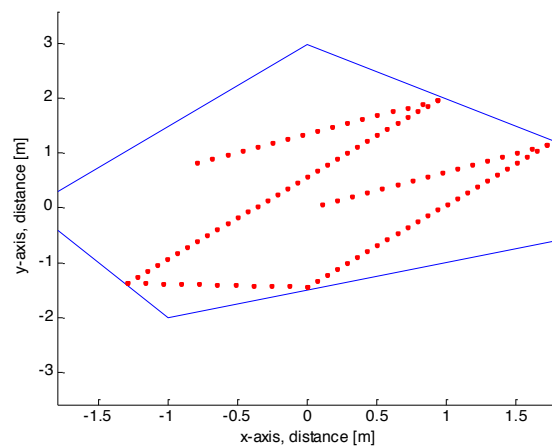
```
cn=dot(c,n1)*n1;
c=c-2*cn;
```

De andra ser ut på motsvarande sätt fast med andra normaler.

Om strålens längd justeras bör nu programmet generera en polygon och en studsande stråle.



Figur 16: Polygon och strålgång i fallet med lika skalor i höjd- och längsled. Här gäller att en stråles infallsvinkel är lika med dess utfallsvinkel.



Figur 17: Polygonen och strålgång i fallet med olika skalor i höjd- och längsled. Observera att i detta fall är en stråles infallsvinkel inte lika med dess utfallsvinkel.

Programmet är nu tänkt att ge ett diagram som det i figur 16 och att se ut ungefär så här:

```
% Ray in a polygon
%Definition of the four vertices
v1=[3;0];
v2=[0;3];
v3=[-2;0];
v4=[-1;-2];
%Definition of the polygon
polygon=[v1 v2 v3 v4 v1];
u1=v2-v1;
u2=v3-v2;
```

```

u3=v4-v3;
u4=v1-v4;
n1=[-u1(2);u1(1)]/sqrt(u1(1)^2+u1(2)^2);
n2=[-u2(2);u2(1)]/sqrt(u2(1)^2+u2(2)^2);
n3=[-u3(2);u3(1)]/sqrt(u3(1)^2+u3(2)^2);
n4=[-u4(2);u4(1)]/sqrt(u4(1)^2+u4(2)^2);
scale=1.2*max(max(abs(polygon)));
figure(gcf)
clf
axis([-scale scale -scale scale]), axis('equal')
hold on
%% Plot the polygon
plot(polygon(1,:),polygon(2,:))
r0=[0;0];
r=r0;
c=[.3;.2];
dt=.1*scale;
kmax=100;
for t=0:dt:dt*kmax;
    r=r+c*dt;
    pause(0.1)
    if dot(n1,r-v1)<=0
        cn=dot(c,n1)*n1;
        c=c-2*cn;
    elseif dot(n2,r-v2)<=0
        cn=dot(c,n2)*n2;
        c=c-2*cn;
    elseif dot(n3,r-v3)<=0
        cn=dot(c,n3)*n3;
        c=c-2*cn;
    elseif dot(n4,r-v4)<=0
        cn=dot(c,n4)*n4;
        c=c-2*cn;
    end
    plot(r(1),r(2),'.r')
end
hold off
title('Ray reflection in a polygone')
xlabel('x-axis, distance [m]')
ylabel('y-axis, distance [m]')

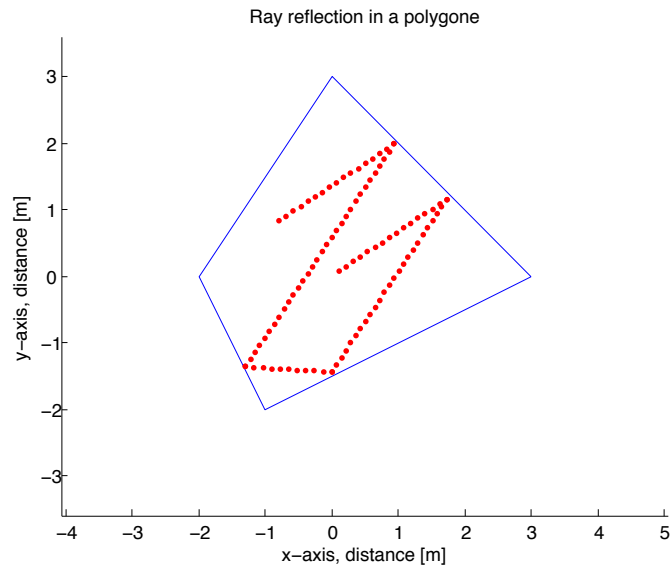
```

5.4 Justering av reflektionspunkt

Så som instruktionerna är givna ovan kommer strålen reflekteras vid polygonens sidor, men en punkt hamnar utanför polygonen (eller på linjen) vid varje reflektionstillfälle. Fundera över vad det beror på.

Ett sätt att lösa problemet är att backa strålen ett steg när en kant detekterats. Detta innebär att om r tilldelas ett nytt värde som visar sig ligga utanför polygonen måste strålen backa ett steg. Vi kan göra detta genom att ha två r -värden, r och rg . När for-satsen börjar är dessa värden lika, r får sedan ett nytt värde, om detta visar sig ligga utanför polygonen kan vi backa tillbaka till det gamla värdet, rg . Skulle if-satsen inte ge någon reflektion tilldelas även rg det nya värdet.

I programmet, före for-satsen, tilldelar vi variabeln rg värdet $r0$



Figur 18

```
rg=r0;
```

Sedan, sist i varje dela av if-satsen lägger vi till

```
r=rg;
```

Ifall Matlab inte detekterar någon kant ska rg få samma värde som r. Vi lägger till följande innan if-satsen slutar

```
else  
rg=r;
```

Programmet är nu tänkt att ge ett diagram som det i figur 18 och att se ut ungefär så här:

```
%% Ray in a polygon  
%Definition of the four vertices  
v1=[3;0];  
v2=[0;3];  
v3=[-2;0];  
v4=[-1;-2];  
%Definition of the polygon  
polygon=[v1 v2 v3 v4 v1];  
u1=v2-v1;  
u2=v3-v2;  
u3=v4-v3;  
u4=v1-v4;  
n1=[-u1(2);u1(1)]/sqrt(u1(1)^2+u1(2)^2);  
n2=[-u2(2);u2(1)]/sqrt(u2(1)^2+u2(2)^2);  
n3=[-u3(2);u3(1)]/sqrt(u3(1)^2+u3(2)^2);  
n4=[-u4(2);u4(1)]/sqrt(u4(1)^2+u4(2)^2);  
scale=1.2*max(max(abs(polygon)));  
figure(gcf)
```

```

clf
axis([-scale scale -scale scale]), axis ('equal')
hold on
%% Plot the polygon
plot(polygon(1,:),polygon(2,:))
r0=[0;0];
r=r0;
c=[.3;.2];
dt=.1*scale;
kmax=100;
rg=r0;
for t=0:dt:dt*kmax;
    r=r+c*dt;
    pause(0.1)
    if dot(n1,r-v1)<=0
        cn=dot(c,n1)*n1;
        c=c-2*cn;
        r=rg;
    elseif dot(n2,r-v2)<=0
        cn=dot(c,n2)*n2;
        c=c-2*cn;
        r=rg;
    elseif dot(n3,r-v3)<=0
        cn=dot(c,n3)*n3;
        c=c-2*cn;
        r=rg;
    elseif dot(n4,r-v4)<=0
        cn=dot(c,n4)*n4;
        c=c-2*cn;
        r=rg;
    else
        rg=r;
    end
    plot(r(1),r(2),'.r')
end
hold off
title('Ray reflection in a polygone')
xlabel('x-axis, distance [m]')
ylabel('y-axis, distance [m]')

```

Lägg in ditt program som en m-fil på itslearning under "inlämningsuppgifter", gärna innehållande dina egna kommentarer.

6 Utforska programmets begränsningar (valfritt moment)

Givet de värden som angetts i anvisningarna ovan: ändra till $c=[0.34, 0.27]$. Försök förklara vad det är som händer med strålen och hitta ett sätt att använda samma värde på c i samma polygon men med ett bättre utfall.

Byt koordinaterna för $v3$ till $(-0.1, 0)$. Varför får strålen en bana där den inte reflekteras i två av polygonens kanter?