```
> Alexander Sage
> Project 1 : least squares fitting
```

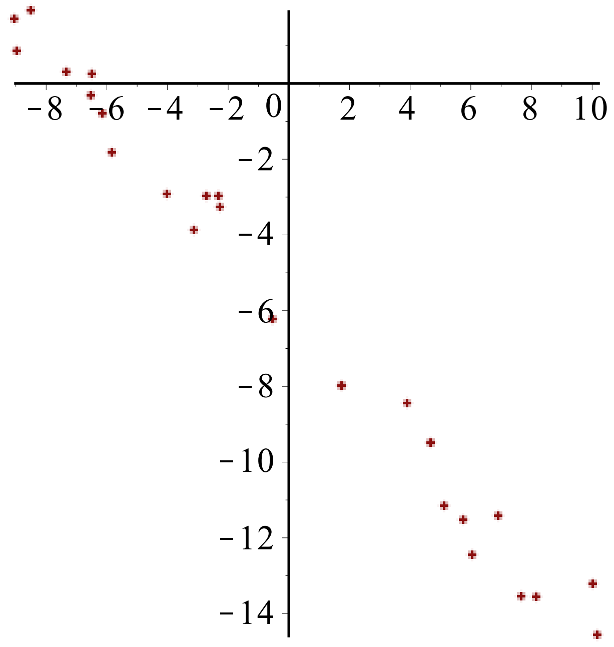
Goal: To find a better way to create a line that describes a set of data points. The process of finding the least squares of a set of data points works well, but it only minimizes the verticle distance between the y-value of the point and the line. Intuitively minimizing the x-value wouldn't produce a significantly better result. Taking into account both the horizontal and verticle distances the best method would be to minimize the euclidean distance. Both methods of minimizing the verticle and horizontal distances (least squares) are done first to truly understand all the mathematical processess that take place when a least squares function is called on a data set, and to use the process as a basis for creating a better fit line.

```
> with(plots) :
```

The data evaluated is listed in the subjection below

and that data is graphed here.

```
> plot(data, style = point)
```



```
> standard fit
                                   standard fit                              (1)
```

The first process below is a typical least squares function extended out to show all the steps taken to fit a line to the data by minimizing the the verticle distance from the line to each individual point

```
> lsqy := proc(data, x)
    local g, m, b, sol;
    g := sum( ( m·data[i][1] + b - data[i][2])^2      , i = 1 ..nops(data));
    sol := solve( {diff(g, m) = 0, diff(g, b) = 0}, {m, b});
    subs(sol, m·x + b);
  end:
```

The subsection below breaks down the short code above with an explicite description of each line

```
> lsqy := proc(data, x)
```

define's the calling name of the function

```
> local g, m, b, sol;
```

defining the names g, m, b, and sol to only have meaning within this program (line of maple)

```
> g := sum( ( m·data[i][1] + b - data[i][2])^2      , i = 1 ..nops(data));
```

g is the sum of distance between the x value of a point evaluated in a typical formula y=m*x+b subtracted by the y value of the same point. This process is looped over all the data points. The result is an equation of the form (m*x1+b-y1)^2+(m*x2+b-y2)^2+...

```
>  sol := solve( {diff(g, m) = 0, diff(g, b) = 0}, {m, b});
```

sol solves two differential equations (one which is the partial derivative of the resultant equation above set equal to 0 with respect to m and the other is the partial derivative of the same equation above set equal to 0 with respect to b) for the values m (slope) and b (y-intercept); which will be the critical value of these derivatives.

```
> subs(sol, m·x + b);
```

subs substitues the values found in the previous step for m and b into an equation of the form m*x+b

```
> end:
```

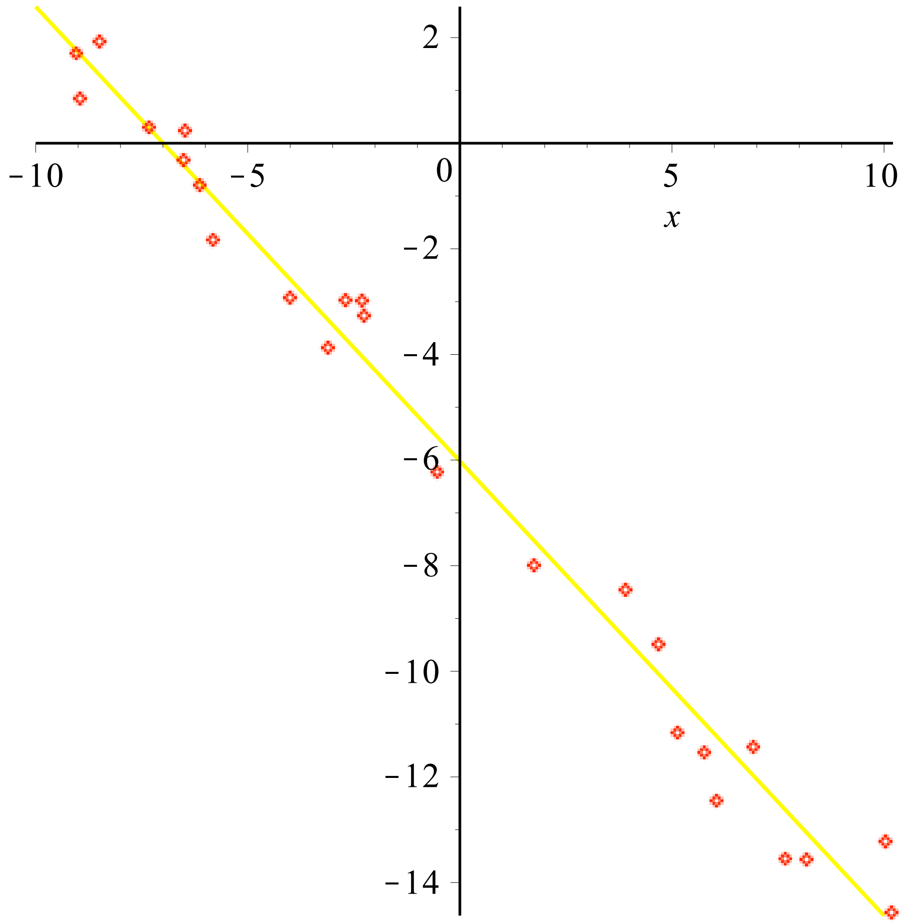
ends the program and stops maple from printing anything to the screen

This is the equation of the line and the graph of the line found by using the least squares function above to minimize the verticle

```
> lsqy(data, x)
```

$$-0.8606108766\,x - 6.020588351 \tag{2}$$

```
> display([plot(lsqy(data, x), color = yellow), plot(data, style = point, color = red)]);
```



> *minimizing the horizontal*

minimizing the horizontal **(3)**

The process below is almost the same, but the base equation is rearranged to minize the horizontal.

> $lsqx := \mathbf{proc}(data, x)$
    local l, m, b, sol;
    $l := sum\left(\left(\frac{(data[i][2]-b)}{m} - data[i][1]\right)^2, i = 1..nops(data)\right)$;
    $sol := solve(\{diff(l, m) = 0, diff(l, b) = 0\}, \{m, b\})$;
    $subs(sol, m \cdot x + b)$;
  end:

The subsection below breaks down the short code above with an explicite description of each line

> $lsqx := \mathbf{proc}(data, x)$

define's the calling name of the function

> **local** l, m, b, sol;

defining the names l, m, b and sol to only have meaning within this program (line of maple)

> $l := sum\left(\left(\frac{(data[i][2]-b)}{m} - data[i][1]\right)^2, i = 1..nops(data)\right)$;

l is the sum of distance between the y value of each point substituted into the equation $\frac{(y-b)}{m} = x$, then subtracted by the x value of the point. This process is looped over all the data points. The result is an equation of the form ((y1-b)/m-x1)^2+((y2-b)/m-x2)^2+...

> $sol := solve(\{diff(g, m) = 0, diff(g, b) = 0\}, \{m, b\})$;

sol solves two differential equations (one which is the partial derivative of the resultant equation above set equal to 0 with respect to m and the other is the partial derivative of the same equation above set equal to 0 with respect to b) for the values m (slope) and b (y-intercept); which will be the critical value of these derivatives.

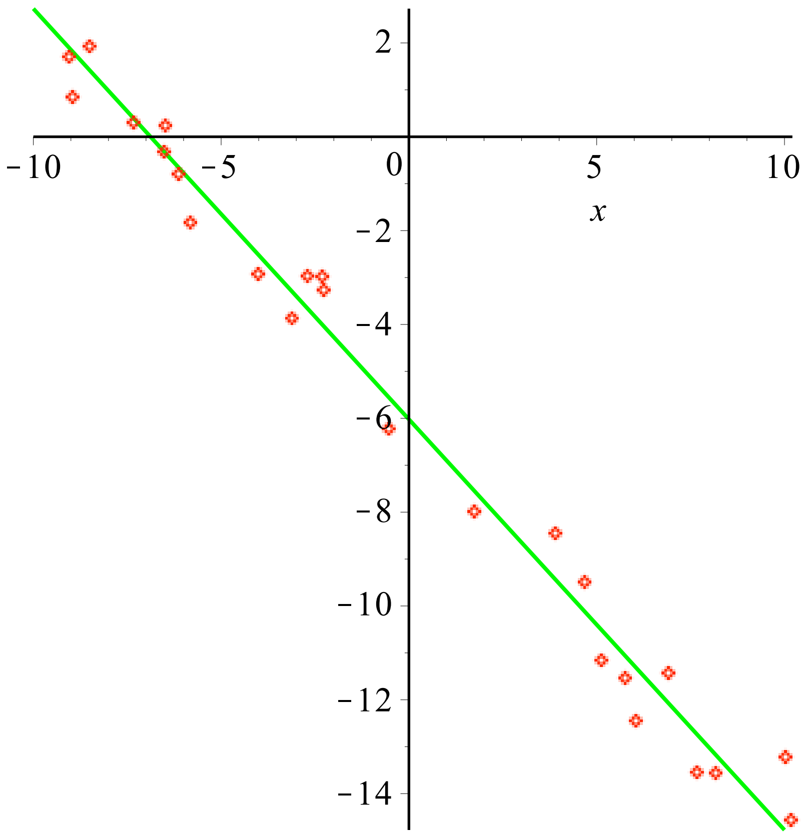
> $subs(sol, m \cdot x + b)$;

subs substitues the values found above for m, and b into an equation of the form m*x+b

> **end**:

ends the program and stops maple from printing anything to the screen

```
> lsqx(data, x)
                         -0.8752620570 x - 6.022148339                         (4)
> display({plot(lsqx(data, x), color = green), plot(data, style = point, color = red)})
```



> *Euclidean*

$$Euclidean \qquad (5)$$

The next process is to derive a completely new formula to minimize. The formula must be the euclidean distance between each point and the line, and must be in terms of the constants (slope, y-intercept) of the line.

start with a formula for the slope of any general line

> $y = \frac{(a \cdot x + c)}{b}$

$$y = \frac{a\,x + c}{b} \qquad (6)$$

this gives an expression for the slope $\frac{a}{b}$=m and the y-intercept $\frac{c}{b}$=t. another line can be drawn perpendicular to this line that also passes through a single point.

the slope of the new line is represented in terms of the first line as the opposite reciprocal of *m*

> $y2 := -\frac{1}{m} \cdot x2 + d$

$$y2 := -\frac{x2}{m} + d \qquad (7)$$

y2 represents every line that passes perpendicular through the best fit line and includes any one of the points. To apply y2 to a specific point the y-intercept d would simply be changed to a different value.

vectors will be used to represent each part

y becomes $v = 1 \cdot \langle 0, t \rangle + u \cdot \langle 1, m \rangle$

the data point would become $w = 1 \cdot \langle x_0, y_0 - t \rangle$

y2 becomes $r = 1 \cdot \langle 0, t \rangle + u \cdot \langle -m, 1 \rangle$

if we project $w \cdot r = \frac{|w \cdot r|}{|r|}$ we get the perpendicular distance between the point and the line y

> $\frac{(-m \cdot x_0 + y_0 - t)}{\sqrt{1 + m^2}}$

$$\frac{-m\,x_0 + y_0 - t}{\sqrt{1 + m^2}} \qquad (8)$$

we square both sides and get

> $d^2 = \frac{(-m \cdot x_0 + y_0 - t)^2}{1^2 + m^2}$

$$d^2 = \frac{(-m\,x_0 + y_0 - t)^2}{1 + m^2} \qquad (9)$$

we use a very simalar method as for lsqx and lsqy to finish the problem, plug in all points in 'data', sum up the equations and expand the equation

```
> dis := sum( (-t + -m·data[i][1] + data[i][2])^2 / (1^2 + m^2), i = 1 ..nops(data) ) :
```

```
> ex := expand(dis)
```

$$ex := \frac{1632.958895}{1+m^2} + \frac{25\,t^2}{1+m^2} + \frac{296.4477186\,t}{1+m^2} + \frac{1001.460112\,m^2}{1+m^2} + \frac{1691.682671\,m}{1+m^2} - \frac{5.32377531\,t\,m}{1+m^2} \qquad \textbf{(10)}$$

Then we take the derivative of the equation (ex) with respect to m and t, and solve for the respective critical points

```
> sol := solve( {diff(ex, m) = 0, diff(ex, t) = 0}, {m, t})
```

$$sol := \{m = 1.153574231, t = -5.806126972\}, \{m = -0.8668709591, t = -6.021254896\} \qquad \textbf{(11)}$$

Sometimes we will get three results, the third set will be close to $\{m = -8.092223063\ 10^{12}, t = -8.616235454\ 10^{11}\}$ and is clearly of the wrong magnitude so it is ignored. The other two are the best fit line desired, and a perpendicular line to that one. This happens because the equation above was squared so there will be multiple roots even though the process was created for just one. The other two results are found and represented as their associated functions below.

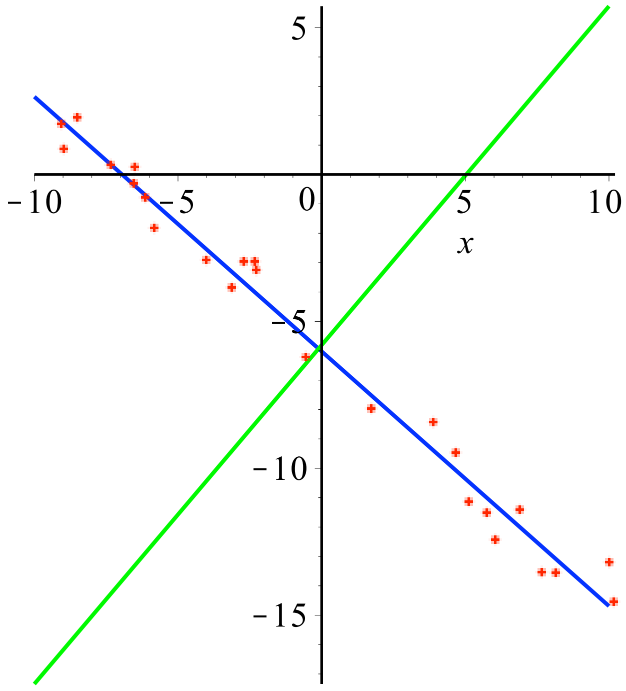
```
> P1 := subs(sol[1], m·x + t)
```

$$P1 := 1.153574231\,x - 5.806126972 \qquad \textbf{(12)}$$

```
> P2 := subs(sol[2], m·x + t)
```

$$P2 := -0.8668709591\,x - 6.021254896 \qquad \textbf{(13)}$$

```
> display({plot(P1, color = green), plot(P2, color = blue), plot(data, style = point, color
        = red)})
```



It is clear that the best fit line, in blue, is the one that is desired.

Below is all the steps used to calculate the best fit line found by minimizing the euclidean distance between each point and the line. The lines of code are essentially the same as the ones used for lsqx and lsqy, with the exception of sel; which has maple look at the values found for slope, and picks the set of m and b with a negative slope (m).

```
> mineuclid := proc(data, x)
    local dis, ex, sel, m, b, sol;
    dis := sum( (-t + -m·data[i][1] + data[i][2])^2 / (1^2 + m^2), i = 1 ..nops(data) ) :
   ex := expand(dis) :
    sol := solve( {diff(ex, m) = 0, diff(ex, t) = 0}, {m, t} );
    sel := select(x→(subs(x, m) < 0), [sol]);
   subs(op(sel), m·x + t)
  end:
```

$$\mathit{dis} := \mathit{sum}\left(\frac{(-t + -m\cdot \mathit{data}[i][1] + \mathit{data}[i][2])^2}{1^2 + m^2}, i = 1..\mathit{nops}(\mathit{data})\right)$$

```
> mineuclid(data, x)
```

$$-0.8668709596\, x - 6.021254892 \qquad \textbf{(14)}$$

After trying all three methods the three equations found were.

```
> lsqy(data, x)
```

$$-0.8606108766\,x - 6.020588351 \qquad \textbf{(15)}$$

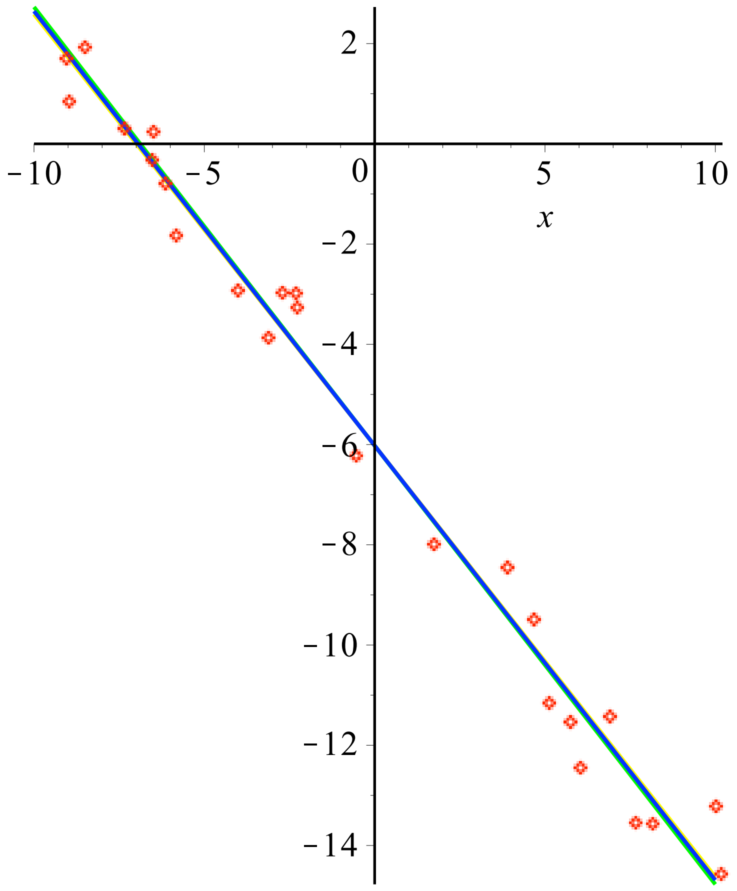
```
> lsqx(data, x)
```

$$-0.8752620570\,x - 6.022148339 \qquad \textbf{(16)}$$

```
> mineuclid(data, x)
```

$$-0.8668709595\,x - 6.021254891 \qquad \textbf{(17)}$$

```
> display({plot(P2, color = blue), plot(lsqx(data, x), color = green), plot(lsqy(data, x), color
      = yellow), plot(data, style = point, color = red)})
```



The differences between lsqx, lsqy and *mineuclid* are very small (as seen on the graph), but the best fit line to the curve (euclidian) would be used in replace of the standard least square method when very high accuracy is needed.