EEE 208 – Programming for EEE

Assist. Prof. Dr. Engin Mendi

Recursive Functions I

- What would happen if a function has to call itself?
- You are familiar with the factorial: 0! = 1, 1! = 1, 2! = 2*1,… and n! = n*(n-1)*…*2*1 for any integer n. MATLAB has a function called **factorial** that calculates these values.
- Now we want to write a recursive function called **rfac**:

```
function A = rfac(N)
if N==0 % The N=0 case is "obvious"
   A = 1;
else     % The solution for N is easily written in
  terms of the  %solution for the N-1 case.
   A = N*rfac(N-1);
end
```

- Suppose we want to find C = rfac(3). Can you explain how this function works?

Recursive Functions 2

```
function A = fac3(N)
if N==0
   A = 1;
else
   A = N*fac3(N-3);
end
```

```
So fac3(12) = 12*9*6*3*1 = 1944
```

- Recursive solutions generally use more memory, because at the deepest level, several function workspaces exist at the same time.

Recursive Functions 3

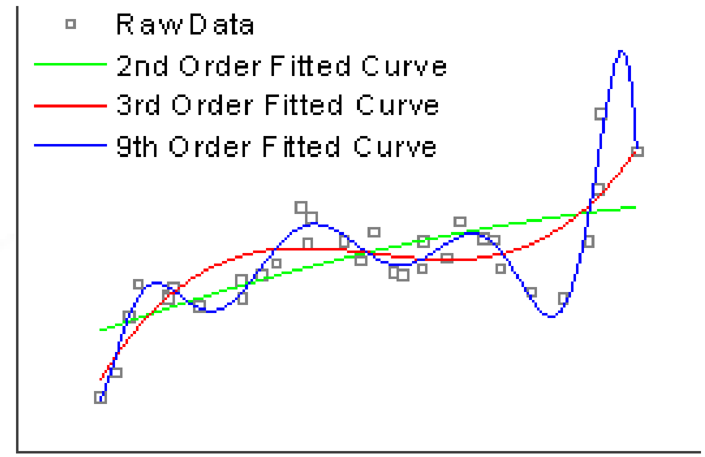
- What is wrong with this function?

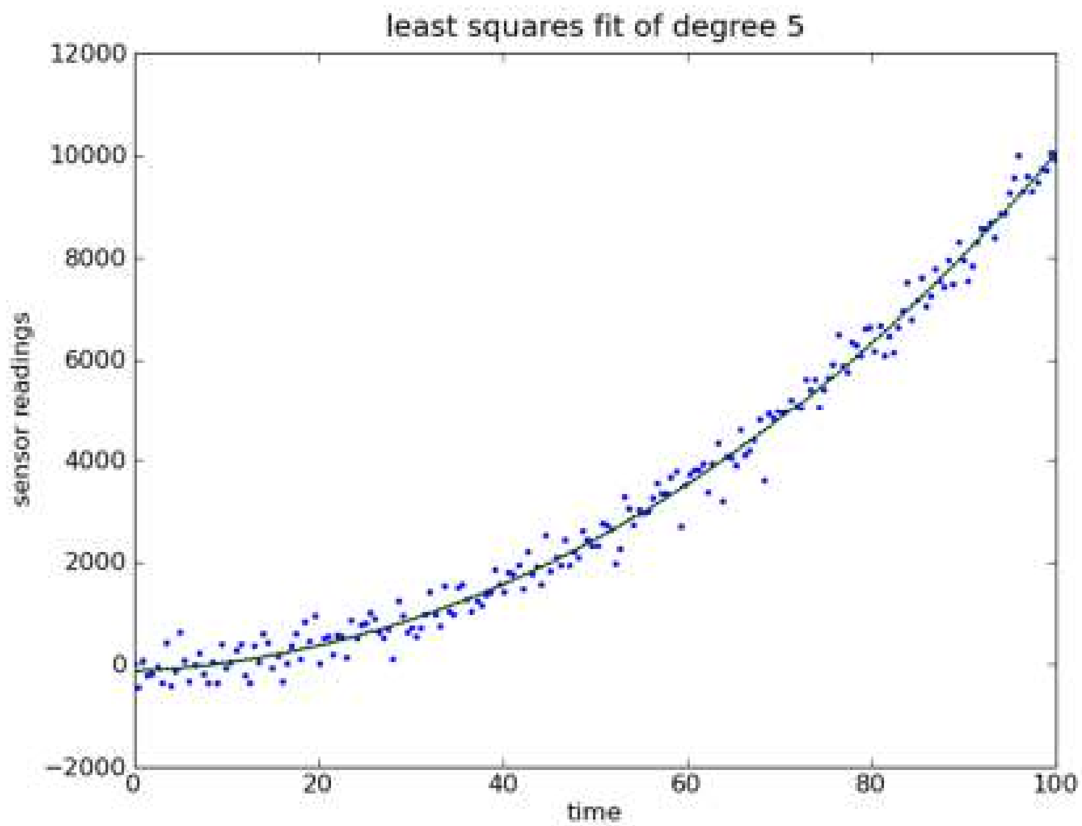
```
function A = rfac(N)
% The solution for N is easily
% written in terms of the solution
% for the N-1 case.
A = N*rfac(N-1);
```

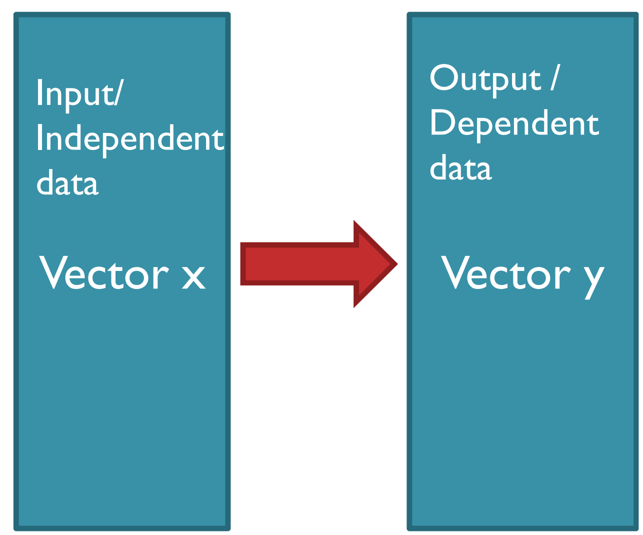
Regression and Curve Fitting

Applications:

1. To Model the Data (to find a function that would explain the changes, behavior, or relationship between independent and dependent variables)
2. To forecast the Data







Regression: Fitting a Line to the Data

We can fit a line to our data:

- Given m pairs of data: (x1,y1), (x2,y2), … , (xm,ym) or in general, *(xi, yi), i = 1, . . . , m,*

find the coefficients α and β such that $F(x) = \alpha x + \beta$

- Next, the residuals *r(i)* are defined as the actual output minus the estimation:

$$
\begin{aligned}
r_1 &= F(x_1) - y_1 = \alpha x_1 + \beta - y_1 \\
r_2 &= F(x_2) - y_2 = \alpha x_2 + \beta - y_2 \\
&\dots \\
r_i &= F(x_i) - y_i = \alpha x_i + \beta - y_i \\
&\dots \\
r_m &= F(x_m) - y_m = \alpha x_m + \beta - y_m
\end{aligned}
$$

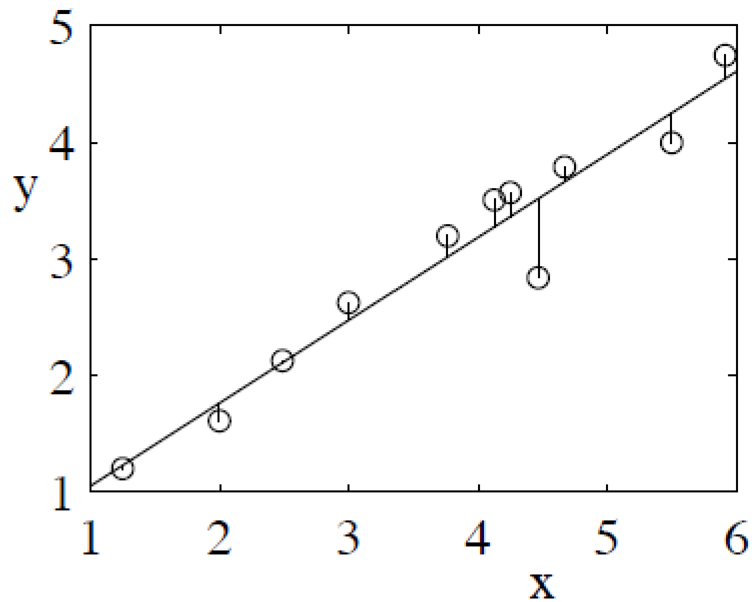
- Sorting it in the matrix form:

$$
\begin{bmatrix} r1 \\ r2 \\ \vdots \\ rm \end{bmatrix} = \begin{bmatrix} x1 & 1 \\ x2 & 1 \\ \vdots & \vdots \\ xm & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} - \begin{bmatrix} y1 \\ y2 \\ \vdots \\ ym \end{bmatrix}
$$

Regression: Curve-fitting with minimum error

- If we define the vector of residuals r, we want to find α and β such that $\sum (r(i)^2)$ is minimized.
- This technique is known as the **Method of Least Squares**

- F(x) can be a polynomial of any degree, a sin/cos function, exponential, power, etc.



Regression and Curve Fitting

- First of all, please read the following notes:
- http://en.wikipedia.org/wiki/Curve_fitting

- In MATLAB, we can use the function polyfit and the Curve Fitting Toolbox (cftool) to find the best fits for our data

- See help **polyfit and cftool** for more information

Example: First-Degree Fitting In Matlab

This table shows the prices of different sizes of SD memory cards advertised in the February 19, 2012

M-file: example_polyfit_1

Memory Capacity in GB	Price in USD
2	9.99
4	10.99
8	19.99
16	29.99

- Question: Is there a linear function that can explain the change in price of SD cards in terms of their capacity?
- Let's define two vectors x and y:

```
x = [2 4 8 16] ;  y = [9.99 10.99 19.99 29.99] ;
p1=polyfit(x,y,1);
plot(x,y,'o', 'MarkerSize', 10, 'LineWidth',2);  grid on;
hold on;
z = 0:0.1: 24;% generate a vector around the input interval
plot(z,polyval(p1,z), 'r--', 'LineWidth',2);
```

- Check the output: p1 = 1.4913   6.5552 so the line equation is:

$F(x) = 1.49\,x + 6.56$

Example: Second-Degree Fitting In Matlab

- Let's find the best second degree polynomial that fits the points (-1,0),(0,-1), and (2,3)

- Solution: Define data vectors
- `X=[-1 0 2]; Y=[0 -1 3],`
- `p2=polyfit(X,Y,2);`
- `plot(X,Y,'o', 'MarkerSize', 10, 'LineWidth',2);`
- `grid on; hold on;`
- `z = -3:0.01:3;`
- `plot(z,polyval(p2,z), 'r-', 'LineWidth',2);`

- `M-file: example_polyfit_2`

- For more options, check **fittype, fitoptions, and fit**

Polynomial Fitting: Practice

1. Evaluate $y=x^2$ for x= -4:0.1:4.
2. Add random noise to these samples. Use randn.
3. Plot the noisy signal with * markers.
4. Fit a 2nd degree polynomial to the noisy data.
5. Plot the fitted polynomial on the same plot, using the same x values and a red line.

```
M-file: example_polyfit_3
```