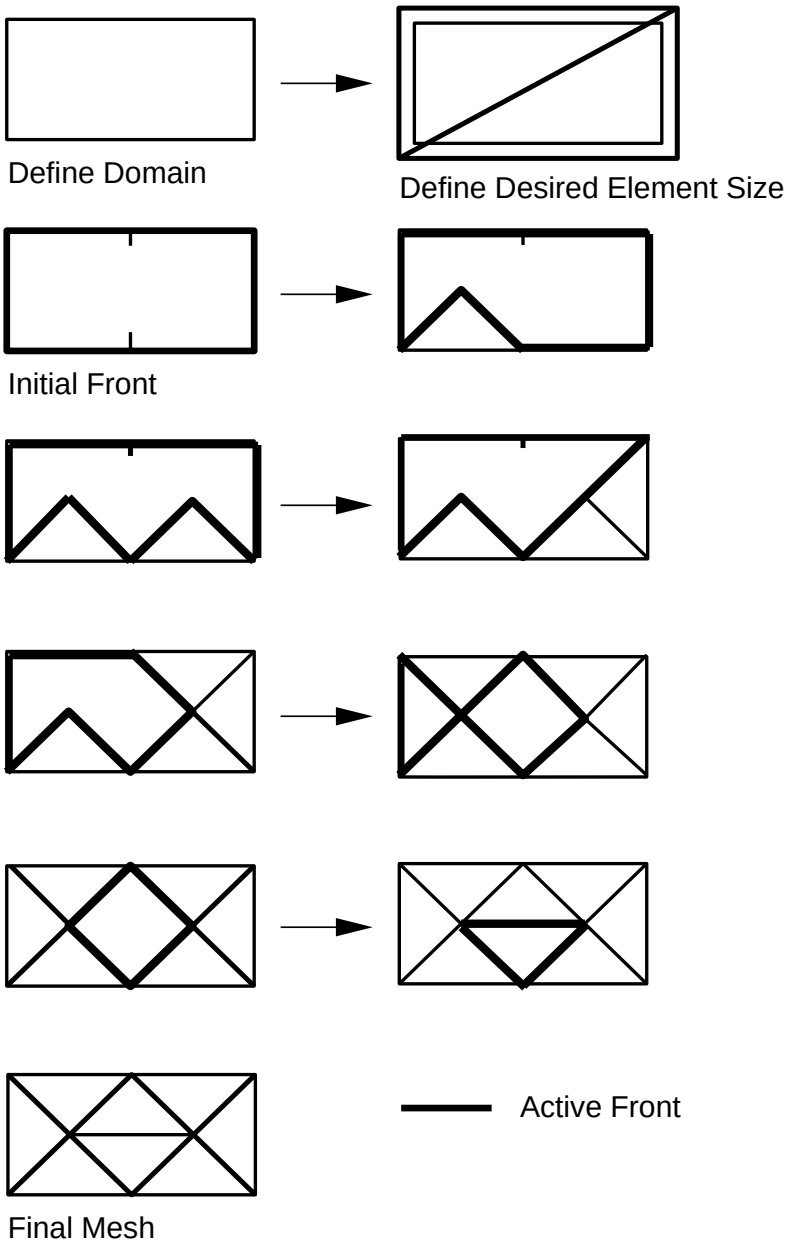
ADVANCING FRONT (1)



ADVANCING FRONT (2)

1. Set up a background grid/sources to determine element sizes
2. Define boundaries of domain to be gridded
3. Set up faces on all boundaries
4. Select the next face to be deleted from the front
5. For this face:
   - 5.1 Select a 'best point' position for the introduction of a new point `IPNEW`
   - 5.2 Determine whether there exists a point in the already generated grid that should be used in lieu of the new point; if so, set this point to `IPNEW` and continue searching (go to 5.2)
   - 5.3 Determine whether the element formed with the selected point `IPNEW` does not cross any given faces; if it does, select a new point as `IPNEW` and try again (go to 5.3)
6. Add the new element, point and faces to their lists
7. Find the generation parameters for the new faces
8. Delete the known faces from the new element
9. If there are any faces left in the front, go to 4

POTENTIALLY CPU-INTENSIVE OPERATIONS

a) Finding the next face to be deleted (step 4)

   ⇒ Use Heap Lists

b) Finding the closest given points to a new point (step 5.2)

   ⇒ Use Octrees

c) Finding the faces adjacent to a given point (step 5.3)

   ⇒ Use Linked Lists

d) Finding for any given location the values of generation parameters from the background grid (step 7) (an interpolation problem on unstructured grids)

   ⇒ Use Near Neighbour Search

INTERPOLATION ON UNSTRUCTURED GRIDS

1. Construct the quadtree for the grid from which the information is to be interpolated. This is the 'background grid'.
2. Construct the linked list that stores the elements surrounding points on the background grid.
3. For each point of the grid to which we want to interpolate:
   - 3.1 Find the closest point on the background grid using the quadtree.
   - 3.2 Find the elements surrounding this point from the linked list.
   - 3.3 Enlarge the region by one layer of elements using the linked list.
   - 3.4 Using the local element coordinates, find the element into which the point to be interpolated falls.
   - 3.5 Interpolate.

GENERATION OF THE INITIAL FRONT

Two main steps:

a) generate sides along line segments

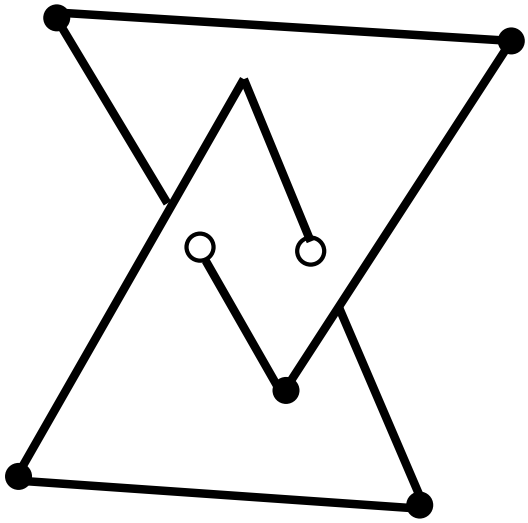
b) triangulate surface segments (2-D advancing front generator) $\Rightarrow$ faces

Use mappings to maintain in 2-D world approximate shape and size of 3-D surface patch

- 3-D surface segment to unit 2-D triangle or square $(x, y, z \rightarrow \xi, \eta)$
- stretching and shearing of unit 2-D triangle or square to approximate 3-D surface segment in a 2-D domain $(\xi, \eta \rightarrow \xi'', \eta'')$.

CHECKING THE INTERSECTION OF FACES (1)

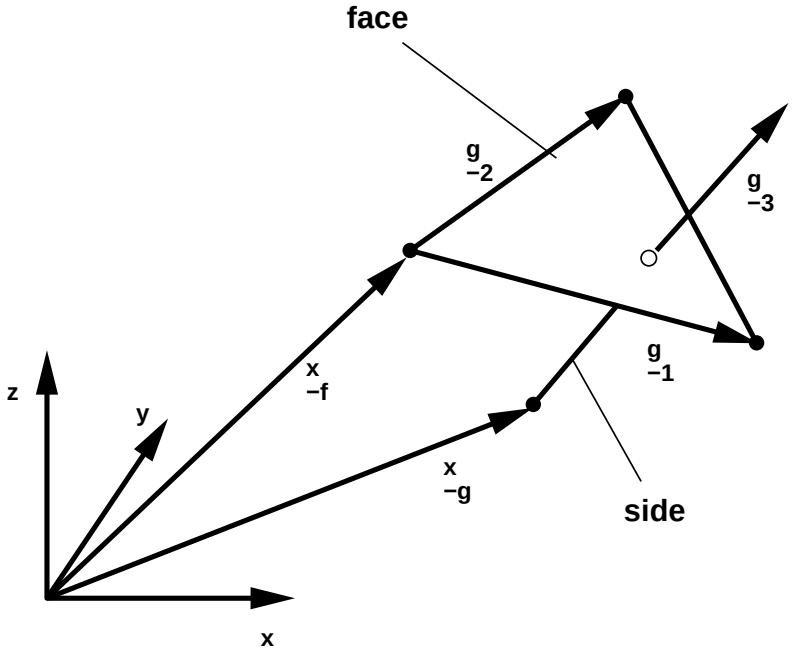
Based on following observation: Two triangular faces do not intersect if no side of either face intersects the other face



⇒

- Build all possible side-face combinations between any two faces and check them in turn;
- If no intersection found:
  ⇒ faces do not cross

CHECKING THE INTERSECTION OF FACES (2)



Face-Side Combination

$$\mathbf{x}_f + \alpha^1 \mathbf{g}_1 + \alpha^2 \mathbf{g}_2 = \mathbf{x}_s + \alpha^3 \mathbf{g}_3$$

$\mathbf{g}_i$: covariant basis

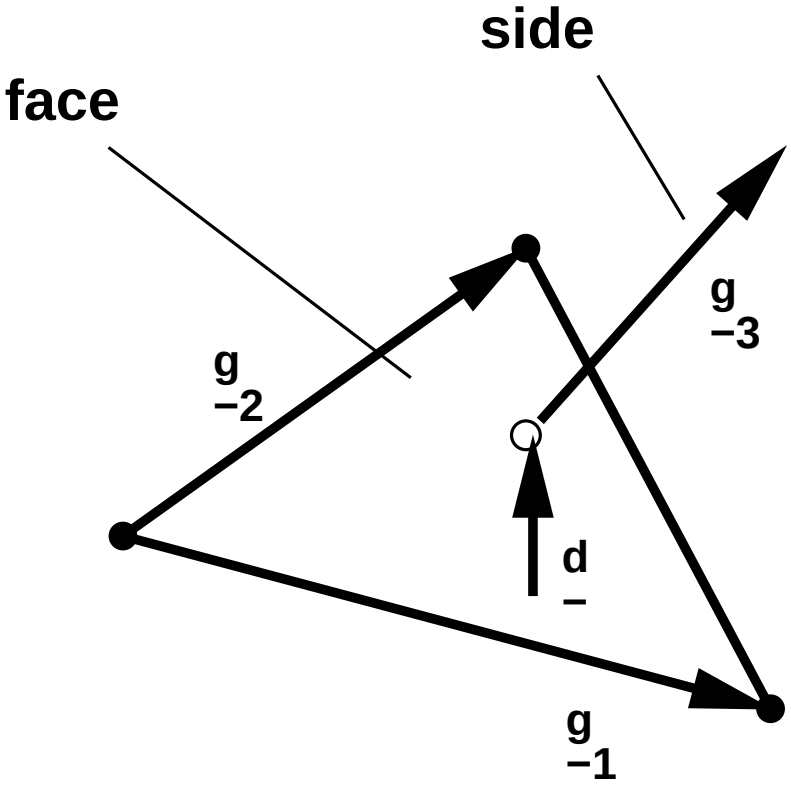
$\mathbf{g}^i$: contravariant basis

$$\mathbf{g}^i \cdot \mathbf{g}_j = \delta^i_j$$

$\Rightarrow$

$$\alpha^1 = (\mathbf{x}_s - \mathbf{x}_f) \cdot \mathbf{g}^1 \quad , \quad \alpha^2 = (\mathbf{x}_s - \mathbf{x}_f) \cdot \mathbf{g}^2$$
$$\alpha^3 = (\mathbf{x}_f - \mathbf{x}_s) \cdot \mathbf{g}^3 \quad , \quad \alpha^4 = 1 - \alpha^1 - \alpha^2$$

CHECKING THE INTERSECTION OF FACES (3)



Face regarded as 'crossed' if

$$max(-\alpha^i, \alpha^i - 1) < toler$$

Very CPU-intensive ⇒ use a layered approach

a) Min/Max-search

b) Local element coordinates

c) In-depth analysis of side-face combinations

SWEEP AND RETRY

Observation: Sometimes, it becomes impossible to introduce a good tetrahedron

⇒ Complete as much as possible, coarsen and retry

SR.1 Remove elements adjoining remaining faces ⇒ new list of faces ⇒ new front

SR.2 Remove all unused points

SR.3 Attempt to complete mesh using advancing front method

SR.4 If any faces left: Goto SR.1

Difference between algorithm and method