Interpolation and Curve Fitting

Pierre Bézier

De Casteljau construction of Bezier curve
From www.wikipedia.org
Beziers Curves and B-Spline Curves

- Not really “interpolating” curves as they normally do not pass through data points
- Curves stay within the polygon defined by the given points
- Local control: effect of changing a knot is limited to a local region in Bezier and b-spline
- Global control: moving a knot in a cubic spline affects the whole curve
Global vs. Local Control

- Does a small change affect the whole curve or just a small segment?
- Local is usually more intuitive and provided by composite curves
Parametric Form

- Explicit function
  \[ y = f(x) \]

- Parametric form
  \[ x = F_1(u) \]
  \[ y = F_2(u) \]

- Example: unit circle
  \[ x = \cos(u) \]
  \[ y = \sin(u) \]
Interpolation in Parametric Form

- Given \( n+1 \) control points,

\[
p_i = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix}, \quad i = 0,\ldots,n
\]

- Compute interpolation functions

\[
P_j(u) = \begin{bmatrix} x(u) \\ y(u) \\ z(u) \end{bmatrix}, \quad 0 \leq u \leq 1
\]

\( j: \) index for each segment of curve
Composite Segments

- Divide a curve into multiple segments
- Represent each in a parametric form
- Maintain continuity between segments
  - position
  - tangent
  - curvature

\[ P_1(u) \quad P_2(u) \quad P_3(u) \quad P_n(u) \]
Cubic 3D Curves

- Three cubic polynomials, one for each coordinate
  \[ x(u) = a_x u^3 + b_x u^2 + c_x u + d_x \]
  \[ y(u) = a_y u^3 + b_y u^2 + c_y u + d_y \]
  \[ z(u) = a_z u^3 + b_z u^2 + c_z u + d_z \]

- In matrix notation
  \[
  \begin{bmatrix}
  x(u) \\
  y(u) \\
  z(u)
  \end{bmatrix} =
  \begin{bmatrix}
  u^3 & u^2 & u & 1 \\
  \end{bmatrix}
  \begin{bmatrix}
  a_x & a_y & a_z \\
  b_x & b_y & b_z \\
  c_x & c_y & c_z \\
  d_x & d_y & d_z \\
  \end{bmatrix}
  \]
Hermite Interpolation

- Hermite interpolation requires:
  - values at endpoints
  - derivatives at endpoints

- To create a composite curve, use the end of one as the beginning of the other and share the tangent vector.

\[
x(u) = a_x u^3 + b_x u^2 + c_x u + d_x
\]

Needs 4 equations to solve 4 unknowns.
Hermite Curve Formation—
consider x coordinate first

- Cubic polynomial and its derivative
  \[ x(u) = a_x u^3 + b_x u^2 + c_x u + d_x \]
  \[ x'(u) = 3a_x u^2 + 2b_x u + c_x \]

- Given \( x_i, x_{i+1}, x'_i, x'_{i+1} \), solve for \( a, b, c, d \)

  - 4 equations are given for 4 unknowns
  \[
  \begin{align*}
  x(0) &= x_i = d_x \\
  x(1) &= x_{i+1} = a_x + b_x + c_x + d_x \\
  x'(0) &= x'_i = c_x \\
  x'(1) &= x'_{i+1} = 3a_x + 2b_x + c_x 
  \end{align*}
  \]
Hermite Curve Formation (cont.)

- Problem: solve for $a$, $b$, $c$, $d$
  
  \[
  x(0) = x_i = d_x \\
  x(1) = x_{i+1} = a_x + b_x + c_x + d_x \\
  x'(0) = x'_i = c_x \\
  x'(1) = x'_{i+1} = 3a_x + 2b_x + c_x
  \]

- Solution:
  
  \[
  a_x = 2(x_i - x_{i+1}) + x'_i + x'_{i+1} \\
  b_x = 3(x_{i+1} - x_i) - 2x'_i - x'_{i+1} \\
  c_x = x'_i \\
  d_x = x_i
  \]
Hermite Curve Formation (cont.)

- The coefficients can be expressed as linear combination of geometric information

\[
\begin{align*}
a_x &= 2(x_i - x_{i+1}) + x'_i + x'_{i+1} \\
b_x &= 3(x_{i+1} - x_i) - 2x'_i - x'_{i+1} \\
c_x &= x'_i \\
d_x &= x_i
\end{align*}
\]

\[
\begin{bmatrix}
a_x \\
b_x \\
c_x \\
d_x
\end{bmatrix} =
\begin{bmatrix}
2 & -2 & 1 & 1 \\
-3 & 3 & -2 & -1 \\
0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
x_i \\
x_{i+1} \\
x'_i \\
x'_{i+1}
\end{bmatrix}
\]
Hermite Curve Formation (cont.)

- The $x$ component of Hermite curve is then represented as

\[
x(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} a_x \\ b_x \\ c_x \\ d_x \end{bmatrix}
\]

\[
= \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_i \\ x_{i+1} \\ x'_i \\ x'_{i+1} \end{bmatrix}
\]
Hermite Curve Formation (cont.)

- Including y and z components

\[
\begin{bmatrix}
x(u) \\
y(u) \\
z(u)
\end{bmatrix} = \begin{bmatrix}
u^3 & u^2 & u & 1
\end{bmatrix}
\begin{bmatrix}
a_x & a_y & a_z \\
b_x & b_y & b_z \\
c_x & c_y & c_z \\
d_x & d_y & d_z
\end{bmatrix}
\begin{bmatrix}
x_i & y_i & z_i \\
x_{i+1} & y_{i+1} & z_{i+1} \\
x'_i & y'_i & z'_i \\
x'_{i+1} & y'_{i+1} & z'_{i+1}
\end{bmatrix}
\]

\[
\begin{bmatrix}
2 & -2 & 1 & 1 \\
-3 & 3 & -2 & -1 \\
0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0
\end{bmatrix}
\]
Hermite Interpolation in Matrix Form

\[ P(u) = u^T M p \]

\[ M = \begin{bmatrix}
2 & -2 & 1 & 1 \\
-3 & 3 & -2 & -1 \\
0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0
\end{bmatrix} \]

\[ u^T = [u^3, u^2, u, 1] \] is the parameter

\( M \) is the coefficient matrix

\( p \) is the geometric information

\( i \)th segment in composite curves

control points/knots

Numerical Methods © Wen-Chieh Lin
Blending Functions of Hermite Splines

- Each cubic Hermite spline is a linear combination of 4 blending functions.

\[ P(u) = u^T M p \]

![Hermite Blending Functions](image)

Geometric information

\[
P(u) = \begin{bmatrix}
2u^3 - 3u^2 + 1 \\
-2u^3 + 3u^2 \\
u^3 - 2u^2 + u \\
u^3 - u^2
\end{bmatrix}^T \begin{bmatrix}
p_i \\
p_{i+1} \\
p'_i \\
p'_i
\end{bmatrix}
\]
Beziers Curves

- Another variant of the same game
- Instead of endpoints and tangents, four control points

Points $p_0$ and $p_3$ are on the curve
Points $p_1$ and $p_2$ are off the curve
$P(0) = p_0, \ P(1) = p_3$
$P'(0) = 3(p_1 - p_0), \ P'(1) = 3(p_3 - p_2)$

\[
P(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix}
\]
Bezies Curves (cont.)

- Variant of the Hermite spline
  - basis matrix derived from the Hermite basis
- Gives more uniform control knobs (series of points) than Hermite
- The slope at $u = 0$ is the slope of the secant line between $p_0$ and $p_1$

$$P'(0) = 3(p_1 - p_0)$$
$$dx/du = 3(x_1 - x_0)$$
$$dy/du = 3(y_1 - y_0)$$
$$dy/dx = (y_1 - y_0)/(x_1 - x_0)$$
Blending Functions of Bezier Curves

- Also known as degree 3 Bernstein polynomials
- Weighted sum of Bernstein polynomials (infinite series) converges uniformly to any continuous function on the interval \([0,1]\)

\[
P(u) = \begin{bmatrix}
(1-u)^3 \\
3u(1-u)^2 \\
3u^2(1-u) \\
u^3
\end{bmatrix}^{T}
\begin{bmatrix}
p_i \\
p_{i+1} \\
p_{i+2} \\
p_{i+3}
\end{bmatrix}
\]
Composing Bezier Curves

- Control points at consecutive segments need to be collinear to avoid a discontinuity of slope.
Interpolation, Continuity, and Local Control

- Hermite/Bezier splines
  - Interpolate control points
  - Local control
  - No C2 continuity for cubics

- Cubic (Natural) splines
  - Interpolate control points
  - No local control - moving one control point affects whole curve
  - C2 continuity for cubics
  - Can’t get C2, interpolation and local control with cubics
Cubic B-Splines

- Basis-spline (B-spline)
- Give up interpolation
  - the curve passes near the control points
  - best generated with interactive placement (it’s hard to guess where the curve will go)
- Curve obeys the convex hull property
- C2 continuity and local control are good compensation for loss of interpolation
Cubic B-splines (cont.)

- Any evaluation of conditions of $B_i(1)$ cannot use $p_{i+3}$, because $p_{i+3}$ does not appear in $B_i(u)$
- Evaluation conditions of $B_{i+1}(0)$ cannot use $p_{i-1}$
- Two conditions satisfy this symmetric condition

\[
B_i(1) = B_{i+1}(0) = \frac{p_i + 4p_{i+1} + p_{i+2}}{6}
\]

\[
B'_i(1) = B'_{i+1}(0) = \frac{-p_i + p_{i+2}}{2}
\]
Solving for B-spline Coefficients

- From
  \[ B_i(u) = a_i u^3 + b_i u^2 + c_i u + d_i \]
  \[ B'_i(u) = 3a_i u^2 + 2b_i u + c_i \]

- We can obtain

  \[ B_i(0) = d_i \]
  \[ B_i(1) = a_i + b_i + c_i + d_i \]
  \[ B'_i(0) = c_i \]
  \[ B'_i(1) = 3a_i + 2b_i + c_i \]

- We can solve a set of 4 equations for \( a, b, c, d \)

Then plug in the followings

\[ B_i(1) = B_{i+1}(0) = \frac{p_i + 4p_{i+1} + p_{i+2}}{6} \]
\[ B_i(0) = \frac{p_{i-1} + 4p_i + p_{i+1}}{6} \]
\[ B'_i(1) = B'_{i+1}(0) = \frac{-p_i + p_{i+2}}{2} \]
\[ B'_i(0) = \frac{-p_{i-1} + p_{i+1}}{2} \]
Cubic B-splines in Matrix Form

\[ B_i(u) = \mathbf{u}^T \mathbf{M} \mathbf{p} = \frac{1}{6} \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} p_{i-1} \\ p_i \\ p_{i+1} \\ p_{i+2} \end{bmatrix} \]
Blending Functions of Cubic B-Splines

- Represent cubic b-spline as
  \[ B_i(u) = b_{-1}p_{i-1} + b_0p_i + b_1p_{i+1} + b_2p_{i+2} \]

- Blending functions
  \[
  b_{-1} = \frac{(1-u)^3}{6}, \\
  b_0 = \frac{u^3}{2} - u^2 + \frac{2}{3}, \\
  b_1 = -\frac{u^3}{2} + \frac{u^2}{2} + \frac{u}{2} + \frac{1}{6}, \\
  b_{-1} = \frac{u^3}{6}
  \]
Continuity in Cubic B-splines

- Continuity conditions similar to cubic spline

\[ B_i(1) = B_{i+1}(0) = \frac{p_i + 4p_{i+1} + p_{i+2}}{6} \]

\[ B'_{i}(1) = B'_{i+1}(0) = \frac{-p_i + p_{i+2}}{2} \]

\[ B''_{i}(1) = B''_{i+1}(0) = p_i - 2p_{i+1} + p_{i+2} \]

Prove that this condition is automatically satisfied if the first two hold!
Composing B-splines

- Given points $p_0, \ldots, p_n$, we can construct B-splines $B_1$ through $B_{n-2}$

- We need additional points outside the domain of the given points to construct $B_0$ and $B_{n-1}$
  - Add fictitious points $p_{-2}, p_{-1}, p_{n+1}, p_{n+2}$
  - Set $p_{-2} = p_{-1} = p_0$ and $p_{n+2} = p_{n+1} = p_n$ so that the curve starts and ends at the given extreme points
Comparison of Basic Cubic Splines

<table>
<thead>
<tr>
<th>Type</th>
<th>Control points</th>
<th>Continuity</th>
<th>Interpolation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Natural</td>
<td>n points, shared</td>
<td>C2</td>
<td>Y</td>
</tr>
<tr>
<td>Hermite</td>
<td>2 points, 2 slopes</td>
<td>C1</td>
<td>Y</td>
</tr>
<tr>
<td>Bezier</td>
<td>4 points</td>
<td>C1</td>
<td>Y</td>
</tr>
<tr>
<td>B-Splines</td>
<td>4 points, shared</td>
<td>C2</td>
<td>N</td>
</tr>
</tbody>
</table>
Interpolating on a Surface

- Bezier surfaces
- B-Spline surfaces
Bicubic Surfaces

- To represent surfaces use *bicubic* functions
  - \( x(u,v), y(u,v), z(u,v) \) are cubic polynomials both in \( u \) and \( v \)
  - 16 terms for combination of powers of \( u \) and \( v \)
    \[
    x(u, v) = a_{11}u^3v^3 + a_{12}u^3v^2 + \cdots + a_{44}
    \]
- A bicubic surface can be represented by a 4x4 matrix \( A \)
  \[
  u^T = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix}
  \]
  \[
  v^T = \begin{bmatrix} v^3 & v^2 & v & 1 \end{bmatrix}
  \]
  \[
  x(u, v) = u^T A v
  \]
- Also known as tensor surfaces
Cubic B-Spline Surfaces

\[ x_{ij}(u, v) = u^T A v = u^T M X_{i,j} M^T v \]

\[ B_i(u) = u^T M p = \frac{1}{6} \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} p_{i-1} \\ p_i \\ p_{i+1} \\ p_{i+2} \end{bmatrix} \]

\[ X_{ij} = \begin{bmatrix} x_{i-1,j-1} & x_{i-1,j} & x_{i-1,j+1} & x_{i-1,j+2} \\ x_{i,j-1} & x_{i,j} & x_{i,j+1} & x_{i,j+2} \\ x_{i+1,j-1} & x_{i+1,j} & x_{i+1,j+1} & x_{i+1,j+2} \\ x_{i+2,j-1} & x_{i+2,j} & x_{i+2,j+1} & x_{i+2,j+2} \end{bmatrix} \]
Bezier Surfaces

\[ x_{ij}(u, v) = u^T A v = u^T M X_{i,j} M^T v \]

- Recall Bezier curve:

\[ P_i(u) = u^T M p = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_{i-1} \\ p_i \\ p_{i+1} \\ p_{i+2} \end{bmatrix} \]

\[ X_{ij} = \begin{bmatrix} x_{i-1,j-1} & x_{i-1,j} & x_{i-1,j+1} & x_{i-1,j+2} \\ x_{i,j-1} & x_{i,j} & x_{i,j+1} & x_{i,j+2} \\ x_{i+1,j-1} & x_{i+1,j} & x_{i+1,j+1} & x_{i+1,j+2} \\ x_{i+2,j-1} & x_{i+2,j} & x_{i+2,j+1} & x_{i+2,j+2} \end{bmatrix} \]
Bezier Surfaces

\[ p(0,0) = p_{00} \]

\[ \frac{\partial p}{\partial u}(0,0) = 3(p_{10} - p_{00}) \]

\[ \frac{\partial p}{\partial v}(0,0) = 3(p_{01} - p_{00}) \]
Least Square Approximation

- Measurement errors are inevitable in observational and experimental sciences.
- Errors can be smoothed out by averaging over many cases, i.e., taking more measurements than are strictly necessary to determine parameters of system.
- Resulting system is **overdetermined**, so usually there is no exact solution.
Least Square Approximation

- Higher dimensional data are projected into lower dimensional space to suppress irrelevant details
- Such projection is most conveniently accomplished by least square method

Linear projection from 2D to 1D space:
Original data distributed in a 2D space
They can be approximated as a 1D line
Linear Least Squares

- For linear problems, we obtain overdetermined linear system \( Aa = b \) with \( m \times n \) matrix \( A \), \( m > n \).
- System is better written \( Aa \approx b \), since equality is usually not exactly satisfied when \( m > n \).
- Least squares solution \( a \) minimizes squared Euclidean norm of residual vector \( r = b - Aa \).

\[
\min_a \|r\|_2^2 = \min_a \|b - Aa\|_2^2
\]
Curve Fitting

- Given $m$ data points $(x_i, y_i)$, find $n$-vector $a$ of parameters that gives “best fit” to model function $y = f(x, a)$,

$$\min_{a} \sum_{i=1}^{m} (y_i - f(x, a))^2$$

- Problem is linear if function $f$ is linear in components of $a$

$$f(x, a) = a_1 \phi_1(x) + a_2 \phi_2(x) + \cdots + a_n \phi_n(x)$$

- Problem can be written in matrix form as $Aa \approx b$ with $A_{ij} = \phi_j(x_i)$ and $b_i = y_i$
Curve Fitting (cont.)

- Polynomial fitting
  \[ f(x, a) = a_1 + a_2 x + \cdots + a_n x^{n-1} \]
  is linear, since polynomial linear in coefficients, though nonlinear in independent variable \( x \)

- Fitting sum of exponentials
  \[ f(x, a) = a_1 e^{a_2 x} + \cdots + a_{n-1} e^{a_n x} \]
  is example of nonlinear problem

- We will only consider linear least squares problems for now
Example: curve fitting

- Fitting quadratic polynomial to five data points gives linear least squares problem

\[
Aa = \begin{bmatrix}
1 & x_1 & x_1^2 \\
1 & x_2 & x_2^2 \\
1 & x_3 & x_3^2 \\
1 & x_4 & x_4^2 \\
1 & x_5 & x_5^2 \\
\end{bmatrix}
\begin{bmatrix}
a_1 \\
a_2 \\
a_3 \\
\end{bmatrix} \approx
\begin{bmatrix}
y_1 \\
y_2 \\
y_3 \\
y_4 \\
y_5 \\
\end{bmatrix}
\]
Example: curve fitting (cont.)

- For data
  
  \[
  \begin{array}{cccccc}
  x & -1.0 & -0.5 & 0.0 & 0.5 & 1.0 \\
  y & 1.0 & 0.5 & 0.0 & 0.5 & 2.0 \\
  \end{array}
  \]

  overdetermined linear system is

  \[
  \begin{bmatrix}
  1 & -1.0 & 1.0 \\
  1 & -0.5 & 0.25 \\
  1 & 0.0 & 0.0 \\
  1 & 0.5 & 0.25 \\
  1 & 1.0 & 1.0 \\
  \end{bmatrix} \begin{bmatrix}
  a_1 \\
  a_2 \\
  a_3 \\
  \end{bmatrix} \approx \begin{bmatrix}
  1.0 \\
  0.5 \\
  0.0 \\
  0.5 \\
  2.0 \\
  \end{bmatrix}
  \]

- Solution, which we will see later how to compute,

  \[
  \mathbf{a} = [0.086 \quad 0.40 \quad 1.4]^T
  \]
Example (cont.)

- The computed approximation polynomial is
  \[ p(x) = 0.086 + 0.4x + 1.4x^2 \]

- Resulting curve and original data points are shown in graph
Normal Equations

- To minimize squared Euclidean norm of residual vector
  \[ \|r\|^2 = r^T r = (b - Aa)^T (b - Aa) \]
  \[ = b^T b - 2a^T A^T b + a^T A^T A a \]

- Take derivative with respect to \( a \) and set it to 0
  \[ 2A^T A a - 2A^T b = 0 \]
  which reduces to \( n \times n \) linear system of normal equations
  \[ A^T A a = A^T b \]
Orthogonality

- Space spanned by columns of $m \times n$ matrix $A$, $\text{span}(A) = \{Aa : a \in \mathbb{R}^n\}$, is of dimension at most $n$
- If $m > n$, $b$ generally does not lie in $\text{span}(A)$, so there is no exact solution to $Aa = b$
- Vector $y = Aa$ in $\text{span}(A)$ closest to $b$ in 2-norm occurs when residual $r = b - Aa$ is orthogonal to $\text{span}(A)$,
  
  $$0 = A^T r = A^T (b - Aa)$$

again giving system of 

$$A^T Aa = A^T b$$
Orthogonality (cont.)

- Geometric relationships among $b$, $r$, and $\text{span}(A)$ are shown in diagram

\[ r = b - Aa \]

$y = Aa$

$\text{span}(A)$
Pseudoinverse and Condition Number

- Nonsquare $m \times n$ matrix $A$ has no inverse in usual sense

- If $\text{rank}(A) = n$, pseudoinverse is defined by
  \[ A^+ = (A^T A)^{-1} A^T \]
  and condition number by
  \[ \text{cond}(A) = \|A\| \|A^+\| \]

- By convention, $\text{cond}(A) = \infty$ if $\text{rank}(A) < n$
Pseudoinverse and Condition Number

- Just as condition number of square matrix measure closeness to singularity, condition number of rectangular matrix measures closeness to rank deficiency

- Least squares solution of $Aa \approx b$ is given by $a = A^+ b$
Sensitivity and Conditioning

- Sensitivity of least squares solution to $\mathbf{Aa} \approx \mathbf{b}$ depends on $\mathbf{b}$ as well as $\mathbf{A}$
- Define angle $\theta$ between $\mathbf{b}$ and $\mathbf{y} = \mathbf{Aa}$ by
  \[
  \cos(\theta) = \frac{\|\mathbf{y}\|_2}{\|\mathbf{b}\|_2} = \frac{\|\mathbf{Aa}\|_2}{\|\mathbf{b}\|_2}
  \]
- Bound on perturbation $\Delta \mathbf{x}$ in solution $\mathbf{x}$ due to perturbation $\Delta \mathbf{b}$ in $\mathbf{b}$ is given by
  \[
  \frac{\|\Delta \mathbf{x}\|_2}{\|\mathbf{x}\|_2} \leq \text{cond}(\mathbf{A}) \frac{1}{\cos(\theta)} \frac{\|\Delta \mathbf{b}\|_2}{\|\mathbf{b}\|_2}
  \]
Sensitivity and Conditioning

- Similarly, for perturbation E in matrix A,

\[
\frac{\|\Delta x\|_2}{\|x\|_2} \leq \text{cond}(A)(\tan(\theta)\text{cond}(A) + 1) \frac{\|E\|_2}{\|A\|_2}
\]

- Condition number of least squares solution is about \(\text{cond}(A)\) if residual is small, but can be squared or arbitrarily worse for large residual
Normal Equation Method

- If $m \times n$ matrix $A$ has rank $n$, then symmetric $n \times n$ matrix is positive definite, so its Cholesky factorization
  \[ A^T A = LL^T \]
can be used to obtain solution $a$ to system of normal equations
  \[ A^T A a = A^T b \]
which has same solution as linear least squares problem $Aa \approx b$

- Normal equations method involves transformations:
  rectangular $\rightarrow$ square $\rightarrow$ triangular
Example: Normal Equation Method

- For polynomial data-fitting on

\[
\begin{align*}
  x & \quad -1.0 \quad -0.5 \quad 0.0 \quad 0.5 \quad 1.0 \\
  y & \quad 1.0 \quad 0.5 \quad 0.0 \quad 0.5 \quad 2.0
\end{align*}
\]

- Normal equations method gives

\[
A^T A = \begin{bmatrix}
  1 & 1 & 1 & 1 & 1 \\
  -1.0 & -0.5 & 0.0 & 0.5 & 1.0 \\
  1.0 & 0.25 & 0.0 & 0.25 & 1.0 \\
  1 & 1.0 & 1.0
\end{bmatrix}
\begin{bmatrix}
  1 & -1.0 & 1.0 \\
  1 & -0.5 & 0.25 \\
  1 & 0.0 & 0.0 \\
  1 & 0.5 & 0.25 \\
  1 & 1.0 & 1.0
\end{bmatrix}
= \begin{bmatrix}
  5.0 & 0.0 & 2.5 \\
  0.0 & 2.5 & 0.0 \\
  2.5 & 0.0 & 2.125
\end{bmatrix}
\]
Example: Normal Equation Method

- For polynomial data-fitting on
  \[
  \begin{array}{cccccc}
  x & -1.0 & -0.5 & 0.0 & 0.5 & 1.0 \\
  y & 1.0 & 0.5 & 0.0 & 0.5 & 2.0 \\
  \end{array}
  \]

- Normal equations method gives
  \[
  A^Tb = \begin{bmatrix}
  1 & 1 & 1 & 1 & 1 \\
  -1.0 & -0.5 & 0.0 & 0.5 & 1.0 \\
  1.0 & 0.25 & 0.0 & 0.25 & 1.0 \\
  \end{bmatrix}
  \begin{bmatrix}
  1.0 \\
  0.5 \\
  0.0 \\
  0.0 \\
  0.0 \\
  0.5 \\
  2.0 \\
  \end{bmatrix}
  = \begin{bmatrix}
  1.0 \\
  0.0 \\
  0.5 \\
  2.0 \\
  \end{bmatrix}
  = \begin{bmatrix}
  4.0 \\
  1.0 \\
  3.25 \\
  \end{bmatrix}
  \]
Example (cont.)

- Choleskey factorization of symmetric positive definite matrix $A^TA$ gives

\[
\begin{bmatrix}
5.0 & 0.0 & 2.5 \\
0.0 & 2.5 & 0.0 \\
2.5 & 0.0 & 2.125
\end{bmatrix} = \begin{bmatrix}
2.236 & 0.0 & 0.0 \\
0.0 & 1.581 & 0.0 \\
1.118 & 0.0 & 0.935
\end{bmatrix} \begin{bmatrix}
2.236 & 0.0 & 1.118 \\
0.0 & 1.581 & 0.0 \\
0.0 & 0.935 & 0.935
\end{bmatrix}
\]

- Solving lower triangular system $Lz = A^Tb$ by forward substitution gives $z = [1.789 0.632 1.336]^T$

- Solving lower triangular system $La = z$ by forward substitution gives $a = [0.086 0.400 1.429]^T$