MA3457/CS4033:
Numerical Methods for Calculus and Differential Equations

Course Materials

P A R T II

B'14
2014-2015

Approximation – Key Idea

Function approximation is *closely related to the idea of function interpolation*.

In F.A., ***we do not require the approximating function to match the given data exactly***.

This helps *avoid some of the difficulties* observed with interpolation:

- when trying to match a *large amount of data*,
- when working with *"noisy" data* (or data containing *error in measurements*),
- etc.

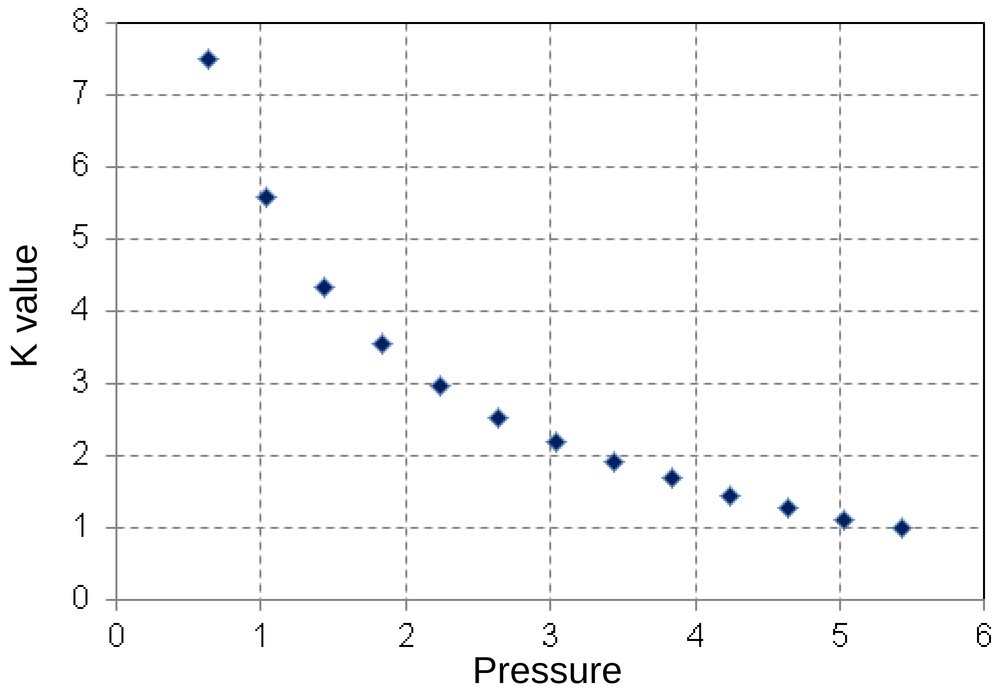
There are many applications in which:

a functional form is known and ***the best function of that form*** *(accommodating the data point)* ***is required***.

In situations like this, *approximation is an ultimate choice*.

Introductory Illustration

Equilibrium Constant of a Reaction vs Pressure



Experimental Data:

Pressure:

[0.635 1.035 1.435 1.835 2.235 2.635 3.035 3.435 3.835 4.235 4.635 5.035 5.435]

K value:

[7.5 5.58 4.35 3.55 2.97 2.53 2.2 1.93 1.7 1.46 1.28 1.11 1.0]

Question: ***What function does represent this behavior?***

Concept of the Method of Least Squares

The most common approach to "best fit" approximation is ***<u>to minimize the sum of the squares of the differences between the data values and the values of the approximating function</u>***:

```
Minimize {(Data value 1 – approx. f)² + … + (Data value n – approx. f)²}
```

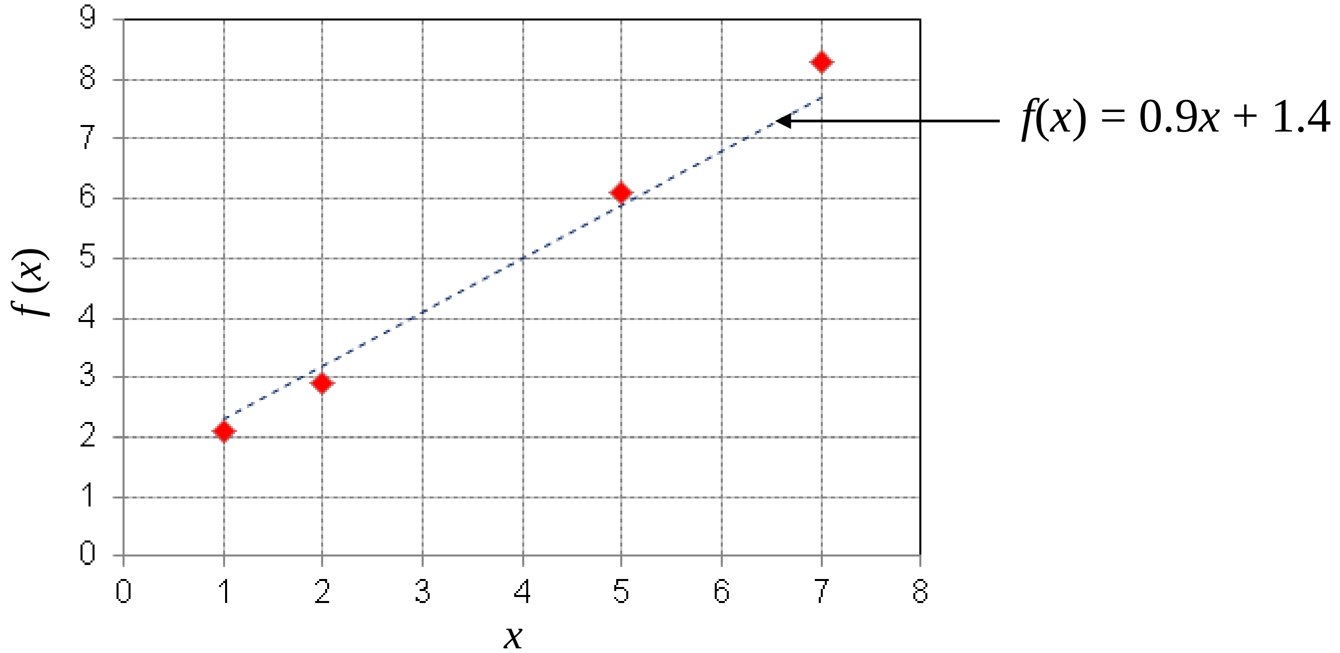
<u>Advantages</u> in using ***<u>the square of the differences</u>***

(*rather than the difference, or the absolute value of the difference, or some other measure of the error*):

- positive differences do not cancel negative differences;
- differentiation is not difficult, and
- small differences become smaller and larger differences are magnified.

Linear Least Squares Approximation

Example – Four Data Points



x_i	y_i	ax_i+b	$d_i=y_i-(ax_i+b)$
1	2.1	1.9791	0.1209
2	2.9	3.0231	-0.1231
3	6.1	6.1549	-0.0549
4	8.3	8.2429	0.0571

Linear LS Approximation – MATLAB Script

LIBRARY OF MATLAB PROCEDURES

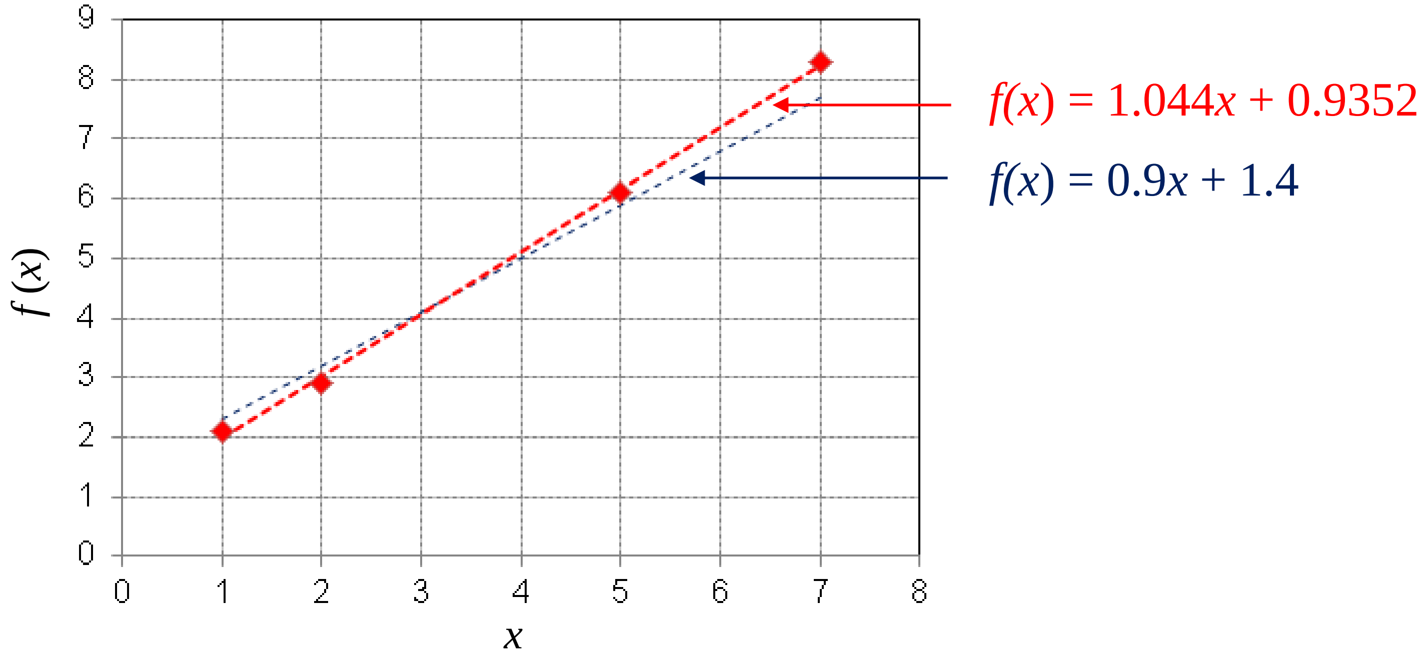
Lin_LS

Gives the linear least squares approximation

```
function s = Lin_LS(x, y)
%
% Linear Regression Function
%
% Input x and y as row or column vectors
% (they are converted to column form is necessary)
%
m = length(x); x = x(:); y = y(:);
sx = sum(x); sy = sum(y);
sxx = sum(x.*x); sxy = sum(x.*y);
%
a = (m*sxy - sx*sy) / (m*sxx - sx^2)
b = (sxx*sy - sxy*sx) / (m*sxx - sx^2)
%
table = [x y (a*x+b) (y-(a*x+b))];
disp('    x          y          a*x+b      y-(a*x+b)')
disp(table), err = sum(table( : , 4) .^ 2)
s(1) = a; s(2) = b;
```

Linear Least Squares Approximation

Example – Four Data Points



Quadratic LS Approximation – MATLAB Script

LIBRARY OF MATLAB PROCEDURES

Quad_LS

Gives the quadratic least squares approximation

```
function z = Quad_LS(x, y)
%
% Quadratic Regression Function
%
% Input x and y as row or column vectors
%
n = length(x); x = x(:); y = y(:);
%
sx = sum(x); sx2 = sum(x.^2);
sx3 = sum(x.^3); sx4 = sum(x.^4);
sy = sum(y); sxy = sum(x.*y);
sx2y = sum(x.*x.*y);
A = [ sx4 sx3 sx2,
      sx3 sx2 sx,
      sx2 sx n]
r = [sx2y sxy sy]'
z = A\r;
%
a = z(1), b = z(2), c = z(3)
%
table = [x y (a*x.^2+b*x+c) (y-(a*x.^2+b*x+c))];
disp('pi=a*x.^2+b*x+c')
disp('    x         y         pi        y-pi')
disp(table)
err = sum(table( : , 4) .^ 2)
```

Cubic LS Approximation

The *Normal Equations* to Determine the Coefficients for the "Best Fit" Cubic Function

$$a\sum_{i=1}^{n} x_i^6 + b\sum_{i=1}^{n} x_i^5 + c\sum_{i=1}^{n} x_i^4 + d\sum_{i=1}^{n} x_i^3 = \sum_{i=1}^{n} x_i^3 y_i$$

$$a\sum_{i=1}^{n} x_i^5 + b\sum_{i=1}^{n} x_i^4 + c\sum_{i=1}^{n} x_i^3 + d\sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} x_i^2 y_i$$

$$a\sum_{i=1}^{n} x_i^4 + b\sum_{i=1}^{n} x_i^3 + c\sum_{i=1}^{n} x_i^2 + d\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} x_i y_i$$

$$a\sum_{i=1}^{n} x_i^3 + b\sum_{i=1}^{n} x_i^2 + c\sum_{i=1}^{n} x + d\sum_{i=1}^{n} 1 = \sum_{i=1}^{n} y_i$$

MATLAB Function `polyfit`

Built-in MATLAB function **`polyfit`** *finds the coefficients of the polynomial of a specified n-th degree* that best fits a set of data, in a least squares sense:

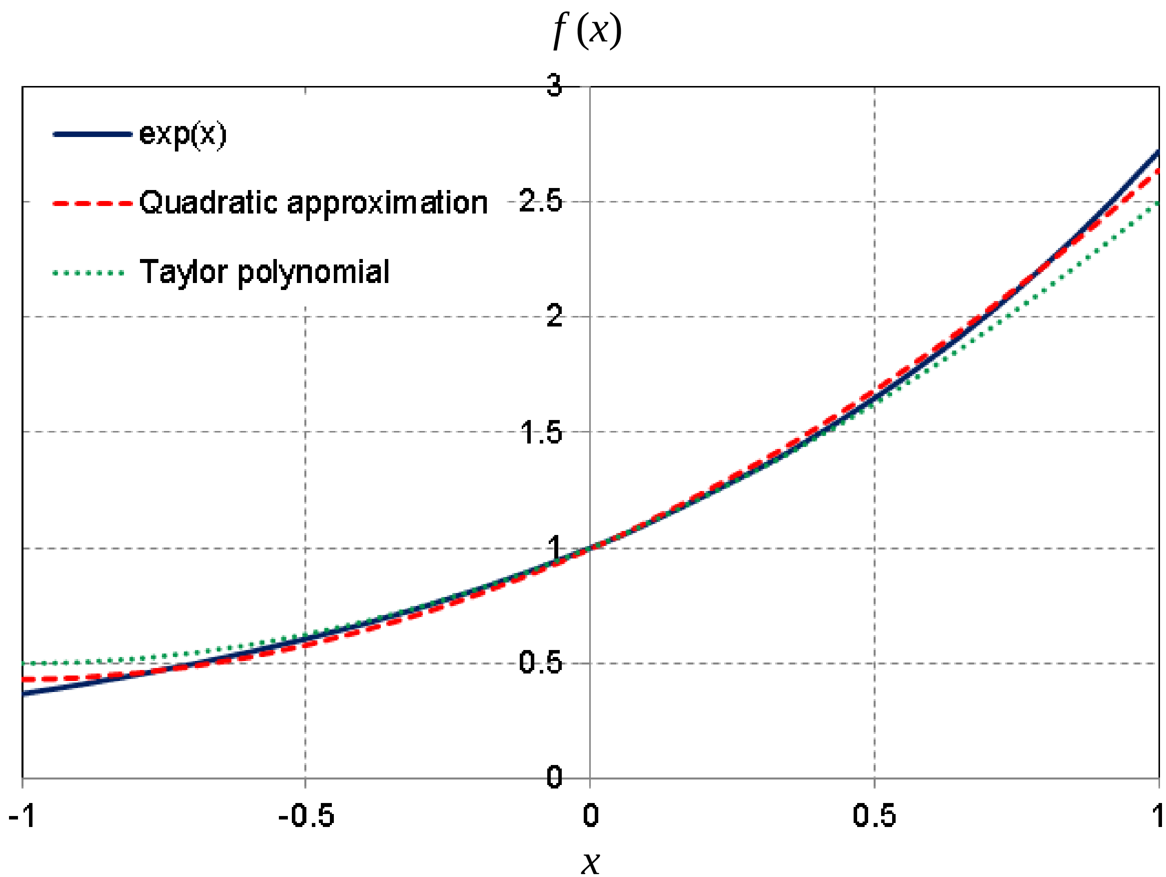
```
> [p, S] = polyfit(x, y, n)
```

where **`x`** and **`y`** are vectors containing the data, and
`n` is the degree of the polynomial desired.

The function returns *the coefficients of the polynomial in the vector* **`p`**.

The returned structure **`S`** can be used (with the function **`polyval`**) to obtain error bound.

Continuous LS Approximation



Quadratic approximation: $p(x) = 0.5368x^2 + 1.1037x + 0.9963$

Taylor polynomial: $t(x) = 0.5x^2 + x + 1$

Continuous LSA & Matrix [P]

$$[P][z] = [q]$$

- In some applications, matrix [P] may be ***ill-conditioned***.

This term "ill-conditioned matrix" is linked with the *matrix condition number* k(**A**) of a square matrix **A** defined as

$$k(\mathbf{A}) = ||\mathbf{A}||\ ||\mathbf{A}^{-1}||$$

where ||.|| is any valid matrix norm. **The condition number is a measure of stability or sensitivity of a matrix to numerical operations**.

Example of an ill-conditioned matrix:

✓ Hilbert Matrix, i.e., a matrix defined as $H_{ij} = 1/(i+j-1)$).

- Matrices with condition numbers near 1 are said to be ***well-conditioned***.
- Matrices with condition numbers ***much greater than one*** (such as around 10^5 for a 5x5 Hilbert matrix) are said to be ***ill-conditioned***.