

# **Lagrange Interpolation and Neville's Algorithm**

**Ron Goldman**

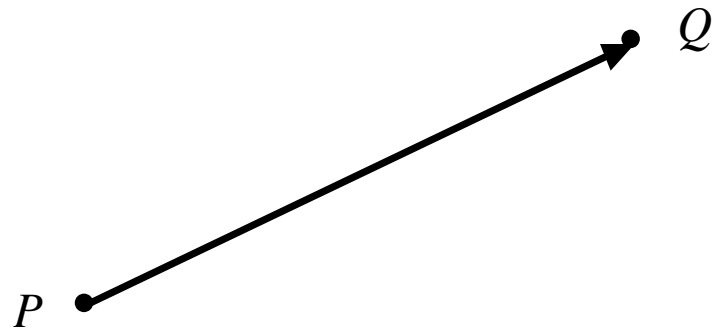
**Department of Computer Science**

**Rice University**

## Tension between Mathematics and Engineering

1. How do Mathematicians actually represent curves and surfaces?
  - Algebra -- Formulas and Algorithms
2. How do Scientists and Engineers want to represent curves and surfaces?
  - Geometry -- Interpolation and Approximation

## Straight Lines



Equation = ?

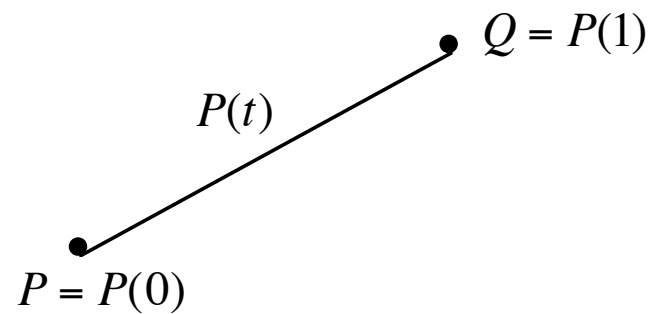
## Linear Interpolation

### *Straight Line*

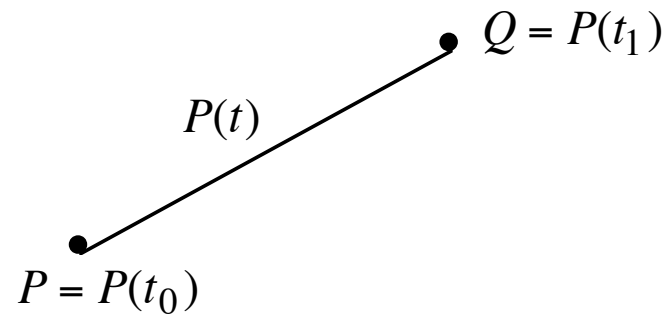
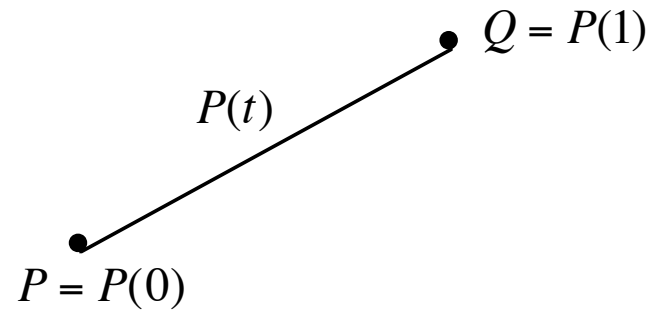
- $P(t) = P + t(Q - P)$
- $P(t) = (1 - t)P + tQ$

### *Observations*

- $P(0) = P$
- $P(1) = Q$



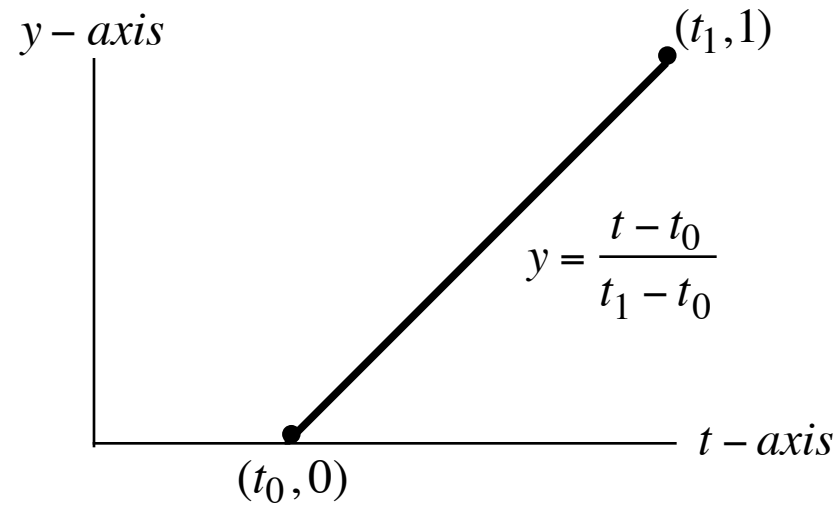
## Linear Interpolation



## Linear Interpolation Revisited

### *Straight Line*

- $P_{01}(t) = (1 - f(t))P_0 + f(t)P_1$
- $f(t_0) = 0$  and  $f(t_1) = 1$



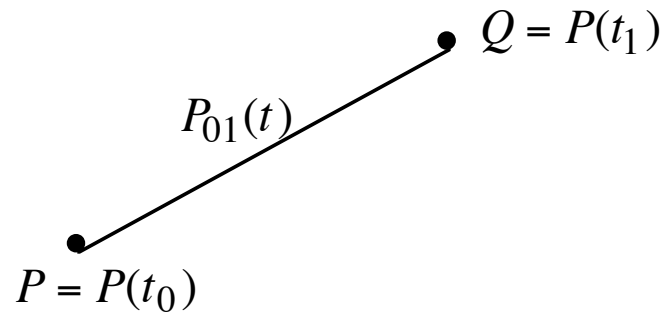
## Linear Interpolation Revisited

### *Linear Interpolation*

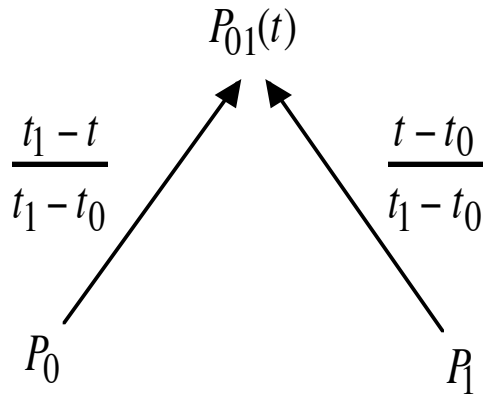
- $f(t) = \frac{(t - t_0)}{(t_1 - t_0)}$
- $f(t_0) = 0$  and  $f(t_1) = 1$ .

### *Straight Line*

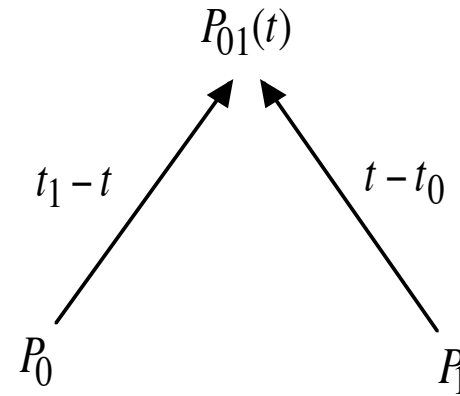
- $P_{01}(t) = \frac{t_1 - t}{t_1 - t_0} P_0 + \frac{t - t_0}{t_1 - t_0} P_1$
- $P(t_0) = P$        $P(t_1) = Q$



## Linear Interpolation



Normalized



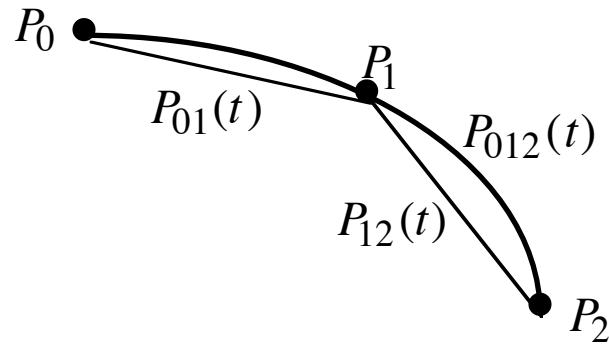
Unnormalized

$$P_{01}(t) = \frac{t_1 - t}{t_1 - t_0} P_0 + \frac{t - t_0}{t_1 - t_0} P_1$$

$$P_{01}(t_0) = P_0$$

$$P_{01}(t_1) = P_1$$

## Quadratic Interpolation



*Problem*

Find a smooth curve  $P_{012}(t)$  such that:

$$P_{012}(t_0) = P_0$$

$$P_{012}(t_1) = P_1$$

$$P_{012}(t_2) = P_2$$

## Quadratic Interpolation

### *Linear Interpolation*

- $$P_{01}(t) = \frac{t_1 - t}{t_1 - t_0} P_0 + \frac{t - t_0}{t_1 - t_0} P_1$$
- $$P_{12}(t) = \frac{t_2 - t}{t_2 - t_1} P_1 + \frac{t - t_1}{t_2 - t_1} P_2$$

### *Quadratic Interpolation*

- $$P_{012}(t) = \frac{t_2 - t}{t_2 - t_0} P_{01}(t) + \frac{t - t_0}{t_2 - t_0} P_{12}(t)$$

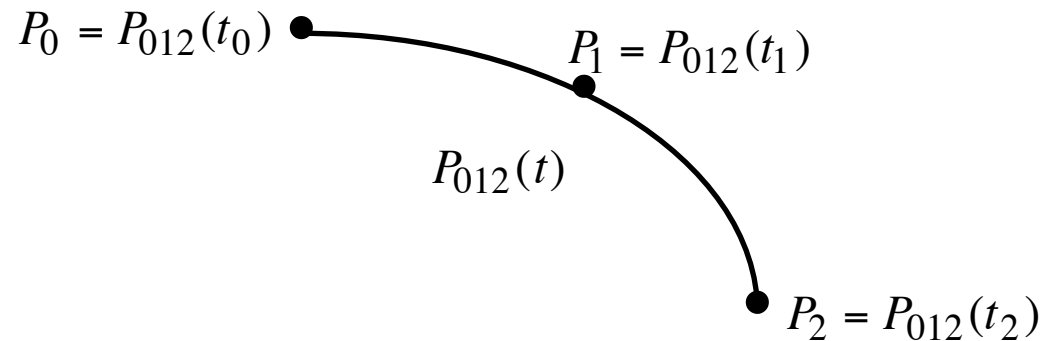
## Verification of Quadratic Interpolation

$$P_{012}(t) = \frac{t_2 - t}{t_2 - t_0} P_{01}(t) + \frac{t - t_0}{t_2 - t_0} P_{12}(t)$$

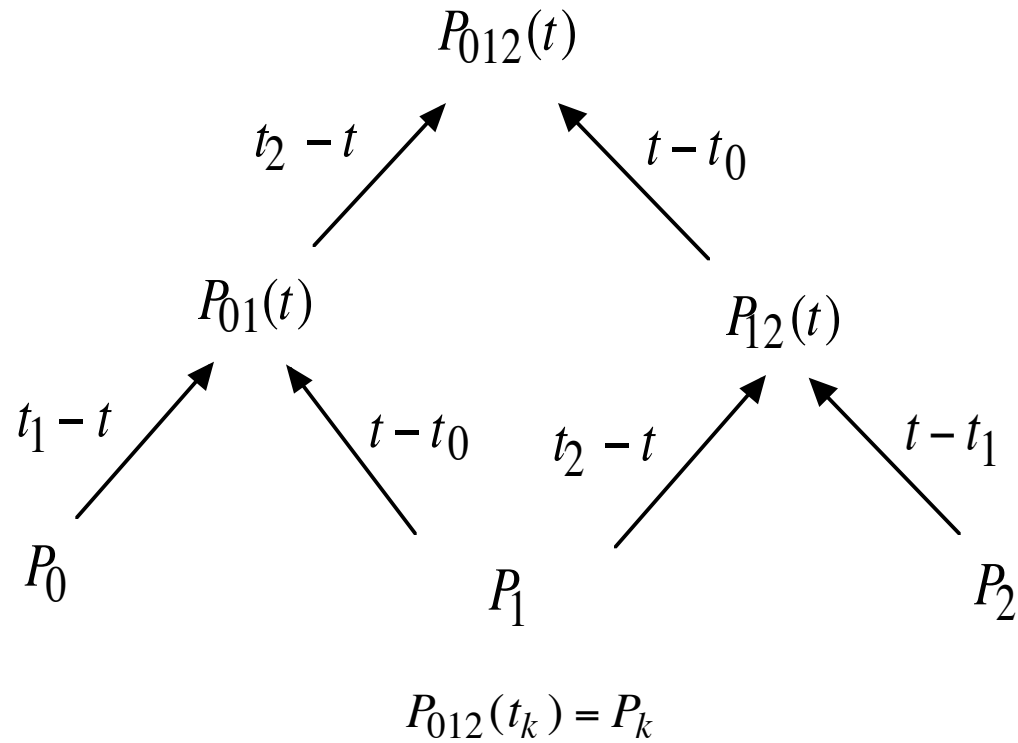
i.  $P_{012}(t_0) = P_{01}(t_0) = P_0$

ii.  $P_{012}(t_2) = P_{12}(t_2) = P_2$

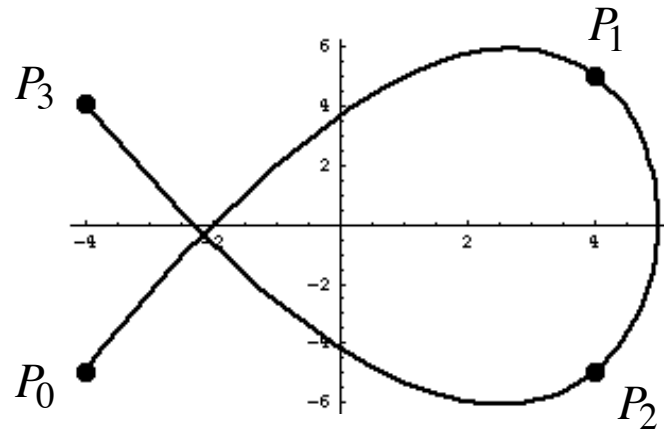
iii. 
$$P_{012}(t_1) = \frac{t_2 - t_1}{t_2 - t_0} P_{01}(t_1) + \frac{t_1 - t_0}{t_2 - t_0} P_{12}(t_1)$$
$$= \frac{t_2 - t_1}{t_2 - t_0} P_1 + \frac{t_1 - t_0}{t_2 - t_0} P_1 = P_1$$



## Neville's Algorithm for Quadratic Interpolation



## Cubic Interpolation



*Problem*

Find a smooth function  $P_{0123}(t)$  such that:

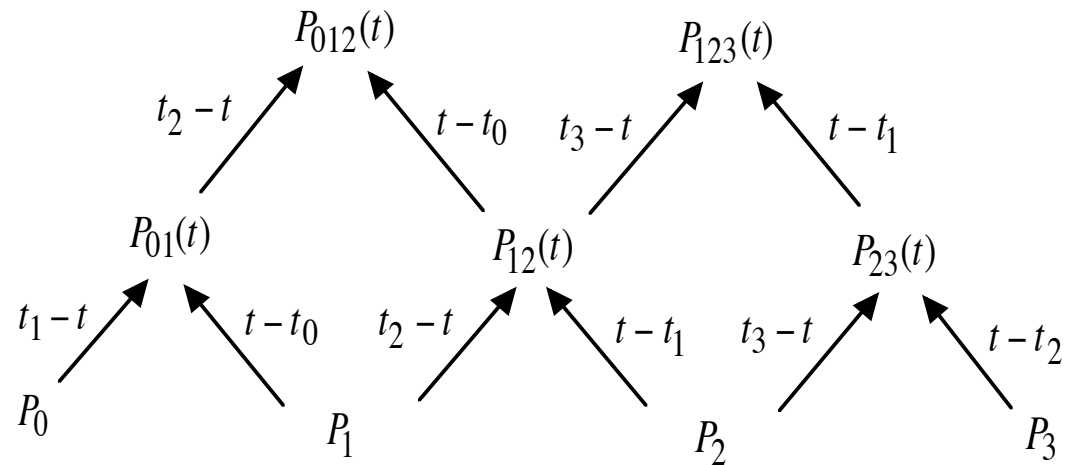
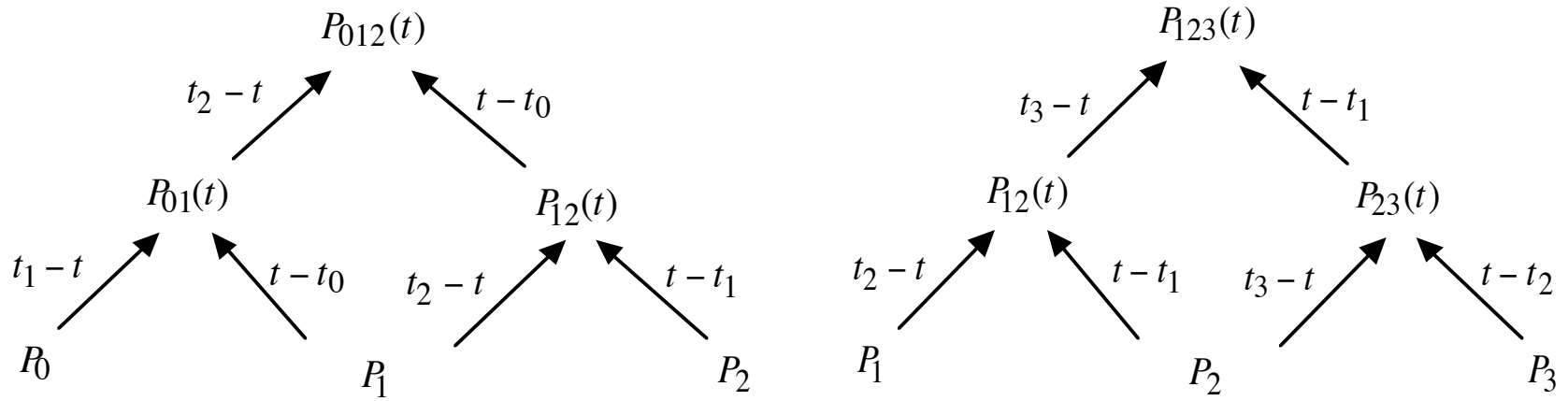
$$P_{0123}(t_0) = P_0$$

$$P_{0123}(t_1) = P_1$$

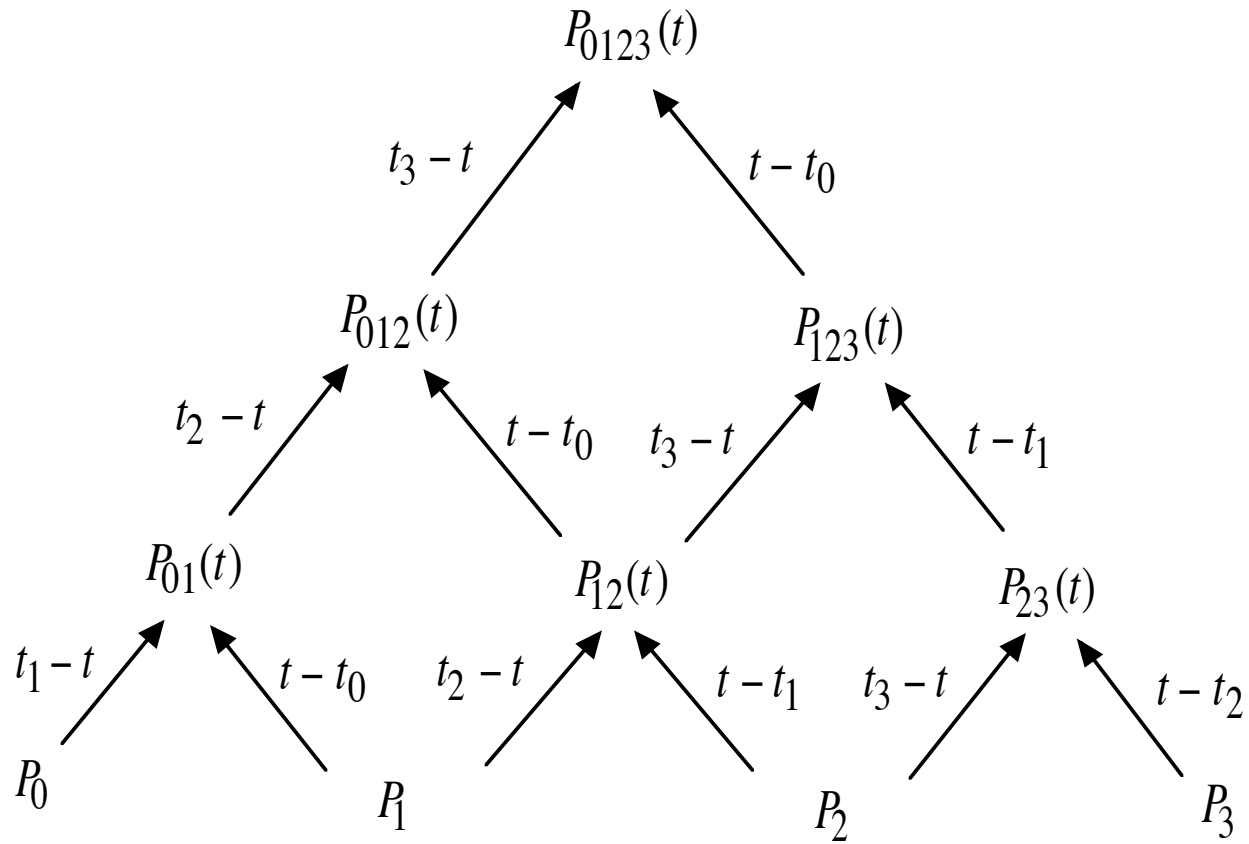
$$P_{0123}(t_2) = P_2$$

$$P_{0123}(t_3) = P_3$$

## Neville's Algorithm for Two Quadratic Curves



## Neville's Algorithm for Cubic Curves



$$P_{0123}(t_k) = P_k$$

## Verification of Cubic Interpolation

$$P_{0123}(t) = \frac{t_3 - t}{t_3 - t_0} P_{012}(t) + \frac{t - t_0}{t_3 - t_0} P_{123}(t)$$

i.  $P_{0123}(t_0) = P_{012}(t_0) = P_0$

ii.  $P_{0123}(t_3) = P_{123}(t_3) = P_3$

iii. 
$$\begin{aligned} P_{0123}(t_1) &= \frac{t_3 - t_1}{t_3 - t_0} P_{012}(t_1) + \frac{t_1 - t_0}{t_3 - t_0} P_{123}(t_1) \\ &= \frac{t_3 - t_1}{t_3 - t_0} P_1 + \frac{t_1 - t_0}{t_3 - t_0} P_1 \\ &= P_1 \end{aligned}$$

iv.  $P_{0123}(t_2) = P_2$  (same as proof for  $t_1$ )

## Neville's Algorithm for Lagrange Interpolation

**Theorem:** Given points  $P_0, \dots, P_n$  and parameters  $t_0, \dots, t_n$ , there exists a polynomial curve  $P_{0\dots n}(t)$  of degree  $n$  that interpolates the given points at the specified parameter values. That is,

$$P_{0\dots n}(t_k) = P_k \quad k = 0, \dots, n .$$

Proof: By induction on  $n$ . Define

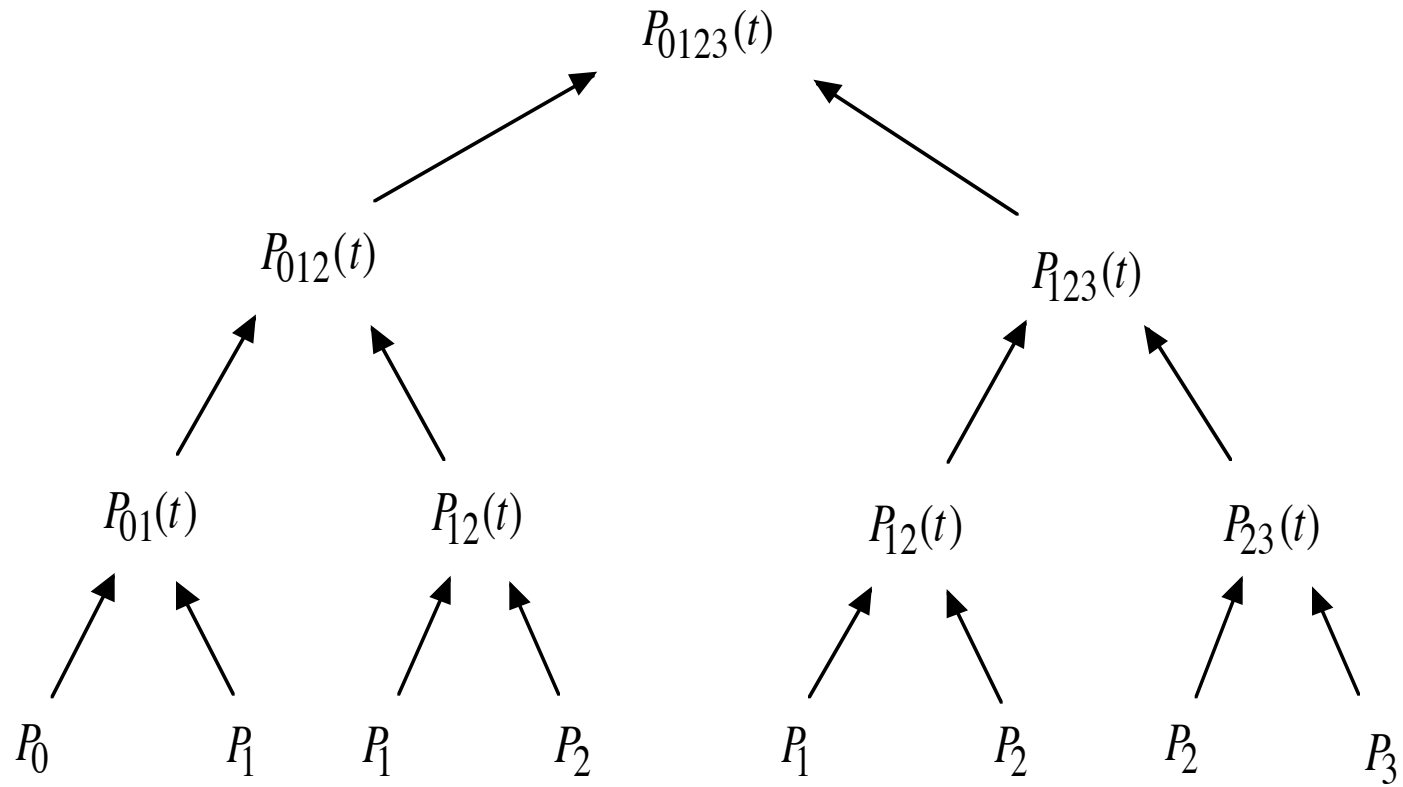
$$P_{0\dots n}(t) = \frac{t_n - t}{t_n - t_0} P_{0\dots n-1}(t) + \frac{t - t_0}{t_n - t_0} P_{1\dots n}(t) .$$

Applying the same arguments we used in the quadratic and cubic cases, you can easily verify that

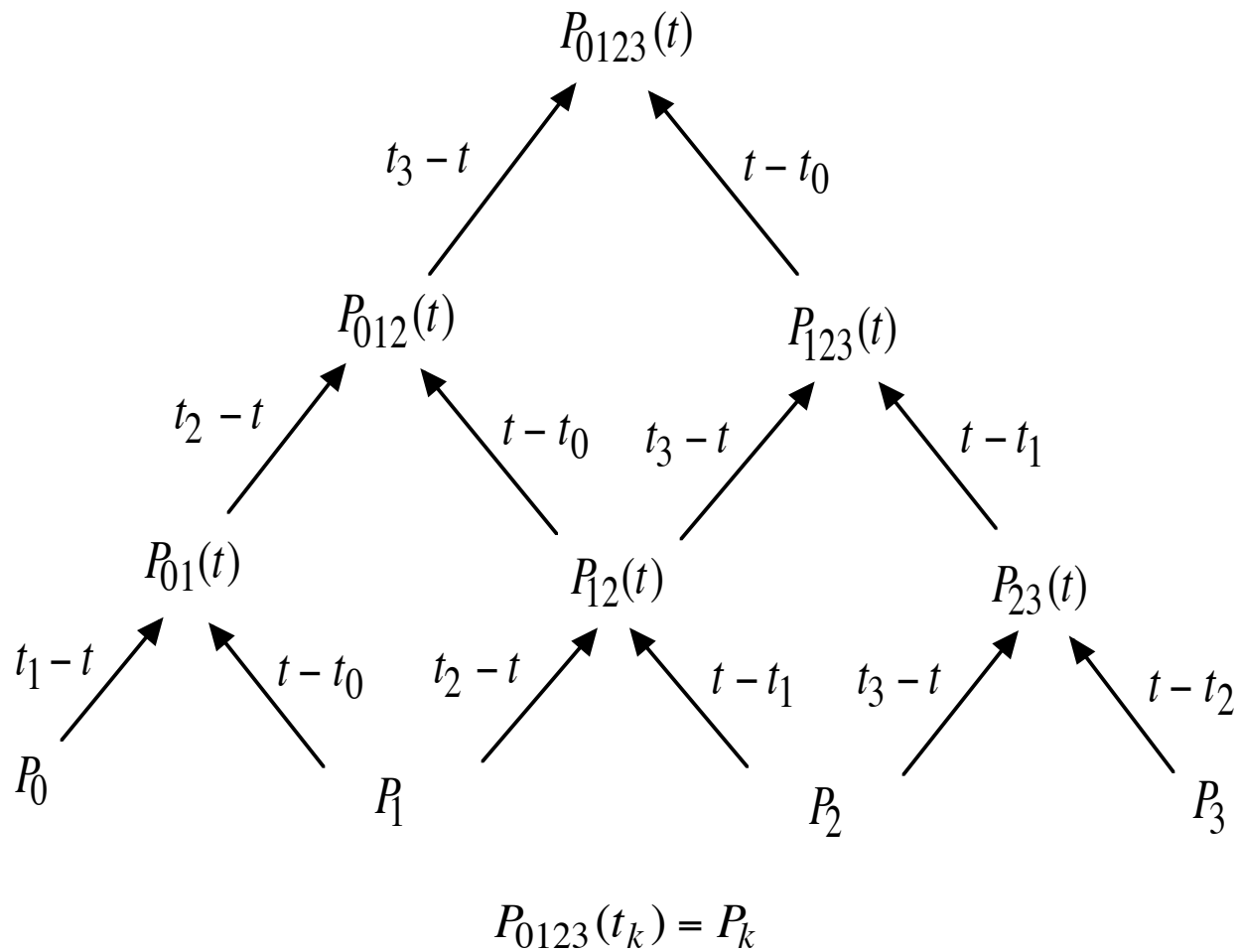
$$P_{0\dots n}(t_k) = P_k \quad k = 0, \dots, n .$$

Since  $P_{0\dots n-1}(t)$  and  $P_{1\dots n}(t)$  are polynomials of degree  $n - 1$ , it follows that  $P_{0\dots n}(t)$  is a polynomial of degree  $n$ . ♦

## Neville's Algorithm -- Recursive Calls



## Neville's Algorithm -- Dynamic Programming



## Advantages of Neville's Algorithm

### 1. *Numerically Stable*

- Uses the Given Data Directly
- No Need to Represent the Polynomial in the Basis  $1, t, t^2, \dots$

### 2. *Fast*

- Dynamic Programming
- $O(n^2)$  vs.  $O(2^n)$

### 3. *Simple Structure*

- Parallel Property
- Strip off First and Last Indices

### 4. *Easy to Update*

- Add a Computation of  $O(n)$  Instead of Redoing Work of  $O(n^2)$ .

## Polynomial Algebra

**Theorem:** *A non-zero polynomial of degree less than or equal to  $n$  can have at most  $n$  roots.*

Proof: Recall that if  $P(t)$  is a polynomial, then

$r$  is a root of  $P(t) \Leftrightarrow t - r$  is a factor of  $P(t)$ .

Now a polynomial of degree at most  $n$  can have at most  $n$  linear factors.

Therefore a polynomial of degree less than or equal to  $n$  can have at most  $n$  roots.



**Corollary:** *The only polynomial of degree less than or equal to  $n$  with more than  $n$  roots is the zero polynomial.*

## Uniqueness of Lagrange Interpolation

**Theorem:** *Given points  $P_0, \dots, P_n$  and parameters  $t_0, \dots, t_n$ , there exists only one polynomial curve  $P_{0\dots n}(t)$  of degree  $n$  that interpolates the given points at the specified parameter values. That is, the curve generated by Neville's algorithm is unique*

Proof: Suppose that there are two polynomials of degree  $n$  such that

$$P_{0\dots n}(t_k) = P_k \quad k = 0, \dots, n$$

$$Q_{0\dots n}(t_k) = P_k \quad k = 0, \dots, n.$$

Define

$$R_{0\dots n}(t) = Q_{0\dots n}(t) - P_{0\dots n}(t).$$

Then  $R_{0\dots n}(t)$  is a polynomial of degree at most  $n$ . But

$$R_{0\dots n}(t_k) = Q_{0\dots n}(t_k) - P_{0\dots n}(t_k) = 0 \quad k = 0, \dots, n,$$

so  $R_{0\dots n}(t)$  has  $n+1$  roots. Therefore  $R_{0\dots n}(t)$  must be the zero polynomial, so

$$P_{0\dots n}(t) = Q_{0\dots n}(t).$$



## Uniqueness of Lagrange Interpolation (continued)

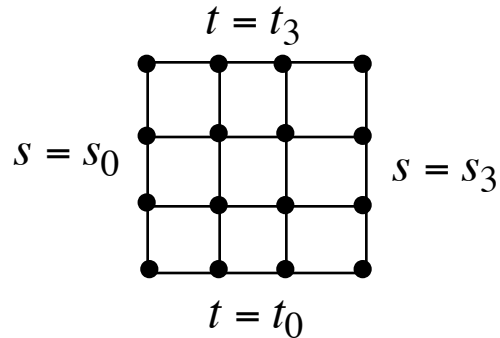
**Corollary:** *The Lagrange interpolant reproduces polynomials. That is, if the points  $P_0, \dots, P_n$  lie at the parameters  $t_0, \dots, t_n$  on a polynomial  $P(t)$  of degree less than or equal to  $n$ , then  $P_{0\dots n}(t) = P(t)$ .*

### *Observations*

- Uniqueness applies only for fixed nodes.
- Changing the nodes  $t_0, \dots, t_n$ , changes the Lagrange interpolant, even if the interpolation points  $P_0, \dots, P_n$  are exactly the same.

## Tensor Product Surface Interpolation

*Setup*



(a) Domain -- Rectangular Grid

$P_{03}$	$P_{13}$	$P_{23}$	$P_{33}$
$P_{02}$	$P_{12}$	$P_{22}$	$P_{32}$
$P_{01}$	$P_{11}$	$P_{21}$	$P_{31}$
$P_{00}$	$P_{10}$	$P_{20}$	$P_{30}$

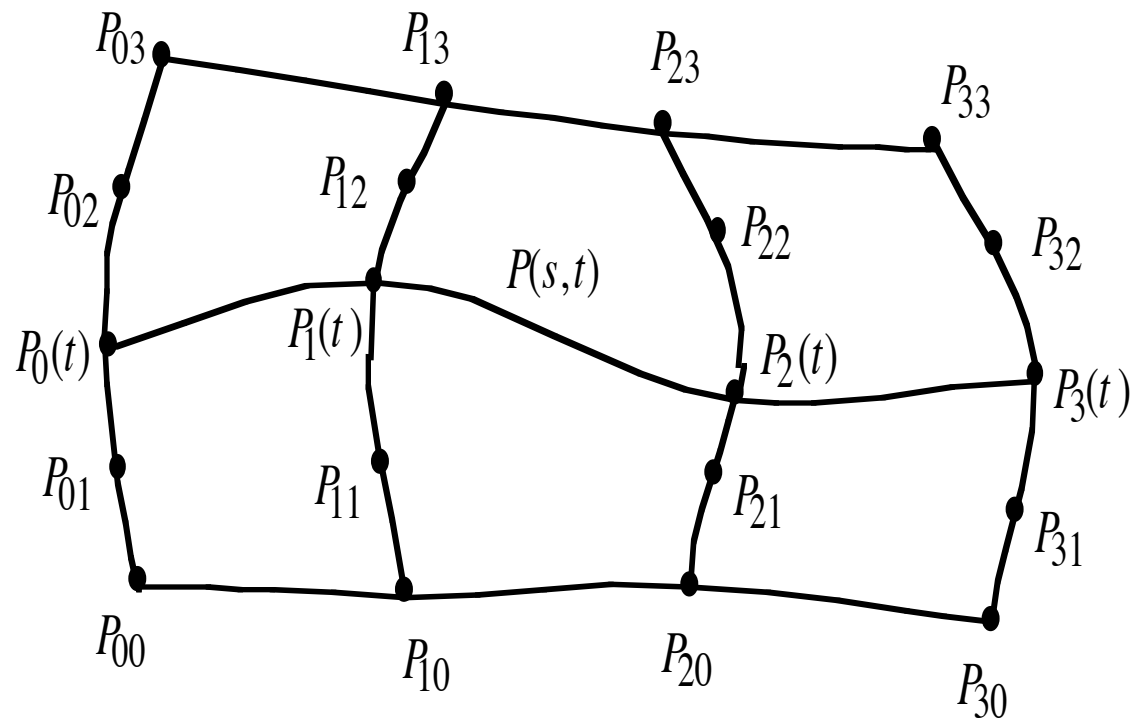
(b) Range -- Rectangular Array of Points

*Problem*

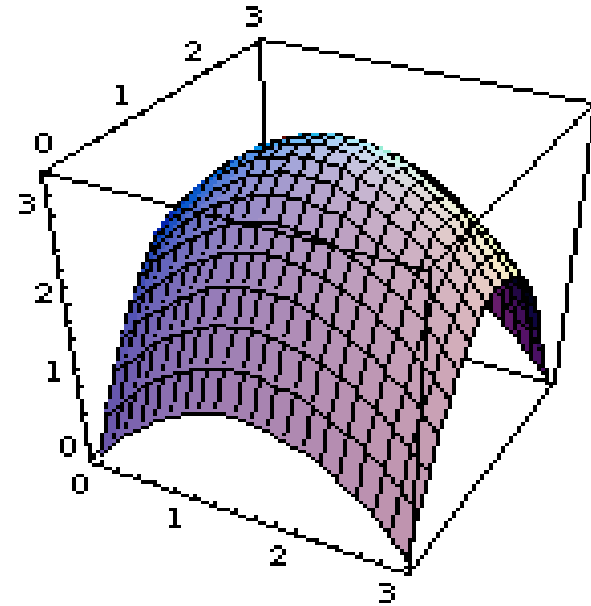
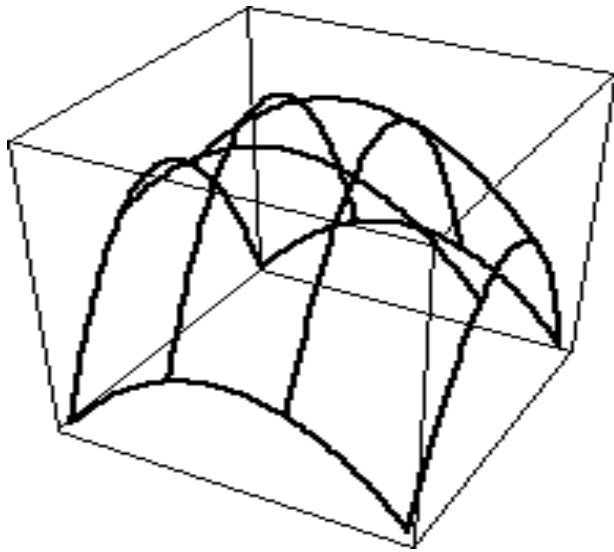
Find a smooth surface  $P(s, t)$  such that:

$$P(s_i, t_j) = P_{ij}$$

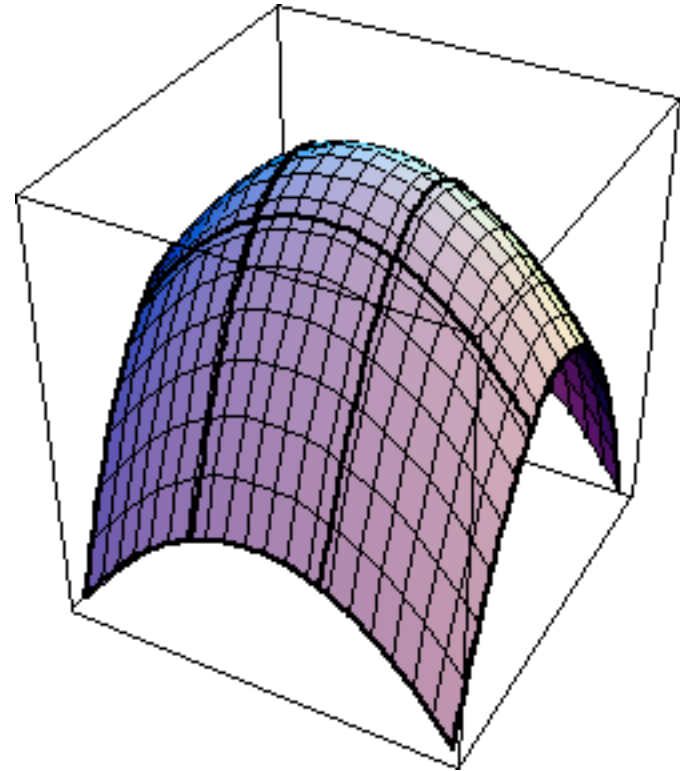
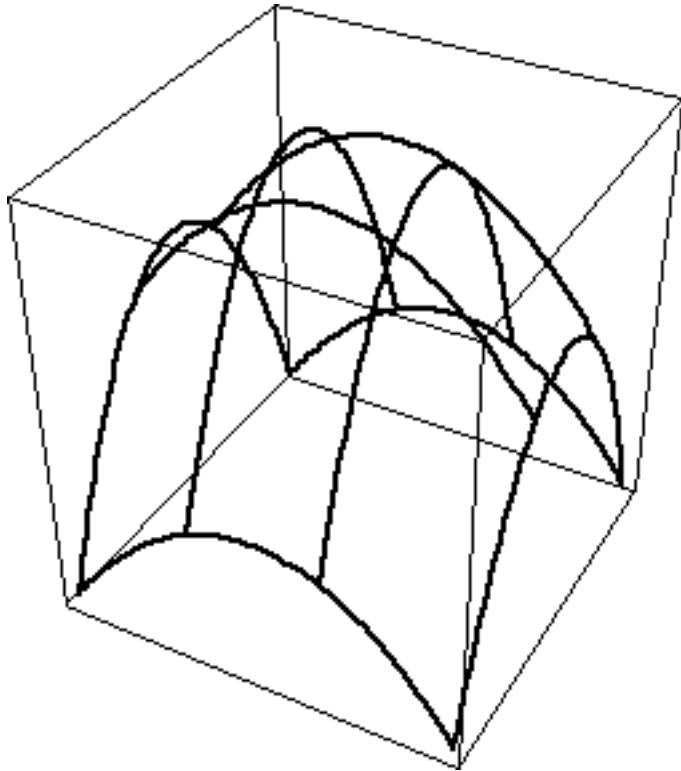
## Surface Interpolation



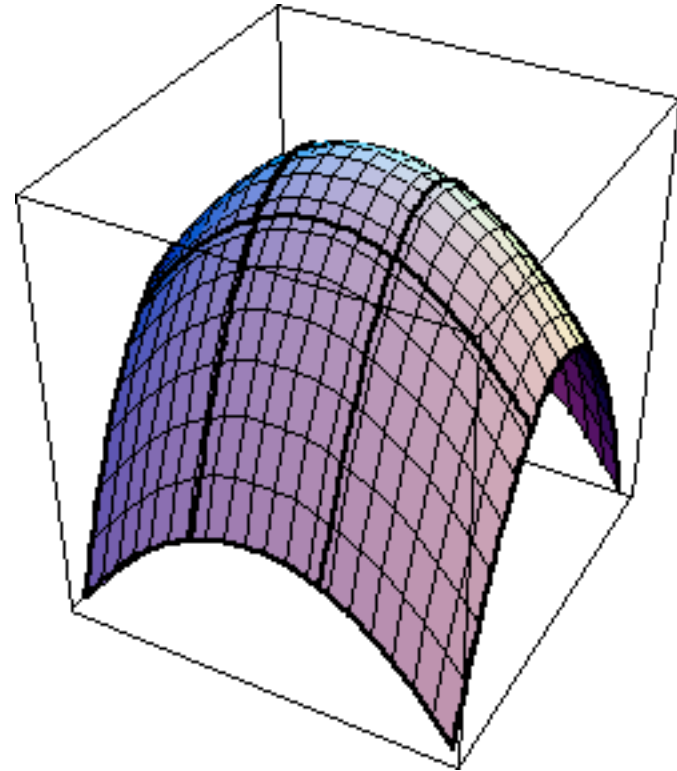
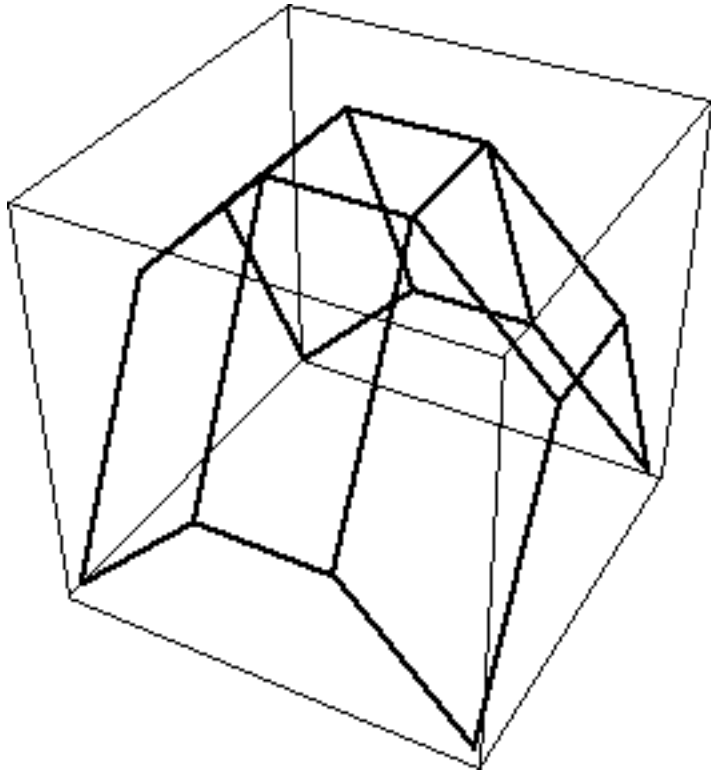
## Surface Interpolation



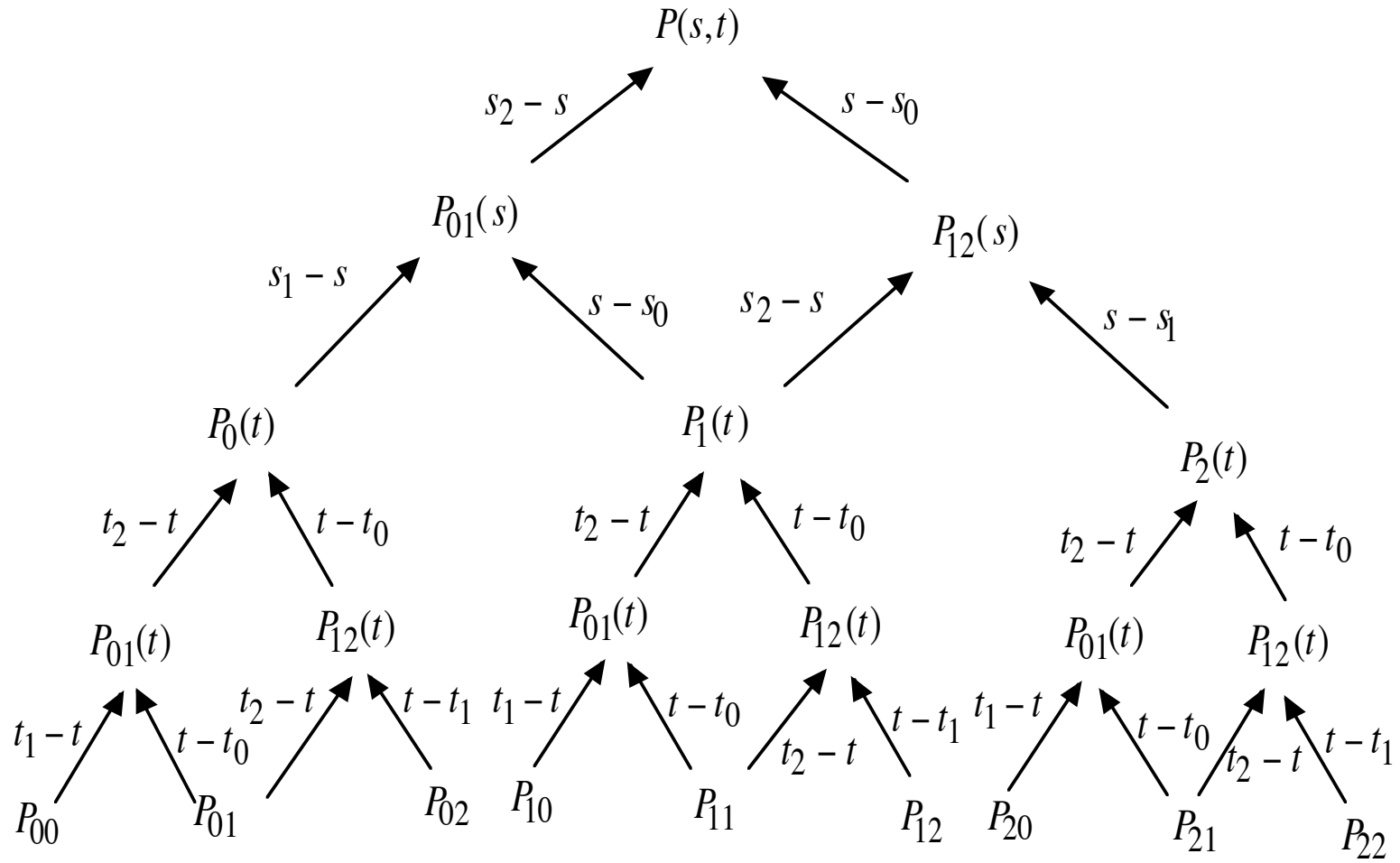
## Surface Interpolation



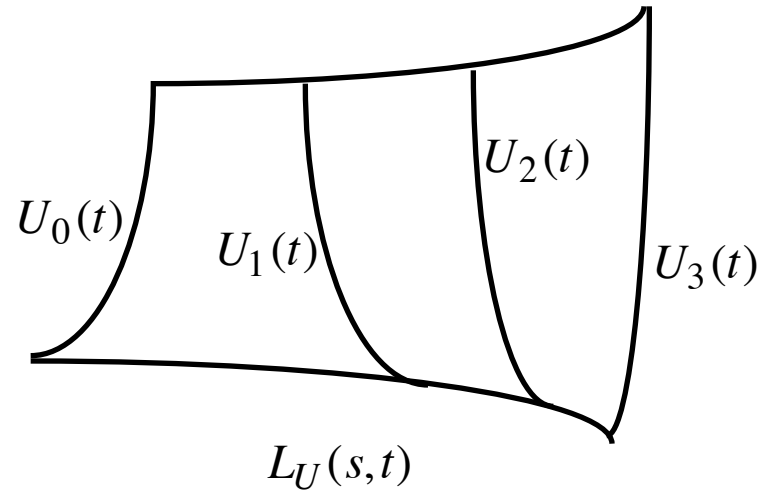
## Surface Interpolation



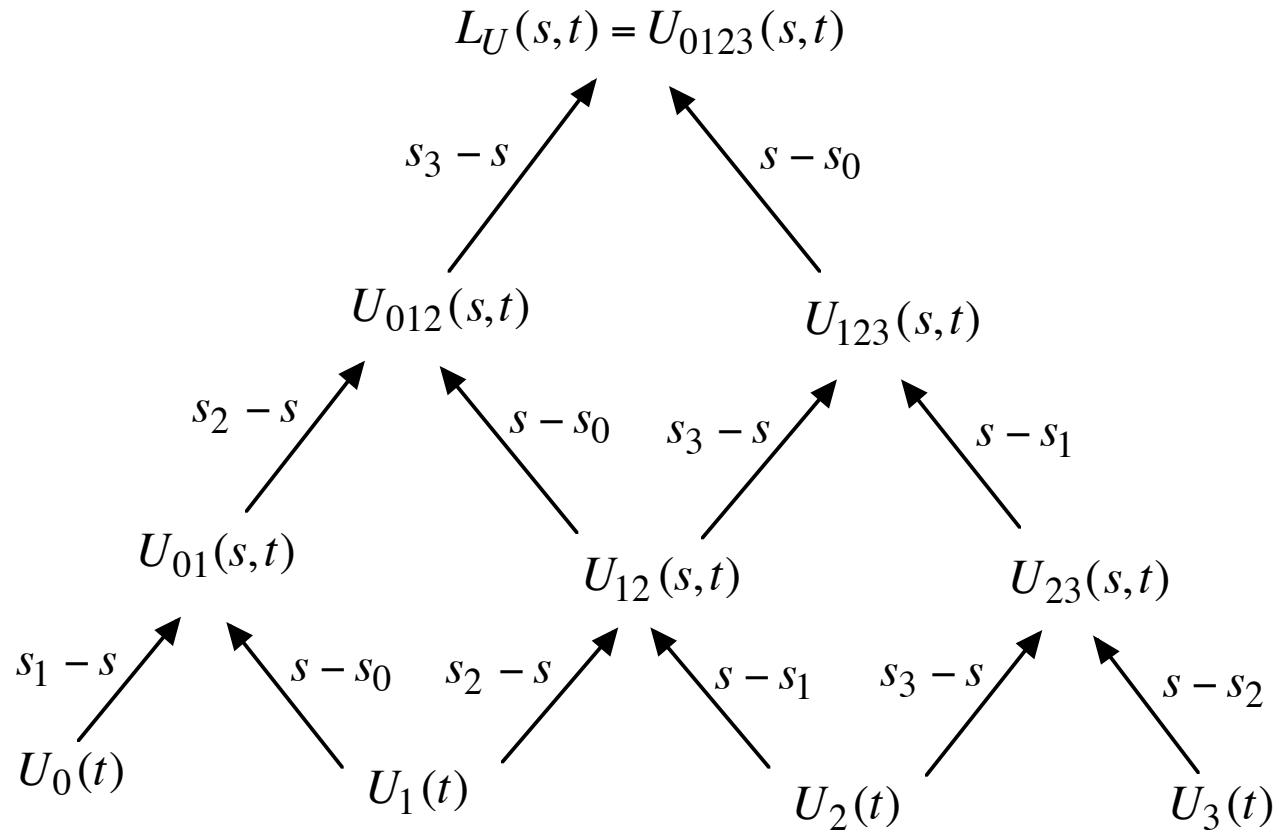
## Neville's Algorithm for Tensor Product Surfaces



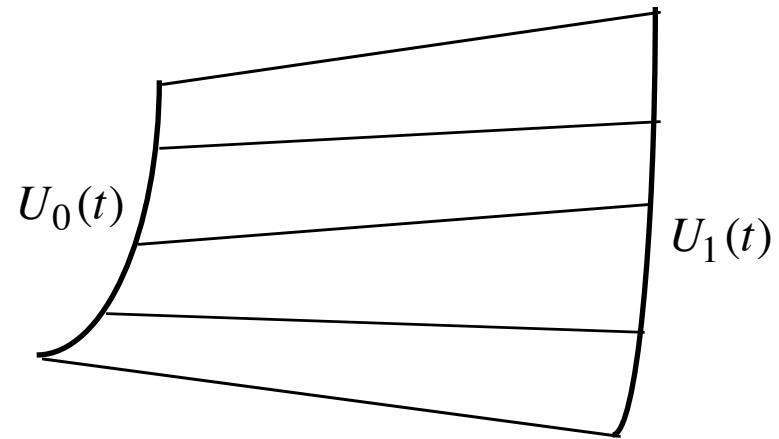
## Lofted Surface



## Neville's Algorithm for Lofted Surfaces



## Ruled Surface



$$R(s, t) = \frac{s_1 - s}{s_1 - s_0} U_0(t) + \frac{s - s_0}{s_1 - s_0} U_1(t)$$

## Summary

### *Key Ideas*

- Linear Interpolation
- Dynamic Programming -- Neville's Algorithm
- Extensions to Surfaces
  - Tensor Product
  - Lofted
  - Ruled

## Themes

### *Linearity*

- Mathematics is Easy
- Represent Complicated Curves and Surfaces by (Successive) Linear Interpolations

### *Polynomial Curves and Surfaces*

- Lagrange Interpolation -- Neville's Algorithm
- Bezier Approximation -- de Casteljau's Algorithm
- B-Splines -- de Boor's Algorithm
- Blossoming