CMSC425 Fall 2019
Homework 1: Geometric exercises
Solutions

Part a. Warm up problems

1. Given the points p1=(-3,4) and p2=(20,15), give the point-vector from of the ray originating at p1 and going through p2. For what value of t is the point (10.8,10.6) on the line?

Point vector form: $p(t) = p1 + t\ (p2 - p1) = p1 + t\ v = (-3,4) + t < 23,11 >$

To solve this you set $p(t) = (-3,4) + t < 23,11 > = (10.8,10.6)$ and solve for t. You can do that in x or y since you have two equations and one unknown.

When $t = 0.6$ we have $p(0.6) = (-3,4) + (0.6) * < 23,11 > = (10.8,10.6)$

2. How far is C=(5,5) from the line through A=(2,5) and B=(4,-1)?

The point vector form is $p(t) = A + t\ (B - A) = (2,5) + t < 2,-6 >$

The perp vector is $v^{\perp} = < 6,2 >$, normalized this is $\hat{v}^{\perp} = \frac{<6,2>}{\sqrt{40}}$
(We normalize because we want a distance).

Distance is $d = v^{\perp} \cdot (C - A) = \frac{<6,2>}{\sqrt{40}} \cdot (3,0) = 2.85$

3. Given the two vectors a=<5,4> and b=<3,4>, give u1 and u2 in the orthogonal projection of a onto b.

We have that $u1 = \frac{a \cdot b}{b \cdot b} b = \frac{<5,4> \cdot <3,4>}{<3,4> \cdot <3,4>} < 3,4 > = \frac{31}{25} < 3,4 >$

And that $u2 = b - u1 = < 3,4 > - \frac{31}{25} < 3,4 > = < -0.72000 , -0.96000 >$

4. Give the angle between the two vectors u=<-1,1,0> and v=<-1,0,1>.

In this case we do have to normalize the two vectors, either before or as part of the calculation.

$\cos(\theta) = \frac{\vec{v} \cdot \vec{u}}{|\vec{v}||\vec{u}|} = \frac{<-1,1,0> \cdot <-1,0,1>}{|<-1,1,0>||<-1,0,1>|} = 0.5000$

$\theta = \mathrm{acos}(0.5000) = 1.0472\ rad$ or 60 degrees

5. Given the three points P1=(1,1,1), P2=(1,2,1), P3=(3,0,4), give a convex combination of the points in the triangle /* any CC is ok */. *** ADDED BUT NOT REQUIRED. What convex combination gives the center?

The question is simple – any coefficients will work here.

Convex combination: ½*P1 + ¼*P2 + ¼ *P3 Any three that add to 1 work.

The center would be: 1/3*P1 + 1/3*P2 + 1/3*P3

6. Starting with the points problem (5), compute the distance of the point P4=(0,0,0) to the plane.

The plan is defined by a normal and a point.

The normal can be computed by the cross product of two vectors. Let
v = P2-P1=<0,1,0> and u=P3-P1=<2,-1,3> then n = u x v = <3,0,-2>
Normalized we have n' = <0.83205,0.00000,-0.55470>

A vector from the plane to the point P4 is w = P4-P1 = <-1,-1,-1>

The distance to the plane is dist = | w • n' | = 0.27735

7. For a vector v=<x,y>, the 2D perp vector $v^{\perp}$ can be defined as vperp=<-y,x>. Will this vector always be 90 degrees counterclockwise from v?

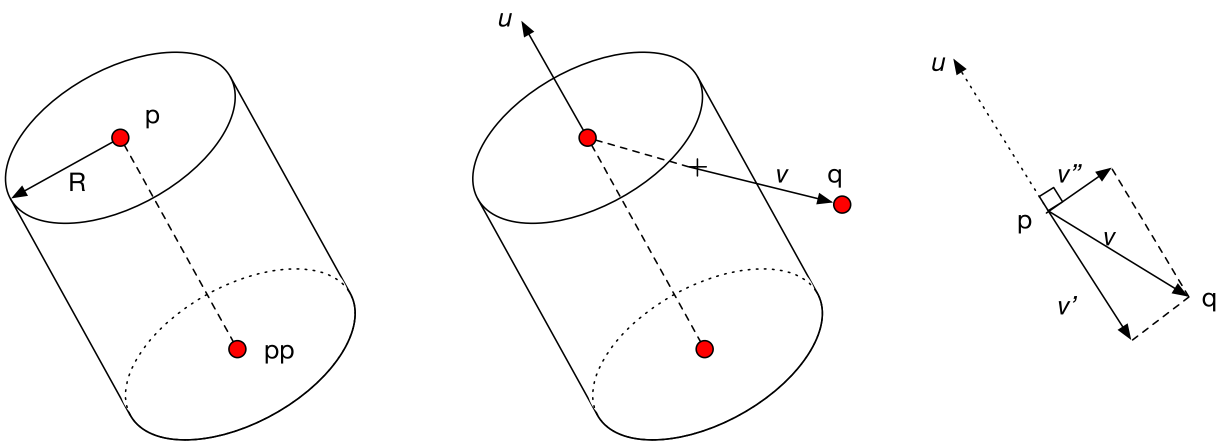
Yes. The easiest argument is by quadrant. Given $v =< x, y >$ and $v^{\perp} =< -y, x >$
Set $v =< 1,1 >$ and $v^{\perp} =< -1,1 >$. Yes.
Set $v =< -1,1 >$ and $v^{\perp} =< -y, x >$. Yes.
Set $v =< -1,-1 >$ and $v^{\perp} =< 1,-1 >$. Yes.
Set $v =< 1,-1 >$ and $v^{\perp} =< 1,1 >$. Yes.
This is not a proof, just a demonstration that quadrants flip as assumed.

A better solution given by students notes that the perp vector is defined by a 90 deg rotation:

$$M * p = \begin{bmatrix} \cos 90 & -\sin 90 \\ \sin 90 & \cos 90 \end{bmatrix} * \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} * \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -y \\ x \end{bmatrix}$$

Part b. Application

1. **Cylinder collider**. Assume you have a cylinder collider defined by 3D points **p** and **pp**, that give the central axis of the cylinder, and R, which is the radius of the cylinder. The points define a unit vector *u* as shown on the middle diagram. **p**, **pp** and R are enough to define the cylinder.



Now assume you have a point $\boldsymbol{q} = (q_x, q_y, q_z)$, somewhere in 3D space (here $\boldsymbol{q}$ is shown outside the cylinder but could be inside.) We want to compute whether $\boldsymbol{q}$ is inside or outside the cylinder as a boolean flag. Show how to do this in mathematical notation (not code).

(a) Given the points P1 and q, show how to compute the coordinates of the vector $\boldsymbol{v} = \langle v_x, v_y, v_z \rangle$ directed from p to q.

Solution: $\boldsymbol{v} = q - p = (q_x, q_y, q_z)\text{-}(p_x, p_y, p_z)$

(b) Given (a), show how to decompose $\boldsymbol{v}$ as the sum of two vectors $\boldsymbol{v}'$ and $\boldsymbol{v}''$ such that $\boldsymbol{v}'$ is parallel to $\boldsymbol{u}$ and $\boldsymbol{v}''$ is perpendicular to u.

Solution:

$\boldsymbol{v}' = \frac{v \bullet u}{u \bullet u} u$ and $\boldsymbol{v}'' = v - v'$

(c) Given your answer to (b), show how to compute the lengths of $\boldsymbol{v}'$ and $\boldsymbol{v}''$

Solution:

$|\boldsymbol{v}'| = \left|\frac{v \bullet u}{u \bullet u} u\right|$ and $|\boldsymbol{v}''| = |v - v'|$

We also need the vector u which we'll define as $\boldsymbol{u} = p - pp$

2. **Cylinder collider in code**. Assuming your answer to (1) above is correct, convert the mathematical equations into a Unity C# method that takes p, pp, R and q, and returns a Boolean true if q is in the cylinder, and false if not.

A point is in the cylinder under two conditions.
a) The point is within a distance R from the line between p and q.
b) The projection of the point onto the line is between the endpoints p and pp.

Pseudocode solution:

1. Compute $v = p - q$
2. Compute v' and v'' as in problem (c)
3. Compute distance d from q to line is the magnitude of v'' or |v-v'|
4. If d > R, reject as not in cylinder
5. Otherwise find projection of q onto line
    - 5a. Let length = distance from p to pp = |p-pp|
    - 5b. Let s = sign of u•v
    - 5c. If s > 0, reject as the projection of q is past p
    - 5d. If s <= 0, then
        - 6a. If $|v'| \leq \mathbf{length}$ then accept since q projects between p and pp
        - 6b. If $|v'| > \mathbf{length}$ reject as q projects on the line past pp.

b) Give a Unity method using Vector3 that implements a solution.

```
int inCylinder(float radius, Vector3 p, Vector3 pp, Vector3 q) {
 // Returns 1 if point in cylinder, 0 otherwise
 int result = 0:
 Vector3 u = p - pp;
 Vector3 v = q - p;
 Vector3 vprime = Vector3.dot(v,u)/Vector3.dot(u,u)*u;
 Vector3 vdblprime = v - vprime;
 float dist = Vector3.magnitude(vdblprime);;
 float length = Vector3.mangitude(u);
 if (dist > R)
    result = 0; // Reject
 else {
        float length = Vector3.magnitude(u);
        float sign   = (Vector3.dot(u,v) > 0): 1 else -1;
        if (sign > 0)
            result = 0; //Reject
        else if (length > Vector3.magnitude(vprime))
            result = 0; //Reject
        else
            result = 1; // Accept
      }
  return result;
  }
```