Programmeringsuppgift 1
Linjer och plan

Innan du börjar:
För att kunna lösa uppgifterna nedan behöver du ha kunskap om

1. Vektorer, linjer och plan i $\mathbb{R}^3$.
2. Hur man hanterar vektorer i Matlab
3. Grundläggande programmeringsstrukturer som `for`- och `while`-slingor samt `if`-satser.

Gör de obligatoriska förberedande uppgifterna i Matlab Grader för Programmeringsuppgift 1 innan du påbörjar arbetet med uppgifterna nedan.

Redovisning:
Koderna för uppgift 1 och 2 skickas in via Canvas senast den 25/9-18 kl 15.00 och redovisas sedan muntligt vid laborationstillfällena den 26/9 eller den 27/9-18. Tider för muntlig redovisning bokas i Canvas.

Uppgift 1

Du ska skriva ett program i Matlab som utför ett antal uppgifter som finns listade nedan. Alla uppgifter ska göras i samma program. Beräkningarna ska göras **utan** att använda inbygga funktioner som `norm`, `cross`, `dot` men du kan såklart använda funktionerna för att kontrollera att du har gjort rätt.

Programmet ska vara välskrivet med kommentarer som upplyser användaren om vad som görs i varje del.

Alla utskrifter från programmet ska vara tydliga. Som exempel kan en utskrift på skärmen se ut som

```
Vinkeln mellan u och v är 30 grader.
```

För utskrifter på skärmen, använd kommandona `disp` och `num2str`.

a) Ditt program ska börja med att be användaren mata in två vektorer $\vec{u}$ och $\vec{v}$ och två punkter P och Q från skärmen. Alla i $\mathbb{R}^3$. Använd kommandot `input` för inmatning från skärmen. *Se tipset i slutet av uppgiften.*

b) Programmet ska sedan beräkna och skriva ut

1. $\|\vec{u}\|$, $\|\vec{v}\|$ samt $\|\vec{u}+\vec{v}\|$.
2. Vinkeln i grader mellan vektorerna $\vec{u}$ och $\vec{v}$.
3. Skalärprodukten $\vec{u}\cdot\vec{v}$ och kryssprodukterna $\vec{u}\times\vec{v}$ och $\vec{v}\times\vec{u}$.

4. Projektionen av $\vec{u}$ på $\vec{v}$, $\text{proj}_{\vec{v}}\vec{u}$, och projektionen av $\vec{v}$ på $\vec{u}$, $\text{proj}_{\vec{u}}\vec{v}$.

b) Låt vektorn $\vec{v}$ definiera riktningen på en linje L som går genom origo. Beräkna avståndet från punkten P till L, skriv ut avståndet och en varning om avståndet är mindre än d_{min} där d_{min} är ett värde som är angivet av användaren.

c) Låt vektorerna $\vec{u}$ och $\vec{v}$ samt punkten Q definiera ett plan S i $\mathbb{R}^3$.

1. Beräkna och skriv ut planets normal.
2. Beräkna och skriv ut avståndet från punkten P till planet.

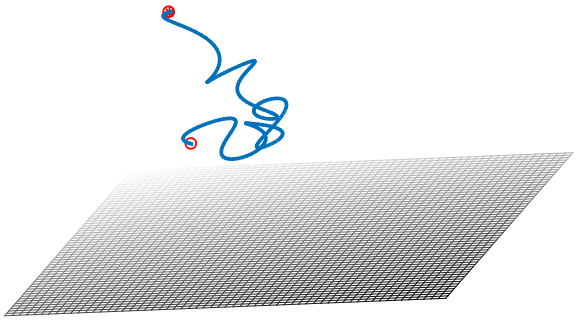
Om användaren väljer $\vec{u}$ och $\vec{v}$ sådana att ett plan inte är väldefinierat ska en varning skrivas ut.

Tips: När du utvecklar programmet kan det vara lämpligt att skriva in vektorer och punkter direkt i programmet istället för att använda `input` och mata in värdena från skärmen. Det blir tidskrävande att testa sitt program om man varje gång man kör programmet ska mata in vektorer och punkter på nytt. När allt fungerar kan man lägga in `input`-satserna.

Uppgift 2

Du har blivit kontaktad av ett datorspelsföretag som behöver hjälp av någon som kan linjär algebra och programmering.

I ett av deras spel rör sig en fisk i havet (i $\mathbb{R}^3$) i en komplicerad bana och ska fångas upp av ett nät (som modelleras med ett oändligt plan), se Figur 1.



Figur 1: Fisken rör sig i en komplicerad bana ovanför nätet. Här kommer nätet inte att fånga in fisken. Notera att denna bana skiljer sig något från banan i uppgiften.

Din uppgift är att skriva ett program som kontrollerar om fisken fångas upp av ett givet nät och i så fall efter hur lång tid det händer. Man vill även att programmet plottar fiskens avstånd till nätet som funktion av tiden fram tills dess att fisken eventuellt fångas upp. Du har fått tre nät som ska kontrolleras.

Fiskens position som funktion av tiden utgör fiskens bana. Den är representerad i form av punkter (koordinater) (x, y, z) uppmätta från tiden $t = 0$ till $t = 10$ med tidsintervallet 0.5 s. Tre vektorer `X`, `Y` och `Z` som innehåller koordinaterna för fisken och en vektor, `T`, med tidpunkterna finns i filen `fiskbana.mat` och finns att ladda ner från Canvas.

Varje plan (som modellerar nätet) är definierat genom två vektorer, $\vec{u}$ och $\vec{v}$, samt en punkt, Q, som alla ligger i det aktuella planet.

Plan 1 ges av $\vec{u} = (0, 4, -1)$, $\vec{v} = (3, -2, 1)$ och $Q = (1, 2, 3)$.

Plan 2 ges av $\vec{u} = (-5, 1, 3)$, $\vec{v} = (-1, 0, 1)$ och $Q = (-10, -1, 1)$.

Plan 3 ges av $\vec{u} = (0, 1, 0)$, $\vec{v} = (-1, 1, 1)$ och $Q = (2, 10, 0)$.

Om fisken inte fångas in innan $t = 10$ så är fisken förlorad.

1. Börja med att ladda in filen med data för fiskens bana. Använd kommandot `load`. Plotta banan och markera startpunkten på banan med en ring och slutpunkten med en stjärna. Använd kommandot `plot3` för att få en 3D-plot.
2. Beräkna sedan avståndet från fiskens position, $P = (x(t), y(t), z(t))$, till nätet. Avståndet ska beräknas för varje tidpunkt från $t = 0$ tills dess att fisken fångas upp av nätet eller tills dess att fisken är förlorad ($t = 10$). Man anser att fisken är fast i nätet om avståndet från fisken till nätet är mindre än 0.5. Spar avstånden i en vektor `distvec`.

   Notera att man inte vet hur länge i tiden avstånden ska beräknas så tänk på vilken typ av slinga som ska använda i programmet.
3. När avstånden har beräknats, plotta avståndet från fisken till nätet som funktion av tiden.
4. I slutet av programmet ska det finnas en utskrift som meddelar om fisken är förlorad eller om fisken har fångats i nätet och i sådant fall vid vilken tidpunkt den fångades upp.

Ovanstående punkter ska göras **för alla tre näten** men utan *koduppprepning*. Det betyder att du skapar ett program som fungerar för ett nät och sedan lägger du en `for`-slinga på lämpligt ställe i programmet så att beräkningarna ovan upprepas för alla tre näten.

Rita nätet(frivilligt)

Om du vill rita ut ett nät (plan) kan du göra det med följande rader MATLAB-kod:

```
[Xp,Yp]=meshgrid(-15:0.5:15,-15:0.5:15);
d = -(a*Q(1)+b*Q(2)+c*Q(3));
Zp=-1/c*(a*Xp+b*Yp+d);
mesh(Xp,Yp,Zp)
```

Kommandot `meshgrid` skapar två matriser med punkter i $x-$ och y-led som går mellan -15 och 15 med steglängden 0.5. z-koordinaterna beräknas sedan utifrån från planets ekvation på formen

$$ax + by + cz + d = 0$$

där (a, b, c) är planets normal och värdet på d bestäms med hjälp av den punkt Q som planet går genom. Notera att detta sätt att lösa ut z ur planets ekvation endast fungerar om $c \neq 0$.