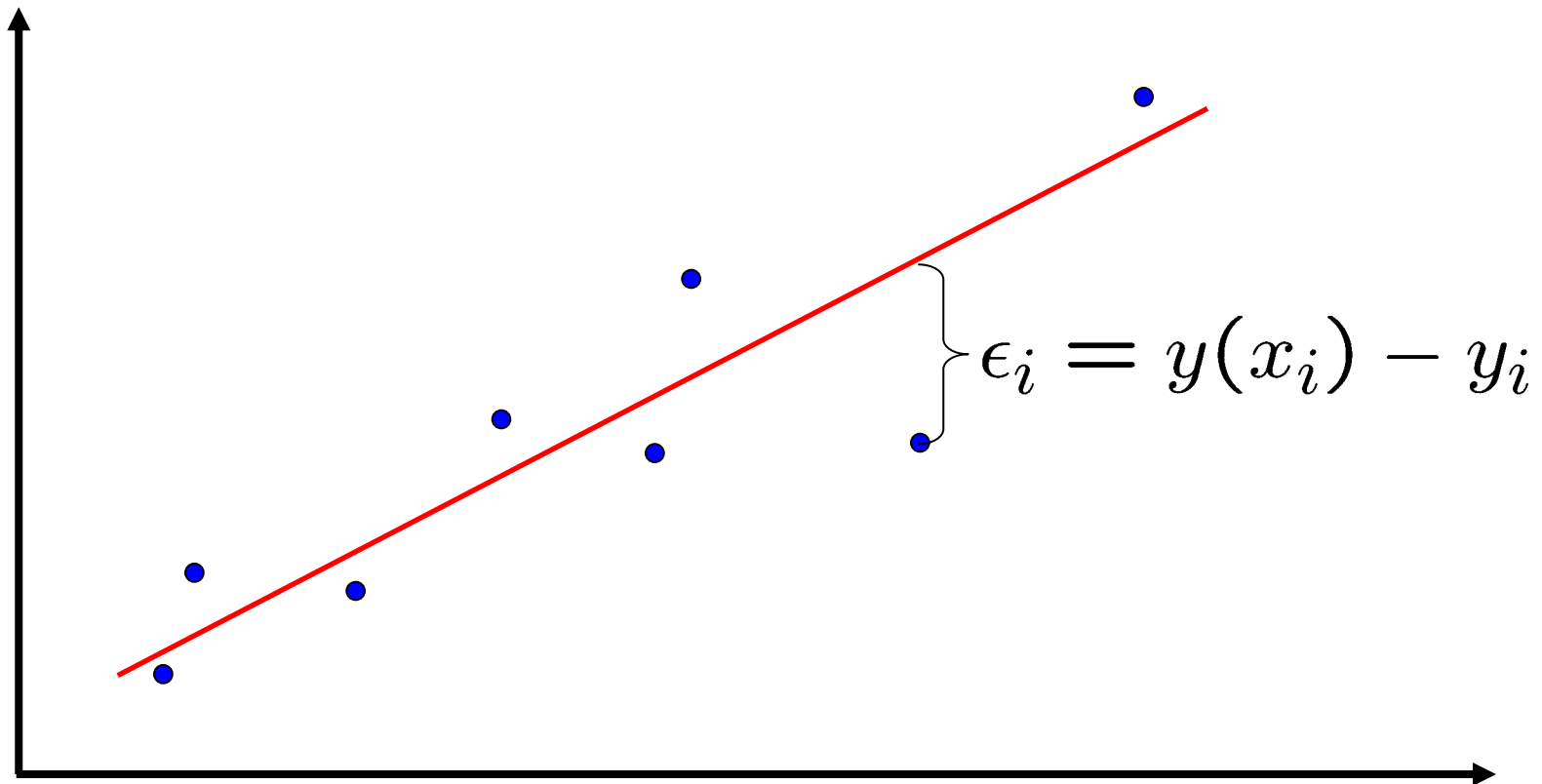


Goals for This Lecture:

- Introduce least-squares techniques for fitting models to data
- Expand the file I/O example code to solve carry out the least squares fit of a line

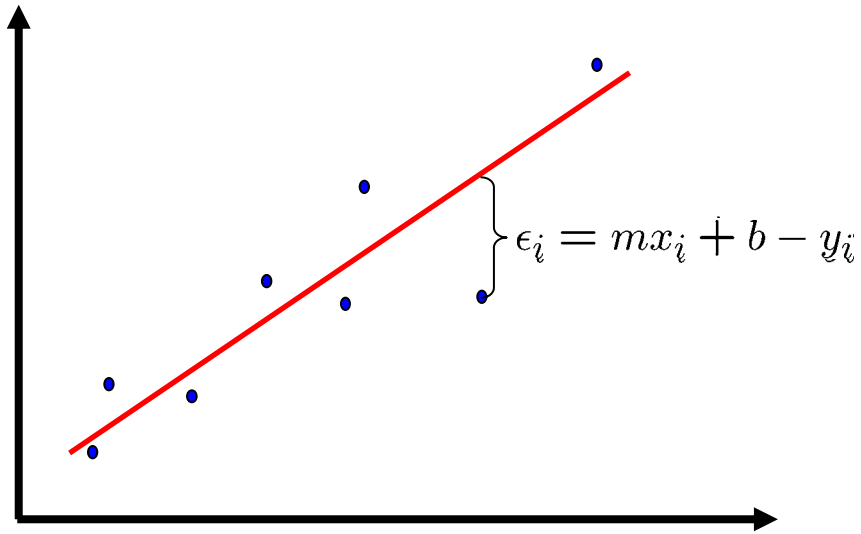
Least-Squares Fitting of Data



- We desire to minimize the overall error in fitting the lines to the data
- A good measure is the sum of the squares of the error at each point

$$\chi^2 = \sum_{i=1}^{i=N} \epsilon_i^2$$

Least-Squares Fitting of a Line



- χ^2 sum:

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

- Conditions for Minimization

$$\frac{\partial \chi^2}{\partial m} = 0 \quad \frac{\partial \chi^2}{\partial b} = 0$$

Least-Squares Fitting of a Line

Conditions for
Minimization:

Resultant equations:

$$\frac{\partial \chi^2}{\partial m} = 0 \quad \sum_{i=1}^{i=N} (mx_i + b - y_i) x_i = m \sum_{i=1}^{i=N} x_i^2 + b \sum_{i=1}^{i=N} x_i - \sum_{i=1}^{i=N} y_i x_i = 0$$

$$\frac{\partial \chi^2}{\partial b} = 0 \quad \sum_{i=1}^{i=N} (mx_i + b - y_i) x_i = m \sum_{i=1}^{i=N} x_i + bN - \sum_{i=1}^{i=N} y_i = 0$$

These equations can be solved for m & b:

$$m = \frac{\sum_{i=1}^{i=N} x_i y_i - \sum_{i=1}^{i=N} x_i \sum_{i=1}^{i=N} y_i}{N \sum_{i=1}^{i=N} x_i^2 - \left(\sum_{i=1}^{i=N} x_i \right)^2}$$

$$b = \frac{\sum_{i=1}^{i=N} y_i - m \sum_{i=1}^{i=N} x_i}{N}$$

C++ ifstream Objects

- C++ ifstream objects have a number of important methods that can be used to control input from the stream

- Open:

```
ifstream.open(file_name_string);
```

- Opens a file stream for reading

- Close:

```
ifstream.close();
```

- Closes a file stream

- Fail:

```
ifstream.fail();
```

- Indicates a failure on the input stream
 - A file open operation failed
 - Extraction of some values from the stream failed

- Clear:

```
ifstream.clear();
```

- Clears the stream for future reuse; should be called after closing file

Least-squares Fit of a Line, pt. 1

```
#include <iostream>
#include <fstream>
using namespace std;
int main(){
    ifstream inStream;           // Declare the input stream
    ofstream outStream;         // Declare the output stream
    double x, y;                 // Declare some variables
    double *xarray, *yarray;     // Declare dynamic arrays
    double sumx=0., sumy=0., sumxy=0., sumx2=0.; // Declare sums
    double slope, intercept ;    // Declare slope & intercept
    int npnts=0;                 // Number of points
    inStream.open("input.dat");  // Open the file for reading
    while(!inStream.fail()) {
        inStream >> x >> y ;    // Read x & y from the file
        if(inStream.fail()) break; // If a failure exit the loop
        npnts = npnts+1 ;       // Increment the data counter
    }
}
```

Least-squares Fit of a Line, pt. 2

```
inStream.close();           // Close the file
inStream.clear();           // Clear the stream for re-use

xarray = new double[npnts] ; // Create the x-array
yarray = new double[npnts] ; // Create the y-array

inStream.open("input.dat"); // Open the file for reading

for(int i=0; i < npnts ; i++) {
    inStream >> xarray[i] >> yarray[i] ; // Read in data
    sumx = sumx+xarray[i];
    sumy = sumy+yarray[i];
    sumxy = sumxy+xarray[i]*yarray[i];
    sumx2 = sumx2+xarray[i]*xarray[i];
}

inStream.close();           // Close the file
inStream.clear();           // Clear the stream for re-use
```

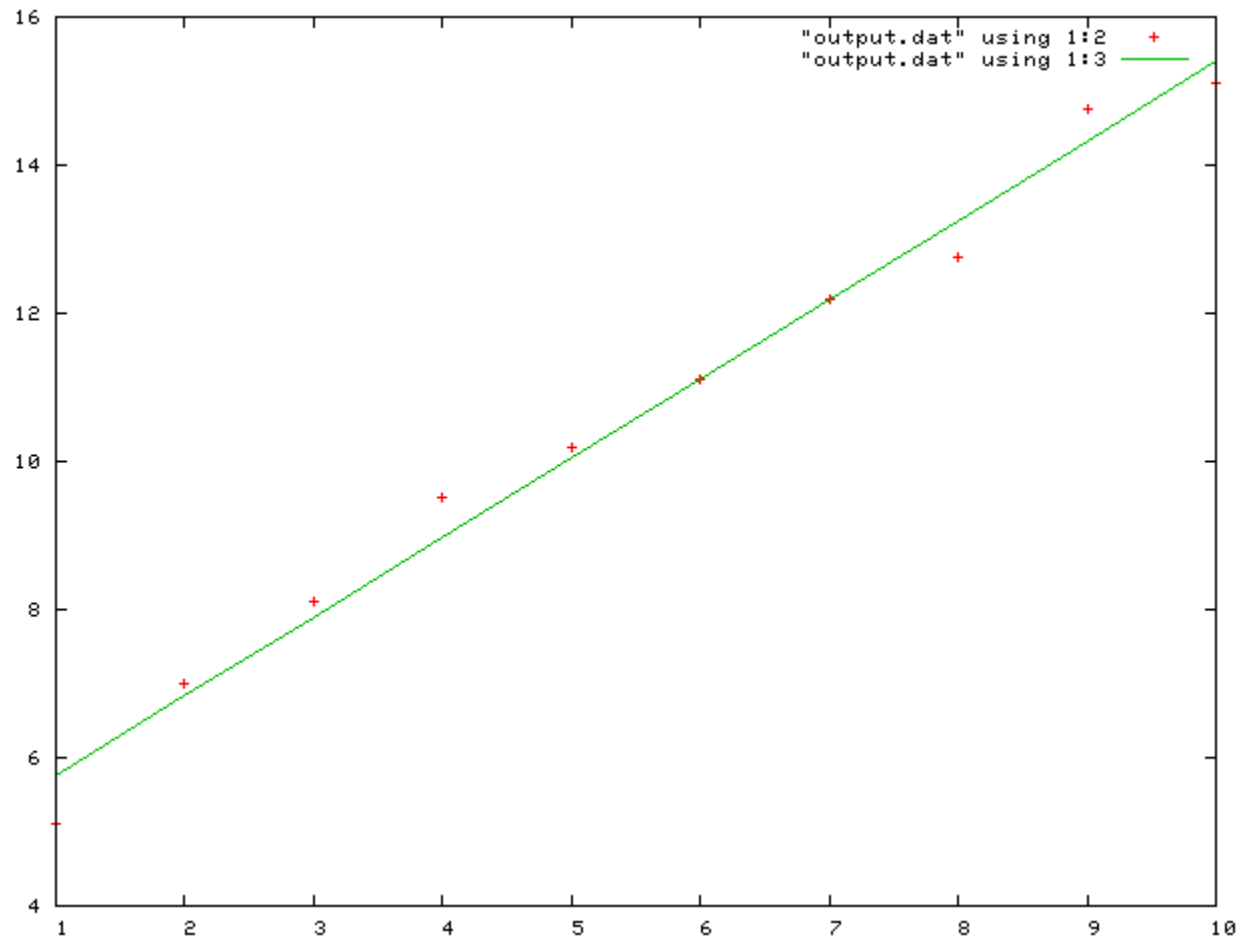
Least-squares Fit of a Line, pt. 3

```
slope = (npnts*sumxy-sumx*sumy)/(npnts*sumx2-sumx*sumx);
intercept = (sumy-slope*sumx)/npnts;
cout << " slope, intercept = " << slope << " "
      << intercept <<endl;

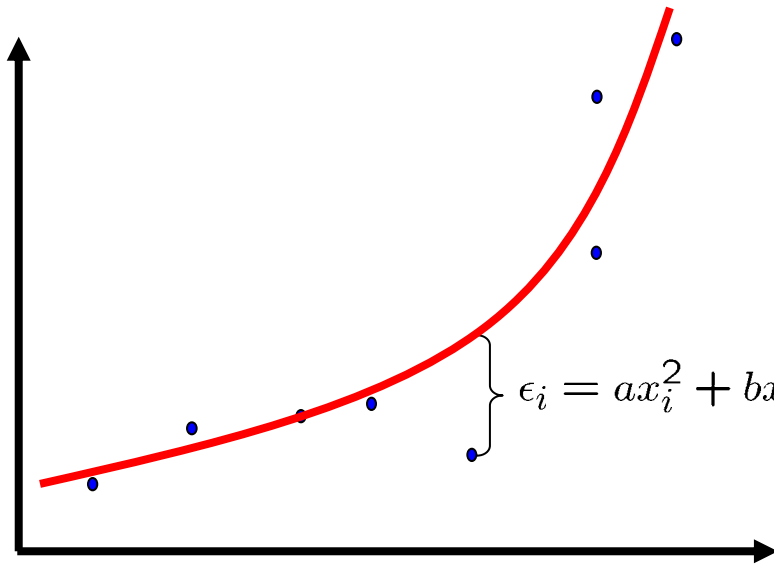
    // Now produce an output file to plot
ostream.open("output.dat");    // Open the file for output
    for(int i=0; i < npnts ; i++) {
        ostream << xarray[i] << " "
                << yarray[i] << " "
                << slope*xarray[i]+intercept << endl;
    }
ostream.close();                // Close the file

return(0);                       // Return a zero error code
}
```

Results



Least-Squares Fitting of a Parabolic Curve



- χ^2 sum:

$$\chi^2 = \sum_{i=1}^{i=N} (ax_i^2 + bx_i + c - y_i)^2$$

- Conditions for Minimization

$$\frac{\partial \chi^2}{\partial a} = 0$$

$$\frac{\partial \chi^2}{\partial b} = 0$$

$$\frac{\partial \chi^2}{\partial c} = 0$$

Least-Squares Fitting of a Parabolic Curve

$$\frac{\partial \chi^2}{\partial a} = \sum_{i=1}^{i=N} (ax_i^2 + bx_i + c - y_i) x_i^2 = a \sum_{i=1}^{i=N} x_i^4 + b \sum_{i=1}^{i=N} x_i^3 + c \sum_{i=1}^{i=N} x_i^2 - \sum_{i=1}^{i=N} y_i x_i^2 = 0$$

$$\frac{\partial \chi^2}{\partial b} = \sum_{i=1}^{i=N} (ax_i^2 + bx_i + c - y_i) x_i = a \sum_{i=1}^{i=N} x_i^3 + b \sum_{i=1}^{i=N} x_i^2 + c \sum_{i=1}^{i=N} x_i - \sum_{i=1}^{i=N} y_i x_i = 0$$

$$\frac{\partial \chi^2}{\partial c} = \sum_{i=1}^{i=N} (ax_i^2 + bx_i + c - y_i) = a \sum_{i=1}^{i=N} x_i^2 + b \sum_{i=1}^{i=N} x_i + cN - \sum_{i=1}^{i=N} y_i = 0$$

- Solve these equations for a, b, & c in terms of sums over powers of x_i & y_i

Non-linear Least-Squares Fitting

- The least-squares examples that we have seen thus far (fitting a line & parabola) are examples of *linear least-squares* methods.
- The minimization conditions produce a set of equations that are *linear in the unknown parameters*.

$$a \sum_{i=1}^{i=N} x_i^4 + b \sum_{i=1}^{i=N} x_i^3 + c \sum_{i=1}^{i=N} x_i^2 - \sum_{i=1}^{i=N} y_i x_i^2 = 0$$

$$a \sum_{i=1}^{i=N} x_i^3 + b \sum_{i=1}^{i=N} x_i^2 + c \sum_{i=1}^{i=N} x_i - \sum_{i=1}^{i=N} y_i x_i = 0$$

$$a \sum_{i=1}^{i=N} x_i^2 + b \sum_{i=1}^{i=N} x_i^1 + cN - \sum_{i=1}^{i=N} y_i = 0$$

- We can solve these equations for the parameters (such as a, b, & c in the case of parabola) in terms of sums over powers of x_i & y_i
- What do we do if the model is non-linear in its unknown parameters?
- Example: fitting $\alpha \sin(\omega x + \phi)$ to a set of data
 - The minimization conditions are non-linear in ω & ϕ

Least-Squares Fitting by Brute-Force & Newton-Raphson

- If you are trying to fit a non-linear model to a set of data you are left with very few choices
- You can do a brute-force search of the parameter space to see which parameters produce the smallest χ^2
 - This is a very unsophisticated approach which is computationally time-consuming
 - Nevertheless, if the number of parameters is small and if the possible range of parameters is reasonably constrained this approach
- You can apply a non-linear root finding method, such as Newton-Raphson iteration, to find the solution of the minimization conditions
 - This can also be computationally demanding
 - The non-linear root finding methods may not converge in some cases
- Bottom line: Non-linear least-squares fitting is a difficult proposition that can require a serious effort!

Assignment

- Read Savitch Sections 12.1 & 12.2.