In science and engineering an important goal is to become a skilled practitioner by doing it yourself. The computer labs provide computational experience related to the analytic theory presented in the lectures.

Linear Least Squares

This lab is about using the the QR factorization to find the least squares fit of a polynomial to noisy data.

When $A \in \mathbf{R}^{m\times n}$ with $m > n$ the system of linear equations $Ax = b$ is said to be over determined. Such systems do not generally have a solution. Therefore, one looks for an x that minimizes the residual error $\|Ax - b\|$. Such an x can be obtained by solving $Rx = Q^T b$ where $A = QR$ is the reduced QR factorization of A. Recall,

- $Q \in \mathbf{R}^{m\times n}$ with $Q^T Q = I$.
- $R \in \mathbf{R}^{m\times m}$ with R upper triangular.

In Julia the left division operator `\` automatically uses the QR factorization to find the least-squares solution for over-determined systems. Thus, one can write `x=A\b` to find the x that minimizes $\|Ax - b\|$.

The following Julia session shows that solving the least-squares problem $Ax = b$ as `x=R\Q'*b` with the QR factorization is the same as `x=A\b` up to minor differences in rounding.

```
julia> A=rand(8,3)
8×3 Matrix{Float64}:
 0.769743  0.76063   0.793801
 0.506999  0.916752  0.436829
 0.969932  0.362005  0.258557
 0.518111  0.762398  0.80526
 0.479167  0.58614   0.583947
 0.299     0.776186  0.779349
 0.990704  0.37363   0.55788
 0.632395  0.545903  0.033063

julia> b=rand(8)
8-element Vector{Float64}:
 0.5397848276026893
 0.1439069926420633
```

```
 0.7684306151674534
 0.41459833254335243
 0.544044142618892
 0.11357772874330574
 0.1554894978784449
 0.1161757504504548

julia> using LinearAlgebra

julia> Q,R=qr(A);

julia> Qreduced=Q[:,1:3]
8×3 Matrix{Float64}:
 -0.397008  -0.13587     0.245731
 -0.261493  -0.48934    -0.448736
 -0.500259   0.402417   -0.15106
 -0.267225  -0.332293    0.296332
 -0.247138  -0.192915    0.162299
 -0.154214  -0.51511     0.270866
 -0.510972   0.407311    0.295185
 -0.326168  -0.0356442  -0.663911

julia> xqr=R\Qreduced'*b
3-element Vector{Float64}:
  0.44242045237489463
 -0.1132690000195104
  0.251348618601191

julia> x=A\b
3-element Vector{Float64}:
  0.4424204523748947
 -0.11326900001951039
  0.251348618601191
```

Note if you repeat the above steps on your computer, the random matrix A and vector b will be different as will the answer for x.

Polynomial Fitting

One application of least squares is fitting a polynomial to noisy data. For example, given data for $i = 1, \ldots, N$ of the form

$$(x_i, y_i) \qquad \text{where} \qquad y_i = p(x_i) + noise$$

suppose the goal is to find the constants c_j such that the polynomial

$$p(x) = c_1 + c_2 x + c_3 x^2 + c_4 x^3$$

is the one from which the data most likely came.

Under the assumption the noise process is independent, identically distributed and Gaussian, the desired c_j are exactly the values which minimize

$$J(c) = \sum_{i=1}^{N} \big| y_i - p(x_i) \big|^2.$$

Happily, minimizing $J(c)$ is the same as solving $Vc = y$ where V is the $N \times 4$ Vandermonde matrix and y the vector such that

$$V = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_N & x_N^2 & x_N^3 \end{bmatrix} \qquad \text{and} \qquad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}.$$

In this lab you will fit a cubit polynomial to the points (x_i, y_i) provided in an individualized data file. These points are stored one per line with x_i the first value and y_i the second. There are $N = 100$ lines in the file. Click on the following link to retrieve your file:

`https://fractal.math.unr.edu/~ejolson/466-22/lsquare/lsdata.cgi`

Please do not use anyone else's data for this lab.

The rest of this discussion presents a step-by-step walk through for the data which appears when I click the above link. That data is different than what you will obtain when you click the same link. To finish this lab please repeat these same steps but for own individualized data.

Upon clicking on the link, I obtained the example

The data was generated from the model

$$y_i = c_1 + c_2 x_i + c_3 x_i^2 + c_4 x_i^3 + \text{noise}.$$

Your data download is here.

In Julia-compatible code the true values of the c_j are

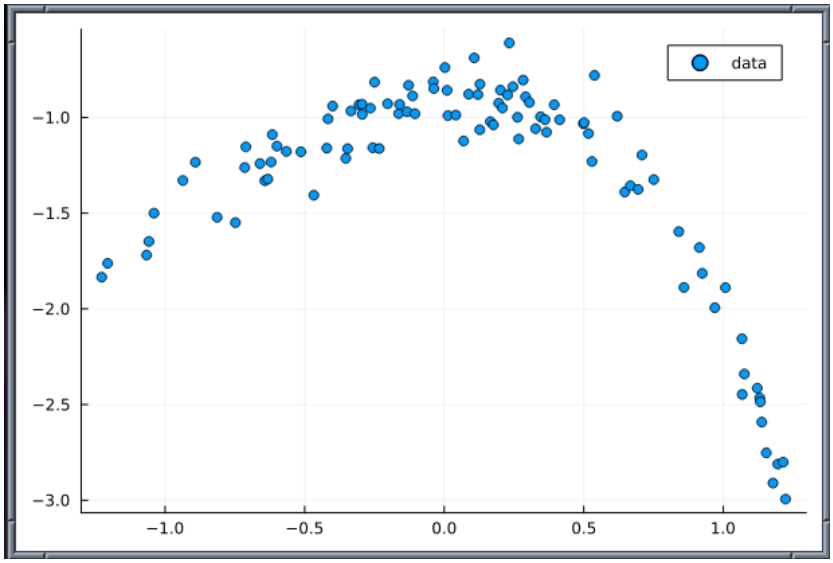
```
ctrue=[-0.9,0.32,-0.92,-0.58]
```

Make a working directory for this lab, for example `lab06`, then right click on the *here* link and save the file in that directory. In my case the name of the data file was `DE229EC9.dat`. It will be different for you.

It's always a good idea to visualize the data before performing any kind of numerical fit. Begin by reading in the data and plotting it. To do this use the `DelimitedFiles` and `Plots` libraries as follows:

```
1 using DelimitedFiles, Plots
2
3 data=readdlm("DE229EC9.dat")
4 x=data[:,1]
5 y=data[:,2]
6 display(scatter(x,y,label="data"))
```

At this point you should obtain a graph that looks similar to



The exact shape depends on your individualized data file.

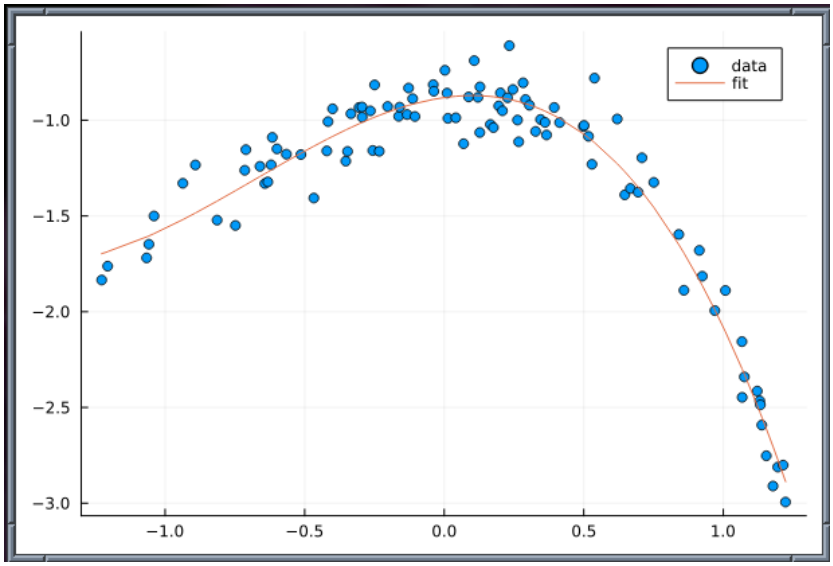
Next create the Vandermonde matrix A and solve the least squares problem $Vc = y$ to obtain the coefficients `cfit` that correspond to the polynomial from which the data most likely came.

```
8 phi=[x->1,x->x,x->x^2,x->x^3]
9 V=[phi[j](x[i]) for i=1:length(x), j=1:4]
10
11 cfit=V\y
12 yfit=V*cfit
13 display(plot!(x,yfit,label="fit"))
```

After executing line 11 the polynomial coefficients were

```
julia> cfit=V\y
4-element Vector{Float64}:
 -0.8818279900787308
  0.20464432548902906
 -0.941495558668323
 -0.46278525707704005
```

The resulting least-squares fit looks like



Again note your data and fit may have a different shape.

In applications the mechanism that generated the data is generally unknown. That's the whole point of finding the least-squares fit in the first place. In such cases after finding the least-squares fit one must subsequently ask whether the fit is statistically good enough.

In this lab the coefficients for the exact polynomial are known. This provides a test case to check that our numerical codes work as expected. To this end, we compare the fitted polynomial with the coefficients on the web page for the polynomial that actually generated the data.

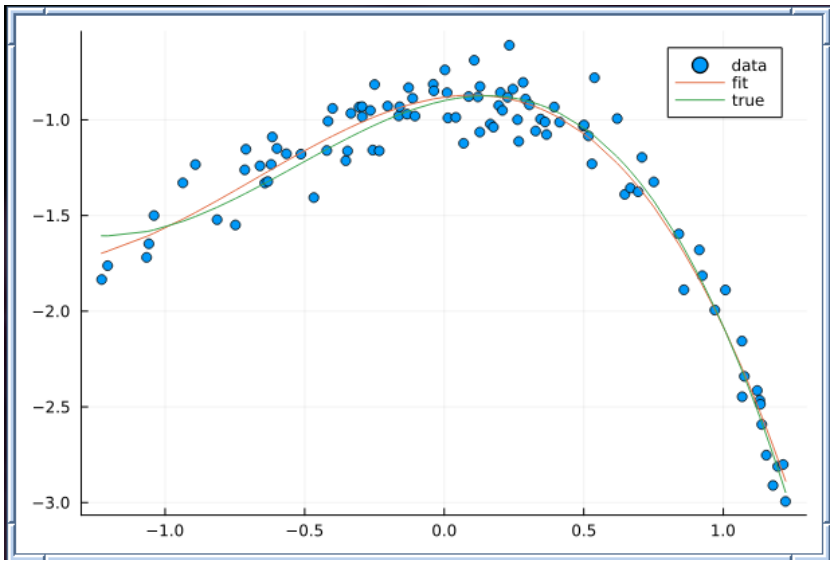
```
15 ctrue=[-0.9,0.32,-0.92,-0.58]
16 norm(cfit-ctrue)/norm(ctrue)
17 ytrue=V*ctrue
18 display(plot!(x,ytrue,label="true"))
```

Note that line 15 is pasted from the web page. The output

```
julia> norm(cfit-ctrue)/norm(ctrue)
0.11526839330427804
```

of line 16 indicates the fitted coefficients differ from the true coefficients by more than 10 percent. This is due to the noise in the data. Your fitted coefficients may be closer or further from the true coefficients.

For the data currently being analyzed the final graph comparing the polynomials looks like



Submitting Your Work

Two PDF files should be submitted for grading. The first file should contain two parts:

- A program that computes and then displays the fitted coefficients of the polynomial, the true coefficients and the norm of the difference between fitted and true coefficients.
- The output from running that program.

The second file should be graph showing the data, the fitted polynomial and the true polynomial. This can be obtained by placing the command `savefig("graph.pdf")` at the end of your program.

After debugging and making sure your program runs correctly, please prepare your submission by typing

```
$ julia lab06.jl >lab06.out
$ j2pdf -o lab06.pdf lab06.jl lab06.out
```

Before uploading, check `lab06.pdf` and `graph.pdf` with

```
$ evince lab06.pdf graph.pdf &
```

to make sure the output and the graph looks correct. Please reboot into Microsoft Windows before leaving the lab.