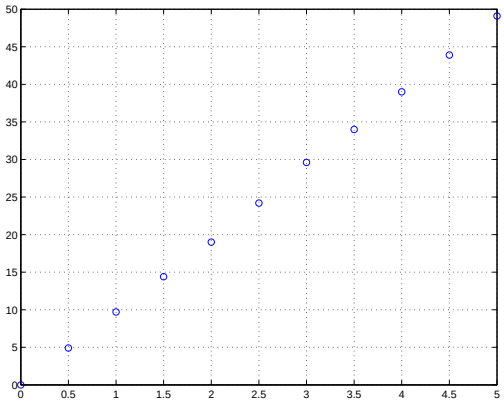
MATH 51: MATLAB HOMEWORK 3

Experimental data generally suffers from imprecision, though frequently one can predict how data 'should' behave by graphing results collected from experiments. For instance, suppose an experimenter observing a falling anvil gathers the following data (measured in seconds and meters per second, as appropriate):

time t	velocity v_t	time t	velocity v_t
0.0	0.0	3.0	29.6
0.5	4.9	3.5	34.0
1.0	9.7	4.0	39.0
1.5	14.4	4.5	43.9
2.0	19.0	5.0	49.1
2.5	24.2		

One can verify quickly that these points don't lie on a line, but when we plot them we see that they nearly do:



Such observations are the critical step which allow experimental data to develop into theories which predict future behavior.

A natural question, then, is to find the line which best fits the given data set. This problem set will deal with solving problems such as this. The technology we will use to solve these problems is multivariable calculus and basic linear algebra. The first section will discuss one of the most common types of 'data fitting.' In the second section we will use MATLAB to fit data points to higher degree polynomials. In the final section, we will experiment with modifications to the 'least squares' approach to curve fitting.

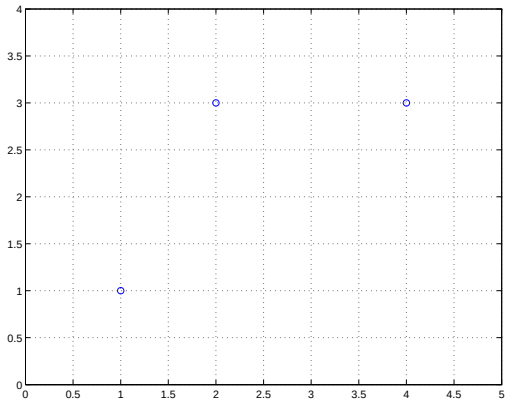
1. Fitting Lines with Least Squares

We first attempt to find the best linear approximation to a collection of points. More specifically, given a collection of points $(x_1, y_1), \cdots, (x_n, y_n)$ in the plane (hereafter called our data set), we seek the line $y = mx + b$ which is 'closest' to our collection. Of course, our answer depends on how we interpret the word 'closest', and there are plenty of viable interpretations. One common interpretation is to say that a line $y = mx + b$ is 'closest' to the data set $(x_1, y_1), \cdots, (x_n, y_n)$ provided the quantity

$$\sum_{i=1}^{n} (y_i - (mx_i + b))^2$$

is minimal. Notice that the term $y_i - (mx_i + b)$ measures – up to sign – the distance between the *actual* value of the the y-variable corresponding to x_i and the *predicted* value of the y-variable corresponding to x_i. Hence each summand measures the square of this distance, and so we are minimizing the sum of these squares. Perhaps not suprisingly, therefore, this technique is called *least squares approximation*. In the rest of this section we'll solve the least squares problem for a given collection of data points using multivariable calculus and MATLAB.

Example. Suppose we are given the points $(1, 1), (2, 3)$ and $(4, 3)$ and asked to find the line which is a best approximation to this data set using the Least Squares technique.



From our discussion above, this is the same as finding values of m and b so that

$$f(m, b) := (1 - (m \cdot 1 + b))^2 + (3 - (m \cdot 2 + b))^2 + (3 - (m \cdot 4 + b))^2$$

is minimized. Since we'll be using MATLAB to minimize $f(m, b)$, we first enter this polynomial into MATLAB. After declaring m and b to be variables using the `syms` command, we define f as the desired polynomial.

```
>> syms m  b;
>> f=(1-(m + b))^2 + (3-(2*m + b))^2 + (3-(4*m + b))^2

f =

(1-m-b)^2+(3-2*m-b)^2+(3-4*m-b)^2
```

Note: If you don't first declare m and b as variables, MATLAB will have a fit.

So we are left with finding values of m and b that minimize the polynomial $f(m,b)$ above. Of course, we use multivariable calculus to solve this problem. Indeed, the minimal value for this polynomial is given by solving $\frac{\partial f}{\partial m} = \frac{\partial f}{\partial b} = 0$, which we can (again!) use MATLAB to compute[1]

First we compute the partial derivatives using the `diff` function in MATLAB. The command `diff(f,m)` computes the partial derivative of f with respect to m, and similarly `diff(f,b)` computes the partial derivative of f with respect to b.

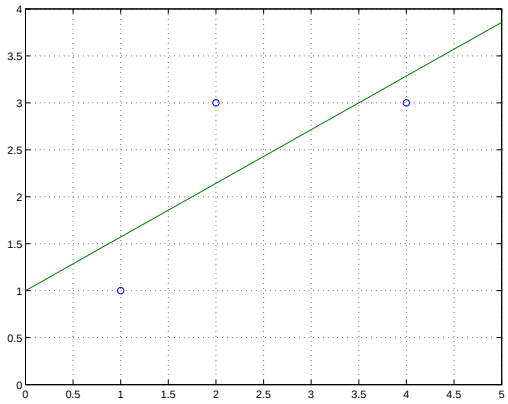
```
>> diff(f,m)
ans =

-38+42*m+14*b
>> diff(f,b)
ans =

-14+14*m+6*b
```

Hence our minimizing values will be solutions to the system $-38+42m+14b = -14+14m+6b = 0$. This is equivalent to solving the matrix equation

$$\begin{pmatrix} 42 & 14 \\ 14 & 6 \end{pmatrix} \begin{pmatrix} m \\ b \end{pmatrix} = \begin{pmatrix} 38 \\ 14 \end{pmatrix}.$$

```
>> A=[42, 14; 14, 6]; B=[38; 14];
>> A^-1*B
ans =
    0.5714
    1.0000
```

Hence the line which solves the least squares problem for the points $(1,1),(2,3)$ and $(4,5)$ is given (approximately) by $y = .5714x + 1$.



[1]Values where $\frac{\partial f}{\partial m} = \frac{\partial f}{\partial b} = 0$ need not be local minima of the function, just as a critical point is not necessarily a minimum of a function of one variable. One should really check that the critical values we compute produce bona fide minima of the function, but we won't here.

Exercises

(1) Use the least squares technique to find the line which best fits the following collection of points

$$(-3,-3),(-2,-3.5),(-1,-2.5),(0,-0.5),(1,-1.5),(2,-0.5),(3,-1),(4,-1.5).$$

(2) One doesn't have to limit the least squares technique to finding lines which fit data sets in the plane. Indeed, if $(x_1,y_1,z_1),\cdots,(x_n,y_n,z_n)$ are points in space, one can try to find the plane $z=c-ax-by$ which best fits the given data set[2]. This is done by finding values of a,b and c which minimize the quantity

$$\sum_{i=1}^{n}\left(z_i-(c-ax_i-by_i)\right)^2.$$

Using techniques analogous to those developed above, find the plane which best fits the data set

$$(-1,0,1),(2,3,4),(5,6,7),(8,10,11),(10,12,15).$$

[2]For the pedants: we're excluding planes which are 'vertical' by considering only those of the form $z=c-ax-by$, just as when using least squares to fit lines we exclude veritical lines by assuming a line is of the form $y=mx+b$.

2. Fitting in Practice

Although fitting lines to data sets can be very useful, in practice we often find that data sets are not well-approximated by straight lines. More often, a data set can be much better approximated by a polynomial of higher degree, such as a parabola or cubic equation. Although fitting higher degree polynomials is straightforward generalization of fitting lines, it can present a computational burden to manually fit polynomials of higher and higher degree. Fortunately for us weary mathematicians, MATLAB can greatly simplify our calculations. We'll learn how to use MATLAB to do this by working through an example.

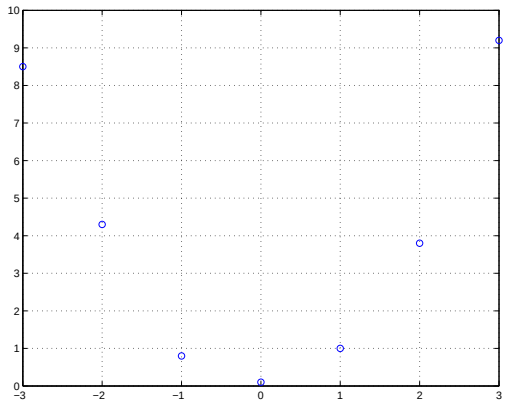
Suppose we wish to find the polynomial which best approximates the points $(-3, 8.5), (-2, 4.3), (-1, 0.8)$, $(0, 0.1), (1, 1), (2, 3.8)$, and $(3, 9.2)$. The first question we should ask ourselves is: What type of polynomial should we guess? For instance, do the points almost seem to lie on a straight line? If not, do they look like they might be better fitted by a parabola? To help answer these questions, we first need to plot the points. We can use MATLAB to do this. First, we store the x- and y-coordinates of the given points:

```
>> x = [-3 -2 -1 0 1 2 3]; y = [8.5 4.3 .8 .1 1 3.8 9.2];
```

(The semi-colon at the end of the line suppresses the output.) We then ask MATLAB to plot our points:

```
>> plot(x,y,'o'), grid on
```

A new window will open, with the following graph:

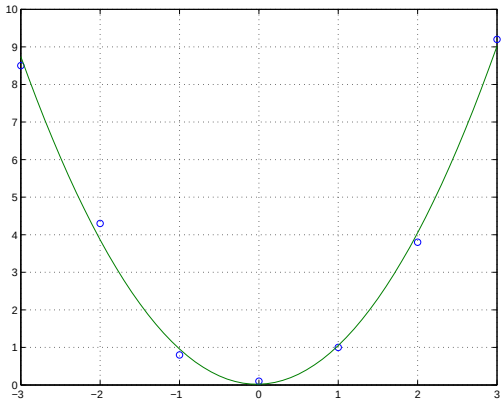


From the graph, it appears that the points don't lie near a straight line, but might lie near a parabola. We can use MATLAB to find the best parabola approximating these points. The appropriate command is called `polyfit`, and it requires three inputs: the x- and y-coordinates of the points to be fit, and the degree of the polynomial we want to fit to the points. In our case, we enter:

```
>> p = polyfit(x,y,2)
p =

    0.9845    0.0464    0.0190
```

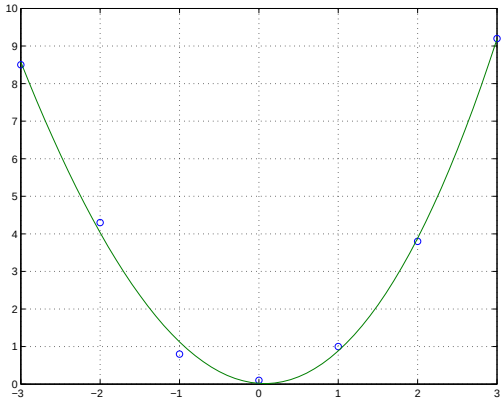
The output reads off the coefficients of the degree 2 polynomial best fitting the data, starting with the highest power. In our case, then, we see that the quadratic best fitting out data set is given by $p(x) = 0.9845x^2 + 0.0464x + 0.0190$. We can plot this polynomial on our previous graph, to see how well it actually fits the data:

```
>> x1 = -3:.1:3; y1 = polyval(p,x1);
>> plot(x,y,'o',x1,y1), grid on
```



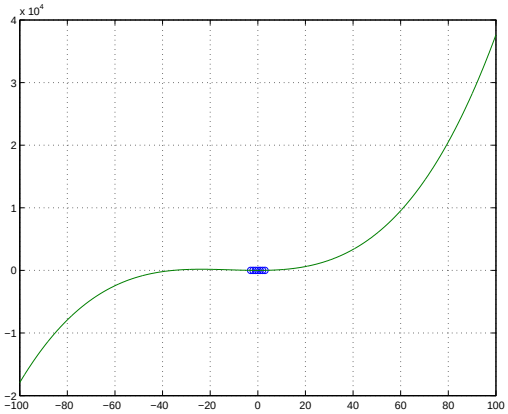
The parabola appears to fit the points quite well. However, suppose we tried to fit a cubic to these points:

```
>> p = polyfit(x,y,3);
>> x1 = -3:.1:3; y1 = polyval(p,x1);
>> plot(x,y,'o',x1,y1), grid on
```



It appears that the cubic fits the data points equally well, if not better! If we expand our view of the graph, however, we see that we may have been deceived:

```
>> x1 = -100:.1:100; y1 = polyval(p,x1);
>> plot(x,y,'o',x1,y1), grid on
```



The long-term behavior of this function is not what we would predict based only on the original data points. For example, based on the data points alone, there is no reason to expect the points should be negative for large negative values of x. As such, the parabola seems to represent a better fit to the points. In general, we look for the polynomial of smallest degree which seems to closely fit the data points.

2.1. **Exercises.**

(1) Find the quartic polynomial which best fits the following points:
$$(-5, 1600), (-3, 250), (-2, 40), (-1, 0), (0, -3), (3, 100), (4, 330), (5, 900)$$

(2) Sketch a graph of this quartic, along with the original data points.

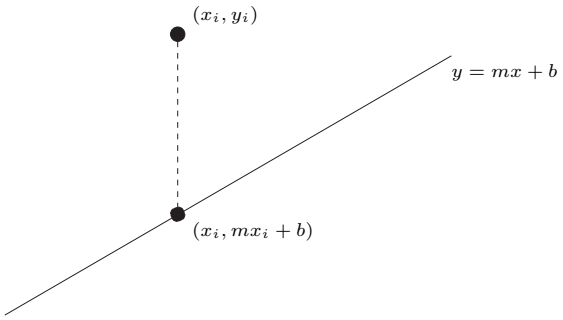
(3) Repeat the previous two exercises for fitting a quadratic curve to the points.

(4) Find the polynomial of smallest degree which appears to best approximate the following points, and sketch a graph of this polynomial along with the data points:

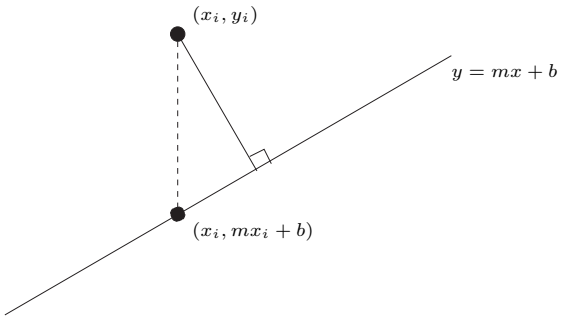
$$(-2.7, 18), (-1.7, 0), (-0.8, 5.5), (0.2, 8.7), (1, 4), (1.6, 0.2), (2.6, 14)$$

3. Bonus: An Approach other than Least Squares

There are many interpretations for what a 'best' approximation to a set of data points can mean. In the Least Squares approach, the 'best' line minimizes the sum of the squares of the differences between the y values of the given points and the corresponding y values of the line.



We could, however, consider minimizing the actual distances between the given points and the line.



(1) For a given line ℓ with equation $y = mx + b$ and point $P = (x_i, y_i)$, find the distance between P and ℓ.

(2) Suppose we are given the points $(1,1), (2,3)$, and $(4,3)$. Find the line ℓ which best approximates these points, in the sense that the sum of the distances between the given points and ℓ is minimized.

(3) Compare this line with the line found using Least Squares.