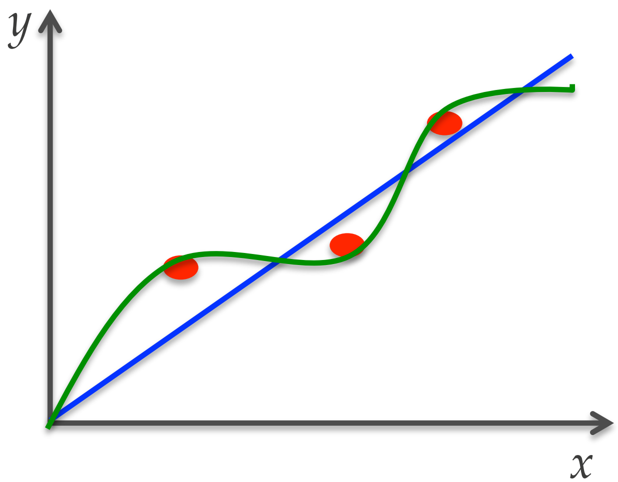
§3.1 Interpolation and Curve Fitting

- **Interpolation** – construct a curve *through* data points; assume data points are __________ and __________.
- **Curve fitting** – find a smooth curve that *approximates* the data points; assume the data has ____________________.

THE UNIVERSITY of TENNESSEE UT KNOXVILLE
Big Orange. Big Ideas.

Interpolation & Curve Fitting



Goal: Given the $n+1$ data points (x_i,y_i), $i=0,1,\ldots,n$, estimate $y(x)$.

§3.2 Lagrange's Method

- Goal: construct a *unique* polynomial of degree *n* that passes through (*n+1*) distinct data points.
- Lagrange formula:

$$P_n(x) = \sum_{i=0}^{n} y_i l_i(x) \text{ , where } l_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{(x - x_j)}{(x_i - x_j)} \text{ , } i = 0,1,\ldots,n.$$

- $l_i(x)$ is called the ____________________.

THE UNIVERSITY of TENNESSEE UT KNOXVILLE

Big Orange. Big Ideas.

Lagrange's Method

- For 2 data points (x_0,y_0) and (x_1,y_1) we have $P_1(x)=y_0l_0(x)+y_1l_1(x)$,

  where

  $l_0(x)=$ ________________, and $l_1(x) =$ __________________.

- What do we know about $P_1(x_0)$ and $P_1(x_1)$?

THE UNIVERSITY of TENNESSEE
KNOXVILLE
Big Orange. Big Ideas.

Lagrange's Method

- For 3 data points (x_0,y_0), (x_1,y_1), and (x_2,y_2) we have $P_2(x)=y_0l_0(x)+y_1l_1(x)+y_2l_2(x)$, where

$l_0(x) =$ ______________ ,

$l_1(x) =$ ______________ , and

$l_2(x) =$ ______________ .

What type of curve is $P_2(x)$?

THE UNIVERSITY of TENNESSEE UT KNOXVILLE

Big Orange. Big Ideas.

Lagrange's Method

- We can specify $l_i(x_j)$ using the Kronecker delta:

$$l_i(x_j) = \delta_{ij} = \begin{cases} 0, \text{if } i \neq j, \\ 1, \text{if } i = j. \end{cases}$$

- Draw cardinal functions for $x_0=0, x_1=2$, and $x_2=3$:

$l_0(x)=$

$l_1(x)=$

$l_2(x)=$

Lagrange's Method

- Does the n^{th} degree Lagrange polynomial interpolate the $(n+1)$ data points? _______

$$P_n(x_j) = \sum_{i=0}^{n} y_i l_i(x_j) = \sum_{i=0}^{n} y_i \delta_{ij} = y_j, \text{ for } j = 0, 1, \ldots, n.$$

- What is the error for approximating $f(x)$ by $P_n(x)$?

$f(x) - P_n(x) =$

where $\xi \in (x_0, x_n)$.

THE UNIVERSITY of TENNESSEE UT KNOXVILLE

Big Orange. Big Ideas.

Newton's Method

- Consider an interpolating polynomial of the form

$$P_n(x) = a_0 + (x - x_0)a_1 + (x - x_0)(x - x_1)a_2$$
$$+ \cdots + (x - x_0)(x - x_1)\cdots(x - x_{n-1})a_n \, .$$

- For 4 data points, the Newton cubic interpolating polynomial can be written as

$$P_3(x) = a_0 + (x - x_0)\left\{a_1 + (x - x_1)\left[a_2 + (x - x_2)a_3\right]\right\}$$

How can you efficiently evaluate $P_3(x)$?

THE UNIVERSITY of TENNESSEE UT KNOXVILLE
Big Orange. Big Ideas.

Newton's Method

- Recurrence relation to build $P_k(x)$:

$$P_0(x) = a_n,$$

$$P_k(x) = a_{n-k} + (x - x_{n-k})P_{k-1}(x),\ k = 1,2,\ldots,n.$$

- Python implementation:

```
P=a[n]
for k in range(1,n+1):
    P=a[n-k]+(x-xData[n-k])*P
```

THE UNIVERSITY of TENNESSEE KNOXVILLE

Big Orange. Big Ideas.

Newton's Method

- Recall that $P_n(x)$ interpolates the data points, $y_i = P_n(x_i), i = 0, 1, \ldots, n.$

$$y_0 = a_0$$

$$y_1 = a_0 + (x_1 - x_0)a_1$$

$$y_2 = a_0 + (x_2 - x_0)a_1 + (x_2 - x_0)(x_2 - x_1)a_2$$

$$\vdots$$

$$y_n = a_0 + (x_n - x_0)a_1 + \cdots + (x_n - x_0)(x_n - x_1)\cdots(x_n - x_{n-1})a_n$$

How can we solve for the a_i's?

THE UNIVERSITY of TENNESSEE UT KNOXVILLE

Big Orange. Big Ideas.

Newton's Method

- Use Newton divided differences to solve for the a_i's:

$$\nabla y_i = \frac{y_i - y_0}{x_i - x_0}, i = 1,2,\ldots,n$$

$$\nabla^2 y_i = \frac{\nabla y_i - \nabla y_1}{x_i - x_1}, i = 2,3,\ldots,n$$

$$\nabla^3 y_i = \frac{\nabla^2 y_i - \nabla^2 y_2}{x_i - x_2}, i = 3,4,\ldots,n$$

$$\vdots$$

$$\nabla^n y_n = \frac{\nabla^{n-1} y_i - \nabla^{n-1} y_{n-1}}{x_i - x_{n-1}}$$

Newton's Method

- Python code to generate the a_i's:

```
a=yData.copy()
for k in range(1,n+1):
   a[i]=(a[i]-a[k-1])/(xData[i]-
                                xData[k-1])
```

- Review `newtonPoly()` on p. 108 of textbook.

THE UNIVERSITY of TENNESSEE UT KNOXVILLE
Big Orange. Big Ideas.

Newton's Method

- **Example 3.4** on pp. 114-115 of textbook:

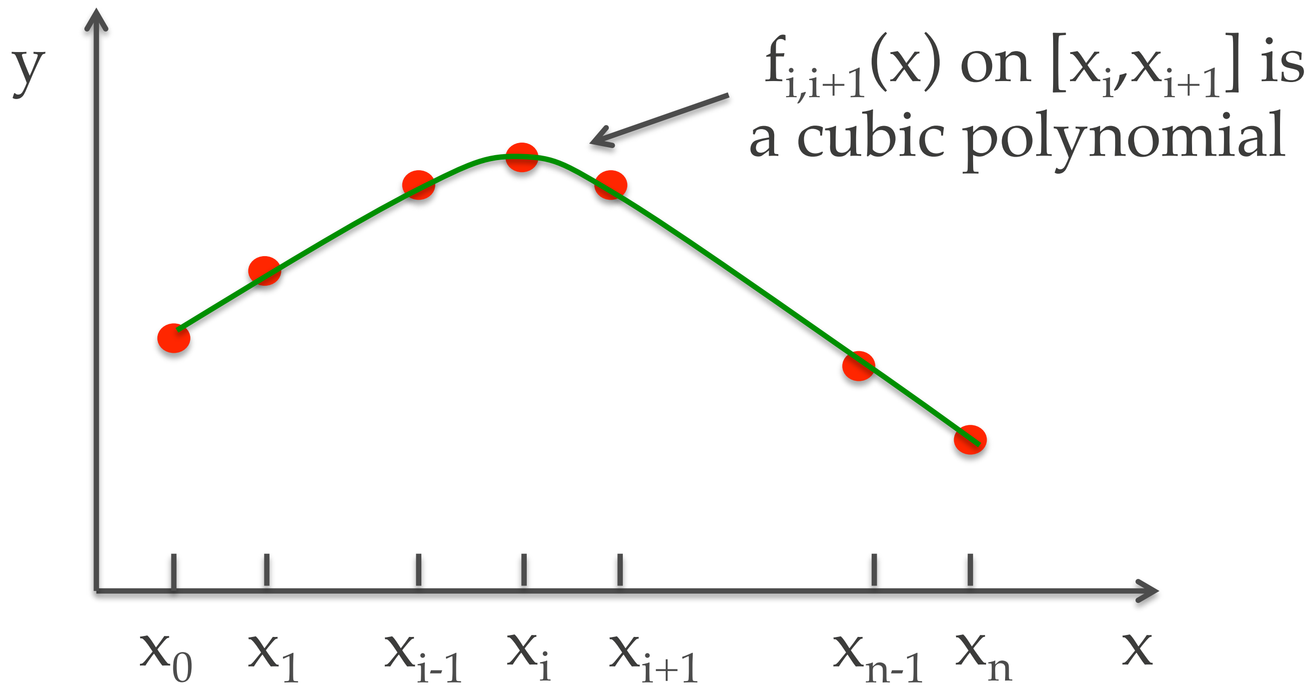
  Interpolate $f(x)=4.8cos(\pi x/20)$ with the following exact values for $y_i=f(x_i)$:

x	0.15	2.30	3.15	4.85	6.25	7.95
y	4.79867	4.49013	4.2243	3.47313	2.66674	1.51909

THE UNIVERSITY of TENNESSEE UT KNOXVILLE
Big Orange. Big Ideas.

§3.3 Cubic Splines

- Cubic splines are *stiffer* than polynomial interpolants, i.e., they oscillate less between the data points (*knots*).



Cubic Splines

- If $f''_{0,1}(x_0) = f''_{n-1,n}(x_n) = 0$ we have a *natural* cubic spline.
- A <u>spline</u> is a *piecewise* cubic curve formed from the n cubics $f_{0,1}(x), f_{1,2}(x), \ldots, f_{n-1,n}(x)$.
- Suppose K_i is the second derivative (*curvature*) of the spline at knot *i*, then the continuity of the spline's second derivative requires $f''_{i-1,i}(x_i) = f''_{i,i+1}(x_i) = K_i$.

What are K_0, K_n for a natural spline?

THE UNIVERSITY of TENNESSEE UT KNOXVILLE

Big Orange. Big Ideas.

Cubic Splines

- So, how do we generate the coefficients of $f_{i,i+1}(x)$?
- Can use Lagrange's 2-point interpolation formula on $[x_i, x_{i+1}]$ to write $f''_{i,i+1}(x) = K_i l_i(x) + K_{i+1} l_{i+1}(x)$, where $l_i(x) = \dfrac{x - x_{i+1}}{x_i - x_{i+1}}$ and $l_{i+1}(x) = \dfrac{x - x_i}{x_{i+1} - x_i}$.
- Via substitution we obtain …

$$f''_{i,i+1}(x) = \frac{K_i(x - x_{i+1}) - K_{i+1}(x - x_i)}{x_i - x_{i+1}}.$$

THE UNIVERSITY of TENNESSEE KNOXVILLE
Big Orange. Big Ideas.

Cubic Splines

- If we integrate (twice with respect to *x*) the expression we just generated for $f''_{i,i+1}(x)$, we produce the following equation for $f_{i,i+1}(x)$:

$$f_{i,i+1}(x) = \frac{K_i(x - x_{i+1})^3 - K_{i+1}(x - x_i)^3}{6(x_i - x_{i+1})} + A(x - x_{i+1}) - B(x - x_i).$$

- Where do the constants *A* and *B* come from?

THE UNIVERSITY of TENNESSEE UT KNOXVILLE
Big Orange. Big Ideas.

Cubic Splines

- Recall that the spline must also interpolate the data points (knots).
- Since $f_{i,i+1}(x_i) = y_i$ we must have

$$\frac{K_i(x_i - x_{i+1})^3}{6(x_i - x_{i+1})} + A(x_i - x_{i+1}) = y_i \,.$$

- What happened to term involving B?

THE UNIVERSITY of TENNESSEE UT KNOXVILLE
Big Orange. Big Ideas.

Cubic Splines

- Similarly we must have $f_{i,i+1}(x_{i+1}) = y_{i+1}$ so that

$$\frac{K_{i+1}(x_{i+1} - x_i)^3}{6(x_i - x_{i+1})} - B(x_{i+1} - x_i) = y_{i+1}\,.$$

- Solving for the constants *A* and *B* yields:

$$A = \frac{y_i}{x_i - x_{i+1}} - \frac{K_i}{6}(x_i - x_{i+1}),\; B = \frac{y_{i+1}}{x_i - x_{i+1}} - \frac{K_{i+1}}{6}(x_i - x_{i+1}).$$

THE UNIVERSITY of TENNESSEE UT KNOXVILLE
Big Orange. Big Ideas.

Cubic Splines

- Now we can substitute those equations for *A* and *B* into the formula for $f_{i,i+1}(x)$ on page 17:

$$f_{i,i+1}(x) = \frac{K_i}{6}\left[\frac{(x-x_{i+1})^3}{x_i - x_{i+1}} - (x-x_{i+1})(x_i - x_{i+1})\right]$$

$$-\frac{K_{i+1}}{6}\left[\frac{(x-x_i)^3}{x_i - x_{i+1}} - (x-x_i)(x_i - x_{i+1})\right]$$

$$+\frac{y_i(x-x_{i+1}) - y_{i+1}(x-x_i)}{x_i - x_{i+1}}.$$

Cubic Splines

- But we require slope continuity at the interior knots, i.e., $f'_{i-1,i}(x_i) = f'_{i,i+1}(x_i),\ i = 1 = 1,2,\ldots,n-1.$
- So, differentiating will yield the following linear system of equations:

$$K_{i-1}(x_{i-1} - x_i) + 2K_i(x_{i-1} - x_{i+1}) + K_{i+1}(x_i - x_{i+1}) =$$

$$6\left(\frac{y_{i-1} - y_i}{x_{i-1} - x_i} - \frac{y_i - y_{i+1}}{x_i - x_{i+1}}\right), i = 1,2,\ldots,n-1.$$

What special form does the coefficient matrix have?

THE UNIVERSITY of TENNESSEE UT KNOXVILLE
Big Orange. Big Ideas.

Cubic Splines

- If the knots are equally spaced at intervals *h,* then

$$x_{i+1} - x_i = h \ ,$$

and the linear system simplifies to:

$$K_{i-1} + 4K_i + K_{i+1}(x_i - x_{i+1}) = \frac{6}{h^2}\left(y_{i-1} - 2y_i + y_{i+1}\right), i = 1,2,\ldots,n-1.$$

- Review `cubicSpine.py` on pp. 122-123 and see **Example 3.7** on pp. 123-124.

THE UNIVERSITY of TENNESSEE KNOXVILLE
Big Orange. Big Ideas.

§3.4 Least-Squares Fit

- Assume experimental data contains *noise;* want smooth curve that fits data (on average). For data points $(x_i, y_i), i = 0,1,\ldots,n$ let us assume the noise is confined to the *y*-coordinates.

- A least-squares fit minimizes (wrt a_j) the function

$$S(a_0, a_1, \ldots, a_m) = \sum_{i=0}^{n} [y_i - f(x_i)]^2 .$$

THE UNIVERSITY of TENNESSEE KNOXVILLE
Big Orange. Big Ideas.

Least-Squares Fit

- The optimal values for the parameters are given by the solution to these equations:

$$\frac{\partial S}{\partial a_k} = 0, k = 0, 1, \ldots, m.$$

- Let $r_i = y_i - f(x_i)$ be the *i*th residual representing the discrepancy between the ith data point and the fitting function at x_i.

THE UNIVERSITY of TENNESSEE KNOXVILLE
Big Orange. Big Ideas.

Least-Squares Fit

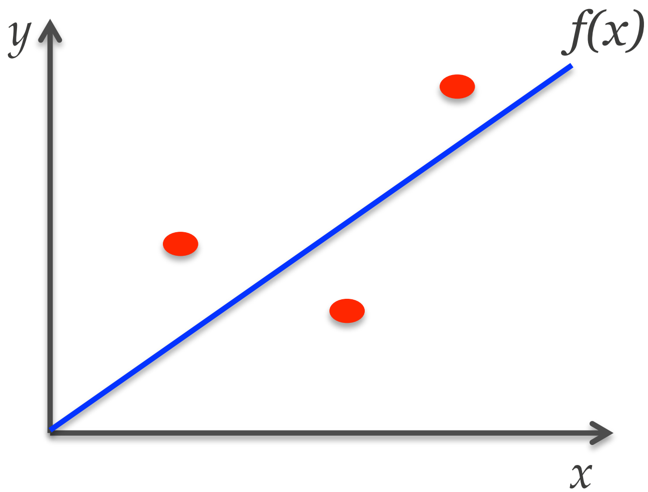
- The fitting function can be chosen as a linear combination of specified functions $f_j(x)$:

$$f(x) = a_0 f_0(x) + a_1 f_1(x) + \cdots + a_m f_m(x).$$

- If we want the fitting function to be a polynomial, then we can choose $f_0(x)=1$, $f_1(x)=x$, $f_2(x)=x^2$, etc.
- How can we visualize the fit (in 2D)?

THE UNIVERSITY of TENNESSEE KNOXVILLE
Big Orange. Big Ideas.

Least-Squares Fit



What can you say about the case when $m=n$?

The spread of data about a fitting curve is quantified by the standard deviation $\sigma = \sqrt{S/(n-m)}$.

Least-Squares Fit

- Let's consider the case when *m*=1 (fitting a straight line); so we try to fit $f(x)=a+bx$ to the data (also called *linear regression*).

- We need to minimize:

$$S(a,b)=\sum_{i=0}^{n}[y_i-f(x_i)]^2=\sum_{i=0}^{n}(y_i-a-bx_i)^2.$$

THE UNIVERSITY of TENNESSEE UT KNOXVILLE
Big Orange. Big Ideas.

Least-Squares Fit

- Taking partial derivatives and setting them to zero yields:

$$\frac{\partial S}{\partial a} = \sum_{i=0}^{n} -2(y_i - a - bx_i) = 2\left[a(n+1) + b\sum_{i=0}^{n} x_i - \sum_{i=0}^{n} y_i\right] = 0.$$

$$\frac{\partial S}{\partial b} = \sum_{i=0}^{n} -2(y_i - a - bx_i)x_i = 2\left(a\sum_{i=0}^{n} x_i + b\sum_{i=0}^{n} x_i^2 - \sum_{i=0}^{n} x_i y_i +\right) = 0.$$

THE UNIVERSITY of TENNESSEE KNOXVILLE
Big Orange. Big Ideas.

Least-Squares Fit

- Now, define $\bar{x} = \frac{1}{n+1}\sum_{i=0}^{n} x_i$ and $\bar{y} = \frac{1}{n+1}\sum_{i=0}^{n} y_i$,

  and divide the equations on the previous slide by *2(n+1)* to obtain: $a + \bar{x}b = \bar{y}$,

$$\bar{x}a + \left(\frac{1}{n+1}\sum_{i=0}^{n} x_i^2\right)b = \frac{1}{n+1}\sum_{i=0}^{n} x_i y_i .$$

THE UNIVERSITY of TENNESSEE UT KNOXVILLE
Big Orange. Big Ideas.

Least-Squares Fit

$$a + \bar{x} b = \bar{y},$$

$$\bar{x} a + \left(\frac{1}{n+1} \sum_{i=0}^{n} x_i^2 \right) b = \frac{1}{n+1} \sum_{i=0}^{n} x_i y_i .$$

- Solving the equations above for *a* and *b* yields:

$$a = \bar{y} - \bar{x} b,$$

$$b = \frac{\sum y_i (x_i - \bar{x})}{\sum x_i (x_i - \bar{x})}.$$

Can now fit $f(x)=a+bx$ to (x_i,y_i), $i=0,1,2,\ldots,n$.

THE UNIVERSITY of TENNESSEE UT KNOXVILLE

Big Orange. Big Ideas.

Fitting Linear Forms

- Assume $f(x) = a_0 f_0(x) + a_1 f_1(x) + \cdots + a_m f_m(x) = \sum_{j=0}^{m} a_j f_j(x)$.

  So that
  $$S = \sum_{i=0}^{n} \left[y_i - \sum_{j=0}^{m} a_j f_j(x_i) \right]^2 ,$$
  then
  $$\frac{\partial S}{\partial a_k} = -2 \left\{ \sum_{i=0}^{n} \left[y_i - \sum_{j=0}^{m} a_j f_j(x_i) \right] f_k(x_j) \right\} = 0, k = 0, 1, \ldots, m.$$

Let's simplify this equation…

THE UNIVERSITY of TENNESSEE UT KNOXVILLE
Big Orange. Big Ideas.

Fitting Linear Forms

- Simplification yields:

$$\sum_{j=0}^{m}\left[\sum_{i=0}^{n} f_j(x_i) f_k(x_i)\right] a_j = \sum_{i=0}^{n} f_k(x_i) y_i, k = 0,1,\ldots,m.$$

or $A\vec{a} = \vec{b},\ A = [A_{kj}], \vec{b} = [b_k],$ "Normal Equations"

where $A_{kj} = \sum_{i=0}^{n} f_j(x_i) f_k(x_i),\ b_k = \sum_{i=0}^{n} f_k(x_i) y_i.$

Fitting Linear Forms

- For a degree m polynomial, assume $f_j(x)=x^j$, for j=$0,1,2,\ldots,m$.

  Then, the resulting linear system is

$$A\vec{a} = \vec{b},\ A = [A_{kj}],\ A_{kj} = \sum_{i=0}^{n} x_i^{j+k},$$

$$\vec{b} = [b_k],\ b_k = \sum_{i=0}^{n} x_i^k y_i.$$

THE UNIVERSITY of TENNESSEE UT KNOXVILLE
Big Orange. Big Ideas.

Fitting Linear Forms

- Let's take a closer look at that linear system:

$$A = \begin{bmatrix} n & \sum x_i & \sum x_i^2 & \cdots & \sum x_i^m \\ \sum x_i & \sum x_i^2 & \sum x_i^3 & \cdots & \sum x_i^{m+1} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \sum x_i^m & \sum x_i^{m+1} & \sum x_i^{m+2} & \cdots & \sum x_i^{2m} \end{bmatrix}, \vec{b} = \begin{bmatrix} \sum y_i \\ \sum x_i y_i \\ \vdots \\ \sum x_i^m y_i \end{bmatrix}.$$

As *m* approaches infinity, **A** becomes more ill-conditioned – what does this say about high-degree polynomial curve fitting?

THE UNIVERSITY of TENNESSEE UT KNOXVILLE
Big Orange. Big Ideas.

Weighting of Data

- Review `PolyFit.py` on pp. 132-133 and see **Example 3.12** on pp. 140-141.
- In some cases it is important to assign a confidence factor or *weight* to each data point (x_i, y_i); we would then need to minimize a sum of squares of weighted residuals, i.e.,

$$r_i = w_i[y_i - f(x_i)],$$ where the w_i's are the weights.

THE UNIVERSITY of TENNESSEE UT KNOXVILLE

Big Orange. Big Ideas.

Weighted Linear Regression

- Let the fitting function be $f(x)=a+bx$, then $S(a,b)=\sum_{i=0}^{n} w_i^2 (y_i - a - bx_i)^2$, and minimizing yields

$$a = \hat{y} - b\hat{x}, \qquad \text{where } \hat{x} = \frac{\sum w_i^2 x_i}{\sum w_i^2}, \hat{y} = \frac{\sum w_i^2 y_i}{\sum w_i^2}.$$

$$b = \frac{\sum w_i^2 y_i (x_i - \hat{x})}{\sum w_i^2 x_i (x_i - \hat{x})},$$

THE UNIVERSITY of TENNESSEE UT KNOXVILLE
Big Orange. Big Ideas.

Bézier Curves

- An alternative to cubic splines for smooth curve generation are Bézier curves.
- They are parametric curves commonly used in computer graphics for generate smooth curves that can be scaled indefinitely (to higher dimensions).
- Combinations of linked Bézier curves are called paths that are easy to manipulate and can even be used in animation as a tool to control motion.

THE UNIVERSITY of TENNESSEE KNOXVILLE
Big Orange. Big Ideas.

Bézier Curves

- Used in the time domain, they can be used to specify the velocity over time of an object such as an icon moving from point A to point B, rather than simply moving a fixed number of pixels per step.
- The mathematical basis for Bézier curves is the Bernstein polynomial (circa 1912), and the French engineer Pierre Bézier demonstrated their use in the design of automobile bodies (for Renault) in 1962.

Bézier Curves

- Applications
  - Computer graphics for smooth curves (quadratic and cubic Bézier curves are the most common); used by Adobe Illustrator, CorelDraw, Inkscape, and Microsoft Excel (smooth curve feature for charts).
  - Animation for outlining movement; used by Adobe Flash and Synfig.
  - TrueType fonts are composed of quadratic Bézier curves (used by font engines like FreeType).

THE UNIVERSITY of TENNESSEE UT
KNOXVILLE
Big Orange. Big Ideas.

Linear Bézier Curves

- A Bézier curve (BC) is defined by a set of control points P_0 through P_n, where n is the order of the (BC). For a linear BC, n=1 and for a quadratic BC, n=2. The first and last control points are <u>always</u> the endpoints of the curve and the intermediate control points usually <u>do not lie</u> on the curve.
- For the case n=1 with points P_0 and P_1, the BC is simply a straight line between those points:

$$B(t) = P_0 + t(P_1 - P_0) = (1-t)P_0 + tP_1, t \in [0,1].$$

THE UNIVERSITY of TENNESSEE UT KNOXVILLE
Big Orange. Big Ideas.

Quadratic Bézier Curves

- A quadratic BC is the path traced by the function *B(t)*, given control points P_0, P_1, and P_2:

$$B(t) = (1-t)\left[(1-t)P_0 + tP_1\right] + t\left[(1-t)P_1 + tP_2\right], t \in [0,1].$$

- The function *B(t)* above linearly interpolates the points that lie on all the linear BCs defined from P_0 to P_1 and all the linear BCs defined from P_1 to P_2.
- Rearranging the right-hand-side for *B(t)* we obtain:

$$B(t) = (1-t)^2 P_0 + 2(1-t)tP_1 + t^2 P_2, t \in [0,1].$$

THE UNIVERSITY of TENNESSEE KNOXVILLE

Big Orange. Big Ideas.

Quadratic Bézier Curves

- Since we have $B(t) = (1-t)^2 P_0 + 2(1-t)tP_1 + t^2 P_2, t \in [0,1]$.

  Suppose we compute its derivative *B′(t)*:

$$B'(t) = 2(1-t)(P_1 - P_0) + 2t(P_2 - P_1).$$

It can be shown that the tangents to the curve *B(t)* at P_0 and P_2 intersect at ___________. As t increases from 0 to 1, the curve departs from P_0 in the direction of _______, and then bends to arrive at P_2 in the direction of _______.

THE UNIVERSITY of TENNESSEE KNOXVILLE
Big Orange. Big Ideas.

Cubic Bézier Curves

- A cubic BC is the path traced by the function *B(t)*, given control points P_0, P_1, P_2, and P_3.
- The curve *B(t)* starts at P_0 going toward P_1 and later arrives at P_3 from the direction of P_2. The curve typically does not pass through the intermediate control points (P_1 and P_2) – they are used for directions.
- Distance between P_0 and P_1 determines how long *B(t)* moves into direction P_2 before turning towards P_3.

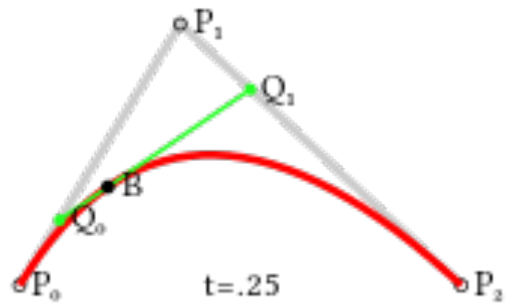
THE UNIVERSITY of TENNESSEE UT KNOXVILLE
Big Orange. Big Ideas.

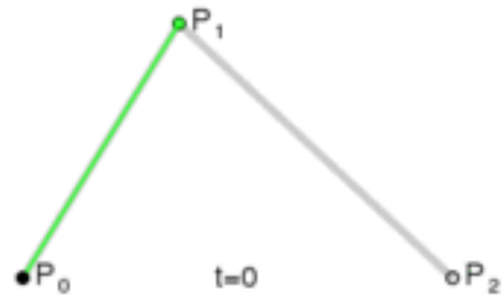
$$B(t) = (1-t)^3 P_0 + 3(1-t)^2 t P_1 + 3(1-t) t^2 P_2 + t^3 P_3, t \in [0,1].$$

Bézier Curve Animations

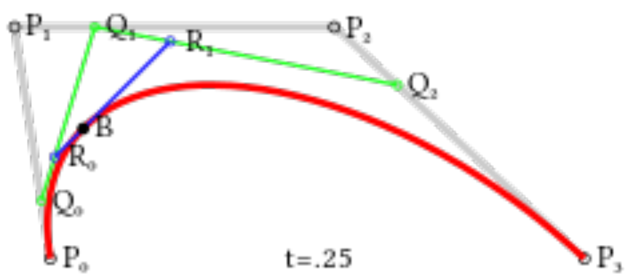
- Images courtesy of Wikipedia:

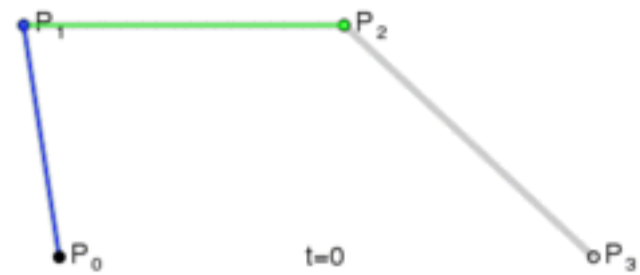
Quadratic





Cubic





THE UNIVERSITY of TENNESSEE KNOXVILLE
Big Orange. Big Ideas.

matplotlib.patch

- The **`patch`** function uses a **`Path`** object to generate Bézier curves via standard curve commands (**`MOVETO`**, **`LINETO`**, **`CLOSEPOLY`**).
- Typically specify a path via

  ```
  path = Path(verts,codes)
  ```

  where `verts` is a list of 2D data points and codes is a list of curve commands.
- See **`bezier.py`** script file for demo of a cubic Bézier curve using the **`patch`** function.

THE UNIVERSITY of TENNESSEE UT KNOXVILLE
Big Orange. Big Ideas.