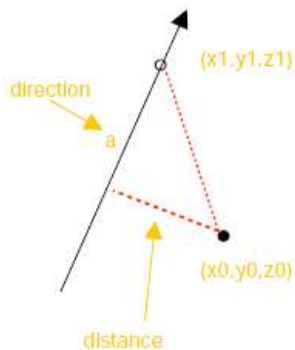


Tutorial Three

1. Distance between a line and a point
 - a. Write the Line class method to determine the distance between a line containing the points (2,1,8) and (3,-1,5) and another point (2,3,2).
 - b. Write the Point class method to determine the distance between the same point and line. Ensure that the Point and the Line methods give an identical result.
2. Rectangle Class: A rectangle object is defined as an area in 2D space consisting of four point objects that determine the corners of the rectangle.
 - a. Open the file **rectangle.h** containing the rectangle class declaration. Given this interface, write the implementation of all the constructor and class methods outlined in **rectangle.cpp**.
 - b. Using the rectangle class determine whether any of the following rectangles intersect:
 - i. Lower left and upper right corners located at (0,0) and (1,1).
 - ii. Lower left point at origin, base 3 units, height 6 units.
 - iii. Lower left point at (1,2), base 3 units, height 6 units.
3. Vector class:
 - a. Open the file **vect.cpp** and complete each of the member functions outlined in the interface (**vect.h**).
 - b. Test the class by writing a program declaring four vectors a, b, c and d. Define a to be a vector with direction (-3,2,4) and b to be a vector through two points (2,1,3) and (6,3,5). Define vector c as a copy of a and assign b to the vector d.
 - c. Ensure that your program calculates the following correctly:
 $a \cdot b = 0$, $a \times b = (-4, 22, -14)$, $a + b = (1, 4, 6)$ and $a - b = (-7, 0, 2)$.
 - d. Ensure that the copy constructor and assignment operator functions were written correctly by confirming that $a \cdot b = c \cdot d$ and $a \times b = c \times d$.
4. Correlation between kinematic variables.
 - a. Write a program which uses the sample class to read the values of the kinematic variable P_1 for 500 simulated detector events (contained in the file P1.cand) and order the sample from the highest to lowest value.
 - b. Write a sample class method to determine the correlation between any two

- kinematic variables. Confirm that the correlation between P_1 and P_2 is 0.43.
- c. Read all of the event values from the eight .cand files into your program. Display the complete correlation matrix between all of the eight kinematic variables.

Hints for Question 1:



- The distance between a point and a line is given by:

$$d = \frac{\| \langle x_1 - x_0, y_1 - y_0, z_1 - z_0 \rangle \times a \|}{\| a \|}$$

- The function prototypes have been set up for you in the supplied **point** and **line** files.

Hints for question 2:

- Start with the constructor methods. The key to writing these methods lies in being able to call the relevant **point** constructors with the correct values.
- Remember the **point** declarations in the private section do not call the correct constructor methods for **point**. They still have to be explicitly declared in the rectangle's constructor methods.
- A **rectangle** is defined in 2D only so always set one of the point coordinates to zero e.g $z=0$.
- The width and height member functions should be just a simple mathematical expression between the member data of the rectangle class. But remember that a point object is not a value. You will have to call accessor methods of the point objects within these functions.

Hints for Question 3:

- The vector member functions **dotProject**, **crossProduct** and **getAngle** will have exactly the same functionality as the code written in tutorial 2. Only the syntax needs to be changed.
- Try to call the private member function **getMagnitude** from **getAngle**.
- The implementation of the empty vector constructor is not important here, just assign **x,y** and **z** with unit values.
- Overload the + and – operators to perform component-wise addition and subtraction.
- The cross product function does not have to be written again when writing the implementation for the overloaded * operator. Just call the cross product function with the relevant arguments.

Hints for Question 4:

- The question uses kinematic variables of 500 simulated events in the DELPHI detector. You do not need to know anything about these variables (or anything about Particle Physics!) to complete this question.
- In the **correlation** class method call the mean and standard deviation member functions already in the class. However, for simplicity do not try and write a separate function to calculate the mean of **xy**.
- The implementation of the correlation matrix can be simplified by placing the sample objects into a sample array. This is no different from placing integers into an integer array. The correlation class method can then be called by iterating over the size of this sample object array.