Representing Meshes
Parametric Curves

Michael Kazhdan

(601.457/657)

Outline

- Representing Meshes
- Parametric Curves

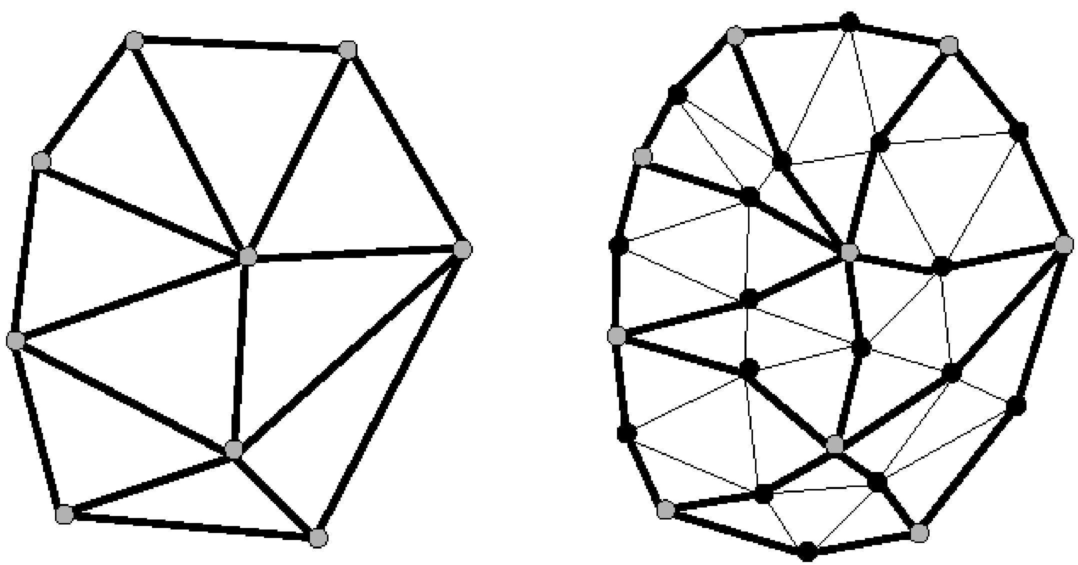
Key Questions

How to refine the mesh?

- Aim for properties like smoothness

How to store the mesh?

- Aim for efficiency in implementing subdivision rules

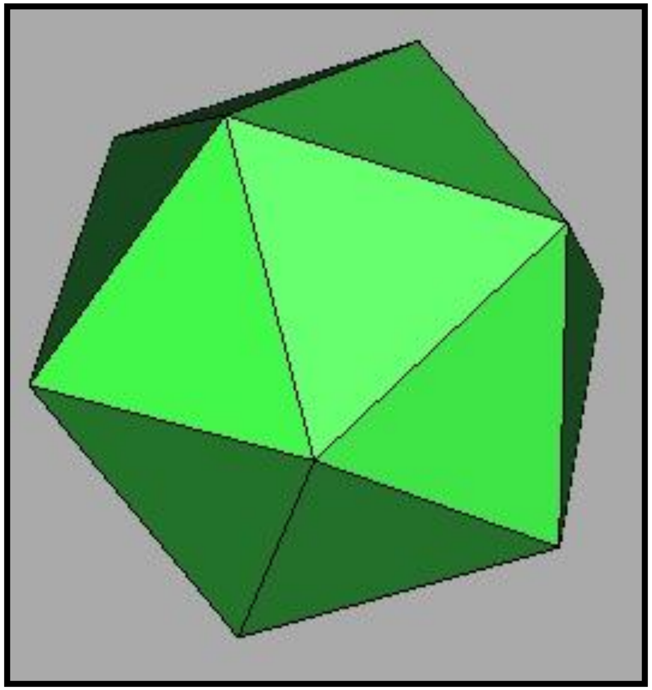


Zorin & Schroeder
SIGGRAPH 99
Course Notes

Polygon Meshes

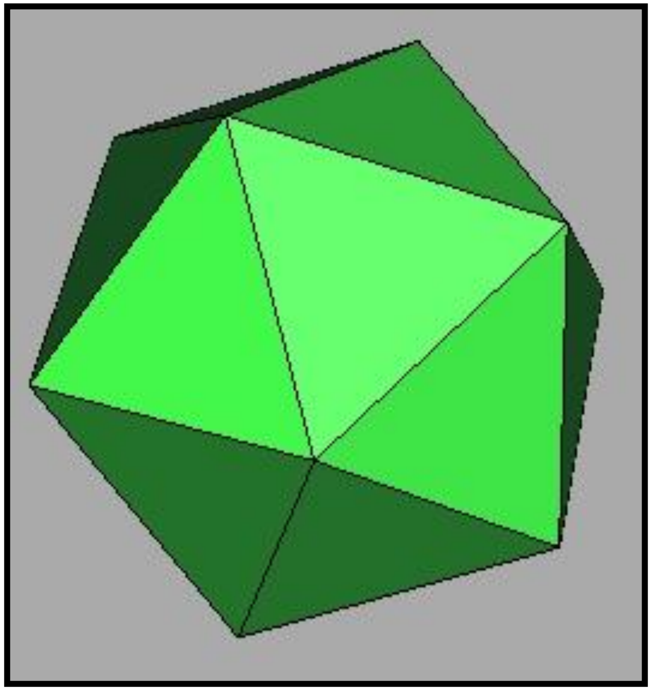
Mesh Representations

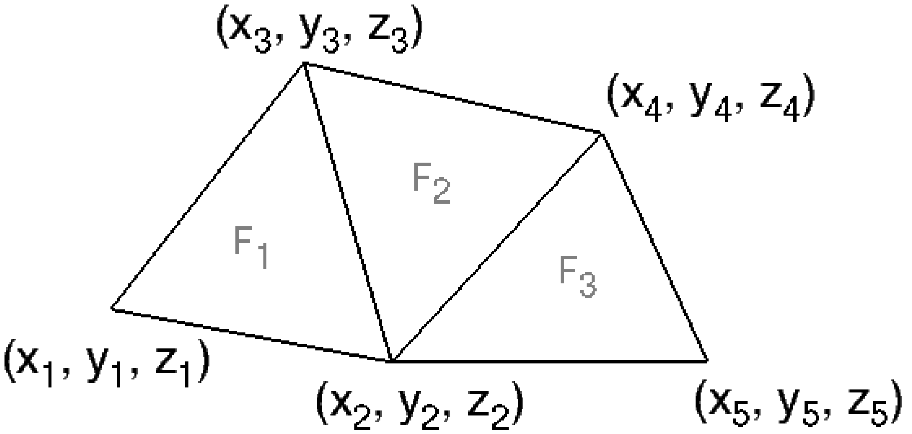
- Independent faces
- Vertex and face tables
- Adjacency lists
- Winged-Edge



Independent Faces

Each face lists vertex coordinates



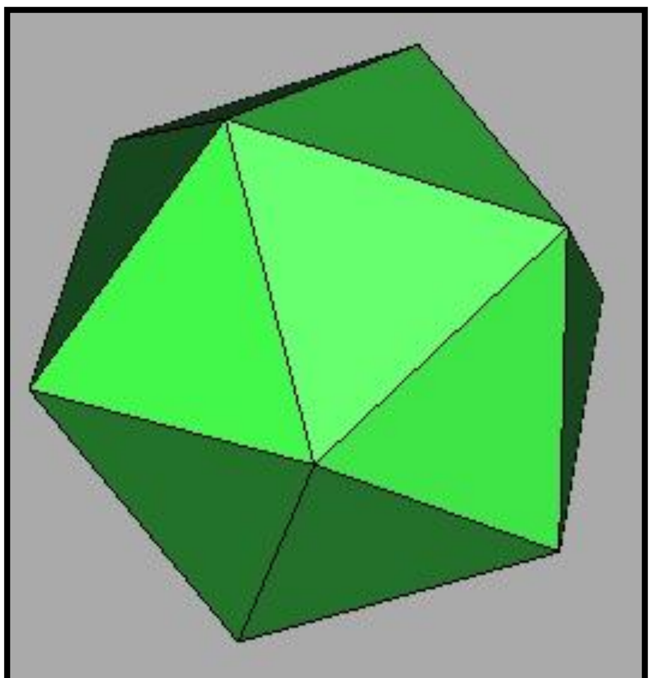


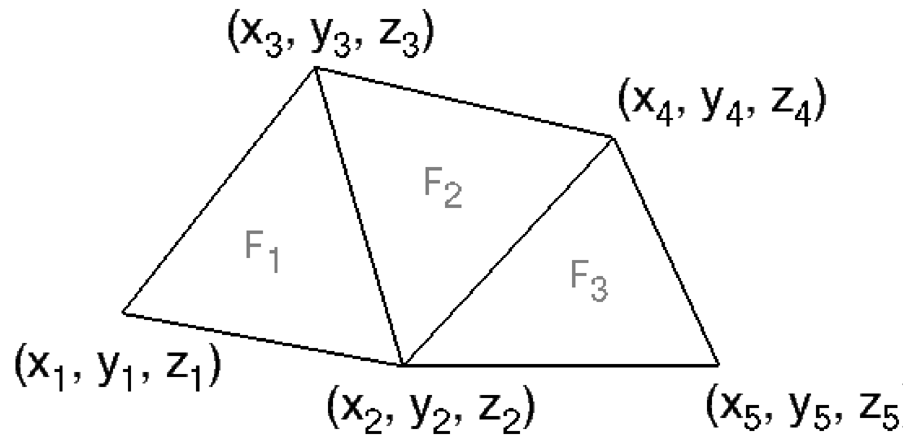
FACE TABLE	
F_1	(x_1, y_1, z_1) (x_2, y_2, z_2) (x_3, y_3, z_3)
F_2	(x_2, y_2, z_2) (x_4, y_4, z_4) (x_3, y_3, z_3)
F_3	(x_2, y_2, z_2) (x_5, y_5, z_5) (x_4, y_4, z_4)

Independent Faces

Each face lists vertex coordinates

- Redundant vertices





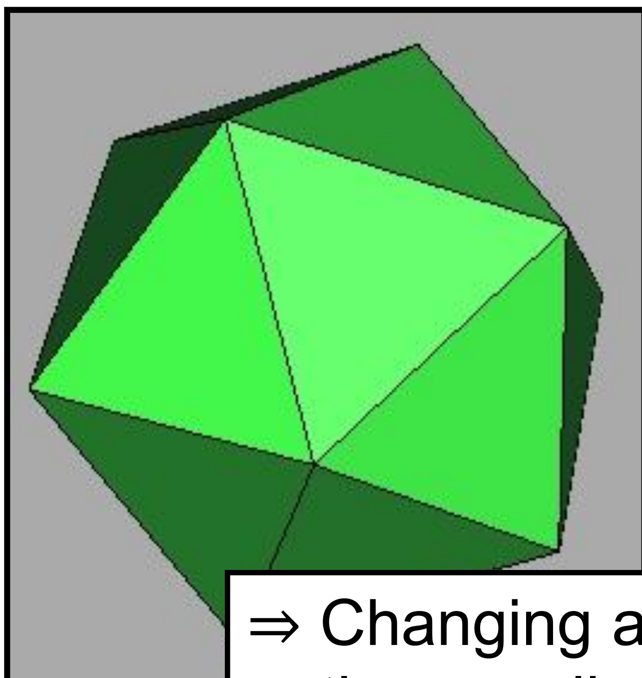
FACE TABLE

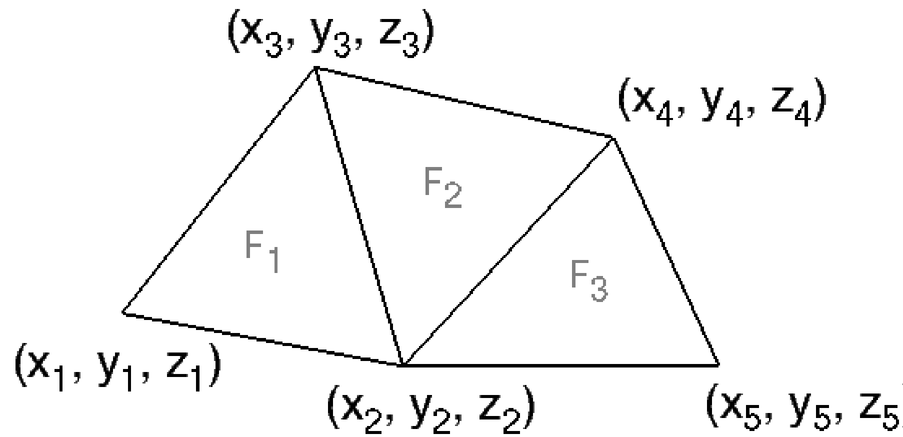
F_1	(x_1, y_1, z_1)	(x_2, y_2, z_2)	(x_3, y_3, z_3)
F_2	(x_2, y_2, z_2)	(x_4, y_4, z_4)	(x_3, y_3, z_3)
F_3	(x_2, y_2, z_2)	(x_5, y_5, z_5)	(x_4, y_4, z_4)

Independent Faces

Each face lists vertex coordinates

- Redundant vertices





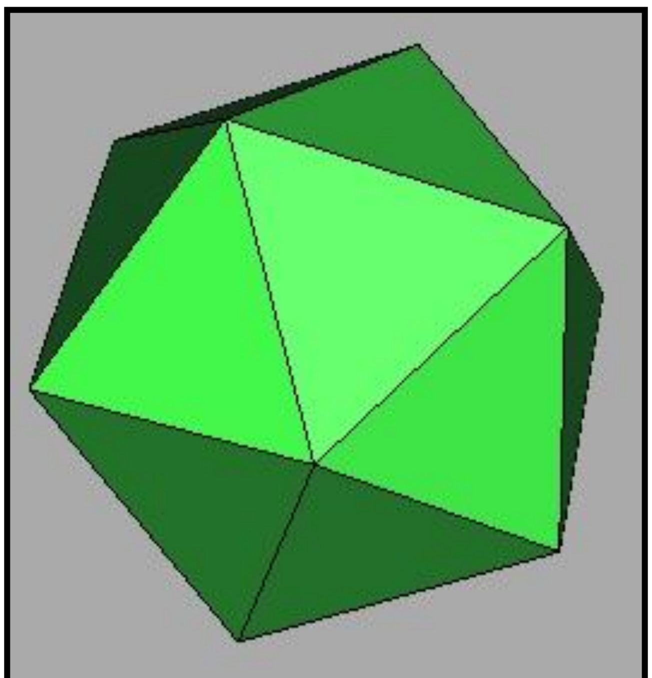
FACE TABLE			
F_1	(x_1, y_1, z_1)	(x_2, y_2, z_2)	(x_3, y_3, z_3)
F_2	(x_2, y_2, z_2)	(x_4, y_4, z_4)	(x_3, y_3, z_3)
[illegible]	[illegible]	[illegible]	(x_4, y_4, z_4)

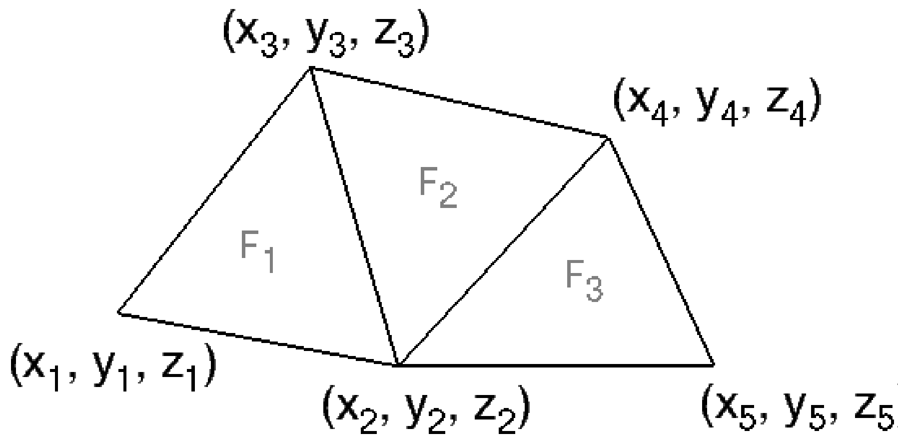
⇒ Changing a vertex requires changing the coordinates of **each** instance.

Independent Faces

Each face lists vertex coordinates

- Redundant vertices
- No (efficient/precise) vertex-adjacency info

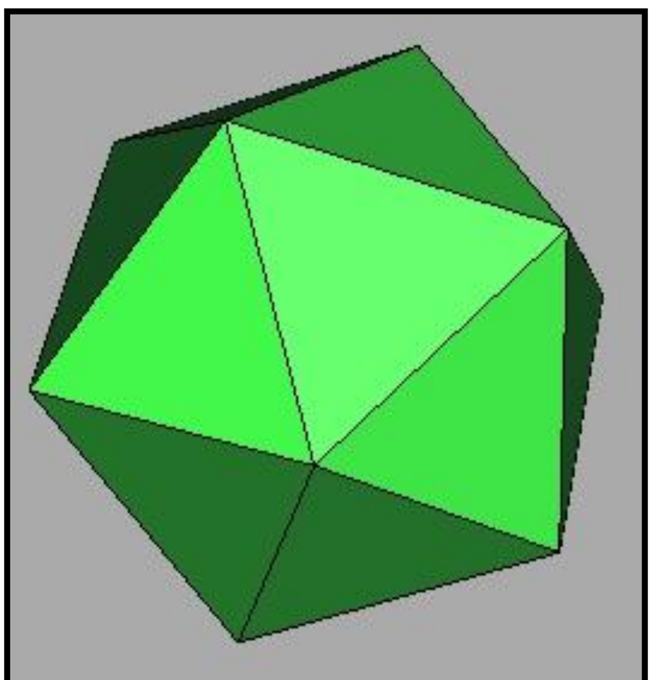


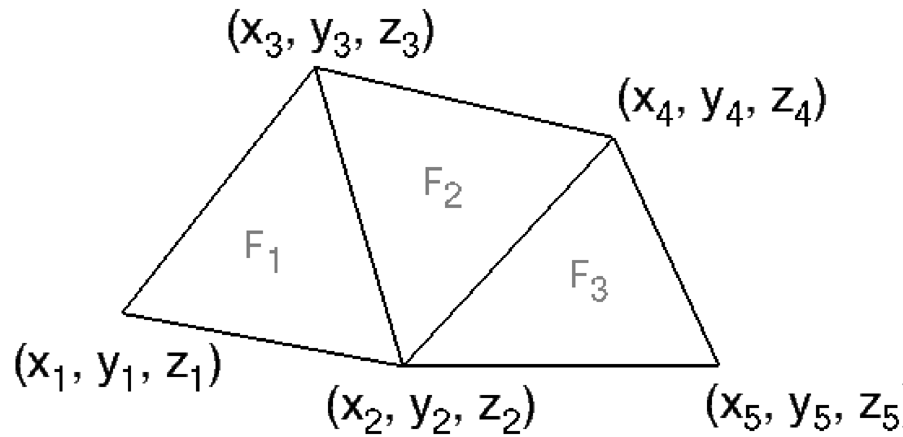


FACE TABLE	
F_1	(x_1, y_1, z_1) (x_2, y_2, z_2) (x_3, y_3, z_3)
F_2	(x_2, y_2, z_2) (x_4, y_4, z_4) (x_3, y_3, z_3)
F_3	(x_2, y_2, z_2) (x_5, y_5, z_5) (x_4, y_4, z_4)

Vertex and Face Tables

Each face lists vertex references





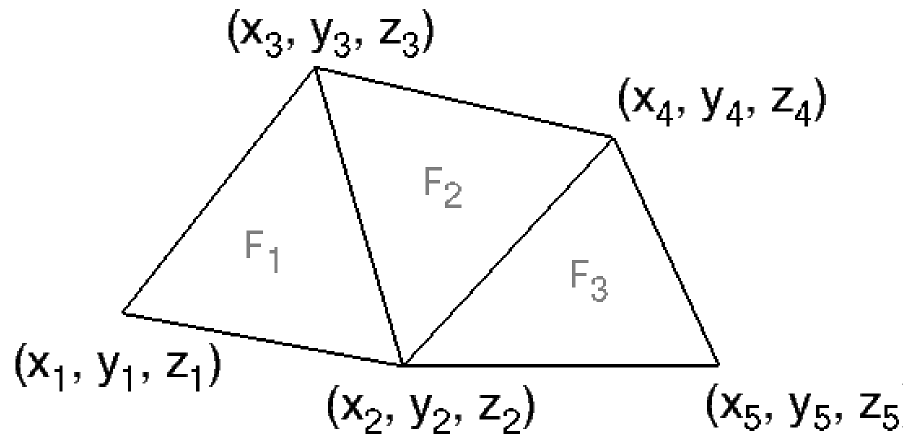
VERTEX TABLE			
V_1	X_1	Y_1	Z_1
V_2	X_2	Y_2	Z_2
V_3	X_3	Y_3	Z_3
V_4	X_4	Y_4	Z_4
V_5	X_5	Y_5	Z_5

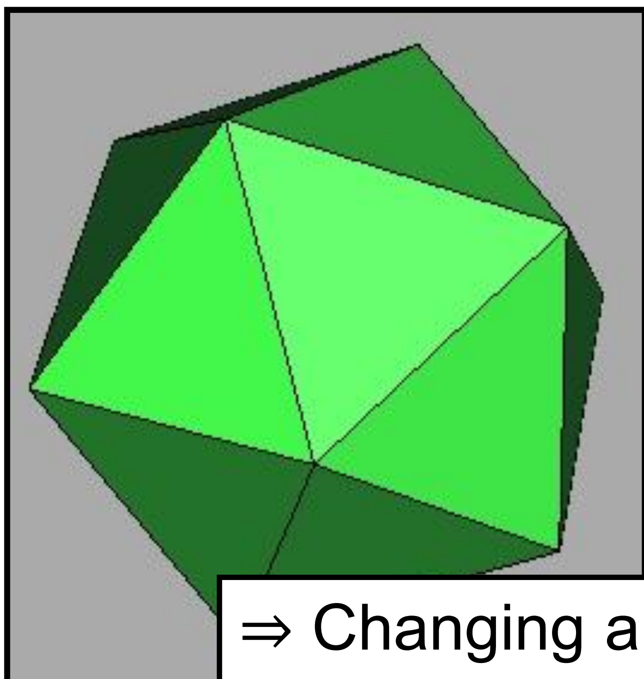
FACE TABLE			
F_1	V_1	V_2	V_3
F_2	V_2	V_4	V_3
F_3	V_2	V_5	V_4

Vertex and Face Tables

Each face lists vertex references

- ✓ Shared vertices





VERTEX TABLE			
V_1	X_1	Y_1	Z_1
V_2	X_2	Y_2	Z_2
V_3	X_3	Y_3	Z_3
V_4	X_4	Y_4	Z_4

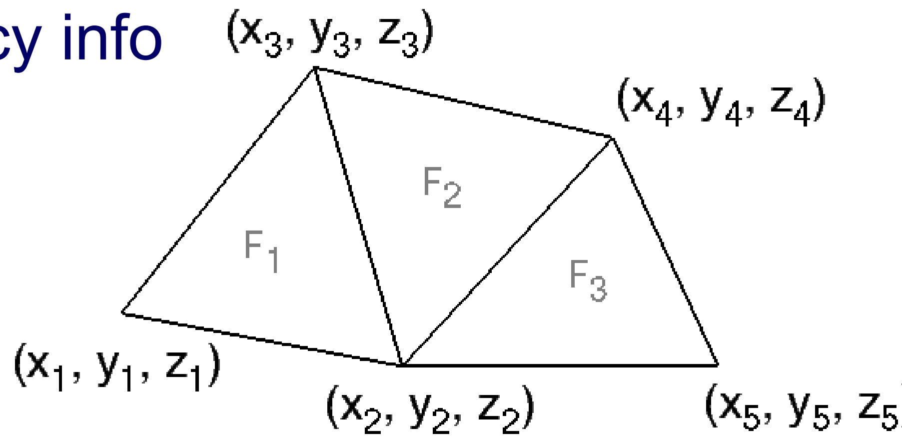
FACE TABLE			
F_1	V_1	V_2	V_3
F_2	V_2	V_4	V_3
F_3	V_2	V_5	V_4

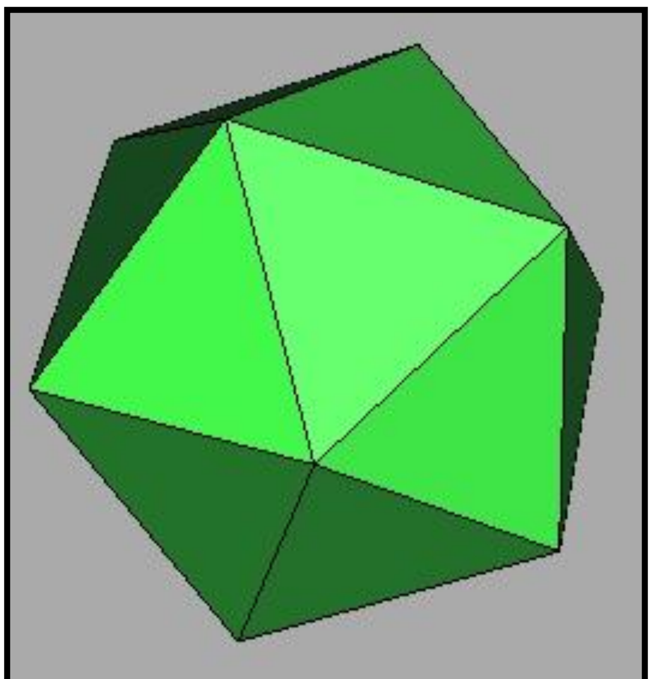
⇒ Changing a vertex requires changing the coordinates of a **single** point.

Vertex and Face Tables

Each face lists vertex references

- ✓ Shared vertices
- ✗ No (efficient) adjacency info





VERTEX TABLE

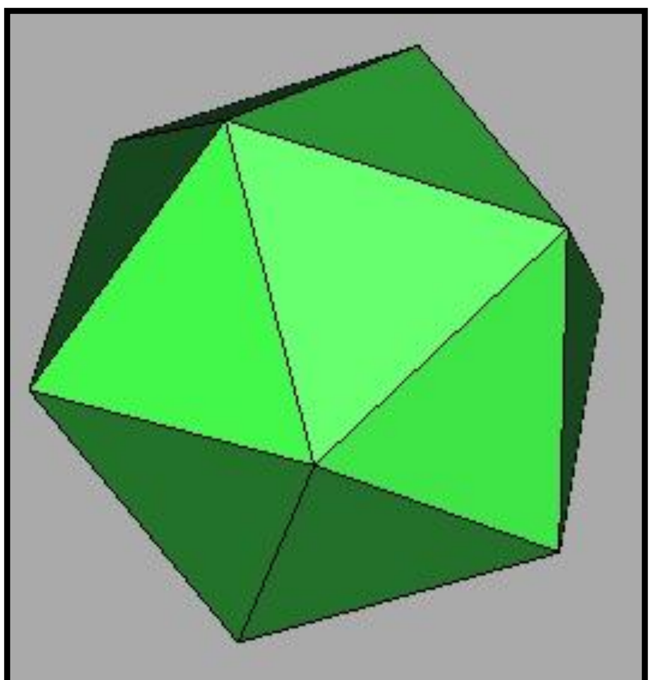
V_1	X_1	Y_1	Z_1
V_2	X_2	Y_2	Z_2
V_3	X_3	Y_3	Z_3
V_4	X_4	Y_4	Z_4
V_5	X_5	Y_5	Z_5

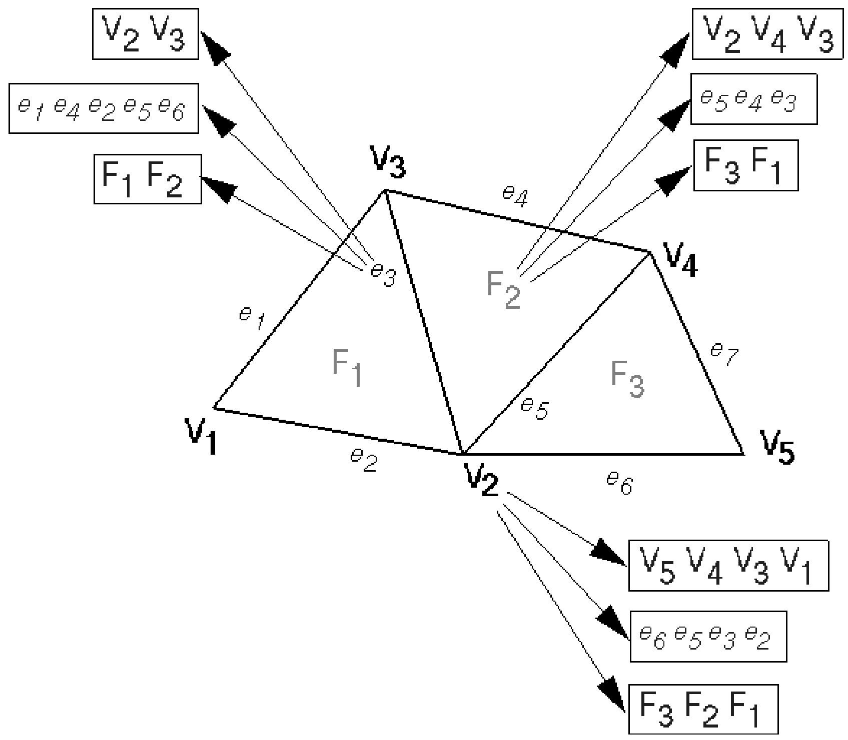
FACE TABLE

F_1	V_1	V_2	V_3
F_2	V_2	V_4	V_3
F_3	V_2	V_5	V_4

Adjacency Lists

Store all vertex, edge, and face adjacencies

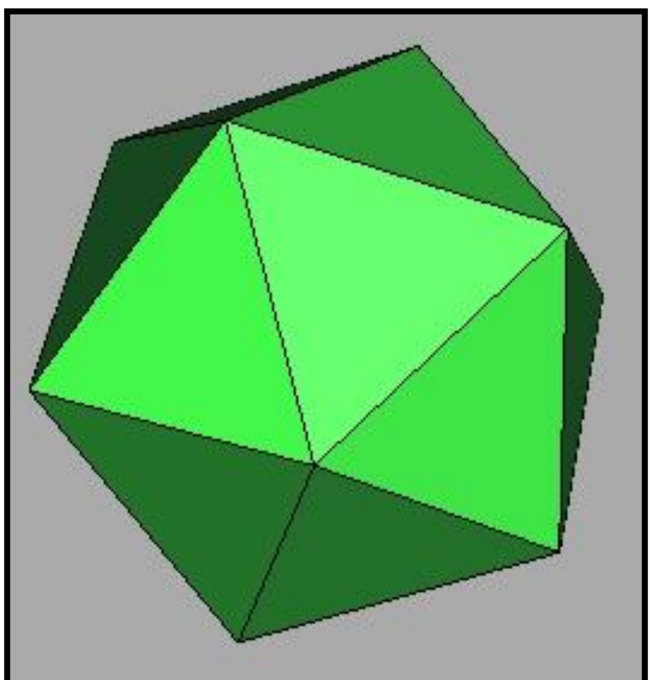


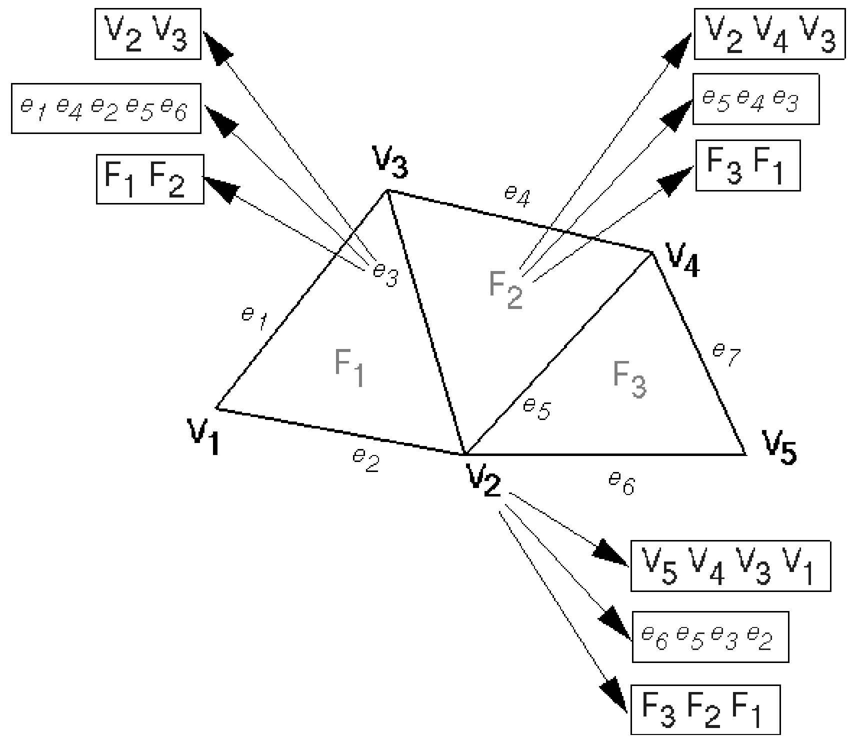


Adjacency Lists

Store all vertex, edge, and face adjacencies

- ✓ Efficient adjacency info
- ✗ Extra storage
- ✗ Variable size arrays

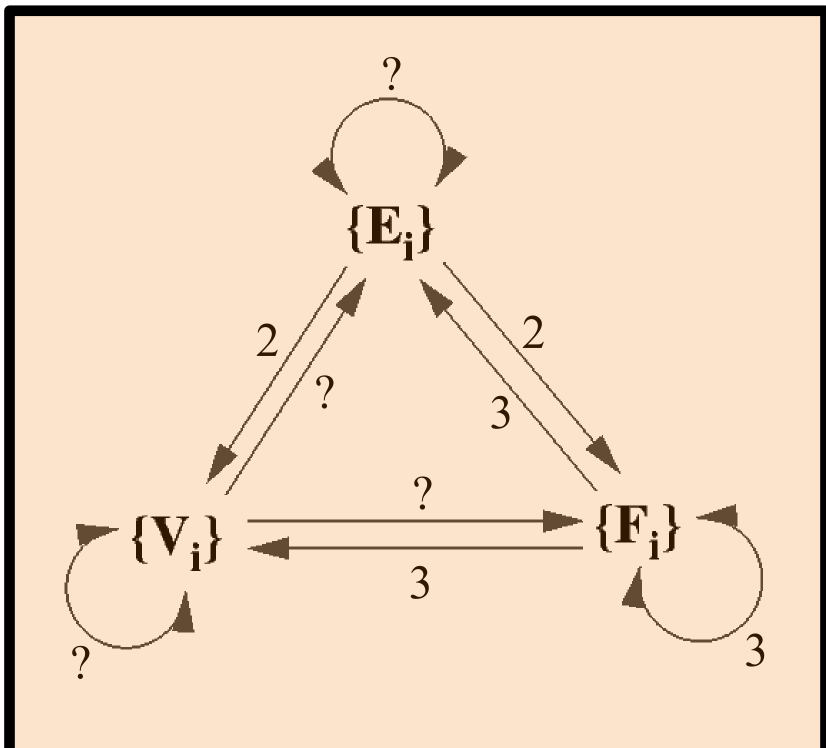


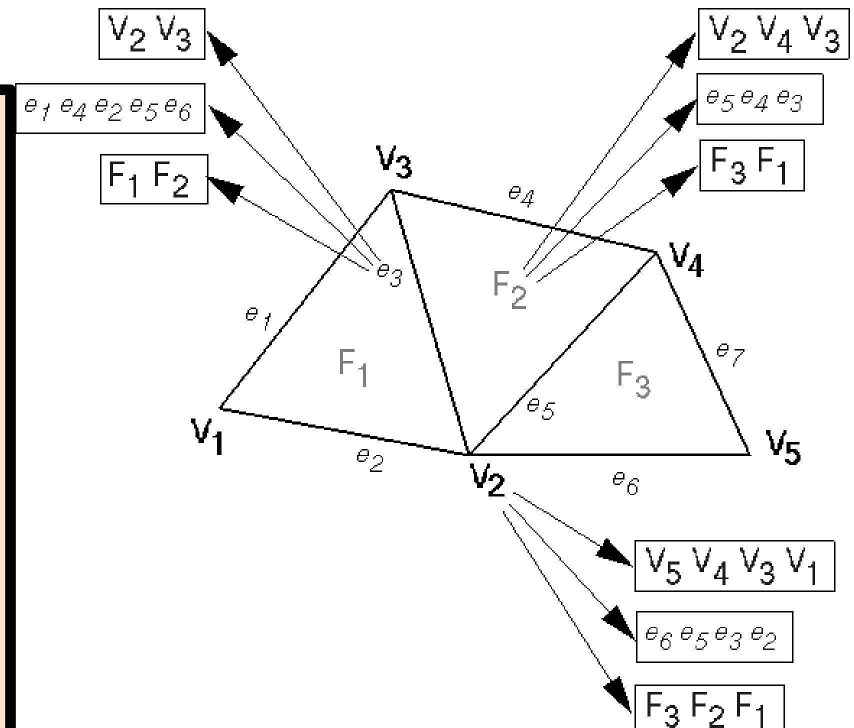


Partial Adjacency Lists

Store all vertex, edge, and face adjacencies

- ✓ Efficient adjacency info
- ✗ Extra storage
- ✗ Variable size arrays

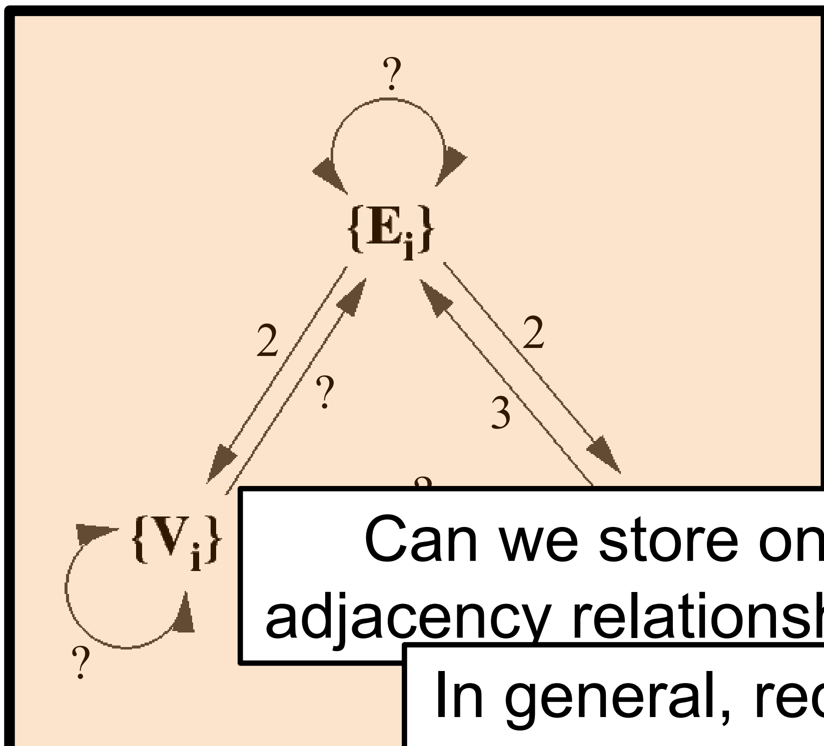


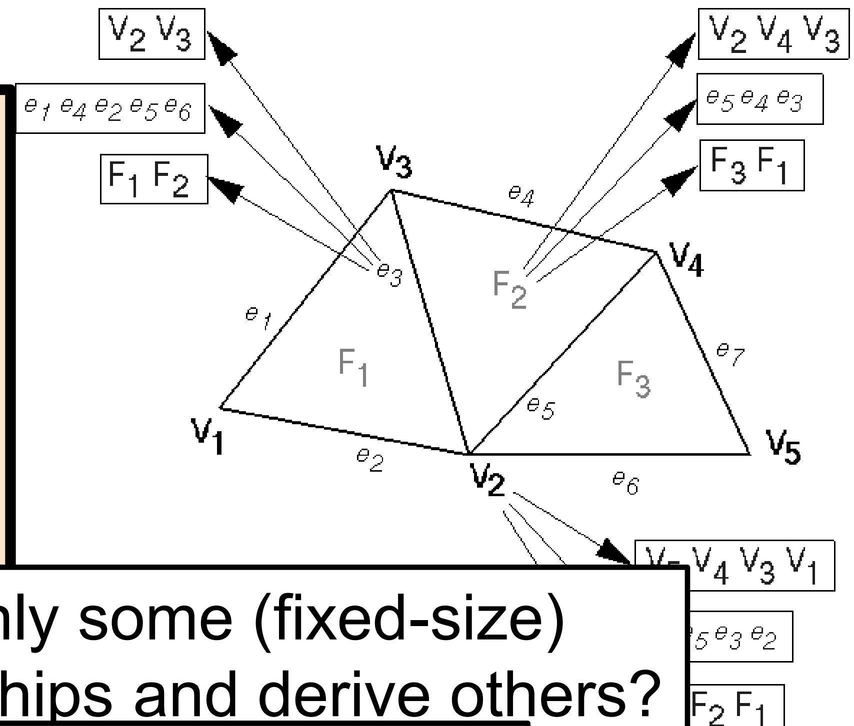


Partial Adjacency Lists

Store all vertex, edge, and face adjacencies

- ✓ Efficient adjacency info
- ✗ Extra storage
- ✗ Variable size arrays





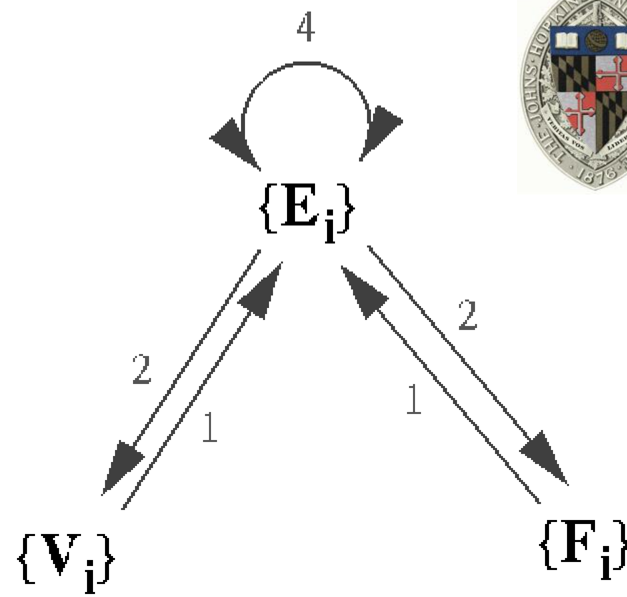
Can we store only some (fixed-size) adjacency relationships and derive others?

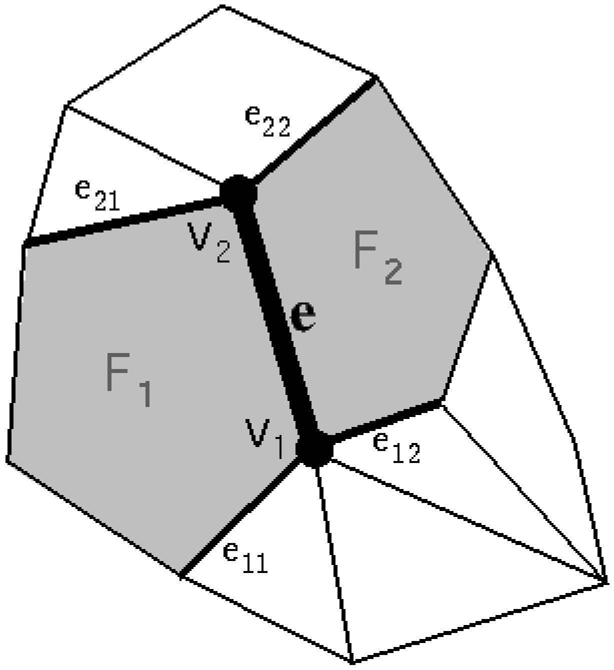
In general, redundancy creates opportunity for inconsistency.

Winged Edge

Adjacency encoded in edges

- ✓ All adjacencies in $O(1)$ time
- ✓ Little extra storage
- ✓ Fixed-size records
- ✓ Supports polygonal faces
- ✗ Mesh needs to be oriented



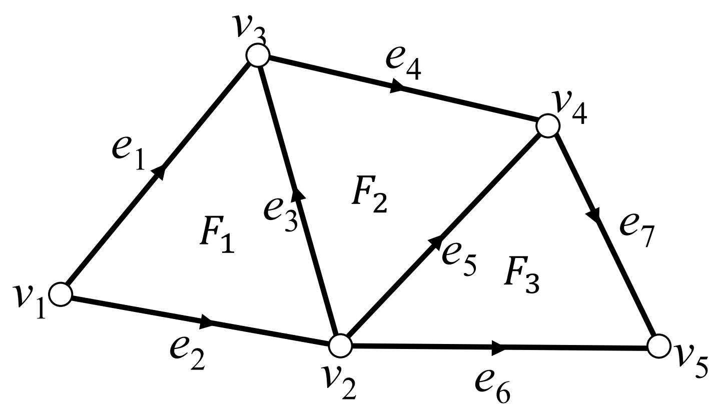


Each edge stores: 4 “wing” edges 2 vertices 2 faces Each face stores: 1 (some) edge Each vertex stores: 1 (some) edge

Winged Edge

Vertex table:

- A reference to some incident edge



VERTEX TABLE				
V_1	X_1	Y_1	Z_1	e_1
V_2	X_2	Y_2	Z_2	e_6
V_3	X_3	Y_3	Z_3	e_3
V_4	X_4	Y_4	Z_4	e_5
V_5	X_5	Y_5	Z_5	e_6

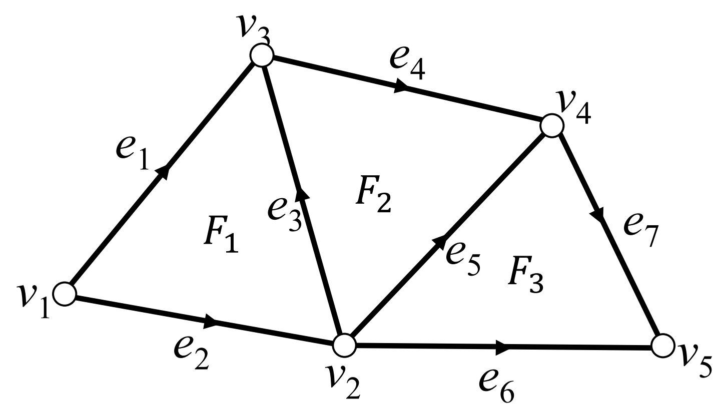
EDGE TABLE	S	E	L	R	S L	S R	E L	E R
e_1	V_1	V_3		F_1	e_2	e_2	e_4	e_3
e_2	V_1	V_2	F_1		e_1	e_1	e_3	e_6
e_3	V_2	V_3	F_1	F_2	e_2	e_5	e_1	e_4
e_4	V3	V_4		F_2	e_1	e_3	e_7	e_5
e_5	V_2	V_4	F_2	F_3	e_3	e_6	e_4	e_7
e_6	V_2	V_5	F_3		e_5	e_2	e_7	e_7
e_7	V_4	V_5		F_3	e_4	e_5	e_6	e_6

FACE TABLE	
F_1	e_1
F_2	e_3
F_3	e_5

Winged Edge

Vertex table:

- A reference to some incident edge
- Vertex positions (and other attributes)



VERTEX TABLE				
V_1	X_1	Y_1	Z_1	e_1
V_2	X_2	Y_2	Z_2	e_6
V_3	X_3	Y_3	Z_3	e_3
V_4	X_4	Y_4	Z_4	e_5
V_5	X_5	Y_5	Z_5	e_6

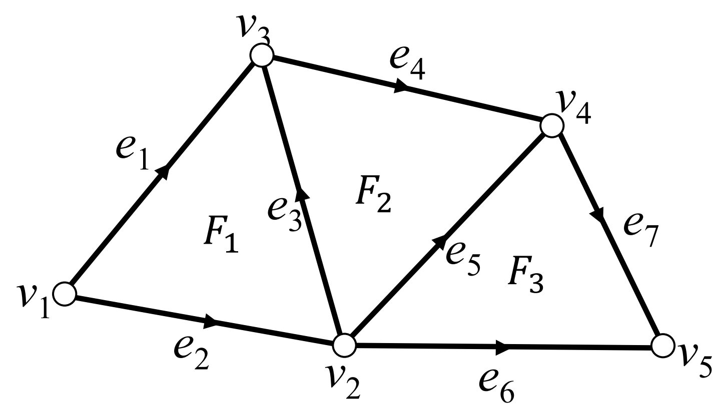
EDGE TABLE					S		E	
	S	E	L	R	L	R	L	R
e_1	V_1	V_3		F_1	e_2	e_2	e_4	e_3
e_2	V_1	V_2	F_1		e_1	e_1	e_3	e_6
e_3	V_2	V_3	F_1	F_2	e_2	e_5	e_1	e_4
e_4	V_3	V_4		F_2	e_1	e_3	e_7	e_5
e_5	V_2	V_4	F_2	F_3	e_3	e_6	e_4	e_7
e_6	V_2	V_5	F_3		e_5	e_2	e_7	e_7
e_7	V_4	V_5		F_3	e_4	e_5	e_6	e_6

FACE TABLE	
F_1	e_1
F_2	e_3
F_3	e_5

Winged Edge

Face table:

- A reference to some incident edge
- (And other attributes)



VERTEX TABLE				
V_1	X_1	Y_1	Z_1	e_1
V_2	X_2	Y_2	Z_2	e_6
V_3	X_3	Y_3	Z_3	e_3
V_4	X_4	Y_4	Z_4	e_5
V_5	X_5	Y_5	Z_5	e_6

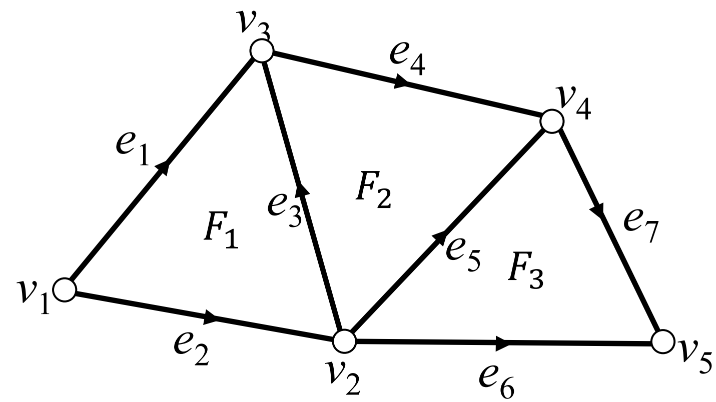
EDGE TABLE					S		E	
	S	E	L	R	L	R	L	R
e_1	V_1	V_3		F_1	e_2	e_2	e_4	e_3
e_2	V_1	V_2	F_1		e_1	e_1	e_3	e_6
e_3	V_2	V_3	F_1	F_2	e_2	e_5	e_1	e_4
e_4	$V3$	V_4		F_2	e_1	e_3	e_7	e_5
e_5	V_2	V_4	F_2	F_3	e_3	e_6	e_4	e_7
e_6	V_2	V_5	F_3		e_5	e_2	e_7	e_7
e_7	V_4	V_5		F_3	e_4	e_5	e_6	e_6

FACE TABLE	
F_1	e_1
F_2	e_3
F_3	e_5

Winged Edge

Edge table:

- References to **S**tart and **E**nd vertices (orientation arbitrary)



VERTEX TABLE				
V_1	X_1	Y_1	Z_1	e_1
V_2	X_2	Y_2	Z_2	e_6
V_3	X_3	Y_3	Z_3	e_3
V_4	X_4	Y_4	Z_4	e_5
V_5	X_5	Y_5	Z_5	e_6

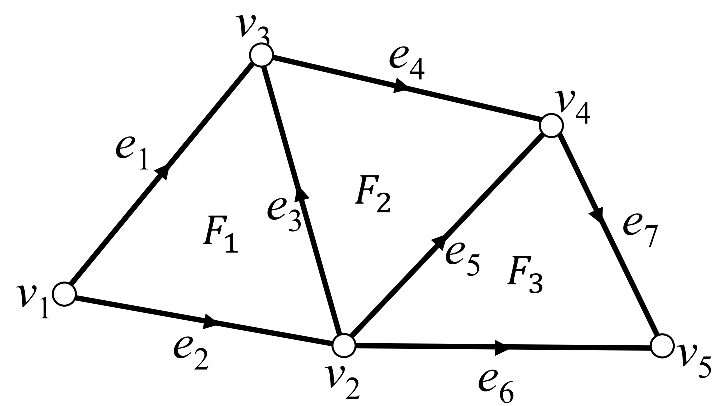
EDGE TABLE					S		E	
	S	E	L	R	L	R	L	R
e_1	V_1	V_3		F_1	e_2	e_2	e_4	e_3
e_2	V_1	V_2	F_1		e_1	e_1	e_3	e_6
e_3	V_2	V_3	F_1	F_2	e_2	e_5	e_1	e_4
e_4	$V3$	V_4		F_2	e_1	e_3	e_7	e_5
e_5	V_2	V_4	F_2	F_3	e_3	e_6	e_4	e_7
e_6	V_2	V_5	F_3		e_5	e_2	e_7	e_7
e_7	V_4	V_5		F_3	e_4	e_5	e_6	e_6

FACE TABLE	
F_1	e_1
F_2	e_3
F_3	e_5

Winged Edge

Edge table:

- References to **S**tart and **E**nd vertices (orientation arbitrary)
- References to **L**eft and **R**ight faces



VERTEX TABLE

V_1	X_1	Y_1	Z_1	e_1
V_2	X_2	Y_2	Z_2	e_6
V_3	X_3	Y_3	Z_3	e_3
V_4	X_4	Y_4	Z_4	e_5
V_5	X_5	Y_5	Z_5	e_6

EDGE TABLE

					S		E	
	S	E	L	R	L	R	L	R
e_1	V_1	V_3		F_1	e_2	e_2	e_4	e_3
e_2	V_1	V_2	F_1		e_1	e_1	e_3	e_6
e_3	V_2	V_3	F_1	F_2	e_2	e_5	e_1	e_4
e_4	$V3$	V_4		F_2	e_1	e_3	e_7	e_5
e_5	V_2	V_4	F_2	F_3	e_3	e_6	e_4	e_7
e_6	V_2	V_5	F_3		e_5	e_2	e_7	e_7
e_7	V_4	V_5		F_3	e_4	e_5	e_6	e_6

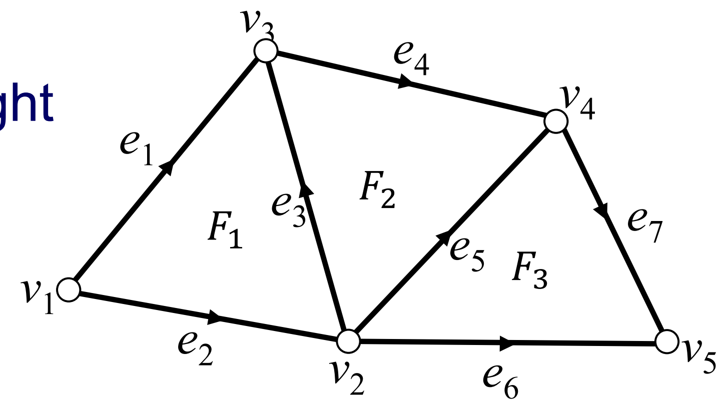
FACE TABLE

F_1	e_1
F_2	e_3
F_3	e_5

Winged Edge

Edge table:

- References to **S**tart and **E**nd vertices (orientation arbitrary)
- References to **L**eft and **R**ight faces
- References to immediate **L**eft and **R**ight edges coming out of the **S**tart vertex



VERTEX TABLE				
V_1	X_1	Y_1	Z_1	e_1
V_2	X_2	Y_2	Z_2	e_6
V_3	X_3	Y_3	Z_3	e_3
V_4	X_4	Y_4	Z_4	e_5
V_5	X_5	Y_5	Z_5	e_6

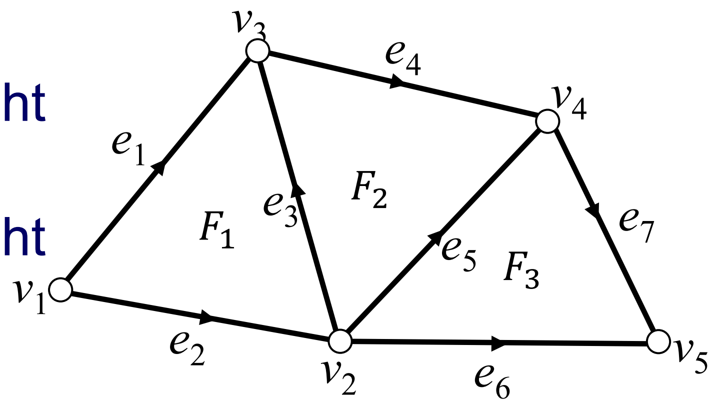
EDGE TABLE					S		E	
	S	E	L	R	L	R	L	R
e_1	V_1	V_3		F_1	e_2	e_2	e_4	e_3
e_2	V_1	V_2	F_1		e_1	e_1	e_3	e_6
e_3	V_2	V_3	F_1	F_2	e_2	e_5	e_1	e_4
e_4	V_3	V_4		F_2	e_1	e_3	e_7	e_5
e_5	V_2	V_4	F_2	F_3	e_3	e_6	e_4	e_7
e_6	V_2	V_5	F_3		e_5	e_2	e_7	e_7
e_7	V_4	V_5		F_3	e_4	e_5	e_6	e_6

FACE TABLE	
F_1	e_1
F_2	e_3
F_3	e_5

Winged Edge

Edge table:

- References to **S**tart and **E**nd vertices (orientation arbitrary)
- References to **L**eft and **R**ight faces
- References to immediate **L**eft and **R**ight edges coming out of the **S**tart vertex
- References to immediate **L**eft and **R**ight edges coming out of the **E**nd vertex



VERTEX TABLE

V_1	X_1	Y_1	Z_1	e_1
V_2	X_2	Y_2	Z_2	e_6
V_3	X_3	Y_3	Z_3	e_3
V_4	X_4	Y_4	Z_4	e_5
V_5	X_5	Y_5	Z_5	e_6

EDGE TABLE

					S		E	
	S	E	L	R	L	R	L	R
e_1	V_1	V_3		F_1	e_2	e_2	e_4	e_3
e_2	V_1	V_2	F_1		e_1	e_1	e_3	e_6
e_3	V_2	V_3	F_1	F_2	e_2	e_5	e_1	e_4
e_4	$V3$	V_4		F_2	e_1	e_3	e_7	e_5
e_5	V_2	V_4	F_2	F_3	e_3	e_6	e_4	e_7
e_6	V_2	V_5	F_3		e_5	e_2	e_7	e_7
e_7	V_4	V_5		F_3	e_4	e_5	e_6	e_6

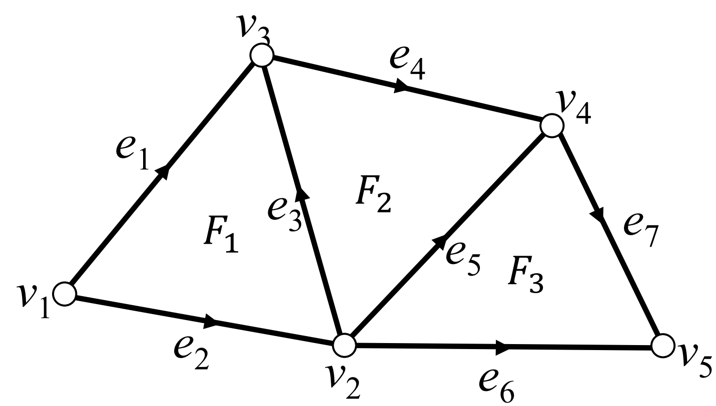
FACE TABLE

F_1	e_1
F_2	e_3
F_3	e_5

Winged Edge

Boundary edges:

- Have only one incident face



VERTEX TABLE				
V_1	X_1	Y_1	Z_1	e_1
V_2	X_2	Y_2	Z_2	e_6
V_3	X_3	Y_3	Z_3	e_3
V_4	X_4	Y_4	Z_4	e_5
V_5	X_5	Y_5	Z_5	e_6

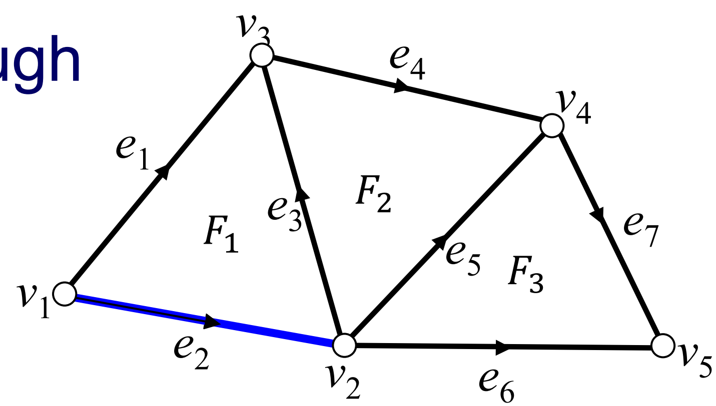
EDGE TABLE					S		E	
	S	E	L	R	L	R	L	R
$\boxed{e_1}$	V_1	V_3		F_1	e_2	e_2	e_4	e_3
$\boxed{e_2}$	V_1	V_2	F_1		e_1	e_1	e_3	e_6
e_3	V_2	V_3	F_1	F_2	e_2	e_5	e_1	e_4
$\boxed{e_4}$	V_3	V_4		F_2	e_1	e_3	e_7	e_5
e_5	V_2	V_4	F_2	F_3	e_3	e_6	e_4	e_7
$\boxed{e_6}$	V_2	V_5	F_3		e_5	e_2	e_7	e_7
$\boxed{e_7}$	V_4	V_5		F_3	e_4	e_5	e_6	e_6

FACE TABLE	
F_1	e_1
F_2	e_3
F_3	e_5

Winged Edge

Boundary edges:

- Have only one incident face
- Wing edges are defined as though the boundary was also a face



VERTEX TABLE				
V_1	X_1	Y_1	Z_1	e_1
V_2	X_2	Y_2	Z_2	e_6
V_3	X_3	Y_3	Z_3	e_3
V_4	X_4	Y_4	Z_4	e_5
V_5	X_5	Y_5	Z_5	e_6

EDGE TABLE	S	E	L	R	S L	S R	E L	E R
e_1	V_1	V_3		F_1	e_2	e_2	e_4	e_3
e_2	V_1	V_2	F_1		e_1	e_1	e_3	e_6
e_3	V_2	V_3	F_1	F_2	e_2	e_5	e_1	e_4
e_4	V_3	V_4		F_2	e_1	e_3	e_7	e_5
e_5	V_2	V_4	F_2	F_3	e_3	e_6	e_4	e_7
e_6	V_2	V_5	F_3		e_5	e_2	e_7	e_7
e_7	V_4	V_5		F_3	e_4	e_5	e_6	e_6

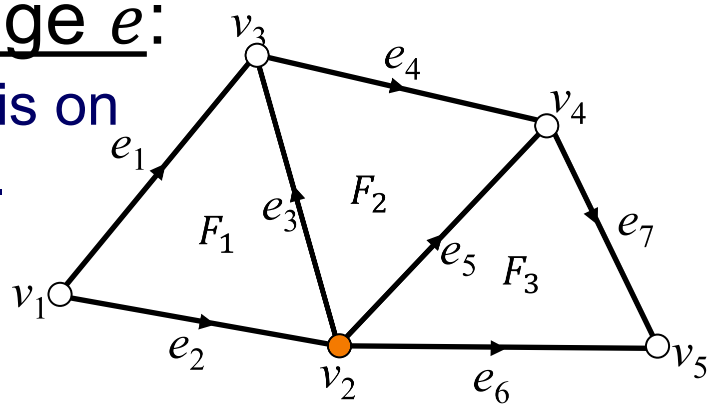
FACE TABLE	
F_1	e_1
F_2	e_3
F_3	e_5

Winged Edge (Example)

Find CCW edges adjacent to v_2.

<u>Note that given a vertex v on edge e</u>:

- If v is the **S**tart, the next CCW edge is on the **L**eft of e, coming out of the **S**tart.
- Otherwise, it is on the **R**ight of e, coming out of the **E**nd.



VERTEX TABLE				
V_1	X_1	Y_1	Z_1	e_1
V_2	X_2	Y_2	Z_2	e_6
V_3	X_3	Y_3	Z_3	e_3
V_4	X_4	Y_4	Z_4	e_5
V_5	X_5	Y_5	Z_5	e_6

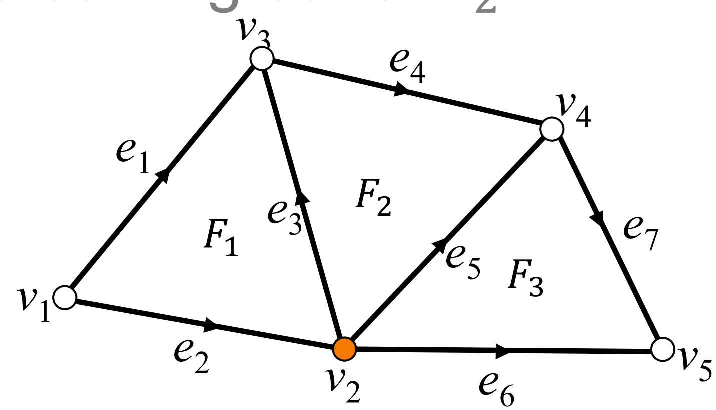
EDGE TABLE					S		E	
	S	E	L	R	L	R	L	R
e_1	V_1	V_3		F_1	e_2	e_2	e_4	e_3
e_2	V_1	V_2	F_1		e_1	e_1	e_3	e_6
e_3	V_2	V_3	F_1	F_2	e_2	e_5	e_1	e_4
e_4	V_3	V_4		F_2	e_1	e_3	e_7	e_5
e_5	V_2	V_4	F_2	F_3	e_3	e_6	e_4	e_7
e_6	V_2	V_5	F_3		e_5	e_2	e_7	e_7
e_7	V_4	V_5		F_3	e_4	e_5	e_6	e_6

FACE TABLE	
F_1	e_1
F_2	e_3
F_3	e_5

Winged Edge (Example)

Find CCW edges adjacent to v_2:

- **Initialize**: Choose the only edge coming out of v_2
- **Do**: Iterate CCW around v_2
- **While**: Haven't cycled back to the start edge



VERTEX TABLE				
V1	X1	Y1	Z1	e1
V2	X2	Y2	Z2	e6
V3	X3	Y3	Z3	e3
V4	X4	Y4	Z4	e5
V5	X5	Y5	Z5	e6

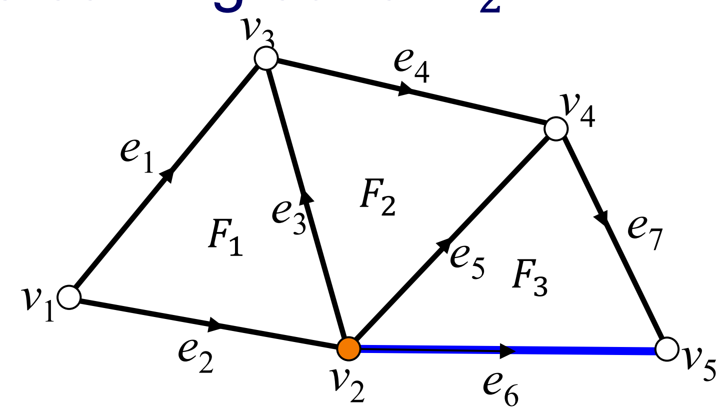
EDGE TABLE					S		E	
	S	E	L	R	L	R	L	R
e1	V1	V3		F1	e2	e2	e4	e3
e2	V1	V2	F1		e1	e1	e3	e6
e3	V2	V3	F1	F2	e2	e5	e1	e4
e4	V3	V4		F2	e1	e3	e7	e5
e5	V2	V4	F2	F3	e3	e6	e4	e7
e6	V2	V5	F3		e5	e2	e7	e7
e7	V4	V5		F3	e4	e5	e6	e6

FACE TABLE	
F1	e1
F2	e3
F3	e5

Winged Edge (Example)

Find CCW edges adjacent to v_2:

- **Initialize**: Choose the only edge coming out of v_2
- **Do**: Iterate CCW around v_2
- **While**: Haven't cycled back to the start edge



VERTEX TABLE				
V_1	X_1	Y_1	Z_1	e_1
V_2	X_2	Y_2	Z_2	e_6
V_3	X_3	Y_3	Z_3	e_3
V_4	X_4	Y_4	Z_4	e_5
V_5	X_5	Y_5	Z_5	e_6

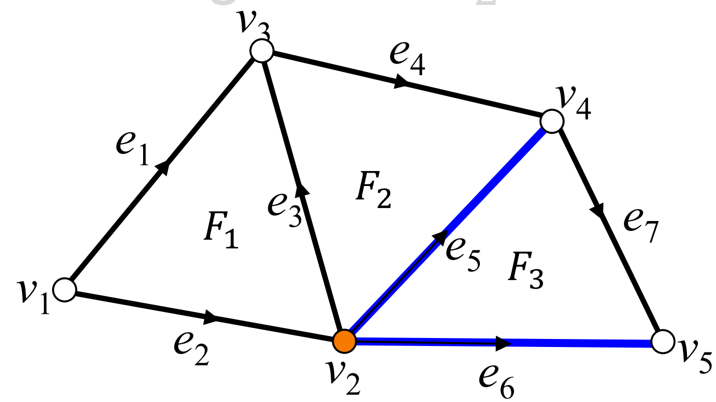
EDGE TABLE					S		E	
	S	E	L	R	L	R	L	R
e_1	V_1	V_3		F_1	e_2	e_2	e_4	e_3
e_2	V_1	V_2	F_1		e_1	e_1	e_3	e_6
e_3	V_2	V_3	F_1	F_2	e_2	e_5	e_1	e_4
e_4	V_3	V_4		F_2	e_1	e_3	e_7	e_5
e_5	V_2	V_4	F_2	F_3	e_3	e_6	e_4	e_7
e_6	V_2	V_5	F_3		e_5	e_2	e_7	e_7
e_7	V_4	V_5		F_3	e_4	e_5	e_6	e_6

FACE TABLE	
F_1	e_1
F_2	e_3
F_3	e_5

Winged Edge (Example)

Find CCW edges adjacent to v_2:

- **Initialize**: Choose the only edge coming out of v_2
- **Do**: Iterate CCW around v_2
  - » If v_2 is the **S**tart...
  - » Otherwise...
- **While**: Haven't cycled back to the start edge



VERTEX TABLE

V_1	X_1	Y_1	Z_1	e_1
V_2	X_2	Y_2	Z_2	e_6
V_3	X_3	Y_3	Z_3	e_3
V_4	X_4	Y_4	Z_4	e_5
V_5	X_5	Y_5	Z_5	e_6

EDGE TABLE

					S		E	
	S	E	L	R	L	R	L	R
e_1	V_1	V_3		F_1	e_2	e_2	e_4	e_3
e_2	V_1	V_2	F_1		e_1	e_1	e_3	e_6
e_3	V_2	V_3	F_1	F_2	e_2	e_5	e_1	e_4
e_4	V_3	V_4		F_2	e_1	e_3	e_7	e_5
e_5	V_2	V_4	F_2	F_3	e_3	e_6	e_4	e_7
e_6	V_2	V_5	F_3		e_5	e_2	e_7	e_7
e_7	V_4	V_5		F_3	e_4	e_5	e_6	e_6

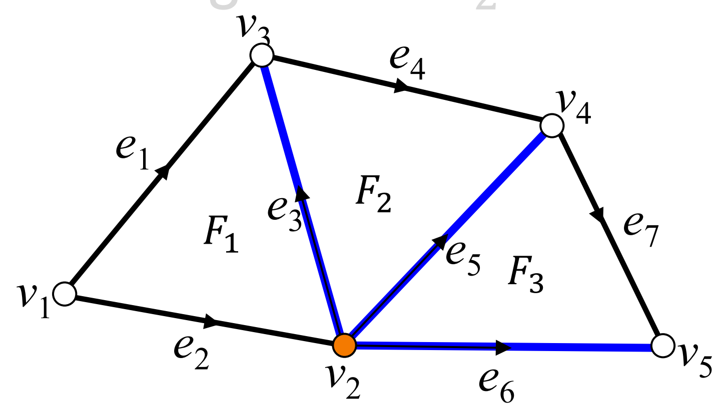
FACE TABLE

F_1	e_1
F_2	e_3
F_3	e_5

Winged Edge (Example)

Find CCW edges adjacent to v_2:

- **Initialize**: Choose the only edge coming out of v_2
- **Do**: Iterate CCW around v_2
  - » If v_2 is the **S**tart...
  - » Otherwise...
- **While**: Haven't cycled back to the start edge



VERTEX TABLE				
V_1	X_1	Y_1	Z_1	e_1
V_2	X_2	Y_2	Z_2	e_6
V_3	X_3	Y_3	Z_3	e_3
V_4	X_4	Y_4	Z_4	e_5
V_5	X_5	Y_5	Z_5	e_6

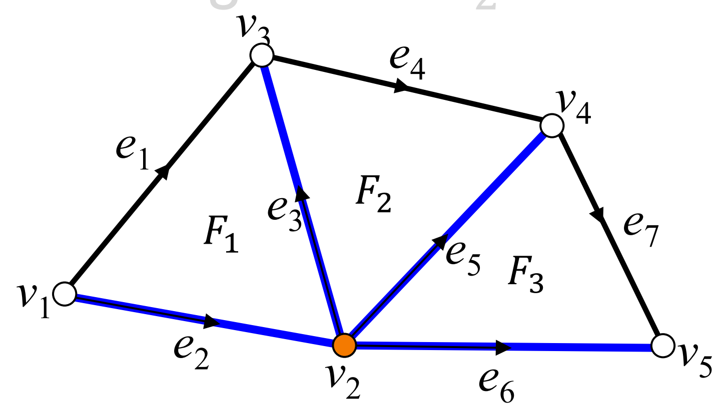
EDGE TABLE					S		E	
	S	E	L	R	L	R	L	R
e_1	V_1	V_3		F_1	e_2	e_2	e_4	e_3
e_2	V_1	V_2	F_1		e_1	e_1	e_3	e_6
e_3	V_2	V_3	F_1	F_2	e_2	e_5	e_1	e_4
e_4	V_3	V_4		F_2	e_1	e_3	e_7	e_5
e_5	V_2	V_4	F_2	F_3	e_3	e_6	e_4	e_7
e_6	V_2	V_5	F_3		e_5	e_2	e_7	e_7
e_7	V_4	V_5		F_3	e_4	e_5	e_6	e_6

FACE TABLE	
F_1	e_1
F_2	e_3
F_3	e_5

Winged Edge (Example)

Find CCW edges adjacent to v_2:

- **Initialize**: Choose the only edge coming out of v_2
- **Do**: Iterate CCW around v_2
  - » If v_2 is the **S**tart...
  - » Otherwise...
- **While**: Haven't cycled back to the start edge



VERTEX TABLE

V_1	X_1	Y_1	Z_1	e_1
V_2	X_2	Y_2	Z_2	e_6
V_3	X_3	Y_3	Z_3	e_3
V_4	X_4	Y_4	Z_4	e_5
V_5	X_5	Y_5	Z_5	e_6

EDGE TABLE

	S	E	L	R	S: L	S: R	E: L	E: R
e_1	V_1	V_3		F_1	e_2	e_2	e_4	e_3
e_2	V_1	V_2	F_1		e_1	e_1	e_3	e_6
e_3	V_2	V_3	F_1	F_2	e_2	e_5	e_1	e_4
e_4	V_3	V_4		F_2	e_1	e_3	e_7	e_5
e_5	V_2	V_4	F_2	F_3	e_3	e_6	e_4	e_7
e_6	V_2	V_5	F_3		e_5	e_2	e_7	e_7
e_7	V_4	V_5		F_3	e_4	e_5	e_6	e_6

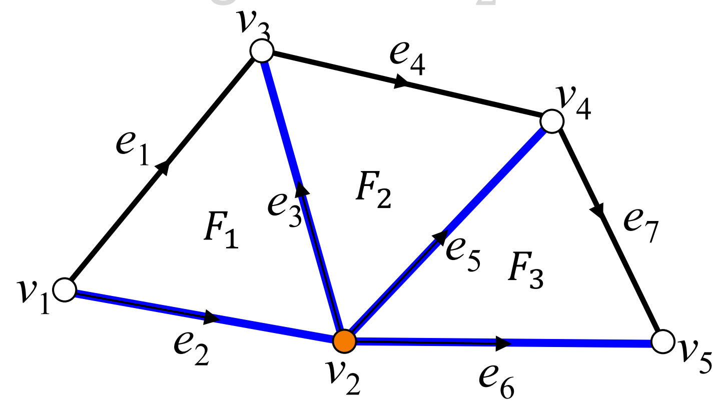
FACE TABLE

F_1	e_1
F_2	e_3
F_3	e_5

Winged Edge (Example)

Find CCW edges adjacent to v_2:

- **Initialize**: Choose the only edge coming out of v_2
- **Do**: Iterate CCW around v_2
  - » If v_2 is the **S**tart...
  - » Otherwise...
- **While**: Haven't cycled back to the start edge



VERTEX TABLE

V_1	X_1	Y_1	Z_1	e_1
V_2	X_2	Y_2	Z_2	e_6
V_3	X_3	Y_3	Z_3	e_3
V_4	X_4	Y_4	Z_4	e_5
V_5	X_5	Y_5	Z_5	e_6

EDGE TABLE

	S	E	L	R	S L	S R	E L	E R
e_1	V_1	V_3		F_1	e_2	e_2	e_4	e_3
e_2	V_1	V_2	F_1		e_1	e_1	e_3	e_6
e_3	V_2	V_3	F_1	F_2	e_2	e_5	e_1	e_4
e_4	V_3	V_4		F_2	e_1	e_3	e_7	e_5
e_5	V_2	V_4	F_2	F_3	e_3	e_6	e_4	e_7
e_6	V_2	V_5	F_3		e_5	e_2	e_7	e_7
e_7	V_4	V_5		F_3	e_4	e_5	e_6	e_6

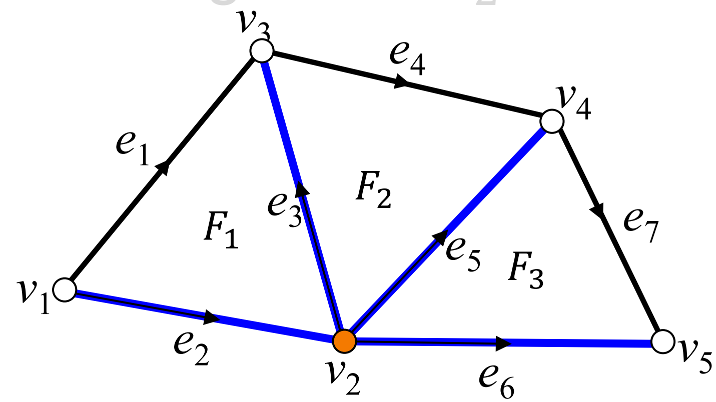
FACE TABLE

F_1	e_1
F_2	e_3
F_3	e_5

Winged Edge (Example)

Find CCW edges adjacent to v_2:

- **Initialize**: Choose the only edge coming out of v_2
- **Do**: Iterate CCW around v_2
- **While**: Haven't cycled back to the start edge



VERTEX TABLE				
V_1	X_1	Y_1	Z_1	e_1
V_2	X_2	Y_2	Z_2	e_6
V_3	X_3	Y_3	Z_3	e_3
V_4	X_4	Y_4	Z_4	e_5
V_5	X_5	Y_5	Z_5	e_6

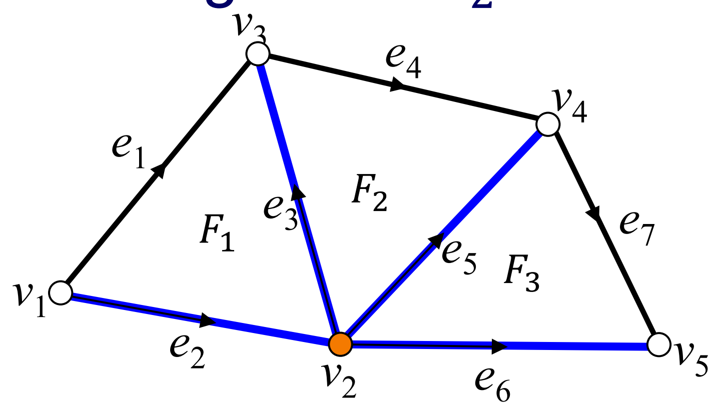
EDGE TABLE					S		E	
	S	E	L	R	L	R	L	R
e_1	V_1	V_3		F_1	e_2	e_2	e_4	e_3
e_2	V_1	V_2	F_1		e_1	e_1	e_3	e_6
e_3	V_2	V_3	F_1	F_2	e_2	e_5	e_1	e_4
e_4	V_3	V_4		F_2	e_1	e_3	e_7	e_5
e_5	V_2	V_4	F_2	F_3	e_3	e_6	e_4	e_7
e_6	V_2	V_5	F_3		e_5	e_2	e_7	e_7
e_7	V_4	V_5		F_3	e_4	e_5	e_6	e_6

FACE TABLE	
F_1	e_1
F_2	e_3
F_3	e_5

Winged Edge (Example)

Find CCW edges adjacent to v_2:

- **Initialize**: Choose the only edge coming out of v_2
- **Do**: Iterate CCW around v_2
- **While**: Haven't cycled back to the start edge



VERTEX TABLE				
V_1	X_1	Y_1	Z_1	e_1
V_2	X_2	Y_2	Z_2	e_6
V_3	X_3	Y_3	Z_3	e_3
V_4	X_4	Y_4	Z_4	e_5
V_5	X_5	Y_5	Z_5	e_6

EDGE TABLE					S		E	
	S	E	L	R	L	R	L	R
e_1	V_1	V_3		F_1	e_2	e_2	e_4	e_3
e_2	V_1	V_2	F_1		e_1	e_1	e_3	e_6
e_3	V_2	V_3	F_1	F_2	e_2	e_5	e_1	e_4
e_4	V_3	V_4		F_2	e_1	e_3	e_7	e_5
e_5	V_2	V_4	F_2	F_3	e_3	e_6	e_4	e_7

FACE TABLE	
F_1	e_1
F_2	e_3
F_3	e_5

Computational complexity is proportional to the size of the output.
(Independent of the size of the mesh.)

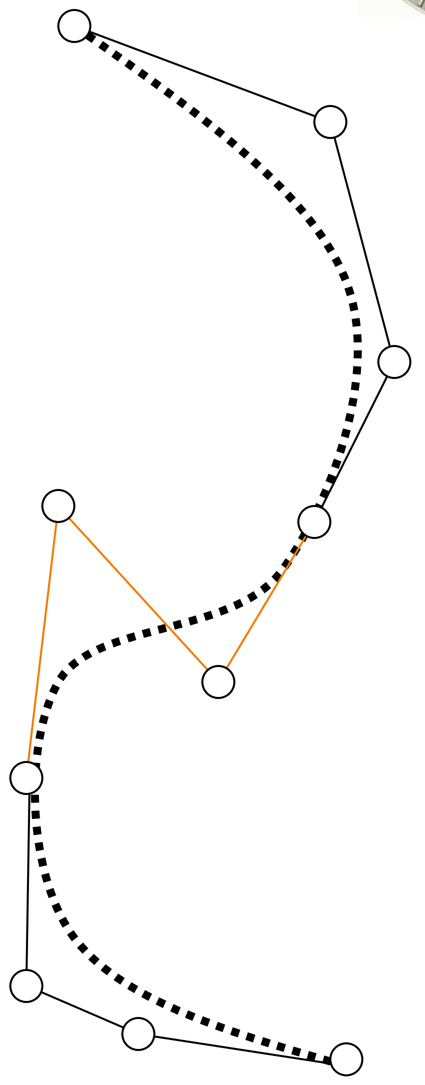
Outline

- Representing Meshes
- Parametric Curves

Parametric Curves

Given a 1D control lattice

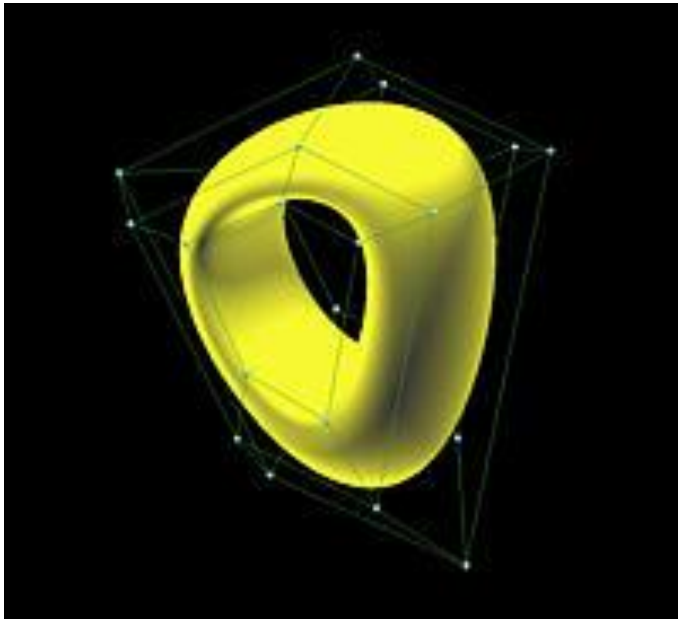
- Compute a smooth curve passing through/near the control points



Parametric Surfaces

Given a 2D control lattice

- Compute a smooth surface passing through/near the control points



Courtesy of C.K. Shene

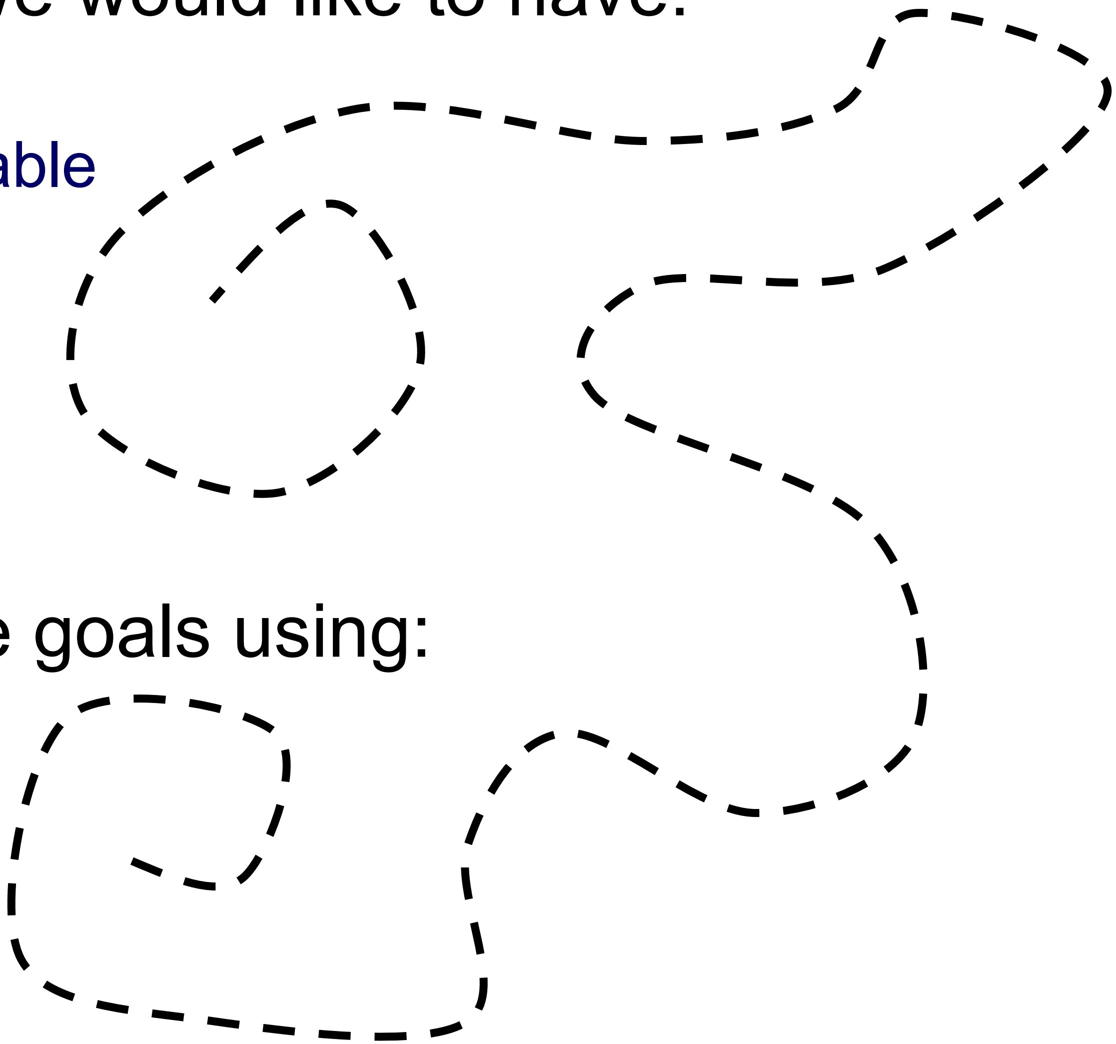
Very closely related to subdivision surfaces!

"Exact Evaluation Of Catmull-Clark Subdivision Surfaces At Arbitrary Parameter Values". [Stam, 1998]

Goals

Some attributes we would like to have:

- Local support
- Simple/predictable
- Continuous

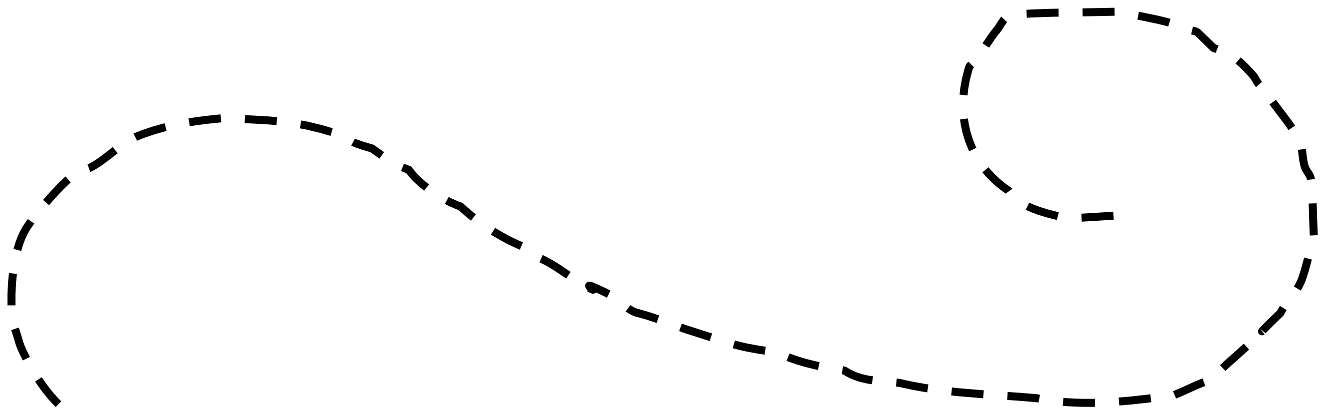


We'll satisfy these goals using:

- Piecewise
- Polynomials

What is a Spline in CG?

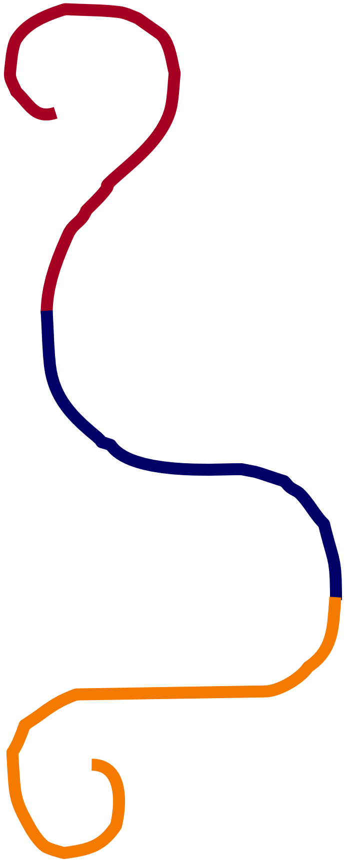
A spline is a *piecewise* *polynomial function* whose derivatives satisfy *continuity constraints* across curve boundaries.



What is a Spline in CG?

Piecewise: the spline is a collection of *parametric curves* segments joined together.

Polynomial functions: each segment is a *parametric polynomial curve*.

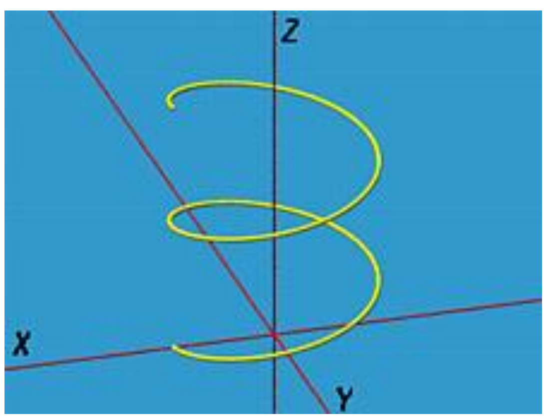


Parametric Curves

A <u>parametric curve</u> in d-dimensions is defined by a collection of coordinate functions in u giving the position of a point on the curve at each u value:

$$\Phi(u) = \big(x_1(u), \cdots, x_d(u)\big)$$

$\Phi(u) = (\cos u, \sin u, u)$



Courtesy of C.K. Shene

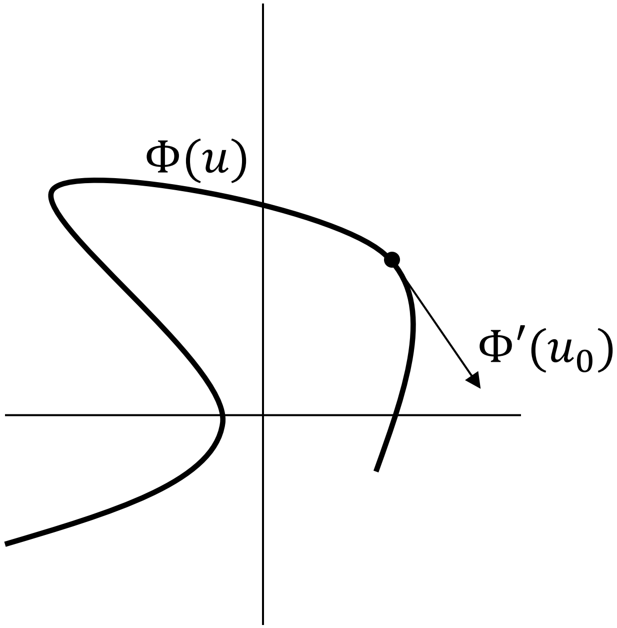
<u>Note</u>:
A parametric curve is **not** the graph of a function.

Derivatives

For a curve $\Phi(u) = \big(x(u), y(u)\big)$, the derivatives of the curve coefficients at a point u_0:

$$\Phi'(u_0) = \big(x'(u_0), y'(u_0)\big)$$

points in a direction tangent to the curve.



<u>Note</u>:
The direction of the derivative is determined by the path that the curve traces out.

The magnitude of the parametric derivative is determined by the tracing speed.

Polynomials

A polynomial in the variable u is:

"An algebraic expression written as a sum of constants multiplied by different powers of a variable."

$$P(u) = a_0 + a_1 \cdot u + a_2 \cdot u^2 + \cdots + a_n \cdot u^n = \sum_{k=0}^{n} a_k \cdot u^k$$

The constant a_k is referred to as the k-th coefficient of the polynomial P.

A polynomial $P(u)$ has degree n if ($a_n \neq 0$ and) for all $k > n$, the coefficients satisfy $a_k = 0$.

Polynomials

A polynomial in the variable u is:

"An algebraic expression written as a sum of constants multiplied by different powers of a variable."

$$P(u) = a_0 + a_1 \cdot u + a_2 \cdot u^2 + \cdots + a_n \cdot u^n = \sum_{k=0}^{n} a_k \cdot u^k$$

A polynomial of degree n has
$n+1$ degrees of freedom

$\Updownarrow$

With $n+1$ pieces of information about a degree-n polynomial, should have enough information to reconstruct its coefficients

Polynomials (Matrices)

$$P(u) = a_0 + a_1 \cdot u + a_2 \cdot u^2 + \cdots + a_n \cdot u^n = \sum_{k=0}^{n} a_k \cdot u^k$$

The polynomial P can be expressed as the matrix multiplication of:

- A row vectors containing the powers of u, and
- A column vector containing the coefficients:

$$P(u) = \begin{pmatrix} u^n & u^{n-1} & \cdots & u^1 & u^0 \end{pmatrix} \cdot \begin{pmatrix} a_n \\ a_{n-1} \\ \vdots \\ a_1 \\ a_0 \end{pmatrix}$$

Polynomials (1st Derivative Matrices)

$$P(u) = a_0 + a_1 \cdot u + a_2 \cdot u^2 + \cdots + a_n \cdot u^n = \sum_{k=0}^{n} a_k \cdot u^k$$

The derivative of the polynomial is:

$$P'(u) = a_1 + 2 \cdot a_2 \cdot u + \cdots + n \cdot a_n \cdot u^{n-1} = \sum_{k=1}^{n} k \cdot a_k \cdot u^{k-1}$$

⇒ The derivative of polynomial P can also be expressed as a matrix multiplication:

$$P'(u) = \begin{pmatrix} n \cdot u^{n-1} & (n-1) \cdot u^{n-2} & \cdots & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} a_n \\ a_{n-1} \\ \vdots \\ a_1 \\ a_0 \end{pmatrix}$$

Polynomials (Matrices)

$$P(u) = \sum_{k=0}^{n} a_k \cdot u^k$$

<u>Example</u>:

Given the values of $P(u)$ at $n+1$ different locations:

$$p_0 = P(u_0), \cdots, p_n = P(u_n)$$

$$p_0 = \begin{pmatrix} u_0^n & \cdots & u_0^0 \end{pmatrix} \cdot \begin{pmatrix} a_n \\ \vdots \\ a_0 \end{pmatrix}, \cdots, p_n = \begin{pmatrix} u_n^n & \cdots & u_n^0 \end{pmatrix} \cdot \begin{pmatrix} a_n \\ \vdots \\ a_0 \end{pmatrix}$$

We can stack into one linear system:

$$\begin{pmatrix} p_0 \\ \vdots \\ p_n \end{pmatrix} = \begin{pmatrix} u_0^n & \cdots & u_0^0 \\ \vdots & \ddots & \vdots \\ u_n^n & \cdots & u_n^0 \end{pmatrix} \begin{pmatrix} a_n \\ \vdots \\ a_0 \end{pmatrix}$$

Polynomials (Matrices)

$$P(u) = \sum_{k=0}^{n} a_k \cdot u^k$$

<u>Example</u>:

Given the values of $P(u)$ at $n+1$ different locations:

$$p_0 = P(u_0), \cdots, p_n = P(u_n)$$

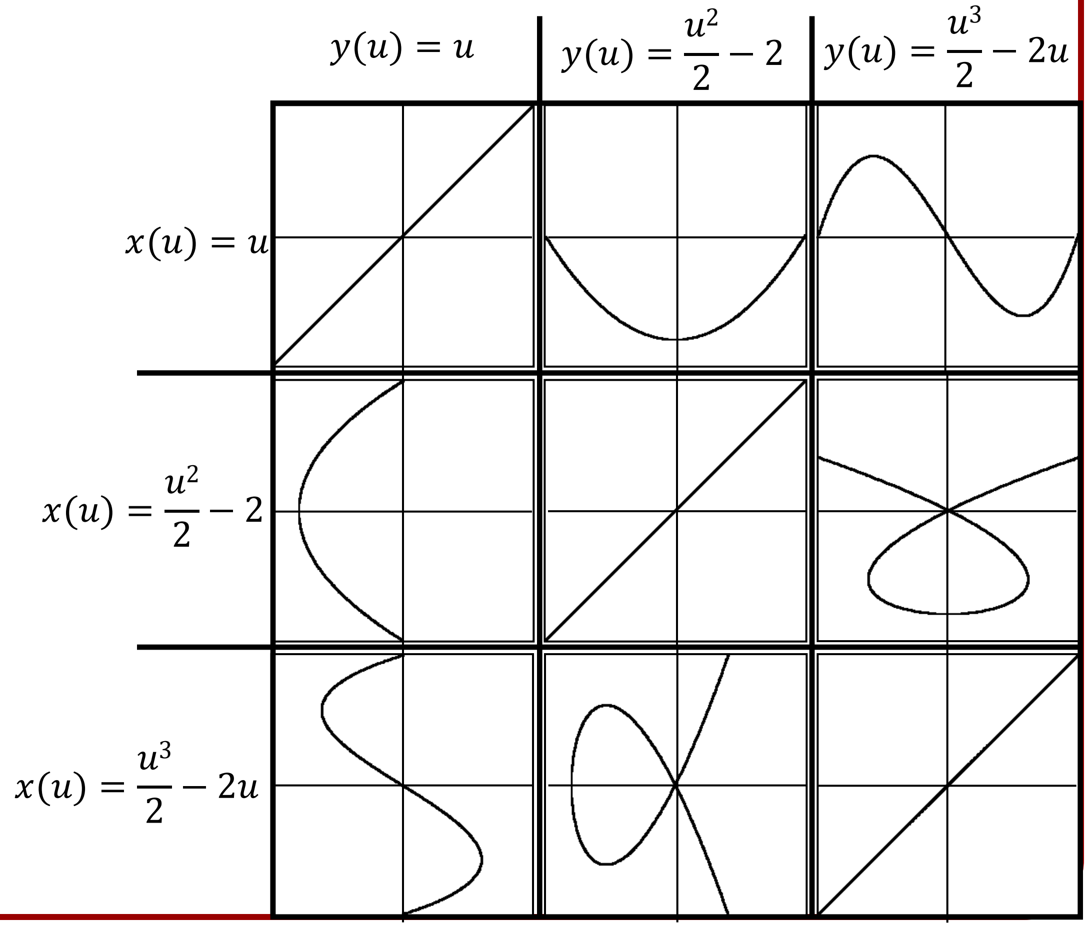
$$p_0 = \begin{pmatrix} u_0^n & \cdots & u_0^0 \end{pmatrix} \cdot \begin{pmatrix} a_n \\ \vdots \\ a_0 \end{pmatrix}, \cdots, p_n = \begin{pmatrix} u_n^n & \cdots & u_n^0 \end{pmatrix} \cdot \begin{pmatrix} a_n \\ \vdots \\ a_0 \end{pmatrix}$$

We can stack into one linear system, and invert to get the coefficients:

$$\begin{pmatrix} p_0 \\ \vdots \\ p_n \end{pmatrix} = \begin{pmatrix} u_0^n & \cdots & u_0^0 \\ \vdots & \ddots & \vdots \\ u_n^n & \cdots & u_n^0 \end{pmatrix} \begin{pmatrix} a_n \\ \vdots \\ a_0 \end{pmatrix} \Rightarrow \begin{pmatrix} a_n \\ \vdots \\ a_0 \end{pmatrix} = \begin{pmatrix} u_0^n & \cdots & u_0^0 \\ \vdots & \ddots & \vdots \\ u_n^n & \cdots & u_n^0 \end{pmatrix}^{-1} \begin{pmatrix} p_0 \\ \vdots \\ p_n \end{pmatrix}$$

Parametric Polynomial Curves

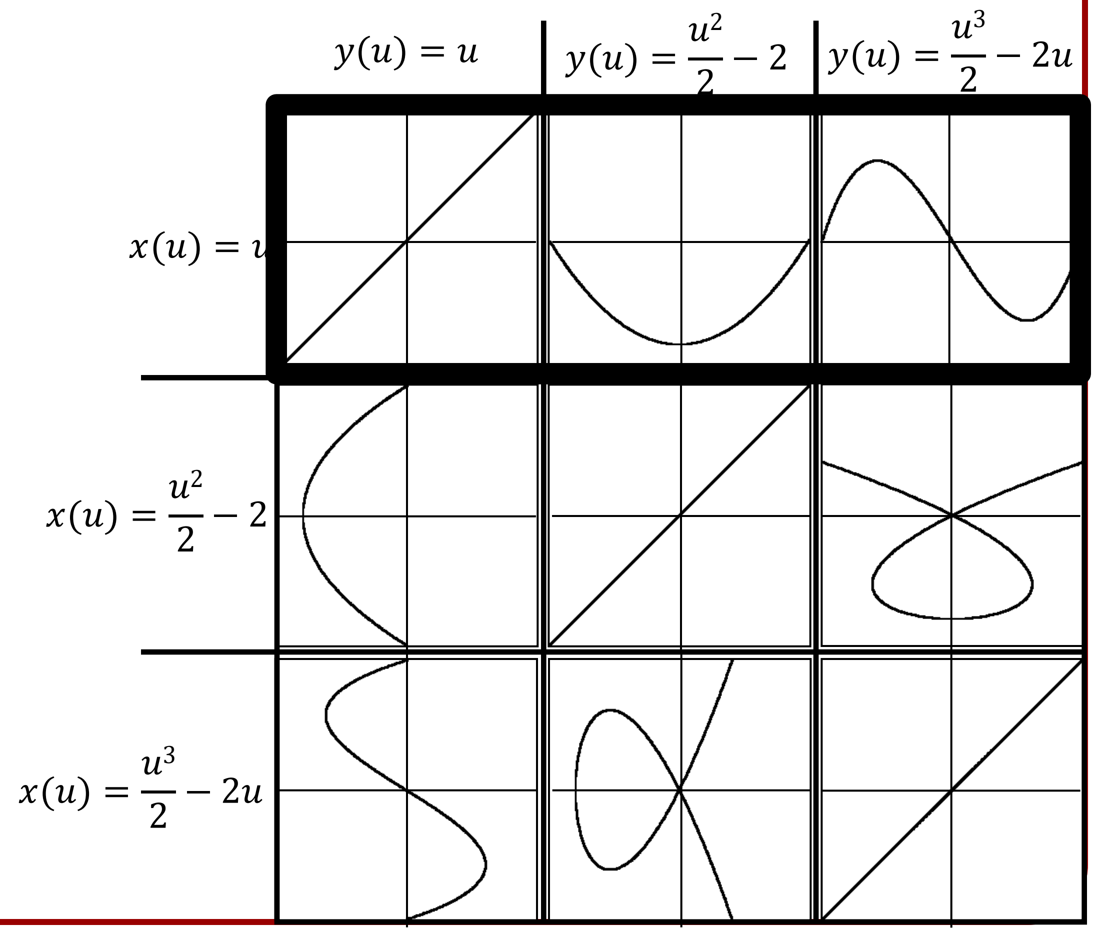
Examples:



Parametric Polynomial Curves

Examples:

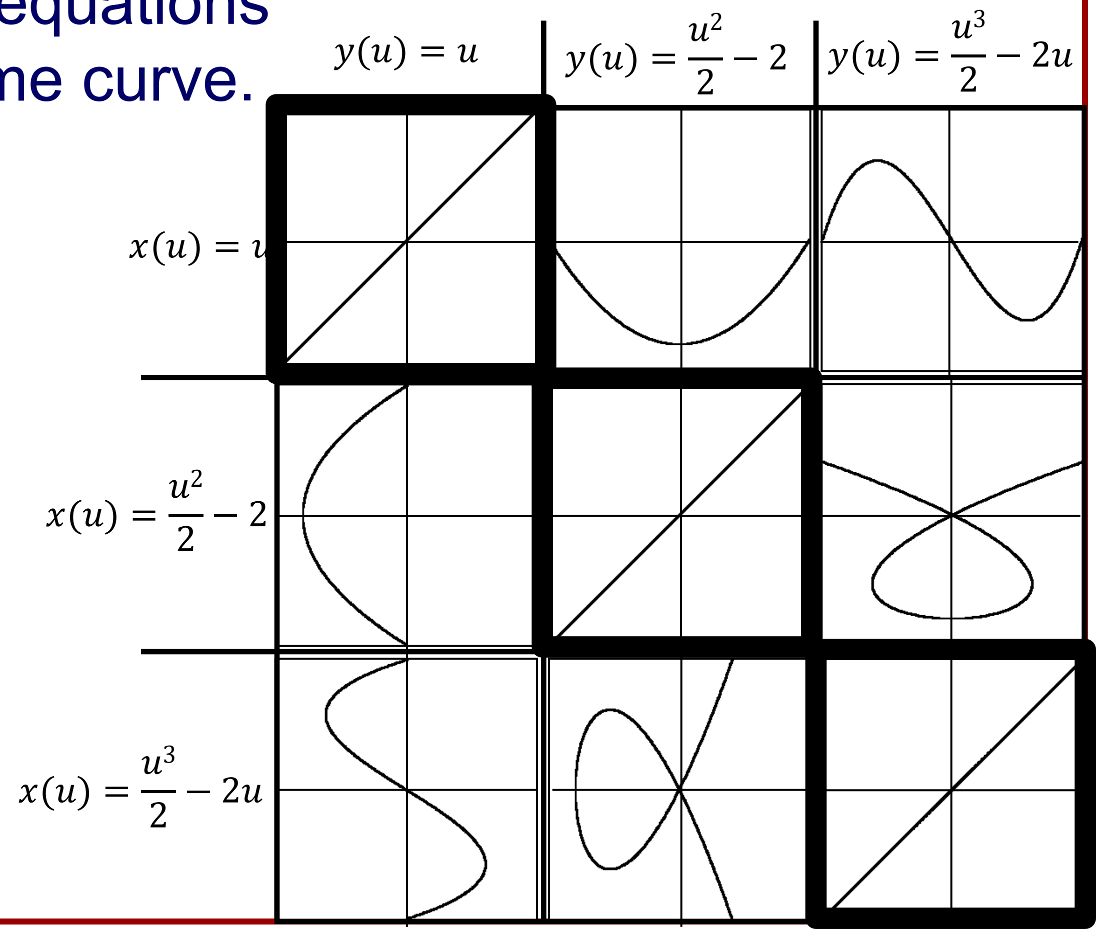
- When $x(u) = u$, the curve is the graph of $y(u)$.



Parametric Polynomial Curves

Examples:

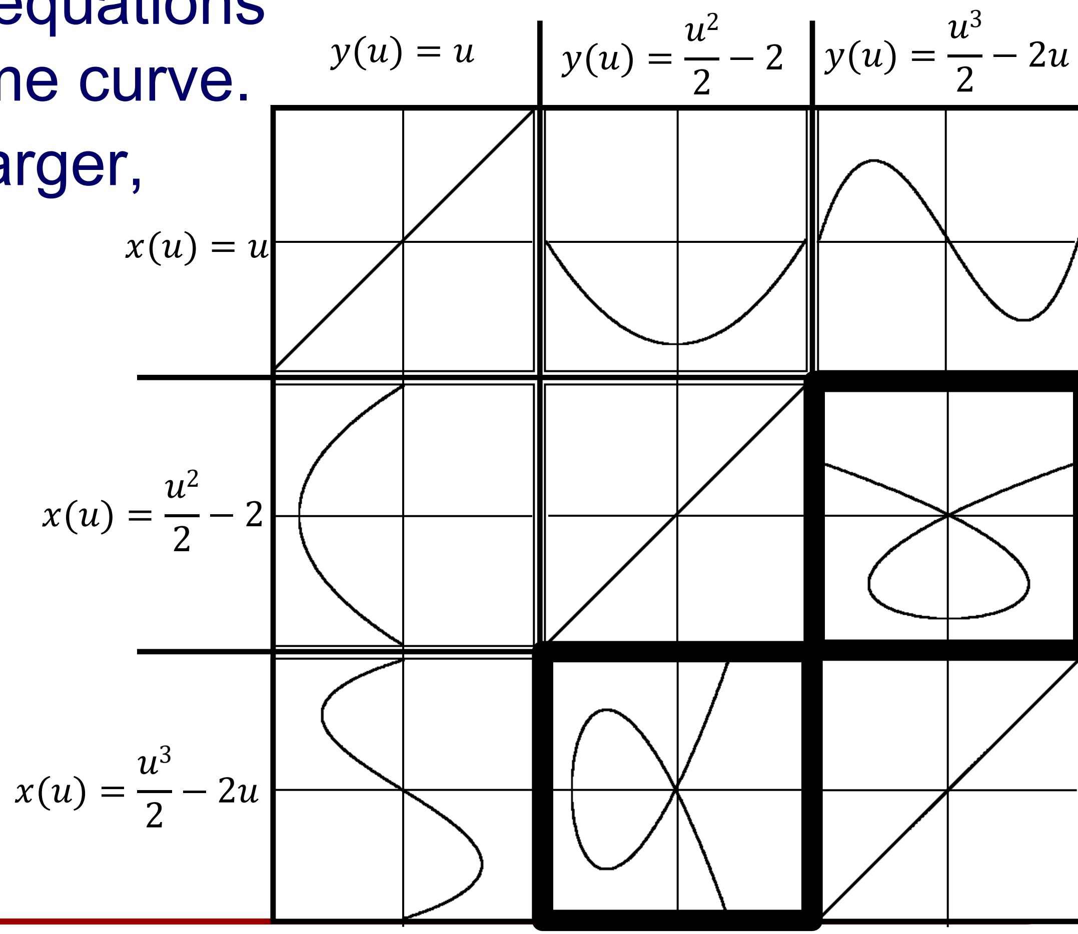
- When $x(u) = u$, the curve is the graph of $y(u)$.
- Different parametric equations can trace out the same curve.



Parametric Polynomial Curves

<u>Examples</u>:

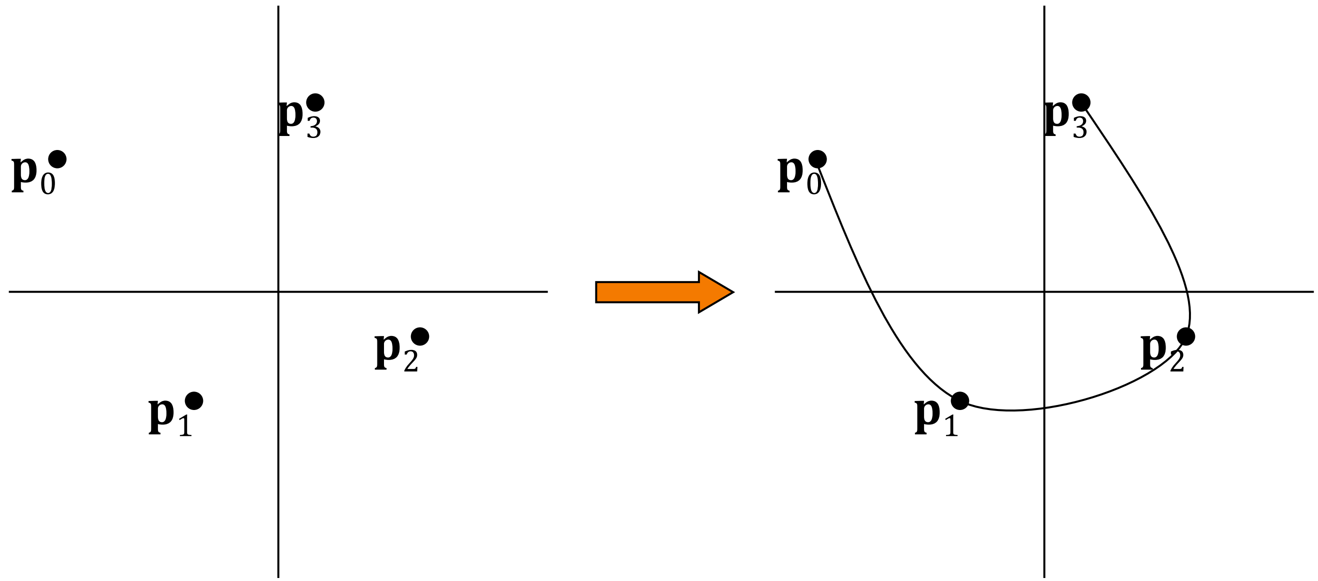
- When $x(u) = u$, the curve is the graph of $y(u)$.
- Different parametric equations can trace out the same curve.
- As the degree gets larger, the complexity of the curve increases.



Parametric Curves (in $\mathbb{R}^d$)

Goal:

Given a sequence of points, $\{\mathbf{p}_1, \cdots, \mathbf{p}_m\} \subset \mathbb{R}^d$, define a parametric curve that passes through/near the points



Parametric Curves (in $\mathbb{R}^d$)

Direct Approach:

Solve for the $d \times m$ coefficients of a parametric polynomial curve of degree $m-1$, passing through the points.

Limitations:

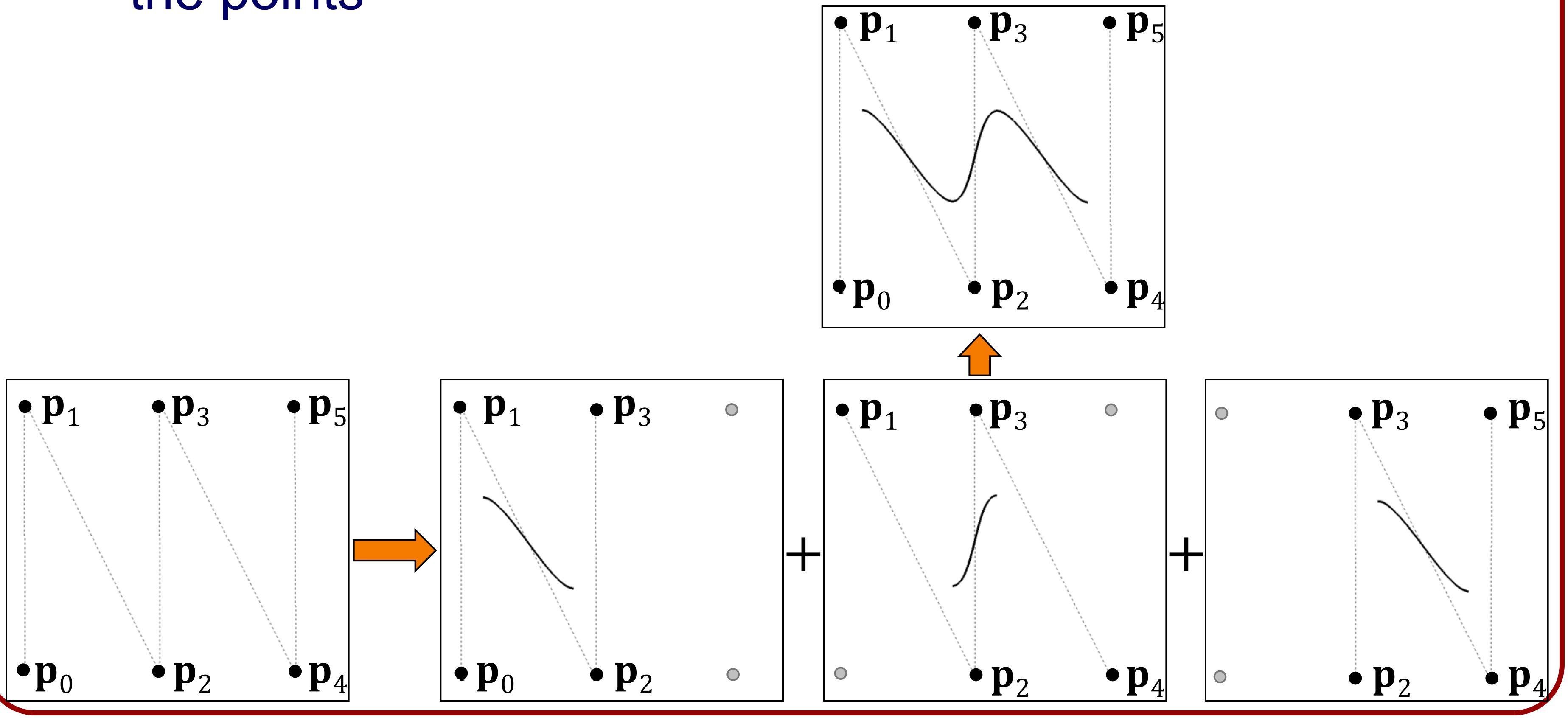
- No local control
- As the number of points increases:
  - The dimension increases and the curve oscillates more
  - Requires inverting a large linear system

Polynomial Fitting Demo

Piecewise parametric polynomials

Approach:

Fit low-degree polynomials to (overlapping) groups of points so that the combined curve passes through/near the points



Piecewise parametric polynomials

Approach:

Fit low-degree polynomials to (overlapping) groups of points so that the combined curve passes through/near the points

Properties:

- Local Control
  - A curve segment only depends on its group of points
- Simplicity
  - Individual curve segments are low-order polynomials
- Continuity/Smoothness
  - How do we guarantee smoothness?

What is a Spline in CG?

Continuity:

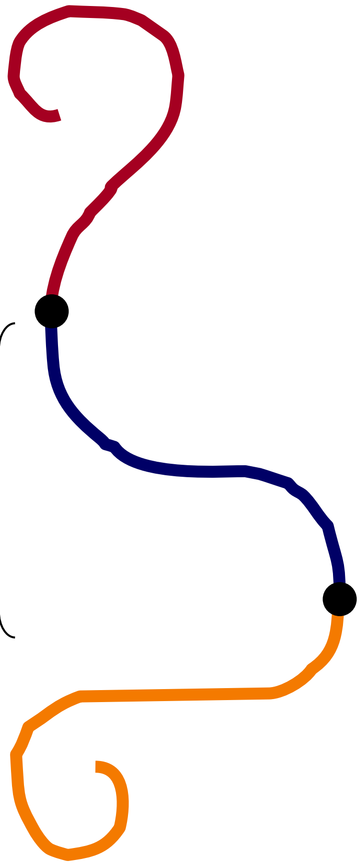
In the interior of the parameterization domain, the polynomial functions are smooth.

The values/derivatives of the polynomials need to match at the boundaries.

$\mathbf{P}_1(u) \quad u \in [0,1]$

$\mathbf{P}_2(u) \quad u \in [0,1]$

$\mathbf{P}_3(u) \quad u \in [0,1]$



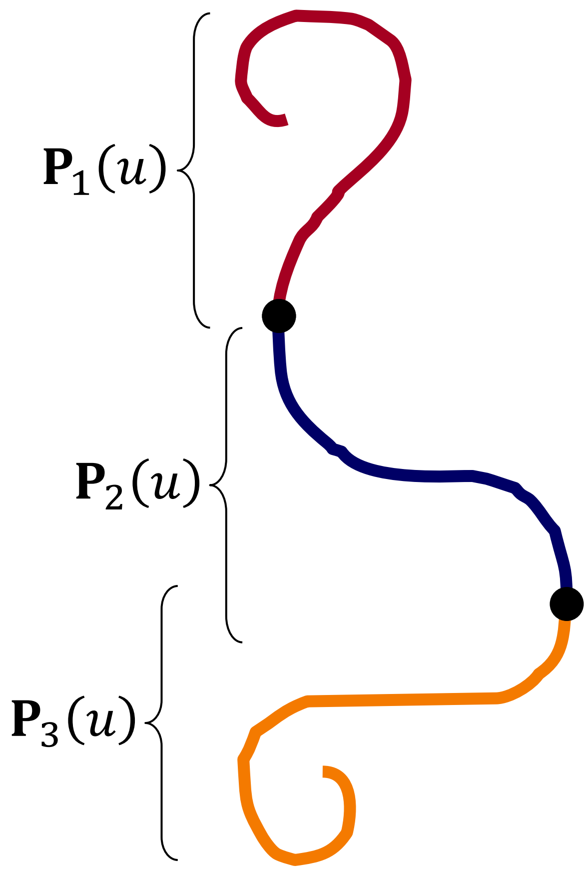
$$\boxed{\mathbf{P}_i(u) = \sum_{j=0}^{n} \mathbf{a}_{ij} \cdot u^j}$$

Continuity/Smoothness

<u>Continuity</u>:

When they meet, values/derivatives of the two curve segments need to be *equal*.

- C^0: function is continuous
  $$\Rightarrow \qquad \mathbf{P}_i(1) = \mathbf{P}_{i+1}(0)$$
- C^1: function is continuous and 1st derivatives equal
  $$\Rightarrow C^0 \quad \text{and} \quad \mathbf{P}_i'(1) = \mathbf{P}_{i+1}'(0)$$
- C^2: function is continuous and 1st and 2nd derivatives are equal
  $$\Rightarrow C^1 \quad \text{and} \quad \mathbf{P}_i''(1) = \mathbf{P}_{i+1}''(0)$$
- C^k: function is continuous and …



Overview

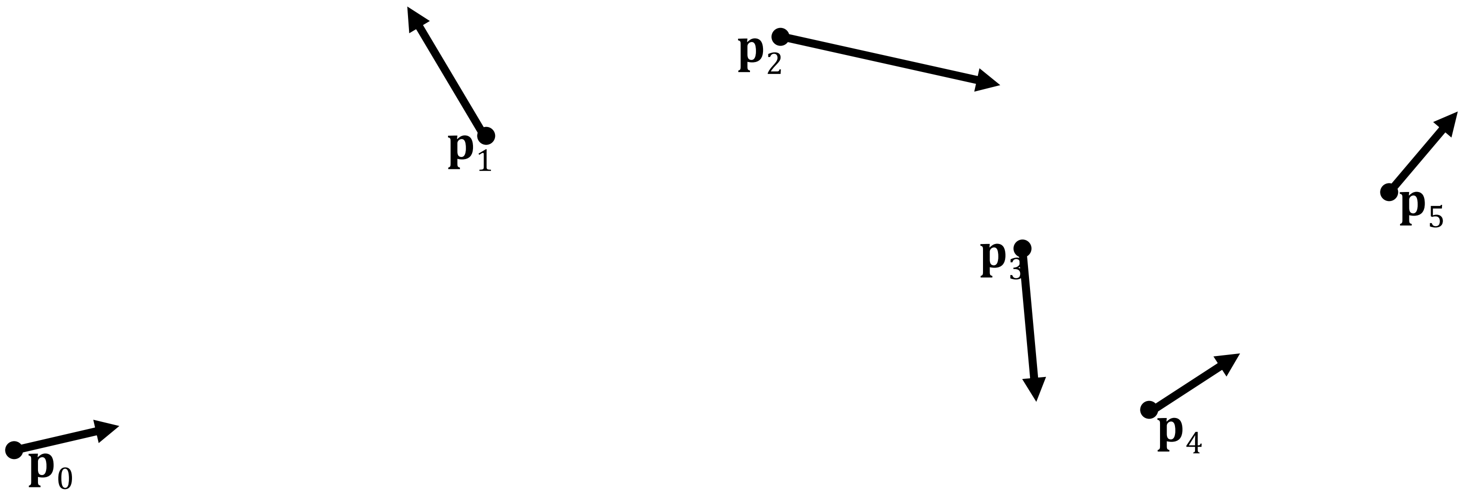
- Representing Meshes
- Parametric Curves
  - Hermite Splines

Specific Example: Hermite Splines

Interpolating piecewise *cubic* polynomial, each specified by:

- Start/end positions
- Start/end tangents

Iteratively construct the curve between adjacent end points that interpolate positions and tangents.

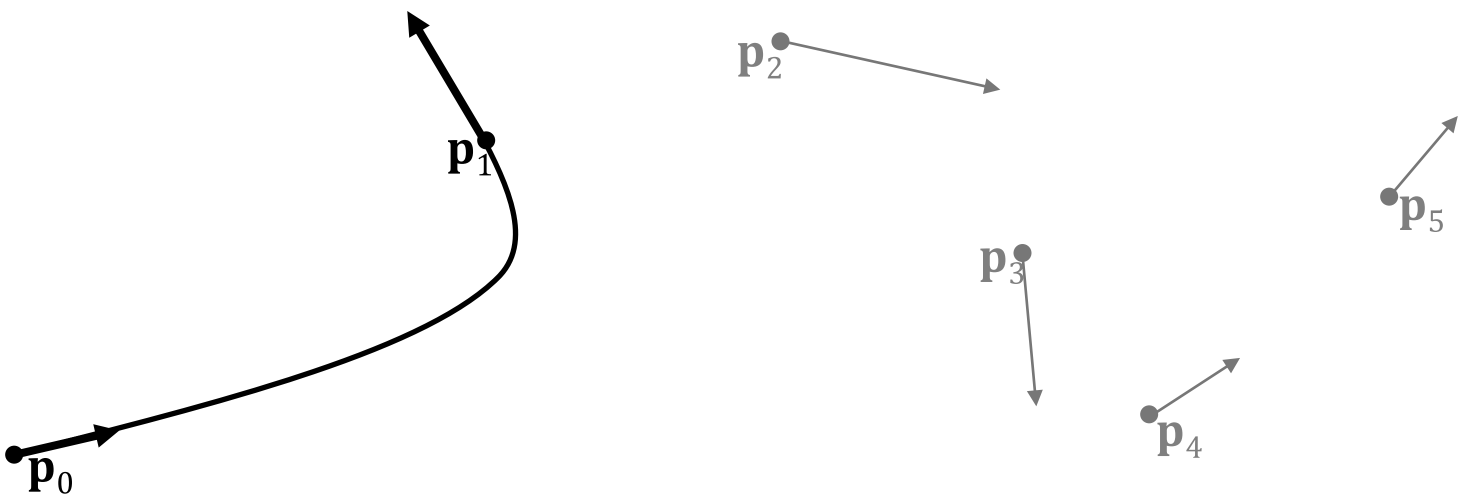


Specific Example: Hermite Splines

Interpolating piecewise *cubic* polynomial, each specified by:

- Start/end positions
- Start/end tangents

Iteratively construct the curve between adjacent end points that interpolate positions and tangents.

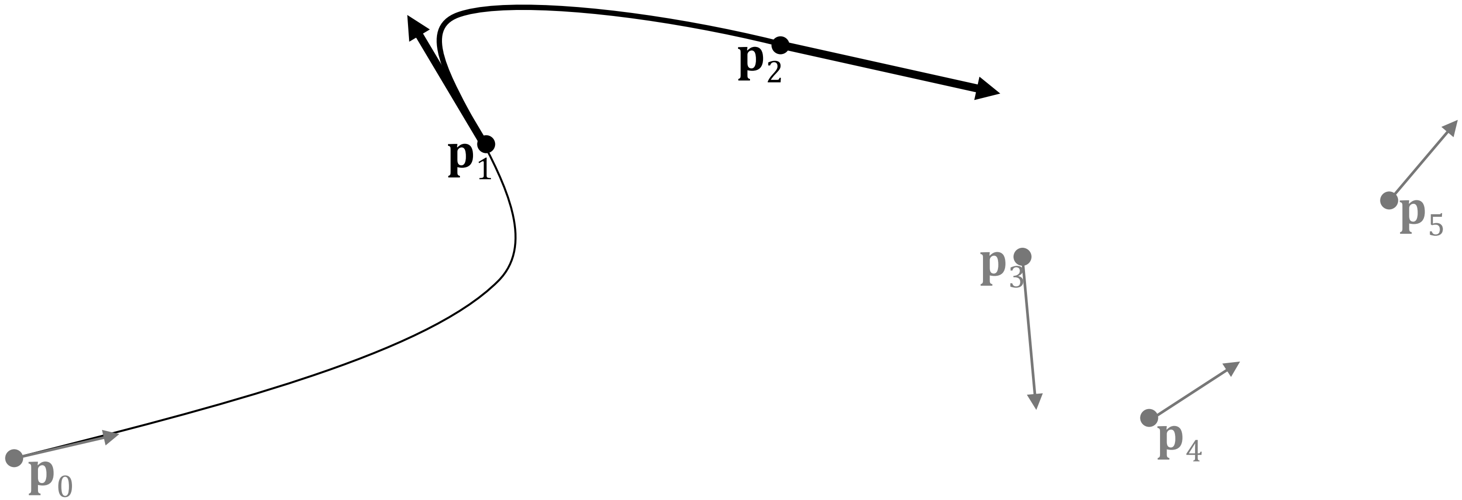


Specific Example: Hermite Splines

Interpolating piecewise *cubic* polynomial, each specified by:

- Start/end positions
- Start/end tangents

Iteratively construct the curve between adjacent end points that interpolate positions and tangents.

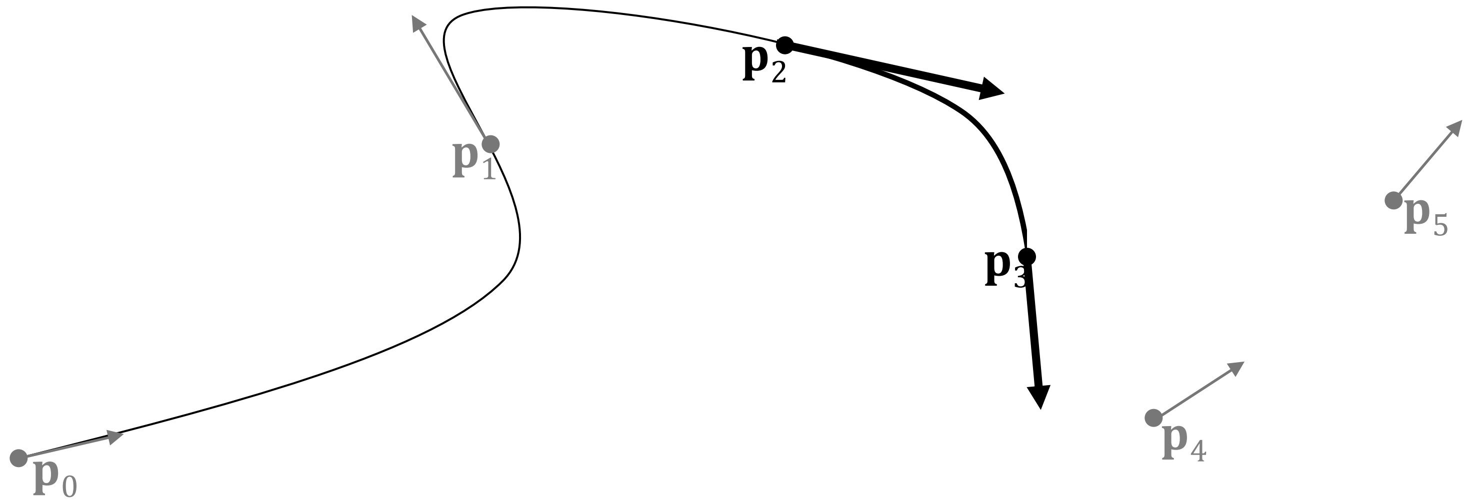


Specific Example: Hermite Splines

Interpolating piecewise *cubic* polynomial, each specified by:

- Start/end positions
- Start/end tangents

Iteratively construct the curve between adjacent end points that interpolate positions and tangents.

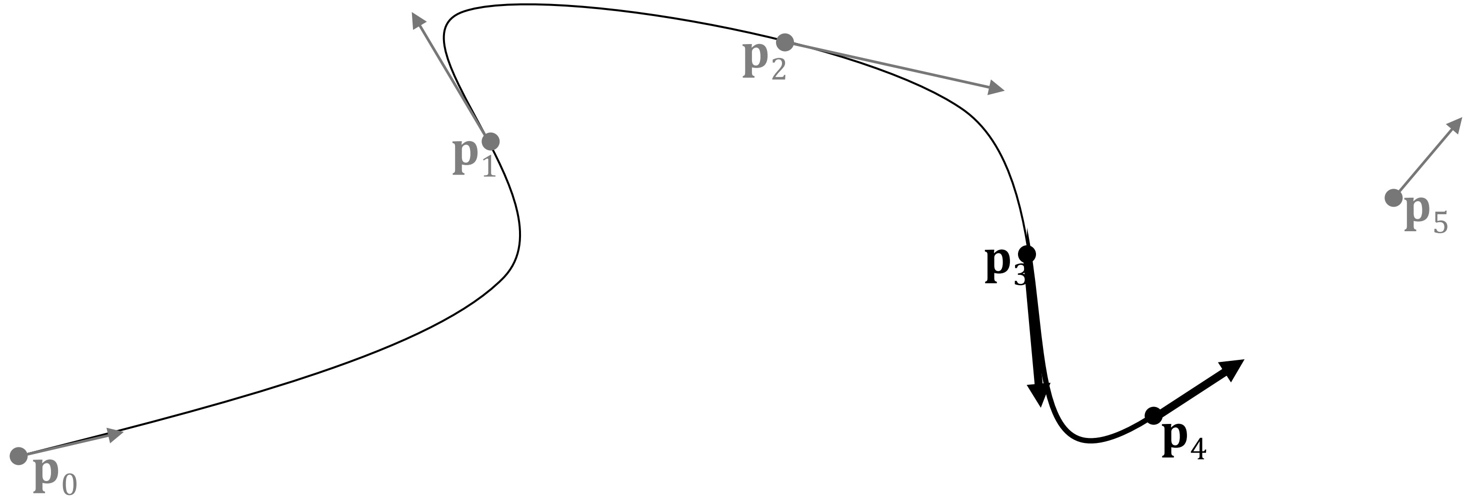


Specific Example: Hermite Splines

Interpolating piecewise *cubic* polynomial, each specified by:

- Start/end positions
- Start/end tangents

Iteratively construct the curve between adjacent end points that interpolate positions and tangents.

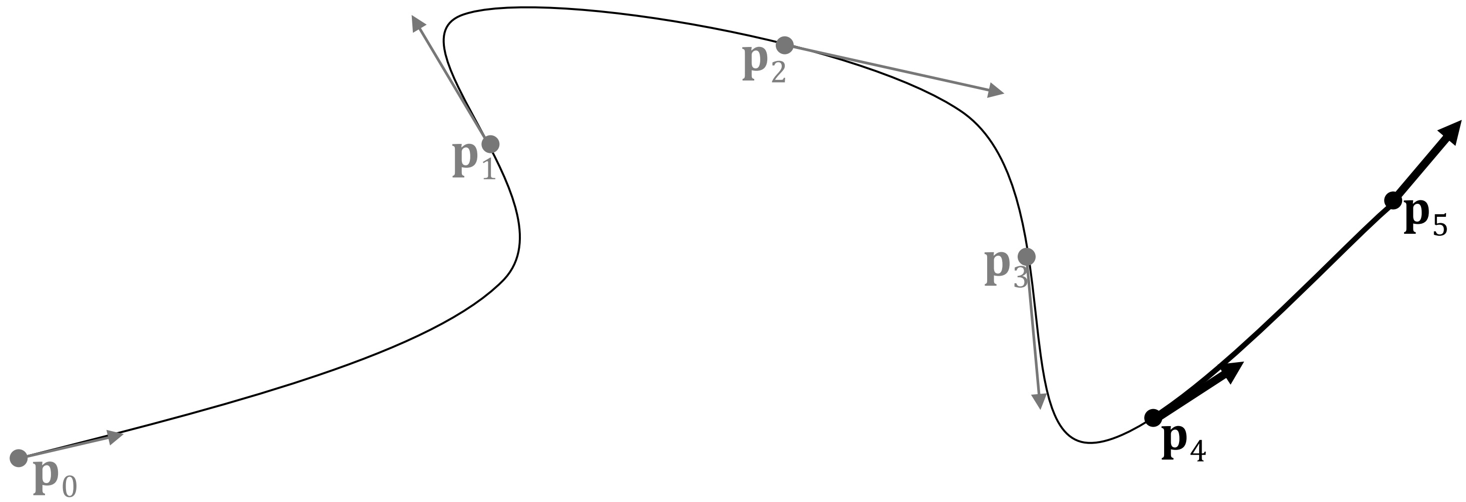


Specific Example: Hermite Splines

Interpolating piecewise *cubic* polynomial, each specified by:

- Start/end positions
- Start/end tangents

Iteratively construct the curve between adjacent end points that interpolate positions and tangents.

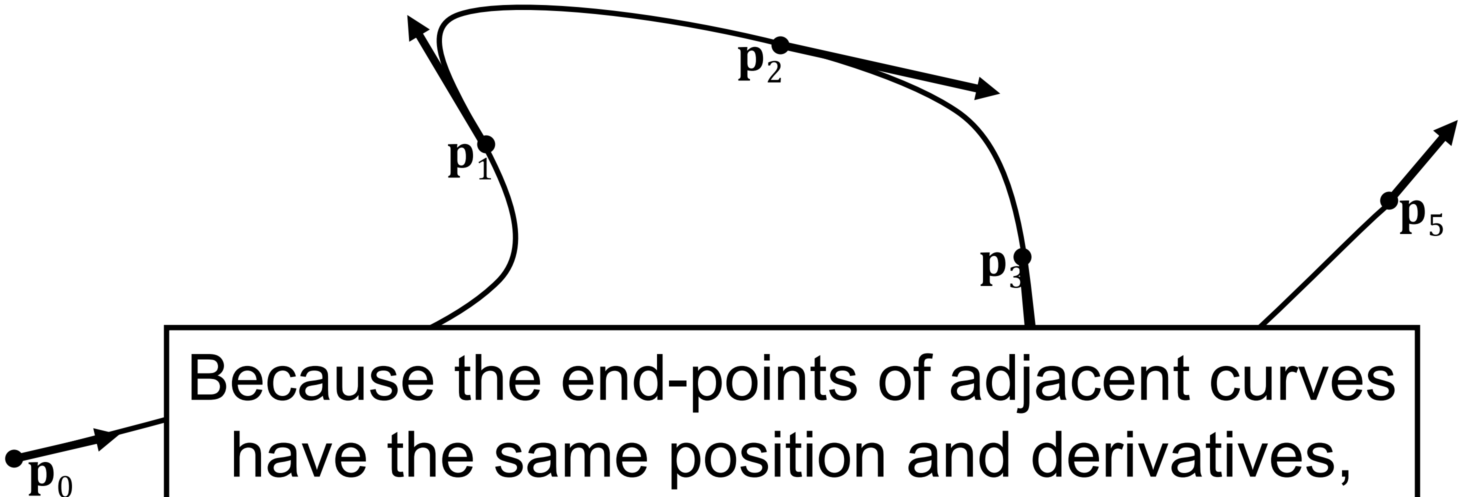


Specific Example: Hermite Splines

Interpolating piecewise *cubic* polynomial, each specified by:

- Start/end positions
- Start/end tangents

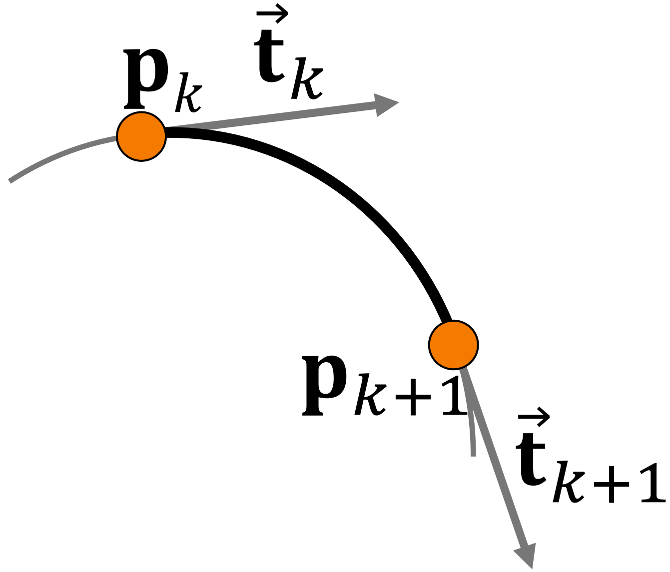
Iteratively construct the curve between adjacent end points that interpolate positions and tangents.



Because the end-points of adjacent curves have the same position and derivatives, the Hermite spline is C^1 by construction.

Specific Example: Hermite Splines

Let $\mathbf{P}_k(u) = \big(x_k(u), y_k(u)\big)$ with $0 \le u \le 1$ be the polynomial curve for the section between control points $\{\mathbf{p}_k, \vec{\mathbf{t}}_k\}$ and $\{\mathbf{p}_{k+1}, \vec{\mathbf{t}}_{k+1}\}$.



Boundary conditions are:

- $\mathbf{P}_k(0) = \mathbf{p}_k$
- $\mathbf{P}_k(1) = \mathbf{p}_{k+1}$
- $\mathbf{P}_k'(0) = \vec{\mathbf{t}}_k$
- $\mathbf{P}_k'(1) = \vec{\mathbf{t}}_{k+1}$

Solve for the coefficients of the polynomials $x_k(u)$ and $y_k(u)$ that satisfy the boundary conditions.

Note:
Four constraints $\Rightarrow$ we need a cubic polynomial.

Specific Example: Hermite Splines

Recall:

For a polynomial:

$$\mathbf{P}_k(u) = \mathbf{a} \cdot u^3 + \mathbf{b} \cdot u^2 + \mathbf{c} \cdot u + \mathbf{d}$$

we have:

$$\mathbf{P}'_k(u) = 3 \cdot \mathbf{a} \cdot u^2 + 2 \cdot \mathbf{b} \cdot u + \mathbf{c}$$

Using the matrix representation:

$$\mathbf{P}_k(u) = \begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{pmatrix} \qquad \mathbf{P}'_k(u) = \begin{pmatrix} 3 \cdot u^2 & 2 \cdot u & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{pmatrix}$$

By abuse of notation, we will think of the coefficients $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, and $\mathbf{d}$ as d-dimensional vectors rather than scalars so that $\mathbf{P}_k(u)$ is a function taking values in

Specific Example: Hermite Splines

$$\mathbf{P}_k(u) = \begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{pmatrix} \qquad \mathbf{P}'_k(u) = \begin{pmatrix} 3 \cdot u^2 & 2 \cdot u & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{pmatrix}$$

The values/derivatives at the end-points are:

$$\mathbf{p}_k = \mathbf{P}_k(0) = \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{pmatrix}$$

Specific Example: Hermite Splines

$$\mathbf{P}_k(u) = \begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{pmatrix} \qquad \mathbf{P}'_k(u) = \begin{pmatrix} 3 \cdot u^2 & 2 \cdot u & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{pmatrix}$$

The values/derivatives at the end-points are:

$$\mathbf{p}_k = \mathbf{P}_k(0) = \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{pmatrix}$$

$$\mathbf{p}_{k+1} = \mathbf{P}_k(1) = \begin{pmatrix} 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{pmatrix}$$

Specific Example: Hermite Splines

$$\mathbf{P}_k(u) = \begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{pmatrix} \qquad \mathbf{P}'_k(u) = \begin{pmatrix} 3 \cdot u^2 & 2 \cdot u & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{pmatrix}$$

The values/derivatives at the end-points are:

$$\mathbf{p}_k = \mathbf{P}_k(0) = \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{pmatrix} \qquad \vec{\mathbf{t}}_k = \mathbf{P}'_k(0) = \begin{pmatrix} 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{pmatrix}$$

$$\mathbf{p}_{k+1} = \mathbf{P}_k(1) = \begin{pmatrix} 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{pmatrix}$$

Specific Example: Hermite Splines

$$\mathbf{P}_k(u) = \begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{pmatrix} \qquad \mathbf{P}'_k(u) = \begin{pmatrix} 3 \cdot u^2 & 2 \cdot u & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{pmatrix}$$

The values/derivatives at the end-points are:

$$\mathbf{p}_k = \mathbf{P}_k(0) = \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{pmatrix} \qquad \vec{\mathbf{t}}_k = \mathbf{P}'_k(0) = \begin{pmatrix} 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{pmatrix}$$

$$\mathbf{p}_{k+1} = \mathbf{P}_k(1) = \begin{pmatrix} 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{pmatrix} \qquad \vec{\mathbf{t}}_{k+1} = \mathbf{P}'_k(1) = \begin{pmatrix} 3 & 2 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{pmatrix}$$

Specific Example: Hermite Splines

$$\mathbf{p}_k = \mathbf{P}_k(0) = \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{pmatrix} \qquad \vec{\mathbf{t}}_k = \mathbf{P}'_k(0) = \begin{pmatrix} 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{pmatrix}$$

$$\mathbf{p}_{k+1} = \mathbf{P}_k(1) = \begin{pmatrix} 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{pmatrix} \qquad \vec{\mathbf{t}}_{k+1} = \mathbf{P}'_k(1) = \begin{pmatrix} 3 & 2 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{pmatrix}$$

Combining into a single matrix gives:

$$\begin{pmatrix} \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \vec{\mathbf{t}}_k \\ \vec{\mathbf{t}}_{k+1} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{pmatrix}$$

Specific Example: Hermite Splines

$$\begin{pmatrix} \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \vec{\mathbf{t}}_k \\ \vec{\mathbf{t}}_{k+1} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{pmatrix}$$

Inverting, we get:

$$\begin{pmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \vec{\mathbf{t}}_k \\ \vec{\mathbf{t}}_{k+1} \end{pmatrix}$$

Specific Example: Hermite Splines

$$\begin{pmatrix} \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \vec{\mathbf{t}}_k \\ \vec{\mathbf{t}}_{k+1} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{pmatrix}$$

Inverting, we get:

$$\begin{pmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \vec{\mathbf{t}}_k \\ \vec{\mathbf{t}}_{k+1} \end{pmatrix} = \begin{pmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \vec{\mathbf{t}}_k \\ \vec{\mathbf{t}}_{k+1} \end{pmatrix}$$

Specific Example: Hermite Splines

$$\begin{pmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{pmatrix} = \begin{pmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \vec{\mathbf{t}}_k \\ \vec{\mathbf{t}}_{k+1} \end{pmatrix}$$

Using the fact that:

$$\mathbf{P}_k(u) = \begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix} \cdot \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{pmatrix}$$

We get:

$$\mathbf{P}_k(u) = \underset{\text{parameters}}{\begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix}} \underset{\mathbf{M}_{\text{Hermite}}}{\begin{pmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}} \underset{\text{boundary info}}{\begin{pmatrix} \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \vec{\mathbf{t}}_k \\ \vec{\mathbf{t}}_{k+1} \end{pmatrix}}$$

Specific Example: Hermite Splines

$$\mathbf{P}_k(u) = \left(\begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix} \begin{pmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \right) \begin{pmatrix} \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \vec{\mathbf{t}}_k \\ \vec{\mathbf{t}}_{k+1} \end{pmatrix}$$

Multiplying out and rearranging terms, we get:

$$\begin{aligned} \mathbf{P}_k(u) = {} & (2u^3 - 3u^2 + 1) \cdot \mathbf{p}_k \\ + {} & (-2u^3 + 3u^2) \cdot \mathbf{p}_{k+1} \\ + {} & (u^3 - 2u^2 + u) \cdot \vec{\mathbf{t}}_k \\ + {} & (u^3 - u^2) \cdot \vec{\mathbf{t}}_{k+1} \end{aligned}$$

Specific Example: Hermite Splines

$$\mathbf{P}_k(u) = (2u^3 - 3u^2 + 1) \cdot \mathbf{p}_k + (-2u^3 + 3u^2) \cdot \mathbf{p}_{k+1} \\ + (u^3 - 2u^2 + u) \cdot \vec{\mathbf{t}}_k + (u^3 - u^2) \cdot \vec{\mathbf{t}}_{k+1}$$

Setting:

- $H_0(u) = 2u^3 - 3u^2 + 1$
- $H_1(u) = -2u^3 + 3u^2$
- $H_2(u) = u^3 - 2u^2 + u$
- $H_3(u) = u^3 - u^2$

we can write $\mathbf{P}_k(u)$ as:

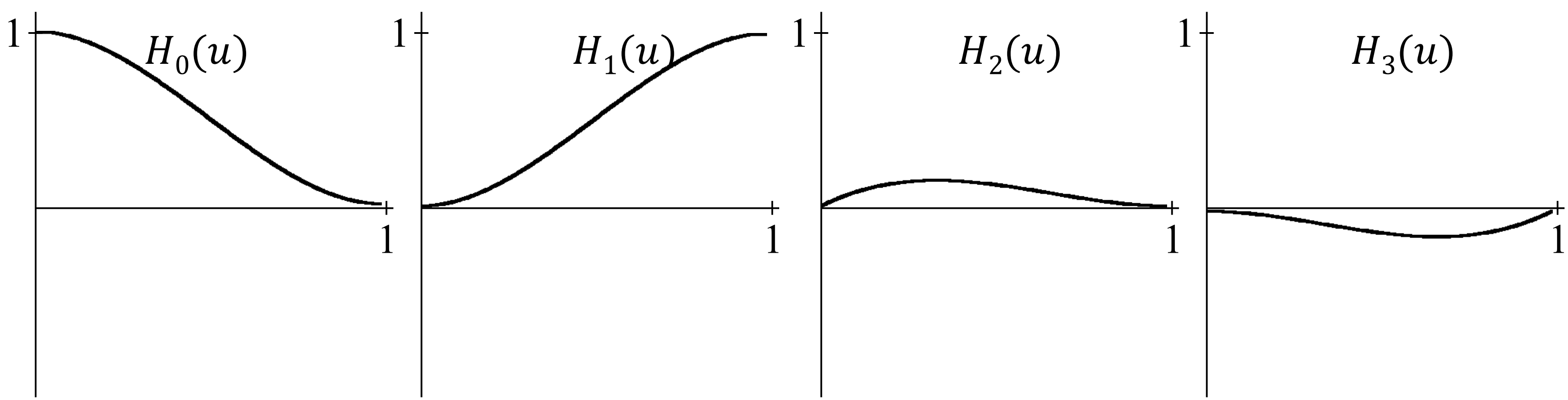
$$\mathbf{P}_k(u) = H_0(u) \cdot \mathbf{p}_k + H_1(u) \cdot \mathbf{p}_{k+1} + H_2(u) \cdot \vec{\mathbf{t}}_k + H_3(u) \cdot \vec{\mathbf{t}}_{k+1}$$

Specific Example: Hermite Splines

Setting:

- $H_0(u) = 2u^3 - 3u^2 + 1$
- $H_1(u) = -2u^3 + 3u^2$
- $H_2(u) = u^3 - 2u^2 + u$
- $H_3(u) = u^3 - u^2$

Blending Functions



$$\mathbf{P}_k(u) = H_0(u) \cdot \mathbf{p}_k + H_1(u) \cdot \mathbf{p}_{k+1} + H_2(u) \cdot \vec{\mathbf{t}}_k + H_3(u) \cdot \vec{\mathbf{t}}_{k+1}$$

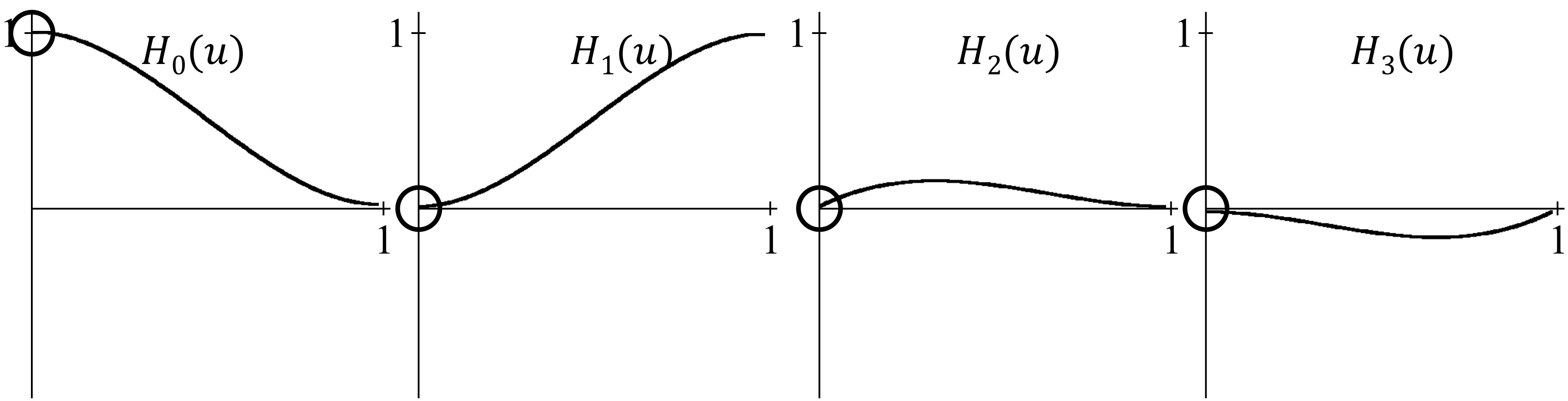
Specific Example: Hermite Splines

Setting:

- $H_0(u) = 2u^3 - 3u^2 + 1$
- $H_1(u) = -2u^3 + 3u^2$
- $H_2(u) = u^3 - 2u^2 + u$
- $H_3(u) = u^3 - u^2$

When $u = 0$:
- $H_0(u) = 1$
- $H_1(u) = 0$
- $H_2(u) = 0$
- $H_3(u) = 0$

So $\mathbf{P}_k(0) = \mathbf{p}_k$



$$\mathbf{P}_k(u) = H_0(u) \cdot \mathbf{p}_k + H_1(u) \cdot \mathbf{p}_{k+1} + H_2(u) \cdot \vec{\mathbf{t}}_k + H_3(u) \cdot \vec{\mathbf{t}}_{k+1}$$

Specific Example: Hermite Splines

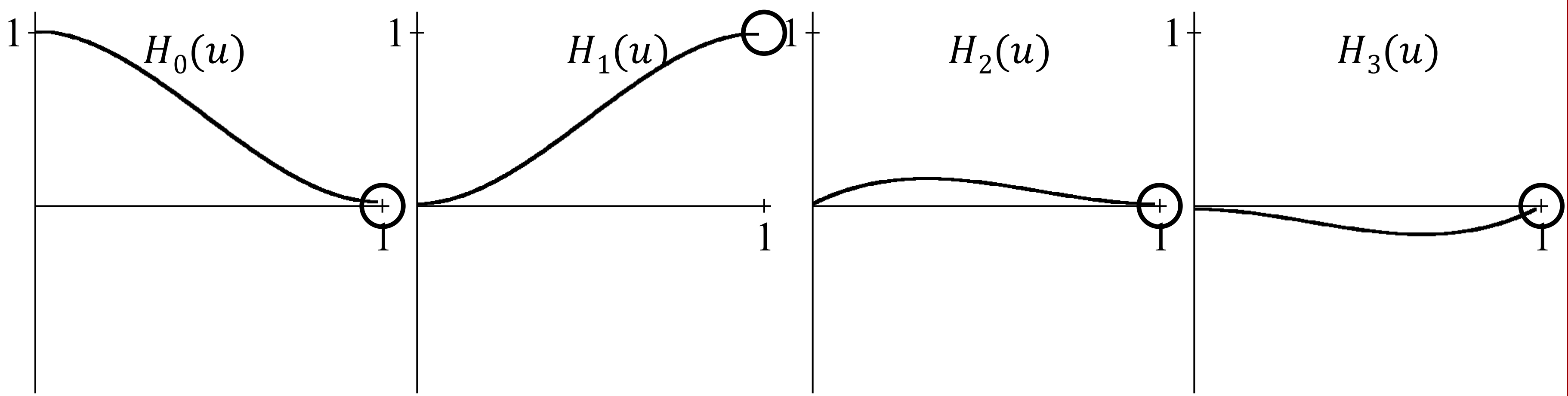
Setting:

- $H_0(u) = 2u^3 - 3u^2 + 1$
- $H_1(u) = -2u^3 + 3u^2$
- $H_2(u) = u^3 - 2u^2 + u$
- $H_3(u) = u^3 - u^2$

When $u = 1$:

- $H_0(u) = 0$
- $H_1(u) = 1$
- $H_2(u) = 0$
- $H_3(u) = 0$

So $\mathbf{P}_k(1) = \mathbf{p}_{k+1}$



$$\mathbf{P}_k(u) = H_0(u)\cdot\mathbf{p}_k + H_1(u)\cdot\mathbf{p}_{k+1} + H_2(u)\cdot\vec{\mathbf{t}}_k + H_3(u)\cdot\vec{\mathbf{t}}_{k+1}$$

Specific Example: Hermite Splines

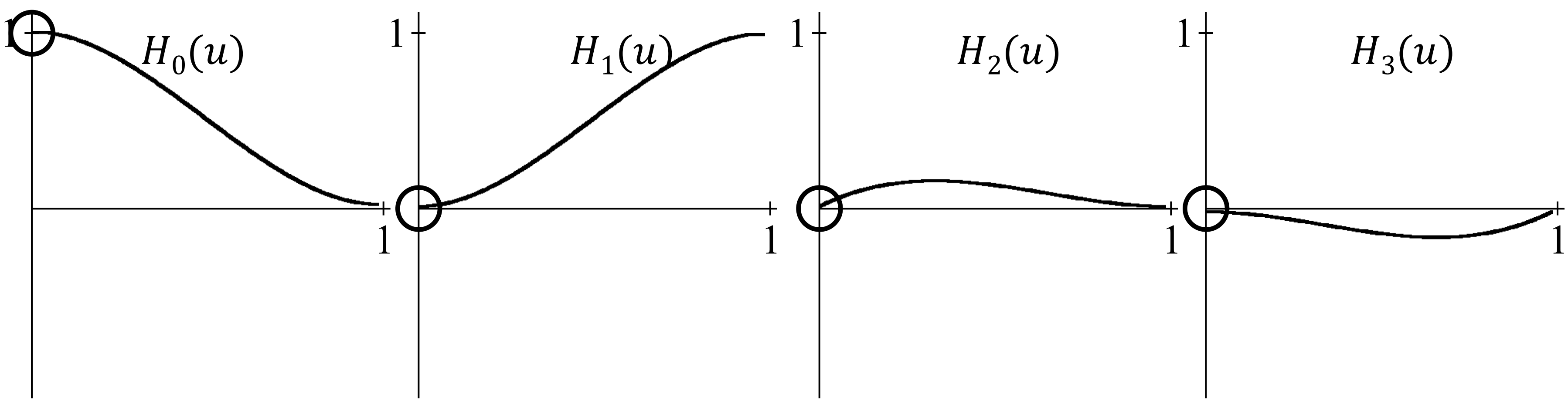
Setting:

- $H_0(u) = 2u^3 - 3u^2 + 1$
- $H_1(u) = -2u^3 + 3u^2$
- $H_2(u) = u^3 - 2u^2 + u$
- $H_3(u) = u^3 - u^2$

When $u = 0$:

- $H_0'(u) = 0$
- $H_1'(u) = 0$
- $H_2'(u) = 1$
- $H_3'(u) = 0$

So $\mathbf{P}_k'(0) = \vec{\mathbf{t}}_k$



$$\mathbf{P}_k'(u) = H_0'(u) \cdot \mathbf{p}_k + H_1'(u) \cdot \mathbf{p}_{k+1} + H_2'(u) \cdot \vec{\mathbf{t}}_k + H_3'(u) \cdot \vec{\mathbf{t}}_{k+1}$$

Specific Example: Hermite Splines

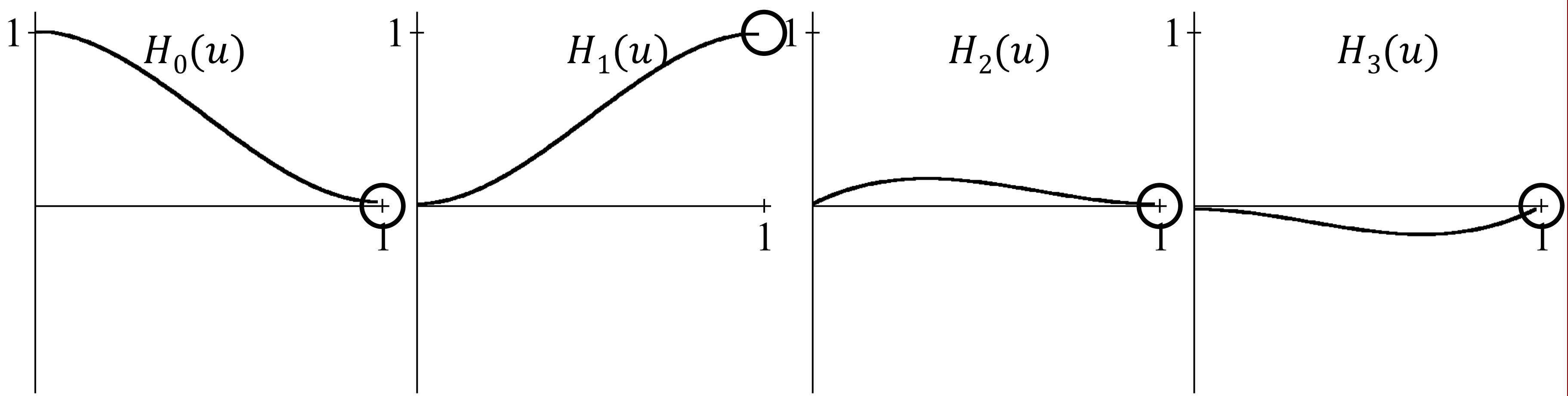
Setting:

- $H_0(u) = 2u^3 - 3u^2 + 1$
- $H_1(u) = -2u^3 + 3u^2$
- $H_2(u) = u^3 - 2u^2 + u$
- $H_3(u) = u^3 - u^2$

When $u = 1$:

- $H_0'(u) = 0$
- $H_1'(u) = 0$
- $H_2'(u) = 0$
- $H_3'(u) = 1$

So $\mathbf{P}_k'(1) = \vec{\mathbf{t}}_{k+1}$



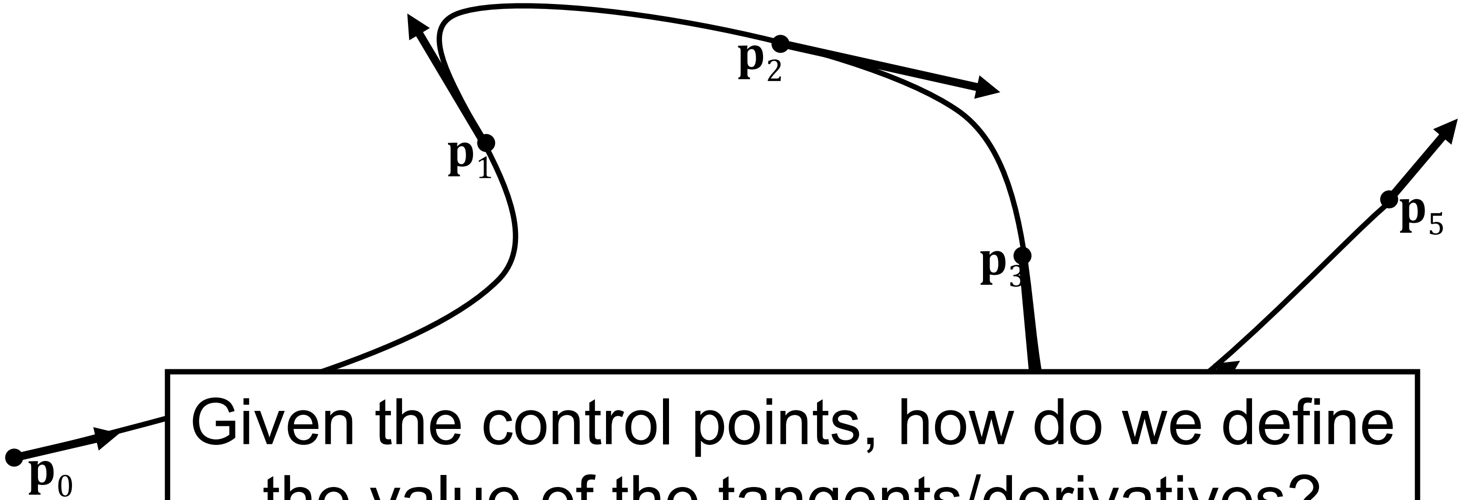
$$\mathbf{P}_k'(u) = H_0'(u) \cdot \mathbf{p}_k + H_1'(u) \cdot \mathbf{p}_{k+1} + H_2'(u) \cdot \vec{\mathbf{t}}_k + H_3'(u) \cdot \vec{\mathbf{t}}_{k+1}$$

Specific Example: Hermite Splines

Interpolating piecewise *cubic* polynomial, each specified by:

- Start/end positions
- Start/end tangents

Iteratively construct the curve between adjacent end points that interpolate positions and tangents.



Given the control points, how do we define the value of the tangents/derivatives?