

# Goals for This Lecture:

- Understand a few more features of LaTeX
- Learn the basics of creating a web page with HTML
- Understand I/O to/from files in C++
- Introduce least-squares techniques for fitting models to data

# Sections & Subsections in LaTeX

```
\documentclass[12pt]{article}
\usepackage{epsfig}
\begin{document}

\section{Introduction}
Blah-blah-blah

\section{Methods}
Yadda-yadda-yadda

\subsection{Techniques}
blah-blah-blah.

\end{document}
```

# Multiline Formulas in LaTeX

```
\documentclass[12pt]{article}
\usepackage{epsfig}
\begin{document}

\begin{eqnarray}{cc}
F & = & a+b+c+d+e+f+g+h \quad \text{\nonumber} \\
& & \alpha + \beta + \gamma + \delta + \epsilon
\end{eqnarray}

\end{document}
```

# Arrays & Conditional Formulas in LaTeX

```
\documentclass[12pt]{article}
\usepackage{epsfig}
\begin{document}

\begin{equation}
f = \left\{
\begin{array}{cc}
x^2 & \mbox{if } x > 0 \\
-x^2 & \mbox{otherwise}
\end{array}
\right.
\end{equation}

\end{document}
```

# Embedding Postscript files into LaTeX

- Including encapsulated postscript files:

```
\documentclass[12pt]{article}
\usepackage{epsfig}
\begin{document}

\begin{figure}
\epsfig{file=plot1.eps}
\caption{A plot of speed versus time}
\end{figure}

\end{document}
```

- There are lots of options for the EPSFIG package. Read the documentation.

# Creating Lists in LaTeX

```
\documentclass[12pt]{article}
\usepackage{epsfig}
\begin{document}
\begin{itemize}
\item Apples
\item Oranges
\item Bannas
\end{itemize}
\begin{enumerate}
\item Horses
\item Cows
\item Pigs
\item Sheep
\end{enumerate}

\end{document}
```

# Centering Tables in LaTeX

```
\documentclass[12pt]{article}
\usepackage{epsfig}
\begin{document}
\begin{center}
\begin{tabular}{|c|c|}
\hline
Theory & Experiment \\
\hline
1.0 & 1.1 \\
\hline
2.0 & 1.99 \\
\hline
3.0 & 3.1 \\
\hline
\end{tabular}
\end{center}
\end{document}
```

# Centering Tables in LaTeX

```
\documentclass[12pt]{article}
\usepackage{epsfig}
\begin{document}
\begin{center}
\begin{tabular}{|c|c|}
\hline
Theory & Experiment \\
\hline
1.0 & 1.1 \\
\hline
2.0 & 1.99 \\
\hline
3.0 & 3.1 \\
\hline
\end{tabular}
\end{center}
\end{document}
```

# Creating Web Pages

```
<html>
  <head>
    <title>
      A Hello World Page!
    </title>
  </head>

  <body>
    <p>
      Hello World!
    </p>
    <p>
      A new paragraph
    </p>
  </body>
</html>
```

# Creating Web Pages

```
<html>

  <head>
    <title>
      A Hello World Page!
    </title>
  </head>

  <body>
    Hello World!
  </body>

</html>
```

# Headings & Horizontal Rules in HTML

```
<h1> This is a Level 1 heading </h1>
```

```
<h2> This is a level 2 heading </h2>
```

```
<h3> This is a level 3 heading </h3>
```

Here is some text.

Now place a horizontal rule.

```
<hr>
```

This is text after the horizontal rule.

# Lists in HTML

Here is an unnumbered list:

```
<ul>  
<li> Apples  
<li> Oranges  
<li> Bannanas  
</ul>
```

Here is an ordered (numbered) list:

```
<ol>  
<li> Horses  
<li> Cows  
<li> Sheep  
</ol>
```

Here is a defintion list:

```
<dl>  
<dt>Dogs<dd>Also known as canines  
<dt>Cats<dd>Also known as felines  
</dl>
```

# Links in HTML

Here is a link to

```
<a href="http://www.astro.sunysb.edu/dswesty">
```

My homepage

```
</a>
```

# Inline Images in HTML

An inline image can be created using the IMG tag:

```

```

# C++ File I/O

- **C++ provides objects that allow I/O to and from files**
- **Instances of these objects are called streams**
- **We have already made use of the STDOUT/STDIN streams cout & cin**
- **We show how to use these file I/O streams by example**

# Reading Values from a File

```
#include <iostream>
#include <fstream>
using namespace std;
main()
{
    ifstream inStream;           // Declare the input stream
    double x, y;                 // Declare some variables

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

    inStream >> x >> y ;        // Read x & y from the file

    cout << " x, y = " << x << " " << y << endl;
    inStream.close();           // Close the file
    return(0);                  // Return a no-error value
}
```

# C++ instream Objects

- C++ instream 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

# Reading an Arbitrary Amount of Data from a File, pt. 1

```
#include <iostream>
#include <fstream>
using namespace std;
main(){
    ifstream inStream;           // Declare the input stream
    double x, y;                // Declare some variables
    double *xarray, *yarray;    // Declare dynamic arrays
    int npnts;                  // Number of points

    inStream.open("input.dat"); // Open the file for reading
    while(!inStream.fail()) {
        inStream >> x >> y ;    // Read x & y from the file
        npnts = npnts+1 ;       // Increment the data counter
        if(inStream.fail()) break; // If a failure exit the loop
    }

    inStream.close();           // Close the file
```

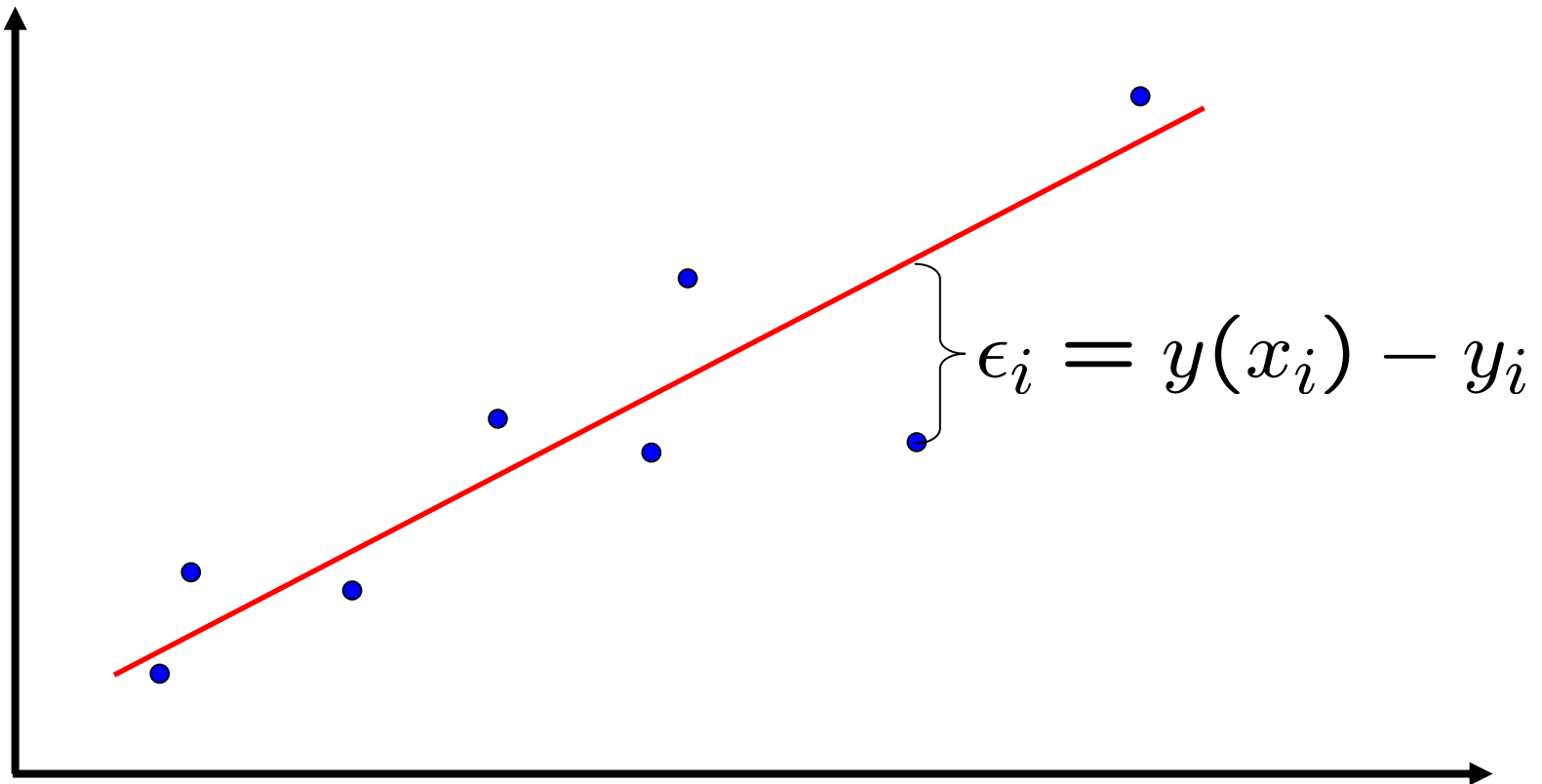
## Reading an Arbitrary Amount of Data from a File, pt. 2

```
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
inStream.close(); // Close the file

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

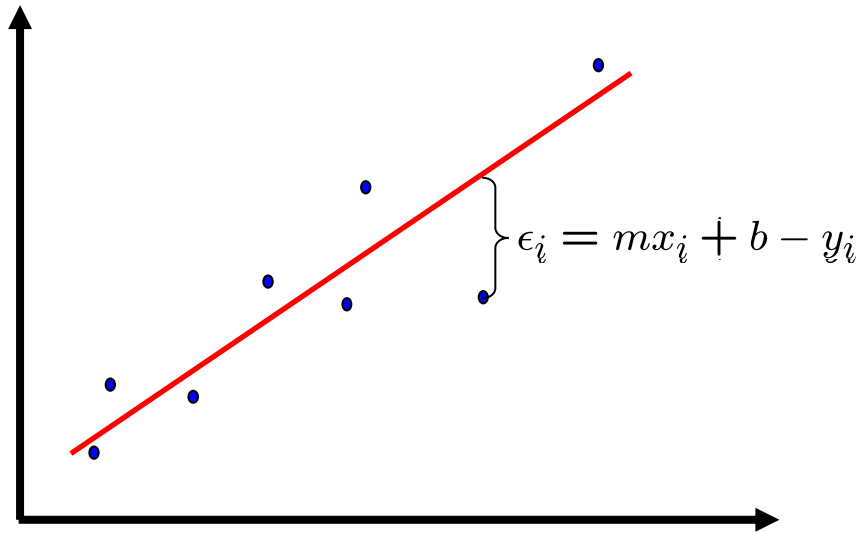
# Least-Squares Fitting of Data



- 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



- $\chi^2$  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:

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

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

# Least-squares Fit of a Line, pt. 1

```
#include <iostream>
#include <fstream>
using namespace std;
main(){
    ifstream inStream;           // Declare the input 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;                   // Number of points
    inStream.open("input.dat");  // Open the file for reading
    while(!inStream.fail()) {
        inStream >> x >> y ;    // Read x & y from the file
        npnts = npnts+1 ;       // Increment the data counter
        if(inStream.fail()) break; // If a failure exit the loop
    }
    inStream.close();           // Close the file
```

## Least-squares Fit of a Line, pt. 2

```
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
slope = (npnts*sumxy-sumx*sumy)/(npnts*sumx2-sumx*sumx);
intercept = (sumy-slope*sumx)/npnts;
cout << " slope, intercept = " << slope << " "
        << intercept <<endl;
return(0); // Return a zero error code
}
```

# Assignment

- Work through the LaTeX tutorials that are linked to on the course web page.
- Work through the NCSA `<html>` tutorials linked to on the course web page.
- Read Savitch Sections 12.1 & 12.2.