Scientific Computing

Maastricht Science Program

Week 3

Frans Oliehoek
<frans.oliehoek@maastrichtuniversity.nl>

Recap

- What is scientific programming?
- Programming
- Finding the zeros of non-linear equations.
- Floating Point Numbers
- Computation: Errors & Cost
- Linear Algebra
  - Vectors & Matrices in Matlab
  - Systems of Linear Equations

This Lecture

- Given data: figure out how variables relate.
  - E.g., given medical symptoms or measurements, what is the probability of some disease?
- Estimating functions from a number of data points.
  - Interpolation
  - Least Squares Regression
- Replacing a complex function by a simpler one

- All this is based on solving linear systems!
  - GEM, LU factorization

Approximations of Functions

- Function approximation:
  Replace a function by a simpler one

- Reasons:
  - Integration: replace a complex function with one that is easy to integrate.
  - Function may be very complex: e.g. result of simulation.
  - Function may be unknown...

“Approximation of Data”

- 'the function unknown'
  - it is only known at certain points $(x_{0,}y_0),(x_{1,}y_1),\ldots,(x_n,y_n)$
  - but we also want the know at other points
  - these points are called the data → “approximation of data”
- Interpolation:
  - find a function that goes exactly through data point
- Regression:
  - find a function that minimizes some error measure
  - better for noisy data.
- Related terms: curve fitting, extrapolation, classification

“Approximation of Data”

- 'the function unknown'
  - it is only known at certain points $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$
  - but we also want the know at other points
  - these points are called the data → “ap
- Interpolation:
  - find a function that goes exactly through data point
- Regression:
  - find a function that minimizes some error measure
  - better for noisy data.
- Related terms: curve fitting, extrapolation, classification

number of data points:

$$N = n + 1$$

Interpolation

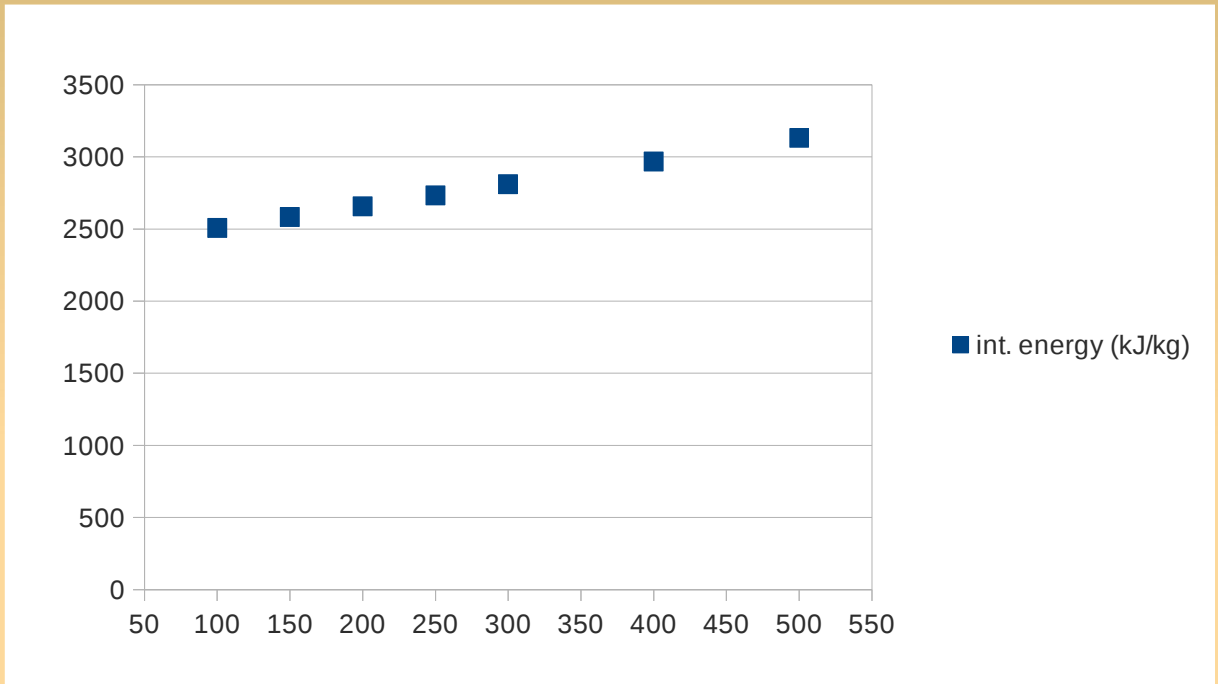
- In the study of Geysers, an important quantity is the internal energy of steam.

Temp. (Celsius)	int. energy (kJ/kg)
100	2506.7
150	2582.8
200	2658.1
250	2733.7
300	2810.4
400	2967.9
500	3131.6

(from Etter, 2011, Introduction to MATLAB)

Temperature Example

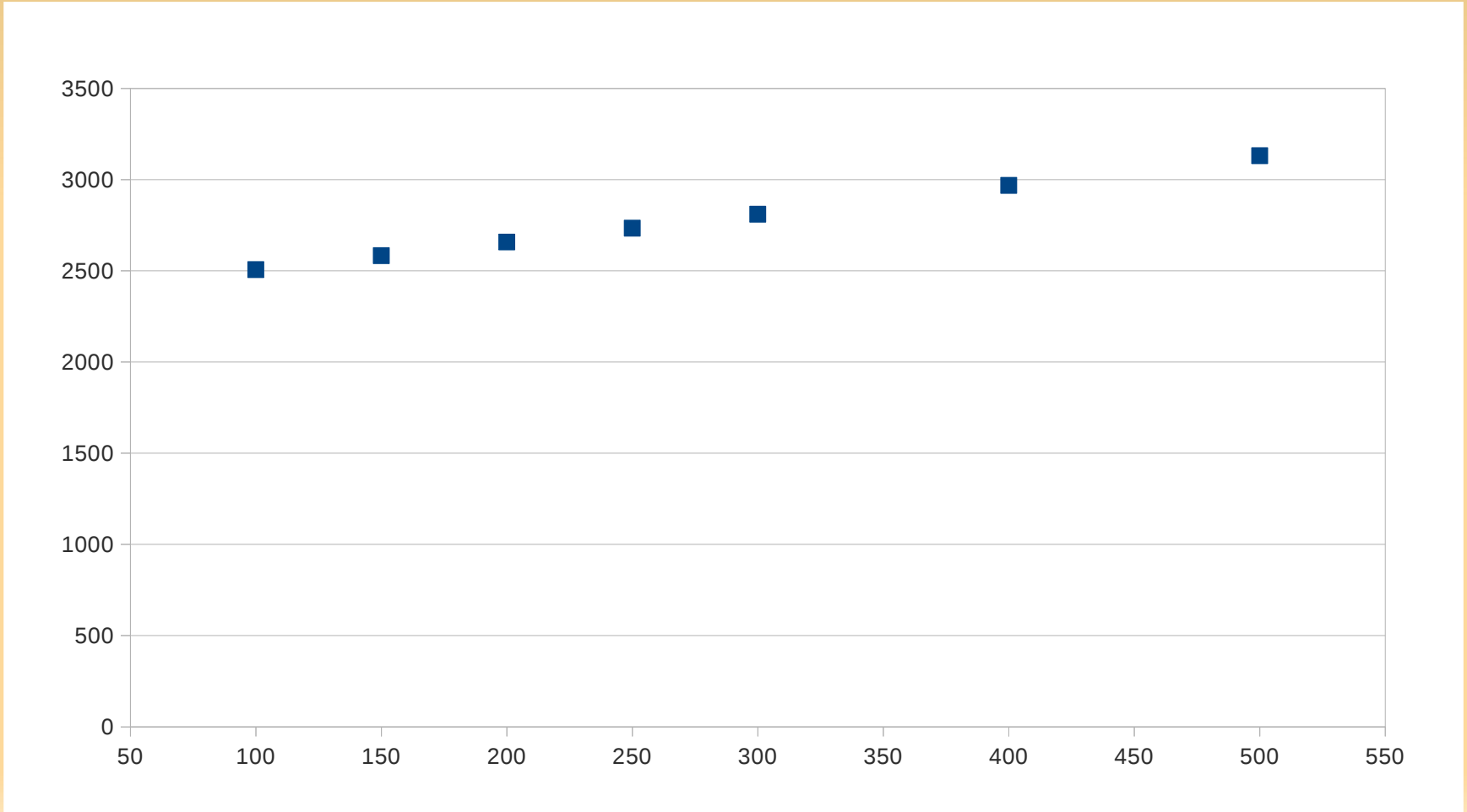
Temp. (Celsius)	int. energy (kJ/kg)
100	2506.7
150	2582.8
200	2658.1
250	2733.7
300	2810.4
400	2967.9
500	3131.6



- Now we want to know the temp. at 430°C...

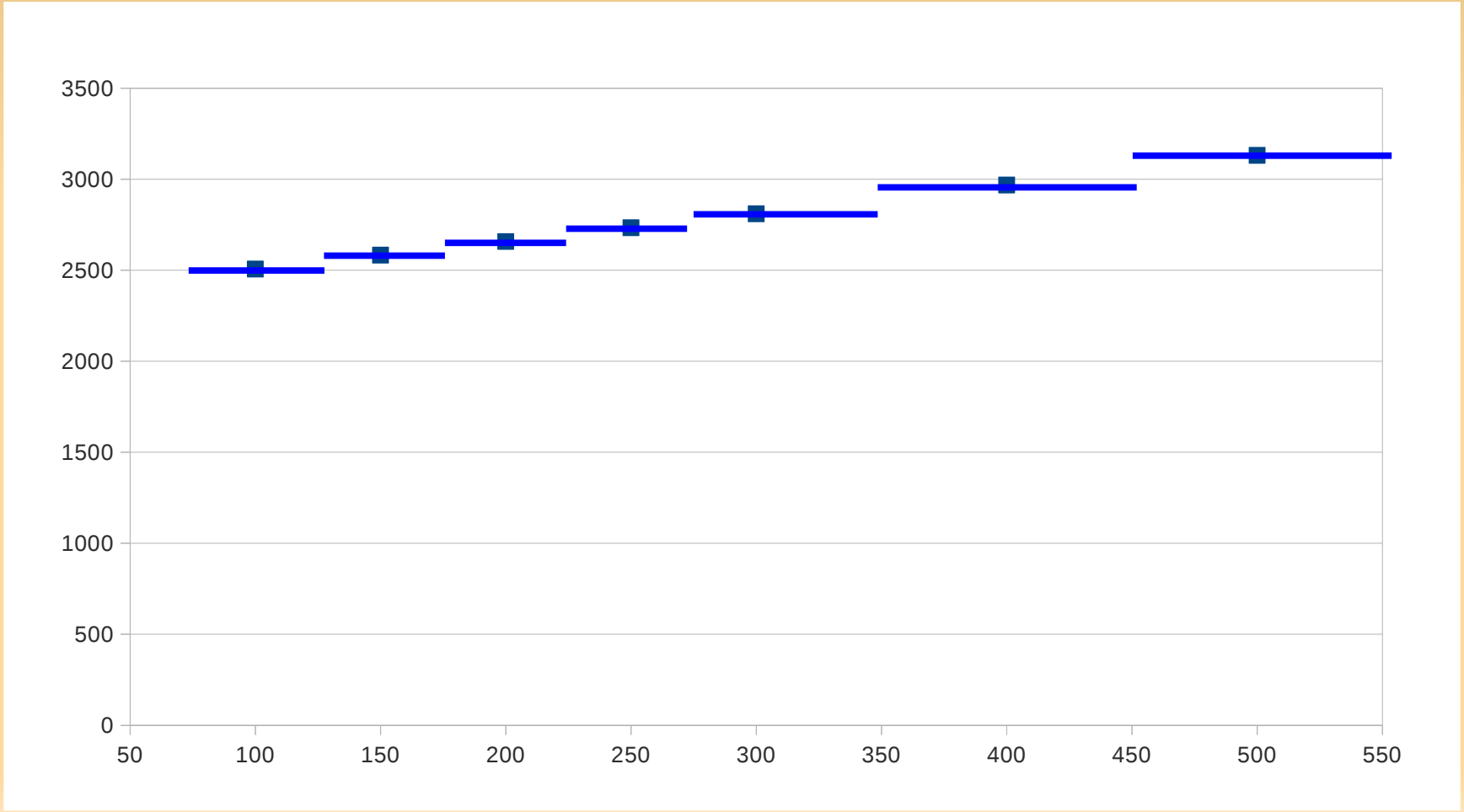
Piecewise Constant Interpolation

- Interpolation: define a function that goes through data
- Piecewise interpolation: use a piecewise function



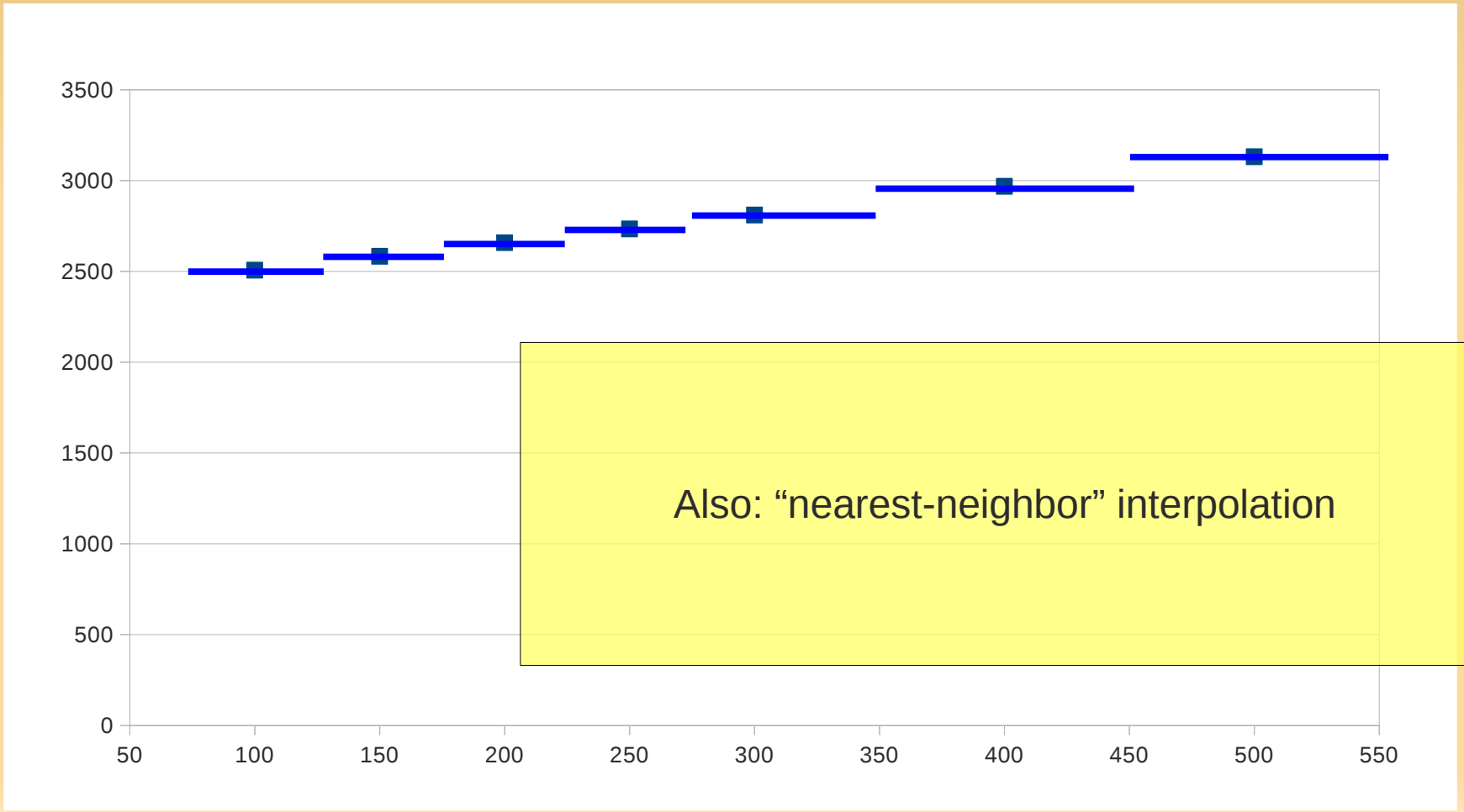
Piecewise Constant Interpolation

- Interpolation: define a function that goes through data
- Piecewise interpolation: use a piecewise function



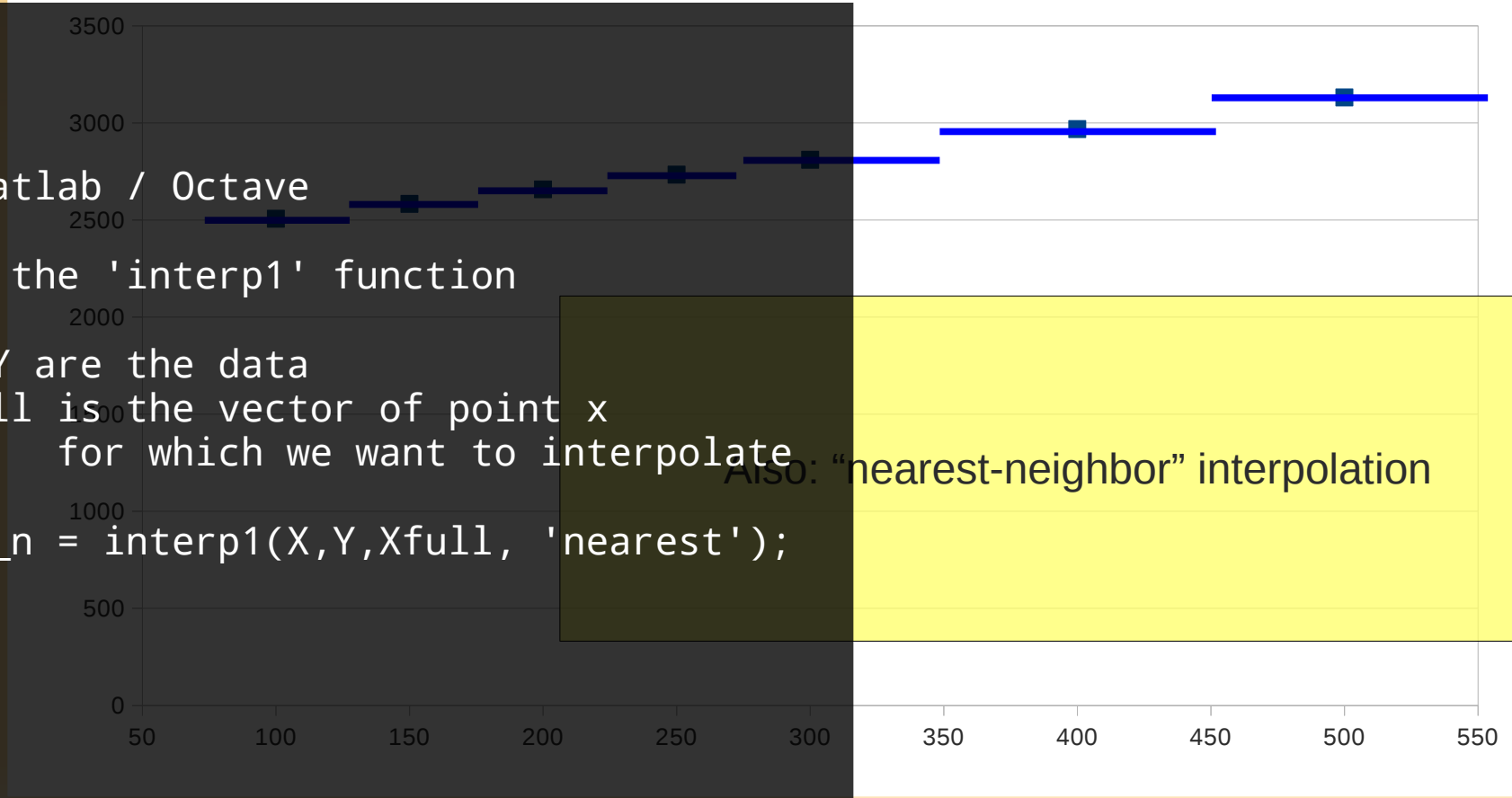
Piecewise Constant Interpolation

- Interpolation: define a function that goes through data
- Piecewise interpolation: use a piecewise function



Piecewise Constant Interpolation

- Interpolation: define a function that goes through data
- Piecewise interpolation: use a piecewise function



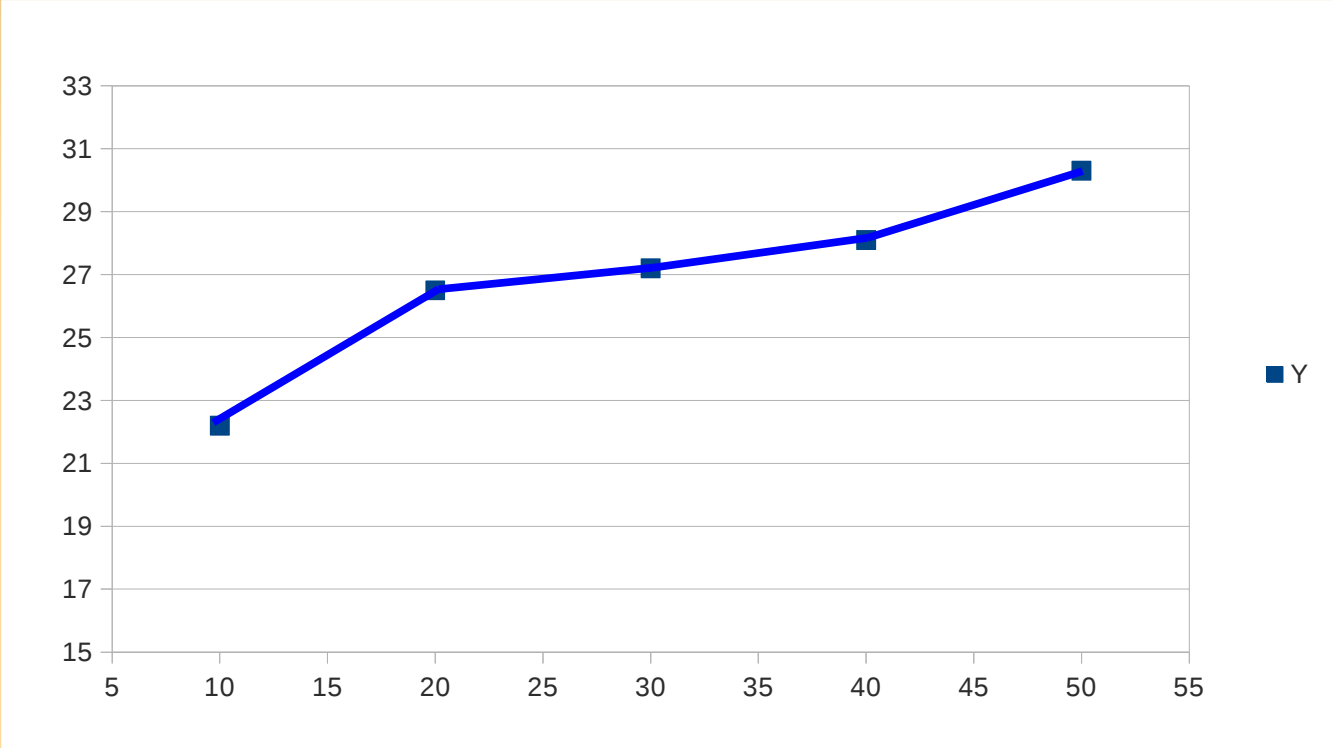

```
%In Matlab / Octave
%
% use the 'interp1' function
%
% X, Y are the data
% Xfull is the vector of point x
%       for which we want to interpolate

Yfull_n = interp1(X,Y,Xfull, 'nearest');
```

Also: “nearest-neighbor” interpolation

Piecewise Linear Interpolation

- Piecewise linear interpolation:
  just connect the data point with lines

X	Y
10	22.2
20	26.5
30	27.2
40	28.1
50	30.3

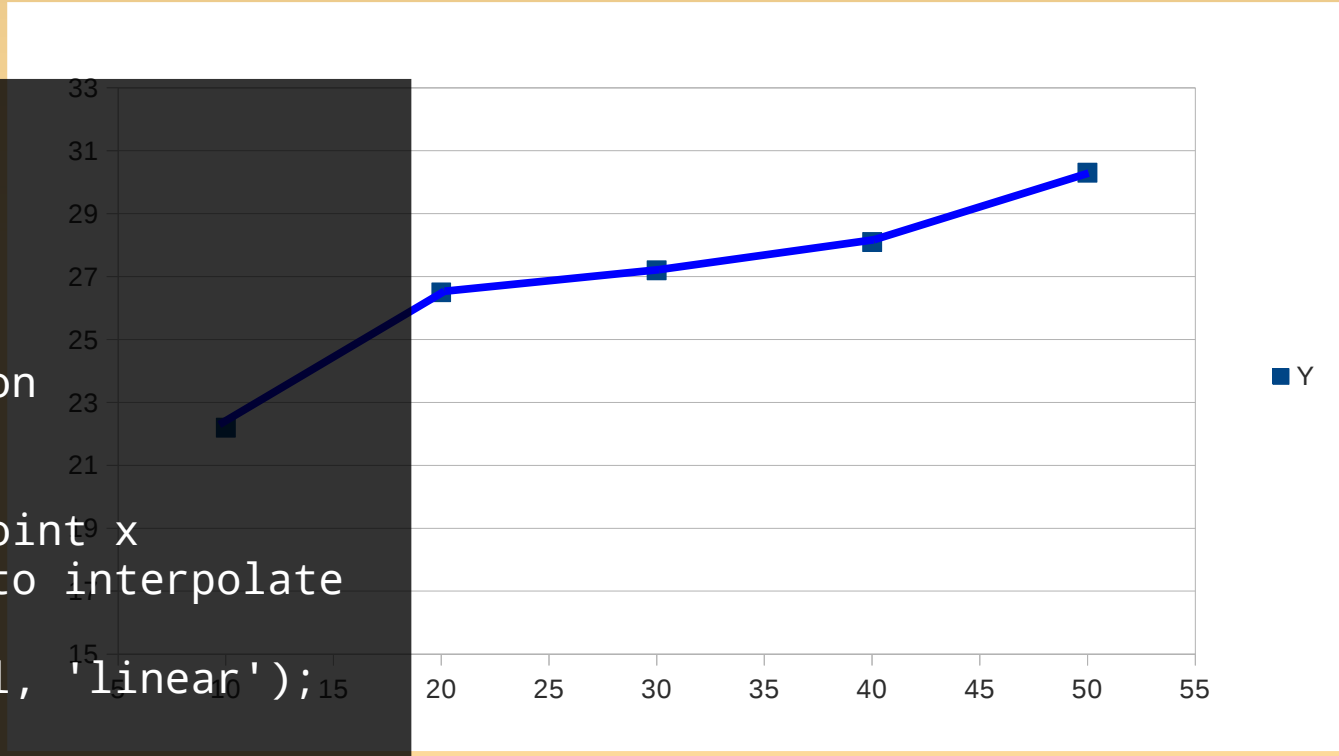


Piecewise Linear Interpolation

- Piecewise linear interpolation:
  just connect the data point with lines

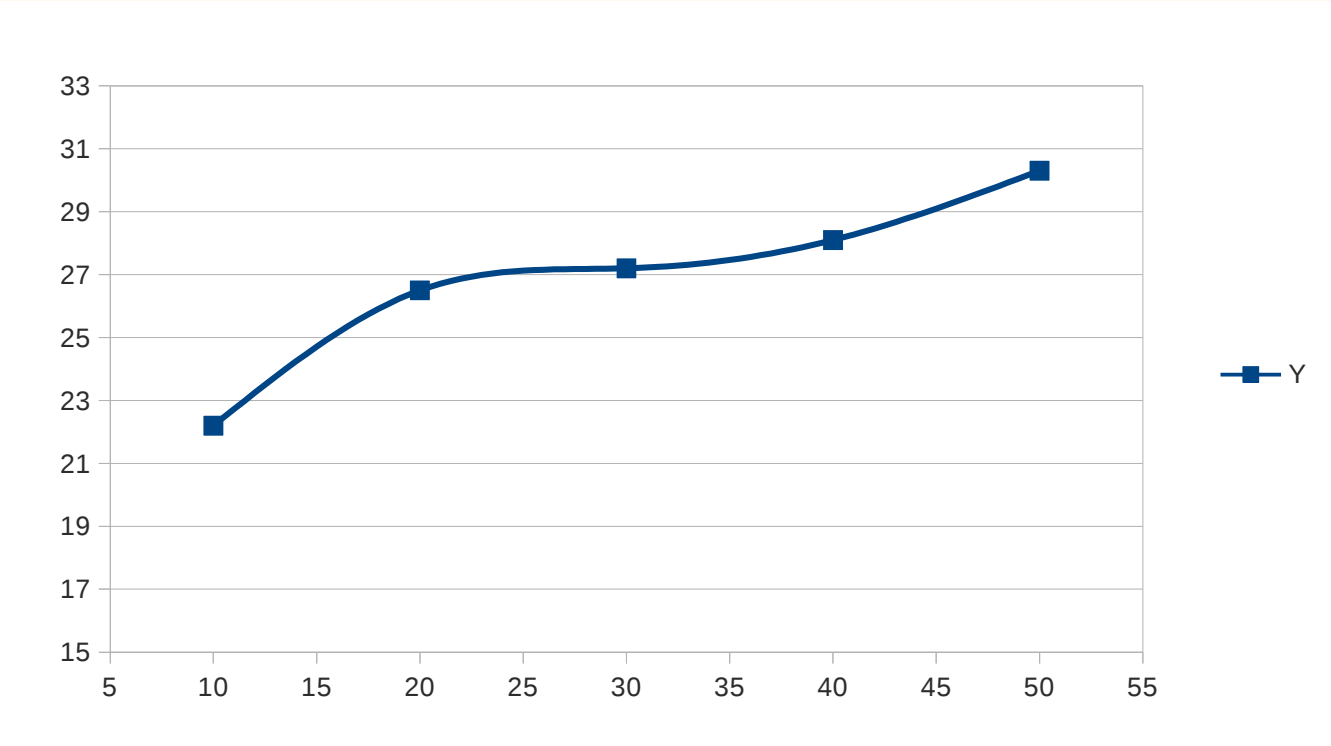
X	Y
10	22.2
20	26.5
30	27.2
40	28.1
50	30.3

```
%In Matlab / Octave
%
% use the 'interp1' function
%
% X, Y are the data
% Xfull is the vector of point x
%       for which we want to interpolate

Yfull_n = interp1(X,Y,Xfull, 'linear');
```



Cubic Splines Interpolation

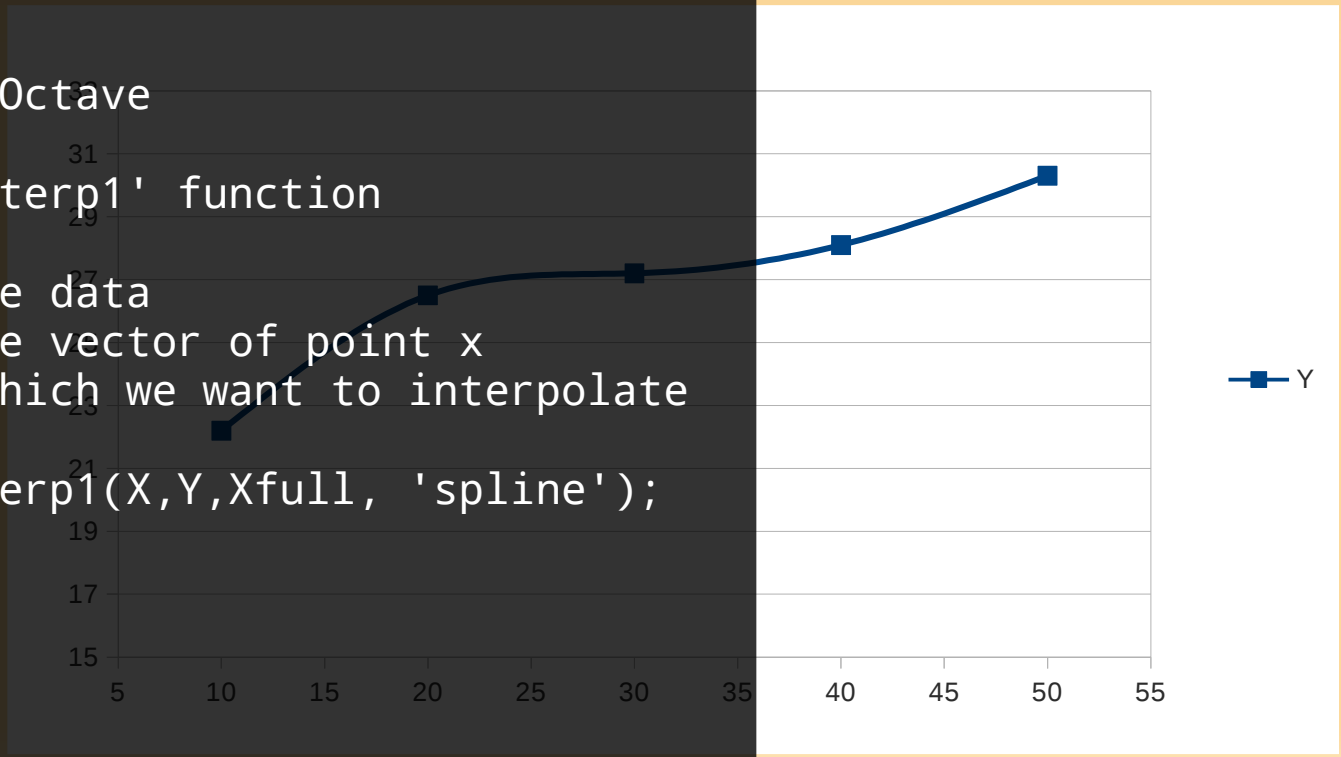
- **cubic-spline** interpolation
  - connect the data point smooth curves (third degree polynomials)
  - still piecewise



Cubic Splines Interpolation

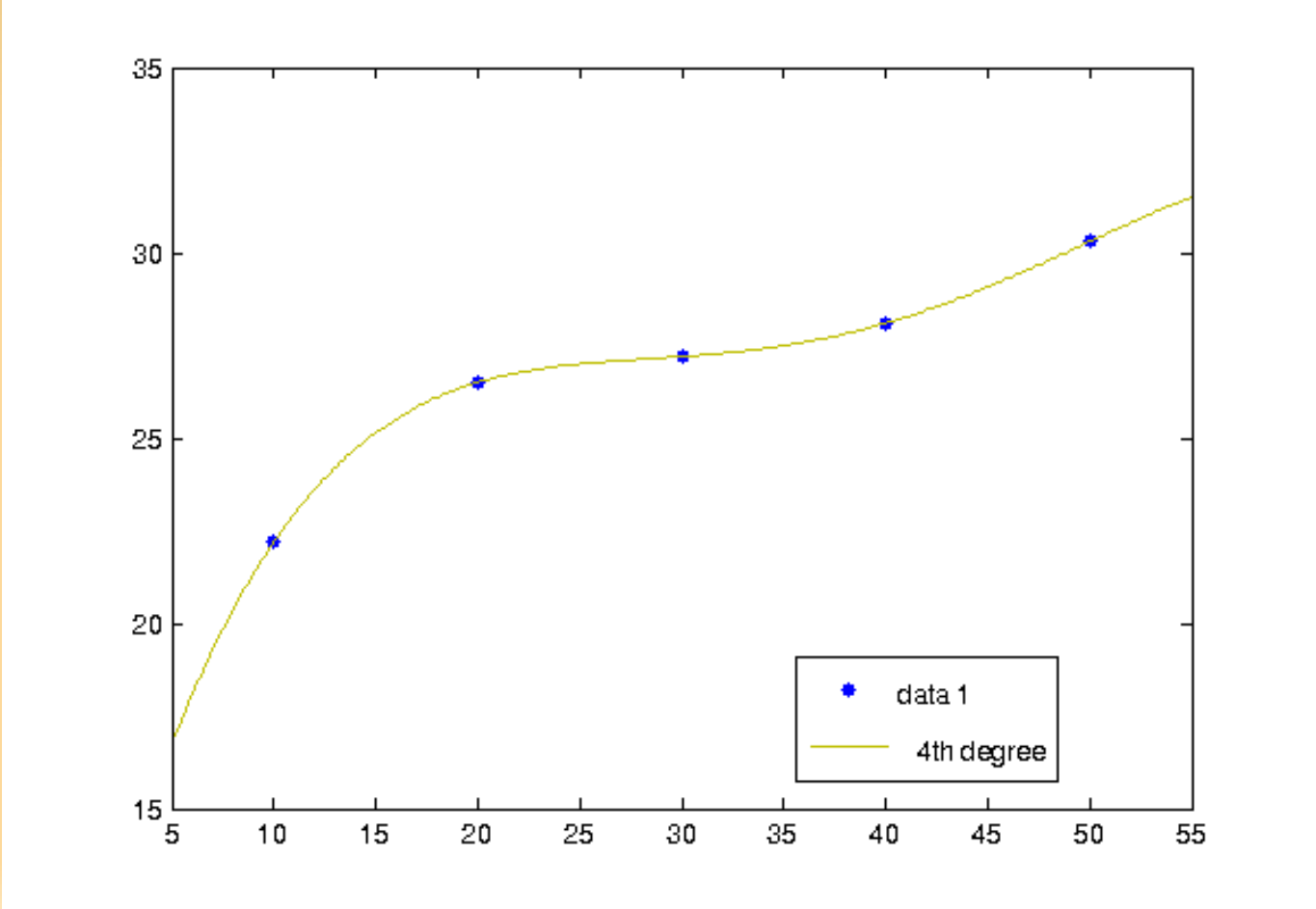
- **cubic-spline** interpolation
  - connect the data point smooth curves (third degree polynomials)
    - still piecewise

```
%In Matlab / Octave
%
% use the 'interp1' function
%
% X, Y are the data
% Xfull is the vector of point x
%       for which we want to interpolate

Yfull_n = interp1(X,Y,Xfull, 'spline');
```



Non-Piecewise Interpolation

- So far: piecewise
- but may want to find a single (non-piecewise) function.



Polynomial Interpolation

- Polynomial interpolation: fit a polynomial

Polynomial Interpolation

- Polynomial interpolation: fit a polynomial

(Prop. 3.1)
given a set of $N = n+1$ data points $(x_{0,} y_0), (x_{1,} y_1), \ldots, (x_n, y_n)$

→ There exist a unique polynomial

$$\Pi_n(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

(of degree n or less) that goes exactly through the points!

- “The interpolating polynomial” (of the 'data' or 'function')

Polynomial Interpolation

- Polynomial interpolation: fit a polynomial

(Prop. 3.1)
given a set of $N = n+1$ data points $(x_{0,} y_0), (x_{1,} y_1), \ldots, (x_n, y_n)$

→ There exist a unique polynomial

$$\Pi_n(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

(of degree n or less) that goes exactly through the points!

- "The interpolating polynomial" (of the 'data' or 'function')

So this is good news -
we can always find such a function.

Uniqueness of the Interpolating polynomial

- Why is this polynomial unique?

Uniqueness of the Interpolating polynomial

- Why is this polynomial unique?
- Suppose not unique: both $\Pi_n(x), \Pi_n'(x)$ perfectly fit the data

  → $\Pi_n(x) - \Pi_n'(x) = 0$ for all the N=n+1 data points
- That is it
  - “vanishes at n+1 points”
  - “has n+1 roots”

- But: a polynomial of degree n has at most n roots!

  → contradiction!

Finding the Interpolating Polynomial

- How do we find it?

Finding the Interpolating Polynomial

- How do we find it?
- The data points define a system of linear equations!

$$a_0 + a_1 x_0 + a_2 x_0^2 + \ldots + a_n x_0^n = y_0$$
$$a_0 + a_1 x_1 + a_2 x_1^2 + \ldots + a_n x_1^n = y_1$$
$$\ldots$$
$$a_0 + a_1 x_n + a_2 x_n^2 + \ldots + a_n x_n^n = y_n$$

Finding the Interpolating Polynomial

- How do we find it?
- The data points define a system of linear equations!

$$a_0 + a_1 x_0 + a_2 x_0^2 + \ldots + a_n x_0^n = y_0$$
$$a_0 + a_1 x_1 + a_2 x_1^2 + \ldots + a_n x_1^n = y_1$$
$$\ldots$$
$$a_0 + a_1 x_n + a_2 x_n^2 + \ldots + a_n x_n^n = y_n$$

$$Da = \begin{bmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ 1 & x_1 & x_1^2 & \ldots & x_1^n \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 1 & x_n & x_n^2 & \ldots & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \ldots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \ldots \\ y_n \end{bmatrix}$$

Finding the Interpolating Polynomial

- How do we find it?
- The data points define a system of linear equations!

$$a_0 + a_1 x_0 + a_2 x_0^2 + \ldots + a_n x_0^n = y_0$$
$$a_0 + a_1 x_1 + a_2 x_1^2 + \ldots + a_n x_1^n = y_1$$
$$\ldots$$
$$a_0 + a_1 x_n + a_2 x_n^2 + \ldots + a_n x_n^n = y_n$$

$$Da = \begin{bmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ 1 & x_1 & x_1^2 & \ldots & x_1^n \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 1 & x_n & x_n^2 & \ldots & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \ldots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \ldots \\ y_n \end{bmatrix}$$

Note:
a is the unknown!

Finding the Interpolating Polynomial

Last week:

$$Ax = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \dots & \dots & \dots \\ a_{m1} & \dots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ \dots \\ x_n \end{bmatrix} = \begin{bmatrix} c_1 \\ \dots \\ c_m \end{bmatrix}$$

…near equations!

$$Da = \begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \dots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \dots \\ y_n \end{bmatrix}$$

Note:
a is the unknown!

Finding the Interpolating Polynomial

Last week:

$$Ax = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \dots & \dots & \dots \\ a_{m1} & \dots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ \dots \\ x_n \end{bmatrix} = \begin{bmatrix} c_1 \\ \dots \\ c_m \end{bmatrix}$$

near equations!

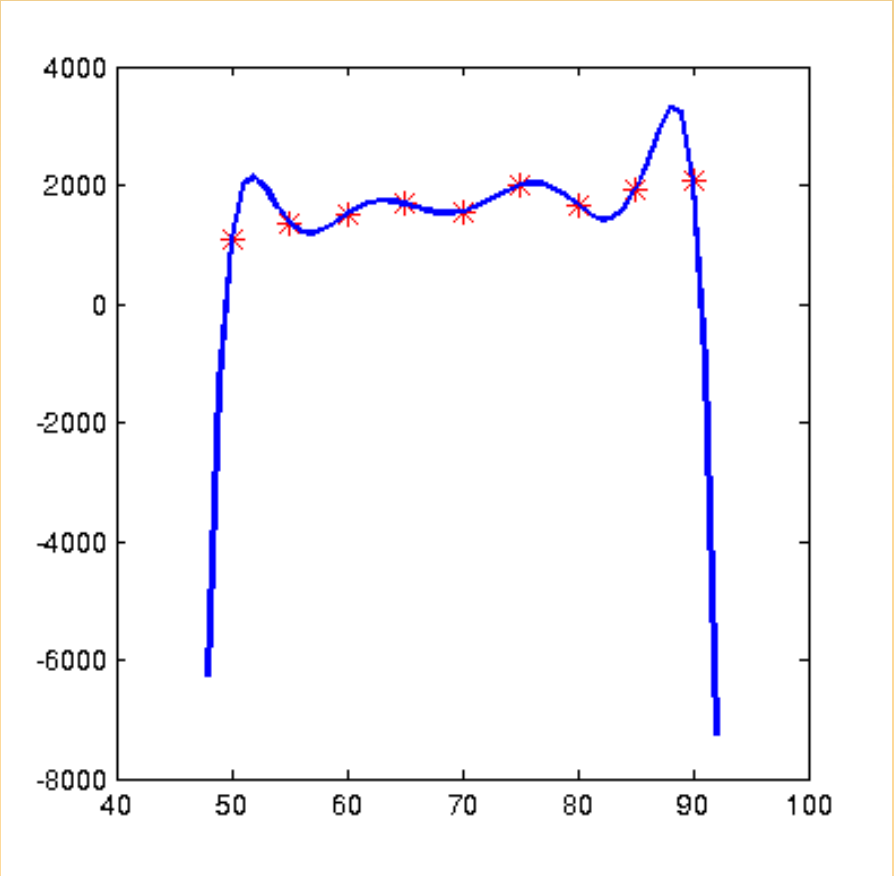
$$Da = \begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \dots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \dots \\ y_n \end{bmatrix}$$

```
%in matlab...
...
% fit a polynomial of degree N
% (X,Y are the N data points
%  c are the coefficients of
%    the polynomial)
c = polyfit(X,Y,N-1);

% now compute the Y values at the
% points X that we want to interpolate
% at:
Yfull_p = polyval(c, Xfull);

% and plot them
plot(Xfull, Yfull_p, '.mo');
```

Note:
a is the unknown!

Limits of Polynomial Interpolation

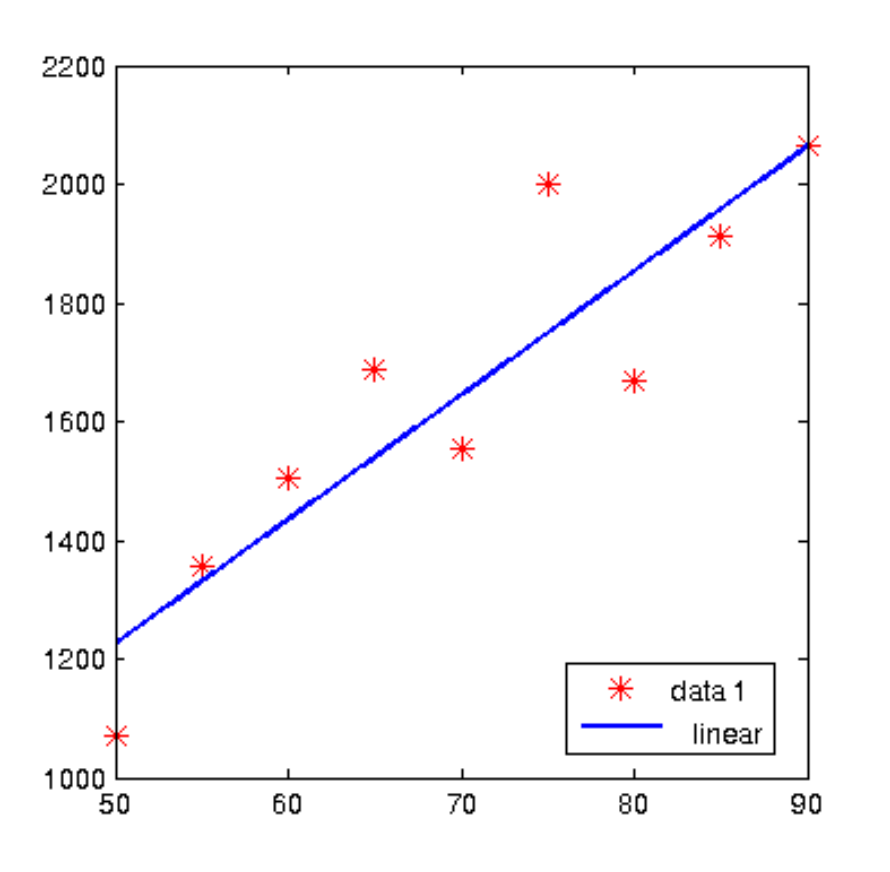
- Does not work very well when N is large.
- Is not very suitable if the data is obtained from noisy measurements.
- “Runge's phenomenon”
- In this case, we would perhaps want to fit a straight line.



Least-Squares Method

- In cases that we made noisy measurements, we don't want to exactly fit the data.

- That is: fit a polynomial** of degree p < n
  - can still use 'polyfit'



** or other function

Least-squares Method - 2

- Mathematically...
- polynomial interpolation:

$$Da = \begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \dots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \dots \\ y_n \end{bmatrix}$$

Least-squares Method - 2

- Mathematically...
- polynomial interpolation:

$$Da = \begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \dots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \dots \\ y_n \end{bmatrix}$$

- Now: (p=2)

$$Da = \begin{bmatrix} 1 & x_0 & x_0^2 \\ 1 & x_1 & x_1^2 \\ \dots & \dots & \dots \\ 1 & x_n & x_n^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \dots \\ y_n \end{bmatrix}$$

Least-squares Method - 2

Problem: No solution!

- D is not square → no inverse exists
- overdetermined system of equations!

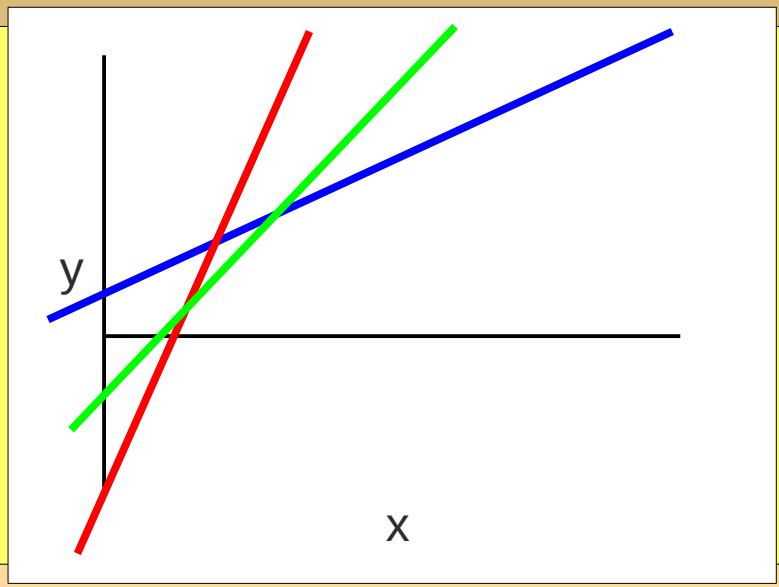
- Now: (p=2)

$$Da = \begin{bmatrix} 1 & x_0 & x_0^2 \\ 1 & x_1 & x_1^2 \\ \dots & \dots & \dots \\ 1 & x_n & x_n^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \dots \\ y_n \end{bmatrix}$$

Least-squares Method - 2

Problem: No solution!

- D is not square → no inverse exists
- overdetermined system of equations!



- Now: (p=2)

$$Da = \begin{bmatrix} 1 & x_0 & x_0^2 \\ 1 & x_1 & x_1^2 \\ \dots & \dots & \dots \\ 1 & x_n & x_n^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \dots \\ y_n \end{bmatrix}$$

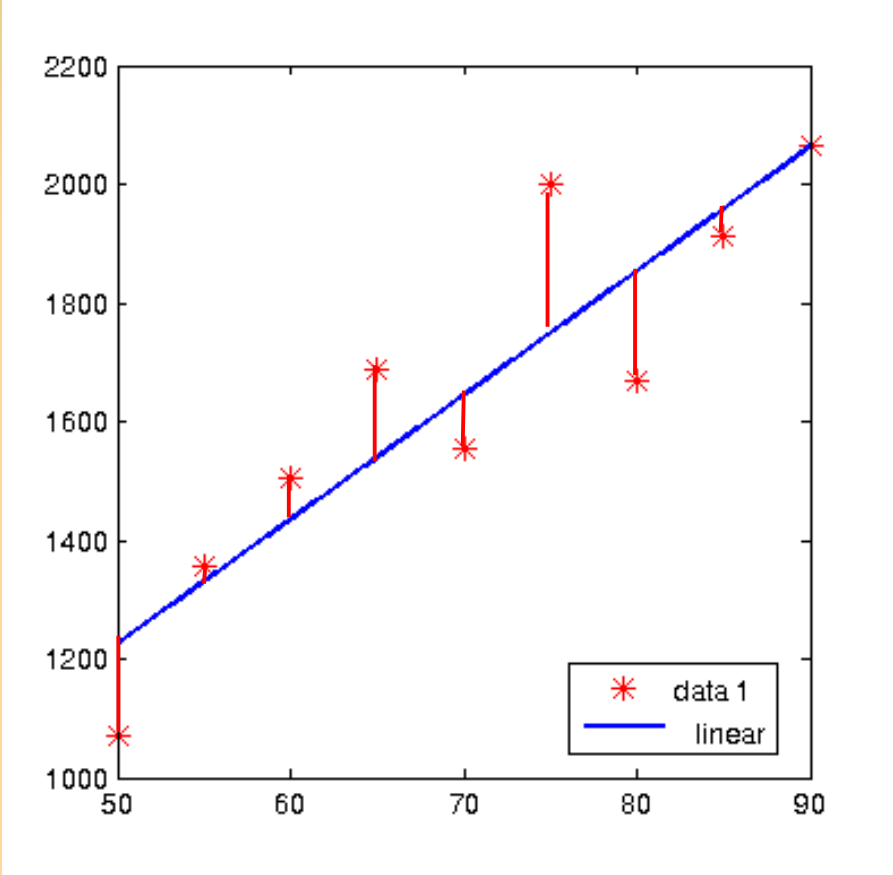
Least-squares Method - 3

- Common approach:
  minimize sum of the squares of the errors

$$\tilde{f}(x) = a_0 + a_1 x$$

$$SSE(\tilde{f}) = \sum_{i=0}^{n} [\tilde{f}(x_i) - y_i]^2$$

- pick the $\tilde{f}$ with min. SSE

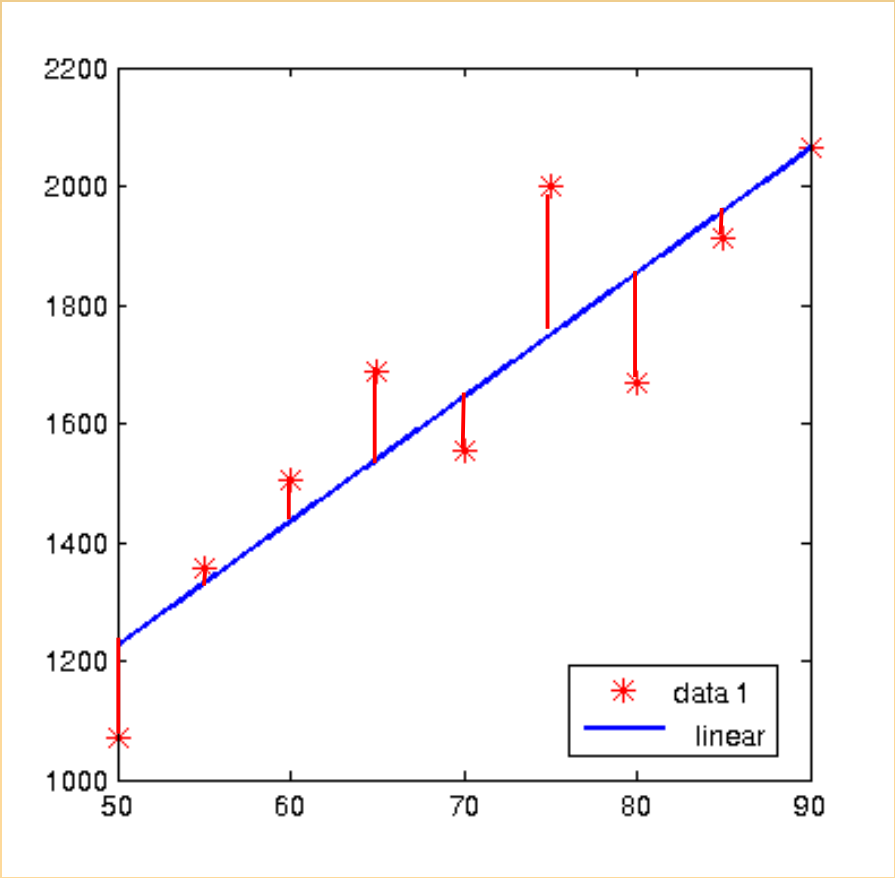


Least-squares Method - 3

- How to find it?
- minimize SSE
  - differentiate and equate to 0 to find minimum.
- Can show that solution *a* satisfies the *normal equations:*

$$D^T D a = D^T y$$

$$SSE(\tilde{f}) = \sum_{i=0}^{n} [\tilde{f}(x_i) - y_i]^2$$

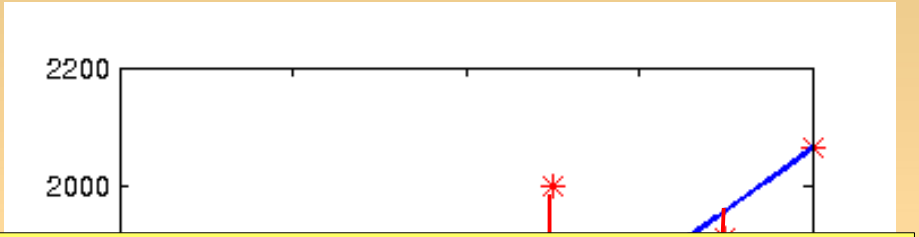


Least-squares Method - 3

- How to find it?
- minimize SSE
  - differentiate and equate to 0 to find minimum.
- Can show that solution *a* satisfies the *normal equations:*

$$D^T D a = D^T y$$

$$SSE(\tilde{f}) = \sum_{i=0}^{n} [\tilde{f}(x_i) - y_i]^2$$



$$Da = \begin{bmatrix} 1 & x_0 & x_0^2 \\ 1 & x_1 & x_1^2 \\ \dots & \dots & \dots \\ 1 & x_n & x_n^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \dots \\ y_n \end{bmatrix}$$

Least-squares Method - 4

- Solving normal equations

$$D^T D a = D^T y$$

- Invert: $a = (D^T D)^{-1} D^T y$

$$Da = \begin{bmatrix} 1 & x_0 & x_0^2 \\ 1 & x_1 & x_1^2 \\ \dots & \dots & \dots \\ 1 & x_n & x_n^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \dots \\ y_n \end{bmatrix}$$

- Solve the system of equations using left division:
  a = (D'D) \ (D'y)

Least-squares Method - 4

- Solving normal equations

$$D^T D a = D^T y$$

- Invert: $a = (D^T D)^{-1} D^T y$

$$Da = \begin{bmatrix} 1 & x_0 & x_0^2 \\ 1 & x_1 & x_1^2 \\ \dots & \dots & \dots \\ 1 & x_n & x_n^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \dots \\ y_n \end{bmatrix}$$

- Solve the system of equations using left division:
  a = (D'D) \ (D'y)

The nice thing: can use '\' directly!
a = D\y

(Remember: “careful! Will also find a solution if none exists!”)

Generalization: basis functions

- The matrix *D* is called the *design matrix.*

$$Da = \begin{bmatrix} f_0(x_0) & \dots & f_k(x_0) \\ \dots & \dots & \dots \\ f_0(x_n) & \dots & f_k(x_n) \end{bmatrix} \begin{bmatrix} a_0 \\ \dots \\ a_k \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \dots \\ y_n \end{bmatrix}$$

- Can use other *basis functions* $\{f_0, \dots, f_k\}$

Solving Linear Systems & LU factorization

Solving Linear Systems

- So how does '\' actually work?
  - Complex!
  - All kinds of special cases

- Two basic algorithms:
  - Gaussian Elimination Method (GEM)
  - LU factorization

Easy cases: Diagonal Matrices

- In case of a diagonal matrix A, the system is easy!

$$\begin{bmatrix} a_{11} & 0 & 0 \\ 0 & a_{22} & 0 \\ 0 & 0 & a_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Easy cases: Diagonal Matrices

- In case of a diagonal matrix A, the system is easy!

$$\begin{bmatrix} a_{11} & 0 & 0 \\ 0 & a_{22} & 0 \\ 0 & 0 & a_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

$$x_1 = c_1 / a_{11}$$
$$x_2 = c_2 / a_{22}$$
$$x_3 = c_3 / a_{33}$$

Easy cases: Diagonal Matrices

- In case of a diagonal matrix A, the system is easy!

$$\begin{bmatrix} a_{11} & 0 & 0 \\ 0 & a_{22} & 0 \\ 0 & 0 & a_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

$$x_1 = c_1 / a_{11}$$
$$x_2 = c_2 / a_{22}$$
$$x_3 = c_3 / a_{33}$$

$$A^{-1} = \begin{bmatrix} 1/a_{11} & 0 & 0 \\ 0 & 1/a_{22} & 0 \\ 0 & 0 & 1/a_{33} \end{bmatrix}$$

Easy cases: Triangular Matrices

- Triangular systems are also easy

$$\begin{bmatrix} 2 & 0 & 0 \\ 6 & 4 & 0 \\ 2 & 4 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 11 \\ 12 \\ -5 \end{bmatrix}$$

Easy cases: Triangular Matrices

- Triangular systems are also easy

$$\begin{bmatrix} 2 & 0 & 0 \\ 6 & 4 & 0 \\ 2 & 4 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 11 \\ 12 \\ -5 \end{bmatrix}$$

$$x_1 = 5.5$$

Easy cases: Triangular Matrices

- Triangular systems are also easy

$$\begin{bmatrix} 2 & 0 & 0 \\ 6 & 4 & 0 \\ 2 & 4 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 11 \\ 12 \\ -5 \end{bmatrix}$$

$$x_1 = 5.5$$

Easy cases: Triangular Matrices

- Triangular systems are also easy

$$\begin{bmatrix} 2 & 0 & 0 \\ 6 & 4 & 0 \\ 2 & 4 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 11 \\ 12 \\ -5 \end{bmatrix}$$

$$\begin{bmatrix} 6 & 4 & 0 \end{bmatrix} \begin{bmatrix} 5.5 \\ x_2 \\ x_3 \end{bmatrix} = 12$$

$$x_1 = 5.5$$

Easy cases: Triangular Matrices

- Triangular systems are also easy

$$\begin{bmatrix} 2 & 0 & 0 \\ 6 & 4 & 0 \\ 2 & 4 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 11 \\ 12 \\ -5 \end{bmatrix}$$

$$\begin{bmatrix} 6 & 4 & 0 \end{bmatrix} \begin{bmatrix} 5.5 \\ x_2 \\ x_3 \end{bmatrix} = 12$$

$$x_1 = 5.5$$

$$33 + 4x_2 = 12$$

$$x_2 = (12 - 33)/4 = -5.25$$

Easy cases: Triangular Matrices

- Triangular systems are also easy

$$\begin{bmatrix} 2 & 0 & 0 \\ 6 & 4 & 0 \\ 2 & 4 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 11 \\ 12 \\ -5 \end{bmatrix}$$

Book (5.9) expresses this in 1 line:

$$x_2 = \frac{1}{4}\left(12 - \left(6 * 5.5\right)\right)$$

$$\begin{bmatrix} 6 & 4 & 0 \end{bmatrix} \begin{bmatrix} 5.5 \\ x_2 \\ x_3 \end{bmatrix} = 12$$

$$x_1 = 5.5$$

$$33 + 4x_2 = 12$$

$$x_2 = \left(12 - 33\right)/4 = -5.25$$

Easy cases: Triangular Matrices

- Triangular systems are also easy

$$\begin{bmatrix} 2 & 0 & 0 \\ 6 & 4 & 0 \\ 2 & 4 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 11 \\ 12 \\ -5 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 4 & 5 \end{bmatrix} \begin{bmatrix} 5.5 \\ -5.25 \\ x_3 \end{bmatrix} = -5$$

$$x_1 = 5.5$$

$$x_2 = -5.25$$

$$11 - 21 + 5x_3 = -10 + 5x_3 = -5$$

$$5x_3 = 5$$

Easy cases: Triangular Matrices

- Triangular systems are also easy

$$\begin{bmatrix} 2 & 0 & 0 \\ 6 & 4 & 0 \\ 2 & 4 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 11 \\ 12 \\ -5 \end{bmatrix}$$

called
'forward substitution'

$$\begin{bmatrix} 2 & 4 & 5 \end{bmatrix} \begin{bmatrix} 5.5 \\ -5.25 \\ x_3 \end{bmatrix} = -5$$

$$x_1 = 5.5$$
$$x_2 = -5.25$$
$$x_3 = 1$$

$$11 - 21 + 5x_3 = -10 + 5x_3 = -5$$

$$5x_3 = 5$$

Easy cases: Triangular Matrices

- Upper triangular matrices work the same.
  - but start at the bottom
  - 'backward substitution'
- basic idea:
  use these simple case to solve general linear systems!

- Gaussian Elimination Method: creates a triangular matrix.
- LU factorization:
  - first decompose a matrix A in L, U
  - then use that to solve the original system

(L)ower and **(U)**pper diagonal

Gaussian Elimination Method (GEM)

- E.g., we want to solve

$$\begin{bmatrix} 3 & 1 & -2 \\ 6 & 4 & 2 \\ 2 & 4 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 8 \\ 6 \\ 2 \end{bmatrix}$$

→ we start by writing this in “augmented form”

Gaussian Elimination Method (GEM)

- Augmented form:

$$\begin{bmatrix} 3 & 1 & -2 \\ 6 & 4 & 2 \\ 2 & 4 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 8 \\ 6 \\ 2 \end{bmatrix}$$

$$\left[\begin{array}{ccc|c} 3 & 1 & -2 & 8 \\ 6 & 4 & 2 & 6 \\ -3 & 4 & 5 & 2 \end{array}\right]$$

→ now we will make it a upper triangular matrix

- eliminate all the non-zeros from the lower left part
- by adding multiples of rows to other rows

Gaussian Elimination Method (GEM)

$$\left[\begin{array}{ccc|c} 3 & 1 & -2 & 8 \\ 6 & 4 & 2 & 6 \\ -3 & 4 & 5 & 2 \end{array}\right]$$

Gaussian Elimination Method (GEM)

$$\left[\begin{array}{ccc|c} 3 & 1 & -2 & 8 \\ 6 & 4 & 2 & 6 \\ -3 & 4 & 5 & 2 \end{array}\right]$$

row2 := row2 - 2*row1

Gaussian Elimination Method (GEM)

$$\left[\begin{array}{ccc|c} 3 & 1 & -2 & 8 \\ 6 & 4 & 2 & 6 \\ -3 & 4 & 5 & 2 \end{array}\right]$$

row2 := row2 - 2*row1

$$\left[\begin{array}{ccc|c} 3 & 1 & -2 & 8 \\ 0 & 2 & 6 & -10 \\ -3 & 4 & 5 & 2 \end{array}\right]$$

Gaussian Elimination Method (GEM)

$$\left[\begin{array}{ccc|c} 3 & 1 & -2 & 8 \\ 6 & 4 & 2 & 6 \\ -3 & 4 & 5 & 2 \end{array}\right]$$

row2 := row2 - 2*row1

$$\left[\begin{array}{ccc|c} 3 & 1 & -2 & 8 \\ 0 & 2 & 6 & -10 \\ -3 & 4 & 5 & 2 \end{array}\right]$$

row3 := row3 + row1

$$\left[\begin{array}{ccc|c} 3 & 1 & -2 & 8 \\ 0 & 2 & 6 & -10 \\ 0 & 5 & 3 & 10 \end{array}\right]$$

Gaussian Elimination Method (GEM)

$$\left[\begin{array}{ccc|c} 3 & 1 & -2 & 8 \\ 6 & 4 & 2 & 6 \\ -3 & 4 & 5 & 2 \end{array}\right]$$

row2 := row2 - 2*row1

$$\left[\begin{array}{ccc|c} 3 & 1 & -2 & 8 \\ 0 & 2 & 6 & -10 \\ -3 & 4 & 5 & 2 \end{array}\right]$$

row3 := row3 + row1

$$\left[\begin{array}{ccc|c} 3 & 1 & -2 & 8 \\ 0 & 2 & 6 & -10 \\ 0 & 5 & 3 & 10 \end{array}\right]$$

row3 := row3 – 2.5*row2

$$\left[\begin{array}{ccc|c} 3 & 1 & -2 & 8 \\ 0 & 2 & 6 & -10 \\ 0 & 0 & -12 & 35 \end{array}\right]$$

Gaussian Elimination Method (GEM)

$$\left[\begin{array}{ccc|c} 3 & 1 & -2 & 8 \\ 6 & 4 & 2 & 6 \\ -3 & 4 & 5 & 2 \end{array}\right]$$

row2 := row2 - 2*row1

Now we have a diagonal system:
→ Solve with backward substitution!

$$\left[\begin{array}{ccc|c} 3 & 1 & -2 & 8 \\ 0 & 2 & 6 & -10 \\ 0 & 0 & -12 & 35 \end{array}\right]$$

row3 := row3 – 2.5*row2

LU factorization

- We want to solve $A\,x = b$
- If $A = LU$
  we get...

find *x*

LU factorization

- We want to solve $A x = b$
- If $A = LU$
  we get...

$$A x = b$$
$$L U x = b$$
$$L\left(U x\right) = b$$

LU factorization

- We want to solve $A\,x = b$
- If $A = LU$
  we get...

$$A\,x = b$$
$$LU\,x = b$$
$$L\left(U\,x\right) = b$$

define: $U\,x \equiv y$

$$L\,y = b$$

LU factorization

- We want to solve $A\,x = b$
- If $A = LU$
  we get...

$$A\,x = b$$
$$L\,U\,x = b$$
$$L\left(U\,x\right) = b$$

define: $U\,x \equiv y$

$$L\,y = b$$

solve

$$y$$

LU factorization

- We want to solve $A\,x = b$
- If $A = LU$
  we get...

$$A\,x = b$$
$$L\,U\,x = b$$
$$L\left(U\,x\right) = b$$

define: $U\,x \equiv y$

$$L\,y = b$$

solve

$$y \longrightarrow U\,x = y$$

LU factorization

- We want to solve $A\,x = b$
- If $A = LU$
  we get...

$$A\,x = b$$
$$L\,U\,x = b$$
$$L\left(U\,x\right) = b$$

define: $U\,x \equiv y$

$$L\,y = b$$

solve

y → $U\,x = y$ solve → x

LU factorization

- How to compute L,U?

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 \\ l_{21} & l_{22} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{bmatrix}$$

- “Gauss factorization”
  - many ways to chose L, U... → arbitrary assignment

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ l_{21} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{bmatrix}$$

  - Now solve the resulting systems of equations
    → $u_{11}=a_{11}$
    → $u_{12}=a_{12}$, etc.
  - (see QSG.)

LU factorization

- How to compute L,U?

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 \\ l_{21} & l_{22} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{bmatrix}$$

- "Gauss factorization"
  - many ways to chose L, U... → arbitrary assignment

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ l_{21} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{bmatrix}$$

  - Now solve the resultin
    → $u_{11}=a_{11}$
    → $u_{12}=a_{12}$, etc.
  - (see QSG.)

Why LU factorization? (rather then GEM)

→ LU factorization is particularly useful when we need to solve Ax=b for multiple b.

Homework Reading

- Recap:
  - CH3: p. 75--81, 93--103 (sec. 3.5 is optional)
  - LU factorization p. 129-142
  - GEM: consult your linear algebra book.