# Interpolation and Curve fitting

Spring 2019

# Interpolation vs Curve fitting

Given some data points $\{x_i, y_i\}_{i=1}^n$ and assuming there is some function $f(x)$ describes the quantity of interest at all points.

**Interpolation**

Find a function satisfying

$$\boxed{P(x_i) = f(x_i), \quad i = 1, \ldots, n}$$

that allows us to approximate $f(x)$ such that the function values between the data sets may be estimated.

# Interpolation vs Curve fitting

Given some data points $\{x_i, y_i\}_{i=1}^{n}$ and assuming there is some function $f(x)$ describes the quantity of interest at all points.

**Interpolation**

Find a function satisfying

$$\boxed{P(x_i) = f(x_i), \quad i = 1, \ldots, n}$$

that allows us to approximate $f(x)$ such that the function values between the data sets may be estimated.

**Curve fitting**

Find a function that is a *good fit* to the original data points The function **does not** have to pass through the original data points.

# Piecewise Linear Interpolation

## Simplest case: 2 points

**Basic algebra problem**

Given two points $(x_1, f(x_1))$ and $(x_2, f(x_2))$, find $P_1(x)$ such that

$$P_1(x_1) = f(x_1)$$
$$P_2(x_2) = f(x_2)$$

$P_1$ can be written in the form

$$\boxed{P_1(x) = \left(\frac{x - x_2}{x_1 - x_2}\right) f(x_1) + \left(\frac{x - x_1}{x_2 - x_1}\right) f(x_2)}$$

# Interpolation in MATLAB `interp1`

```
vq = interp1(x,v,xq,method)
```

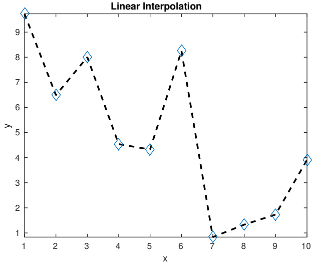
- `x` – sample points    `v` – values $f(x)$
- `xq` – query points on which the polynomial will be evaluated
- `method` – method of interpolation (e.g. `linear`, `cubic`, e.t.c)

# Interpolation in MATLAB `interp1`

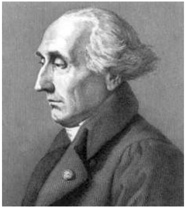
```
vq = interp1(x,v,xq,method)
```

- `x` – sample points   `v` – values $f(x)$
- `xq` – query points on which the polynomial will be evaluated
- `method` – method of interpolation (e.g. `linear`, `cubic`, e.t.c)

**Exercise:** Interpolate 10 random data points with values on $[0, 10]$ and evaluate the polynomial on `1:0.1:10` and plot.



# Higher order interpolation : Lagrange interpolant



Idea:

Given $n+1$ data points $\{(x_1, f(x_1)), (x_2, f(x_2)), \ldots, (x_n, f(x_n)), (x_{n+1}, f(x_{n+1}))\}$, define a polynomial

$$\boxed{P(x) = \sum_{i=1}^{n+1} f(x_j)\varphi_j(x)}$$

where

$$\varphi_j(x_i) = \delta_{ij} = \begin{cases} 0 & \text{if } j = i \\ 1 & \text{if } j \neq i \end{cases}$$

# What does $\varphi(x)$ look like?

- In the linear case:

$$\boxed{P_1(x) = \left(\frac{x - x_2}{x_1 - x_2}\right) f(x_1) + \left(\frac{x - x_1}{x_2 - x_1}\right) f(x_2)}$$

# What does $\varphi(x)$ look like?

- In the linear case:

$$\boxed{P_1(x) = \left(\frac{x - x_2}{x_1 - x_2}\right) f(x_1) + \left(\frac{x - x_1}{x_2 - x_1}\right) f(x_2)}$$

- Notice that:

$$P_1(x_1) = \underbrace{\left(\frac{x_1 - x_2}{x_1 - x_2}\right)}_{1} f(x_1) + \underbrace{\left(\frac{x_1 - x_1}{x_2 - x_1}\right)}_{0} f(x_2) = f(x_1)$$

# What does $\varphi(x)$ look like?

- In the linear case:

$$\boxed{P_1(x) = \left(\frac{x - x_2}{x_1 - x_2}\right) f(x_1) + \left(\frac{x - x_1}{x_2 - x_1}\right) f(x_2)}$$

- Notice that:

$$P_1(x_1) = \cancelto{1}{\left(\frac{x_1 - x_2}{x_1 - x_2}\right)} f(x_1) + \cancelto{0}{\left(\frac{x_1 - x_1}{x_2 - x_1}\right)} f(x_2) = f(x_1)$$

$$P_1(x_2) = \cancelto{0}{\left(\frac{x_2 - x_2}{x_1 - x_2}\right)} f(x_1) + \cancelto{1}{\left(\frac{x_2 - x_1}{x_2 - x_1}\right)} f(x_2) = f(x_2)$$

# What does $\varphi(x)$ look like?

- Therefore:

$$P_1(x) = \varphi_1(x)f(x_1) + \varphi_2(x)f(x_2)$$

where

$$\boxed{\varphi_1(x) = \left(\frac{x - x_2}{x_1 - x_2}\right), \quad \varphi_2(x) = \left(\frac{x - x_1}{x_2 - x_1}\right)}$$

- In the case of 3 data points $\{(x_1, f(x_1)), (x_2, f(x_2)), (x_3, f(x_3))\}$

$$\boxed{P_2(x) = \varphi_1(x)f(x_1) + \varphi_2(x)f(x_2) + \varphi_3(x)f(x_3)}$$

# What does $\varphi(x)$ look like?

- Therefore:

$$P_1(x) = \varphi_1(x) f(x_1) + \varphi_2(x) f(x_2)$$

where

$$\boxed{\varphi_1(x) = \left(\frac{x - x_2}{x_1 - x_2}\right), \quad \varphi_2(x) = \left(\frac{x - x_1}{x_2 - x_1}\right)}$$

- In the case of 3 data points $\{(x_1, f(x_1)), (x_2, f(x_2)), (x_3, f(x_3))\}$

$$\boxed{P_2(x) = \varphi_1(x) f(x_1) + \varphi_2(x) f(x_2) + \varphi_3(x) f(x_3)}$$

where:

$$\boxed{\varphi_1(x) = \frac{(x - x_2)(x - x_3)}{(x_1 - x_2)(x_1 - x_3)}, \quad \varphi_2(x) = \frac{(x - x_1)(x - x_3)}{(x_2 - x_1)(x_2 - x_3)}, \quad \varphi_3(x) = \frac{(x - x_1)(x - x_2)}{(x_3 - x_1)(x_3 - x_2)}}$$

# What does $\varphi(x)$ look like?

- Therefore:

$$P_1(x) = \varphi_1(x)f(x_1) + \varphi_2(x)f(x_2)$$

where

$$\boxed{\varphi_1(x) = \left(\frac{x - x_2}{x_1 - x_2}\right), \quad \varphi_2(x) = \left(\frac{x - x_1}{x_2 - x_1}\right)}$$

- In the case of 3 data points $\{(x_1, f(x_1)), (x_2, f(x_2)), (x_3, f(x_3))\}$

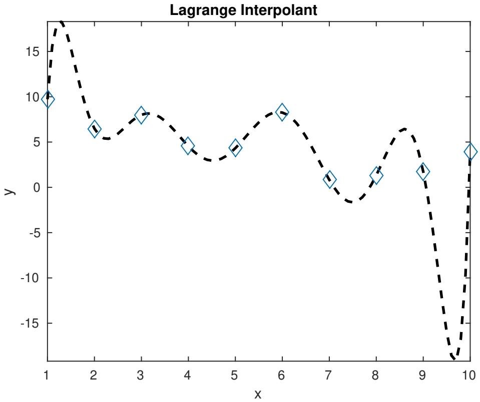
$$\boxed{P_2(x) = \varphi_1(x)f(x_1) + \varphi_2(x)f(x_2) + \varphi_3(x)f(x_3)}$$

where:

$$\boxed{\varphi_1(x) = \frac{(x - x_2)(x - x_3)}{(x_1 - x_2)(x_1 - x_3)}, \quad \varphi_2(x) = \frac{(x - x_1)(x - x_3)}{(x_2 - x_1)(x_2 - x_3)}, \quad \varphi_3(x) = \frac{(x - x_1)(x - x_2)}{(x_3 - x_1)(x_3 - x_2)}}$$

*In the general case, given $n + 1$ data points we can write down the Lagrange interpolating polynomial of degree $n$ of this form*

# Lagrange interpolating polynomial




**Disadvantage**

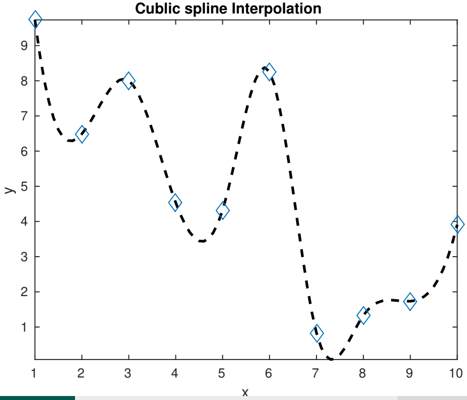
- As the polynomial degree increases, the error near the endpoints increases (MA 427)

# Cubic spline interpolants

A **cubic spline** interpolant is a piecewise cubic that interpolates the function $f$ at $x_1, x_2, x_3, \ldots, x_n$ and has two continuous derivatives.

**Why Splines?**

- Useful in applications that require smooth surfaces, e.g. design of airplane parts or cars.



# Least Squares curve fitting

- A least squares curve fit can be used to obtain a curve such that the squared distance from each point to the curve is minimized.

# Least Squares curve fitting

- A least squares curve fit can be used to obtain a curve such that the squared distance from each point to the curve is minimized.
- The curve can be
  1. Polynomials of degree $n$
  2. Trigonometric
  3. Exponential

# Linear Regression

- Given $N$ data points $(x_k, y_k), \quad k = 1, \ldots, N$ we seek a line $f(x) = mx + b$ so that the **mean square error** (MSE):

$$\begin{aligned} MSE(m,b) &= \frac{1}{N} \sum_{k=1}^{N} \Big( y_k - f(x_k) \Big)^2 \\ &= \frac{1}{N} \sum_{k=1}^{N} \Big( y_k - (mx_k + b) \Big)^2 \end{aligned}$$

- We can compute **partial derivatives** with respect to $m$ and $b$ to find values of $m$ and $b$ that minimize the error.

# Polynomial Regression

- Define

$$\boxed{f(x) = a_1 x^n + a_2 x^{n-1} + \cdots + a_n x + a_{n+1}}$$

that fits the data.

# Polynomial Regression

- Define
  $$\boxed{f(x) = a_1 x^n + a_2 x^{n-1} + \cdots + a_n x + a_{n+1}}$$
  that fits the data.
- In MATLAB, the function `polyfit` performs polynomial regression

## Polynomial Regression

- Define

$$f(x) = a_1 x^n + a_2 x^{n-1} + \cdots + a_n x + a_{n+1}$$

that fits the data.

- In MATLAB, the function `polyfit` performs polynomial regression
- Usage:

```
a = polyfit(X,Y,N)
```

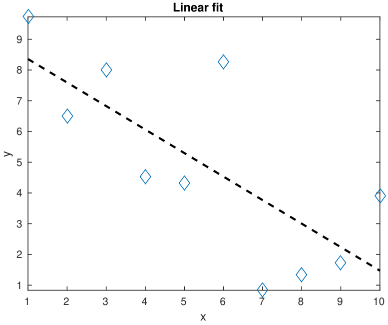
1. a - row vector of coefficients of the regression, i.e.
2. `X` and `Y` - supplied data points
3. `N` - polynomial degree

```
a(1)*X^N + a(2)*X^(N-1)+. . .+ a(N)*X + a(N+1)
```

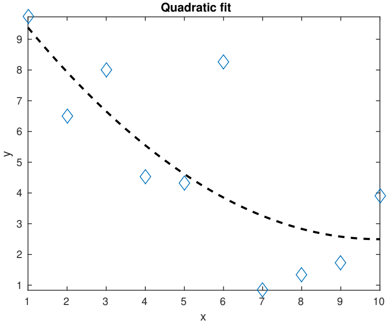
- To evaluate the polynomial, a at any query points `xq`

```
Y = polyval(a,xq)
```

# Linear fit

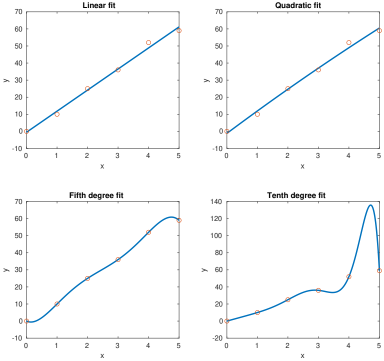


# Quadratic fit



# Exercise

Generate the following plot for `x =0:5;` and `y=[0 10 25 36 52 59];`



# Interpolation in 2D

```
Vq = interp2(X,Y,V,Xq,Yq,method)
```

- Returns `Vq`, the values of a function of two variables at query points `Xq,Yq`

# Interpolation in 2D

```
Vq = interp2(X,Y,V,Xq,Yq,method)
```

- Returns `Vq`, the values of a function of two variables at query points `Xq,Yq`
- The default method is linear interpolation

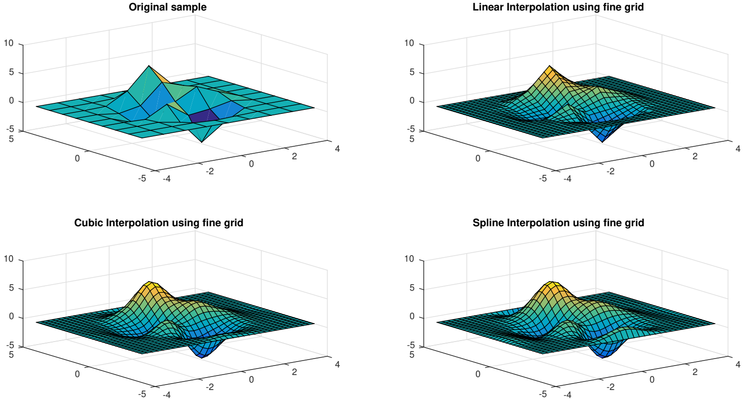
# Interpolation in 2D

```
Vq = interp2(X,Y,V,Xq,Yq,method)
```

- Returns `Vq`, the values of a function of two variables at query points `Xq,Yq`
- The default method is linear interpolation
- Other methods include
  1. `cubic`
  2. `spline`

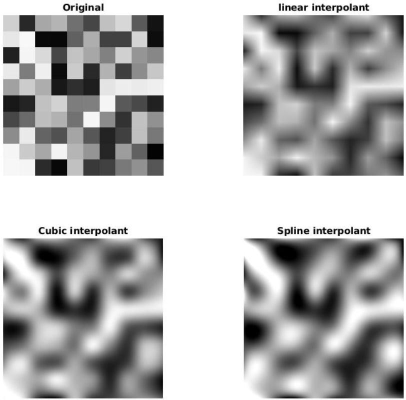
# Sampling and interpolating a Gaussian distribution

- `Z = peaks(X,Y)` – evaluates the `peaks` function at `X` and `Y`.
- We can use `Z` as data points and visualize interpolation using various methods



# Interpolation and image manipulation

- Starting with a random image `pic = rand(10,10);`
- Interpolate the image using 64 times as many points in each direction.



# Extrapolation

**Definition**

Making estimates beyond the obsevation range.

- 1D – `vq = interp1(x,v,xq,method,extrapolation)` allows `xq` to contain points outside the range of `x`.
- 2D – `q = interp2(X,Y,V,Xq,Yq,method,extrapval)` allows for a scalar value `extrapval` to be assigned to all queries outside the domain of sample points.