Distance between point and line

Writing a program to compute the distance between a point and a line (without using a formula for it) illustrates some general principles of computation. Our strategy will be to input the position of the point (c, d) and the parameters m, b determining the equation of the line $y = mx + b$. We will then write a loop which computes several points (x, y) on the line and the distance r of those points to (c, d). We then keep track of which is closest. We do this with a variable we call rmin: whenever we get an r less than rmin we set rmin $= r$. We start rmin our at some huge value.

Here's an analogy for this method: Suppose you want to know the age of the youngest person in a room. You can start out by writing some enormous age (how about 200 years!) on a piece of paper and calling it agemin. Then go around the room asking each person her age. If you find a person with an age less than agemin you replace agemin by that person's age. At the end of the process, you have the youngest age.

The code is on the next page.

You will notice that the code, as written, searches the points (x, y) where $x = -25, -24, -23, \cdots 24, 25$. The question I would like to ask is this: If you wanted to get a more accurate minimum distance, what change would you need to make? The answer is just to ask the code to search a finer mesh of x points by modifying $x = x + 1$ in the for loop to, for example, $x = x + 0.001$.

It is often the case in computational science that a code uses a mesh of points and that by makingthe mesh closer together one gets better answers.

Note: The code also does its search in a restricted range $-25 < x < 25$. One would want to increase that range if the point of closes approach has $|x| > 25$.

```
#include <stdio.h>
#include <math.h>
int main()
{
    double c,d,m,b,rmin,r,x,y;
    rmin=10000.;
    printf("\n Please enter coordinates of point (c,d):  ");
    scanf("%lf %lf",&c,&d);
    printf("\n Please enter parameters of line  m,b:  ");
    scanf("%lf %lf",&m,&b);
    for (x=-25;x<26;x=x+1)
    {
    y = m*x+b;
    r=sqrt( (x-c)*(x-c) + (y-d)*(y-d) );
    if (r<=rmin)
       {
       rmin=r;
       }
    }
    printf("\n The shortest distance is %lf  ",rmin);
    printf("\n The point on the line is %lf %lf \n ",xmin,ymin);
    return 0;
}
```

It is natural to ask if there might be a formula for the distance between a point and a line. There is! There are two ways to derive it. One doesn't require any calculus, so let's do that first.

It should be evident geometrically (draw a picture!) that the shortest distance from a point to a line is obtained by dropping a perpendicular from the point down to the line. The perpendicular has a slope $m' = -1/m$ from that theorem from geometry which states that the product of the slopes of two perpendicular lines is $m\,m' = -1$. So now we have a line which goes through (c, d) whose slope m' we know. We can write down its equation:

$$\begin{aligned} y - d &= m'(x - c) = -\frac{1}{m}(x - c) \\ y &= -\frac{1}{m}(x - c) + d \end{aligned}$$

We can compute the intersection of this new line with the original one $y = mx + b$ since this is just solving two equations in two unknowns. Once that intersection point is known, we just need to compute its distance from (c, d) and we are done.

I leave the algebra details to you.

Those of you who have taken calculus can do the problem another way. We want to minimize the distance between the point and the line. We might as well minimize the square of the distance, since that's the same problem and is simpler because it gets rid of the square roots. Write the square of the distance between the point and the line:

$$d^2 = (x - c)^2 + (y - d)^2$$

If we use the fact that $y = mx + b$ this just becomes a function of x alone:

$$f(x) = d^2 = (x - c)^2 + (mx + b - d)^2$$

We can set the derivative to zero to get the point of closest approach:

$$\begin{aligned} f'(x) &= 2(x - c) + 2m(mx + b - d) = 0 \\ x &= \frac{md - mb + c}{m^2 + 1} \end{aligned}$$

and of course then we get $y = mx + b$ if we want it. We can substitute this value of x into $f(x)$, and take a square root, to get an equation for the smallest distance.