22M:033

J. Simon

Approximate Solution of an Over-Determined Linear System

i.e.

Method of Least Squares

Suppose we have a system of 5 equations in 2 unknowns. Your experience tells you that, in general, there is no solution.

But **how close can we come to a solution**?

This is not just an "academic" question. Here is one reason to care about this question.

Example:

You are collecting data about the relative behavior of two variables, x and y. You observe the following data points.

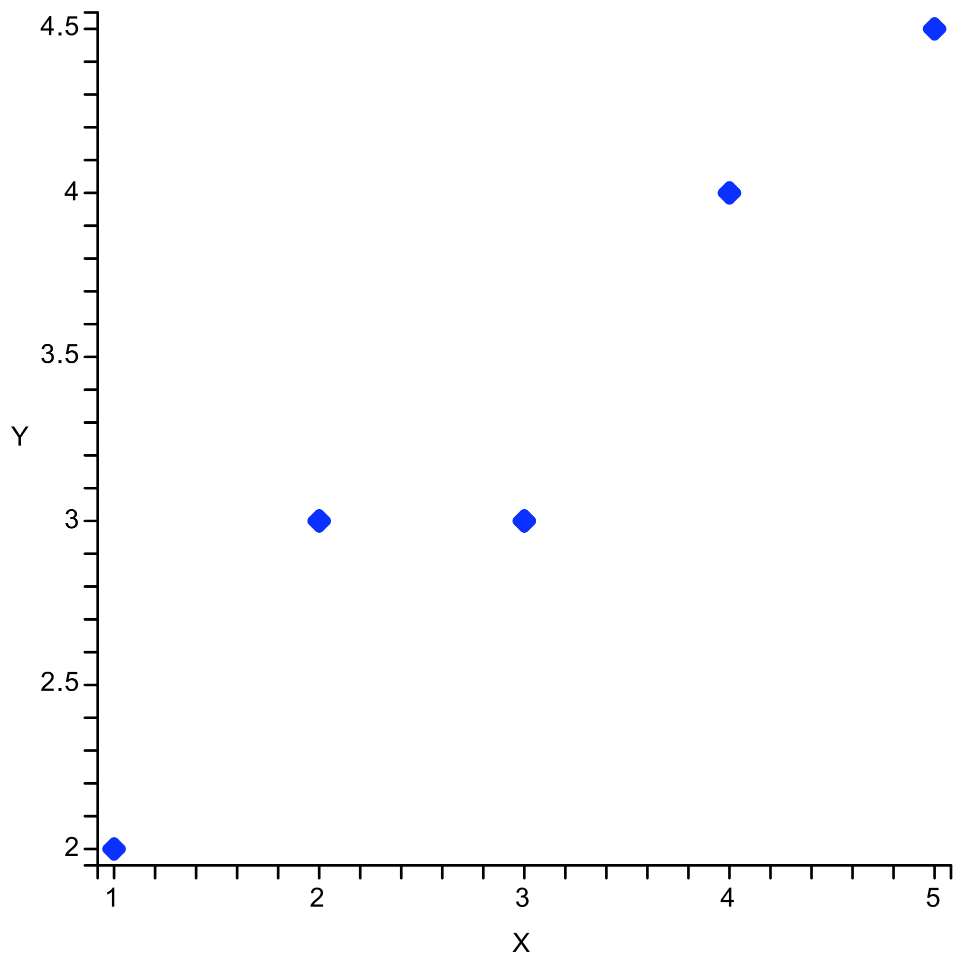
(x,y) = {(1,2), (2,3), (3,3), (4,4), (5,4.5)}.

```
>
> with(linalg):
> with(plots):
> DataList:=[[1,2],[2,3],[3,3],[4,4],[5,4.5]];
```

$$DataList := [[1, 2], [2, 3], [3, 3], [4, 4], [5, 4.5]]$$

```
> PlotThePoints:=plot(DataList, style=point, color=blue, thickness=4,
  labels=[X,Y]):

> display({PlotThePoints});
```



Suppose you believe (either because of some deeper understanding you have of how x and y are connected, or just because you look at the plot of data points and make a geometrically inspired guess) that the quantities x and y actually are related linearly (e.g. y=mx+b for some m and b) and that the fact that these observed points do not lie on a straight line is just an artifact: the variables actually do lie on a line, but there is experimental or observational error in the data recorded. Can we find a "best line" that gives (in some sense) the best estimate of a linear relation between x and y.

For example, here are three lines: y=x+1, y=(0.9)x + 0.9, y=(0.8 x) + (0.7). Which is a better fit to the data? What would be the "best" line?

```
>
> LineEquation1:=x+1;
```

$$LineEquation1 := x + 1$$

```
> LineEquation2:=0.9*x+ 0.9;
```

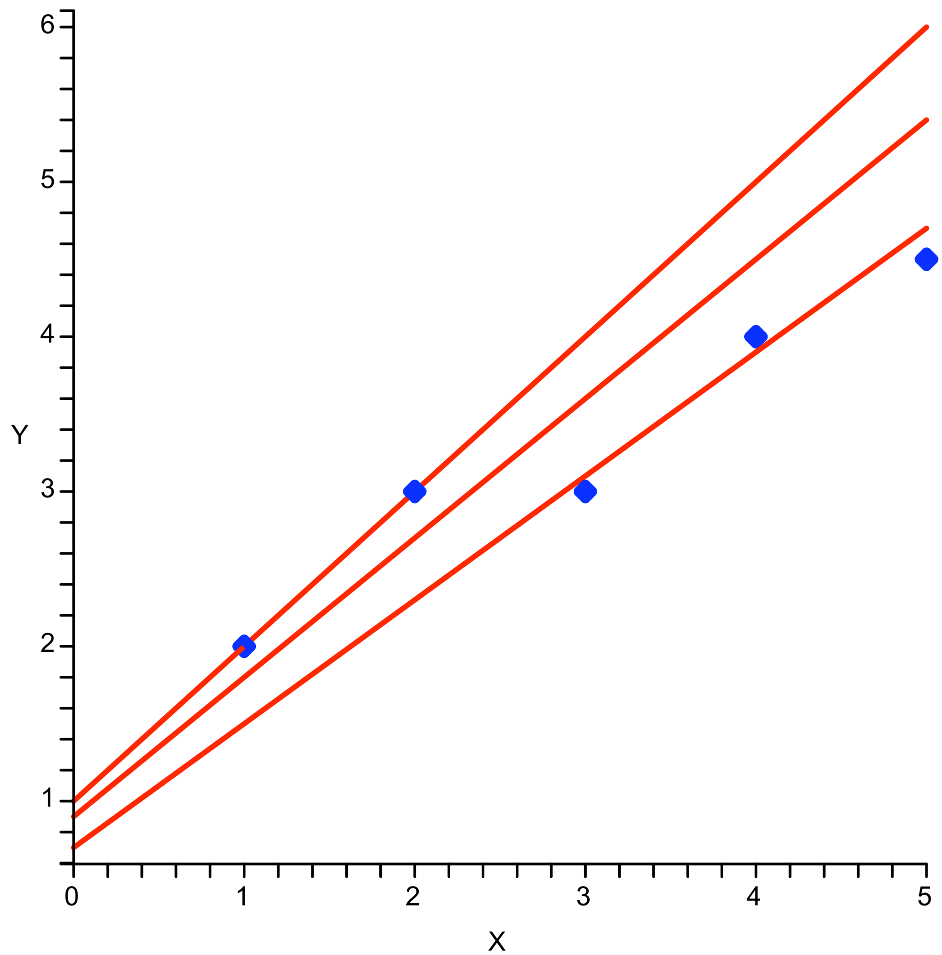
$$LineEquation2 := 0.9\,x + 0.9$$

```
> LineEquation3:=0.8*x+0.7;
```

$$LineEquation3 := 0.8\,x + 0.7$$

```
> PlotTheLines:=plot({LineEquation1, LineEquation2, LineEquation3}, x=0..5,
  color=red, thickness=2):
```

```
> display({PlotThePoints,PlotTheLines});
```



Let's approach this as a linear algebra problem. We seek numbers m,b such that (if it were possible, which it is not) each of the 5 points lies on the line y=mx+b. That means we wish (m,b) was a solution of the linear system (of 5 equations in 2 unknowns):

2=m*1+b

3=m*2+b

3=m*3+b

4=m*4+b

4.5 = m*5+b

Write this system in matrix form:

```
> M:=matrix([[1,1],[2,1],[3,1],[4,1],[5,1]]);
```

$$M := \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \\ 4 & 1 \\ 5 & 1 \end{bmatrix}$$

```
> Y:=matrix([[2],[3],[3],[4],[4.5]]);
```

$$Y := \begin{bmatrix} 2 \\ 3 \\ 3 \\ 4 \\ 4.5 \end{bmatrix}$$

```
> mbMatrix:=matrix([[m],[b]]);
```

$$mbMatrix := \begin{bmatrix} m \\ b \end{bmatrix}$$

```
> LinearSystem:=evalm(M)*evalm(mbMatrix)=evalm(Y);
```

$$LinearSystem := \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \\ 4 & 1 \\ 5 & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ 3 \\ 4 \\ 4.5 \end{bmatrix}$$

There is no point in trying to solve this system in the usual way: adjoin the column of Y values to the coefficient matrix and then row-reduce the augmented coefficient matrix. All you'll get is that the system is inconsistent.

But just to be thorough...

```
> A:=concat(M,Y);
```

$$A := \begin{bmatrix} 1 & 1 & 2 \\ 2 & 1 & 3 \\ 3 & 1 & 3 \\ 4 & 1 & 4 \\ 5 & 1 & 4.5 \end{bmatrix}$$

```
> rref(A);
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

The third row says the system is inconsistent (i.e. there is no straight line that passes through all five points).

Now let's do some magic.

IF we had solutions to the system M*[unknown vector of m,b] = Y vector,

then the [m,b] vector would still satisfy any new system we get by multiplying both sides of this matrix equation by some other matrix. That is

M*[column m,b] = Y ==> W*M*[column m,b] = W*Y.

In particular, do this with M=transpose(M).

```
> W:=transpose(M);
```

$$W := \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

```
> NewCoeffMatrix:=evalm(W&*M);
```

$$NewCoeffMatrix := \begin{bmatrix} 55 & 15 \\ 15 & 5 \end{bmatrix}$$

```
> NewYVector:=evalm(W&*Y);
```

$$NewYVector := \begin{bmatrix} 55.5 \\ 16.5 \end{bmatrix}$$

```
> NewLinearSystem:=evalm(NewCoeffMatrix)*evalm(mbMatrix)=evalm(NewYVector);
```

$$NewLinearSystem := \begin{bmatrix} 55 & 15 \\ 15 & 5 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} = \begin{bmatrix} 55.5 \\ 16.5 \end{bmatrix}$$

THIS system we can solve.

```
> NewAugmentedCoeffMatrix:=concat(NewCoeffMatrix, NewYVector);
```

$$NewAugmentedCoeffMatrix := \begin{bmatrix} 55 & 15 & 55.5 \\ 15 & 5 & 16.5 \end{bmatrix}$$

```
> rref(NewAugmentedCoeffMatrix);
```

$$\begin{bmatrix} 1 & 0 & 0.6000000016 \\ 0 & 1 & 1.499999994 \end{bmatrix}$$

Assuming the decimal bits are due to computer arithmetic, we have

m = 0.6 and b = 1.5 .

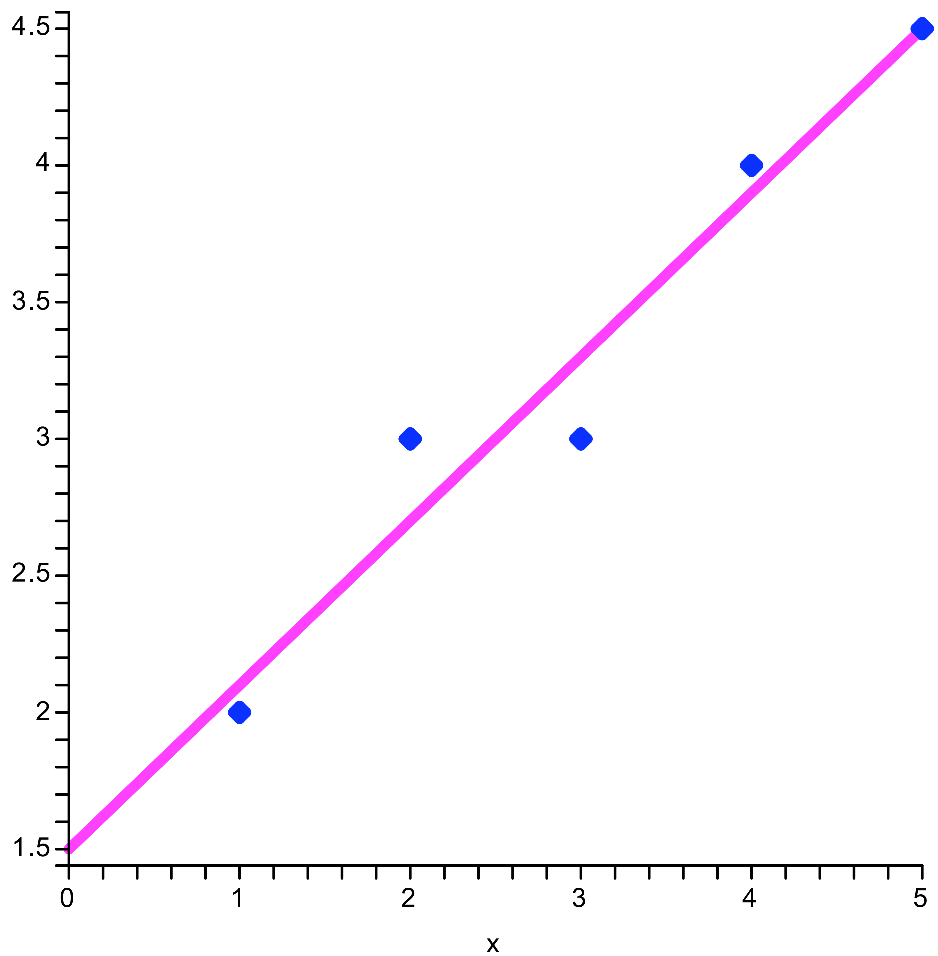
Let's see what that line looks like relative to the data points.

```
> LeastSquaresLine:=0.6*x+1.5;
```

$$LeastSquaresLine := 0.6\, x + 1.5$$

```
> PlotLeastSquaresLine:=plot(LeastSquaresLine, x=0..5, color=magenta,
  thickness=4):
```

```
> display({PlotLeastSquaresLine, PlotThePoints});
```



>

When you have an overdetermined linear system, multiplying both sides by the transpose of the coefficient matrix gives a system whose solution will be an *approximate* solution to the original problem. In fact, it will give you the "best possible" approximate solution in the sense that a measure of error will be smallest.

####################

The line we just found is the same line as you get using the "method of least squares" (as perhaps taught in multivariable calculus class). We consider hypothetical (m,b), and compare the predicted y-value (i.e. mx+b) to the observed y-value at each of the observation points. For each pair of observed vs. predicted data points, we measure the error by taking the square of the difference between the predicted value (mx+b) and the observed value (y). Then add up these squared errors to get a measure of the total error for a given choice of (m,b). This error function E(m,b) is a just a quadratic polynomial in m and b, so it is easy to use partial derivatives (dE/dm = 0, dE/db = 0) to find which (m,b) gives the smallest error. If you do this with the above example of 5 data points, you'll get the same NewLinearSystem!!

```
> Xlist:=convert(col(M,1),list);
```

$$Xlist := [1, 2, 3, 4, 5]$$

```
> Ylist:=convert(col(Y,1),list);
```

$$Ylist := [2, 3, 3, 4, 4.5]$$

```
> PredictedY:=[m*Xlist[1]+b, m*Xlist[2]+b, m*Xlist[3]+b, m*Xlist[4]+b,
  m*Xlist[5]+b];
```

$$PredictedY := [m + b, 2\, m + b, 3\, m + b, 4\, m + b, 5\, m + b]$$

```
> ErrorFunction:=(m,b)-> sum((PredictedY[i]-Ylist[i])^2, i=1..5);
```

$$ErrorFunction := (m, b) \to \sum_{i=1}^{5} (PredictedY_i - Ylist_i)^2$$

```
> E:=ErrorFunction(m,b);
```

$$E := (m + b - 2)^2 + (2\, m + b - 3)^2 + (3\, m + b - 3)^2 + (4\, m + b - 4)^2 + (5\, m + b - 4.5)^2$$

```
> Eq1:=diff(E,m)=0;
```

$$Eq1 := 110\, m + 30\, b - 111.0 = 0$$

```
> Eq2:=diff(E,b)=0;
```

$$Eq2 := 30\, m + 10\, b - 33.0 = 0$$

```
> LeastSquaresAugmentedCoefficientMatrix:=matrix([[110, 30, 111],[30, 10,
  33]]);
```

$$LeastSquaresAugmentedCoefficientMatrix := \begin{bmatrix} 110 & 30 & 111 \\ 30 & 10 & 33 \end{bmatrix}$$

Compare this to the augmented coefficient matrix we got from the "unsolvable" over-determined system by multiplying both sides by transpose(M):

```
> evalm(NewAugmentedCoeffMatrix);
```

$$\begin{bmatrix} 55 & 15 & 55.5 \\ 15 & 5 & 16.5 \end{bmatrix}$$

```
>
```

#################

end of handout

##################