

## Grids 2

torsdag 12. oktober 2017 13:44

1. Here, we consider unstructured grid generation using a spacing function (with applications to adaptivity to follow).

a) Begin with the 1D case. Suppose we have a function  $H: [a, b] \rightarrow \mathbb{R}_+$  (spacing function)

so that  $H(x)$  is desired spacing at  $x$ .

$$\text{then let } N = \int_a^b \frac{1}{H(x)} dx.$$

$$\text{set } N = \max(1, \lfloor N \rfloor) \quad \begin{array}{l} \text{integer part} \\ \text{of } N \\ \text{ie. round down} \end{array}$$

then construct  $x_i$  such that

$$\frac{N}{N} \int_a^{x_i} \frac{1}{H(x)} dx = i.$$

→ check that if  $H = h$  constant, will find  $h = \frac{b-a}{N}$ , assuming  $h$  divides  $b-a$ .

↳ to guarantee that  $N$  is an integer.

b) Now suppose we fix  $N$ . To do this, we solve the IVP

$$y'(x) = \frac{N}{N} \frac{1}{H(x)}, \quad y(a) = 0.$$

then  $x_i$  obtained by  $y(x_i) = i$ .

(side note: when using such methods for adaptive ODE timestepping, can combine with solution of ODE).

→ solve this system numerically.

c) A simplified method:

define  $\tilde{x}_i = \tilde{x}_{i-1} + H(\tilde{x}_{i-1})$  until  $\tilde{x}_M$   
first past  $b$ .

then refine by

$$x_i = x_{i-1} + k H(\tilde{x}_{i-1}), \quad k = \frac{b - \tilde{x}_{N-1}}{\tilde{x}_N - \tilde{x}_{N-1}}$$

$$x_0 = a.$$

where  $N = M$  or  $M-1$   
according to which of  
 $\tilde{x}_M, \tilde{x}_{M-1}$  is close  
to  $b$ .

---

2. Triangulations and spacing:

Sometimes use a generalized notion of a spacing function:

$$H: \Omega \rightarrow \mathbb{R}^{2 \times 2}, \quad \text{positive definite, symmetric}$$

idea: eigenvectors represent max and min stretch directions, eigenvalues give spacings.

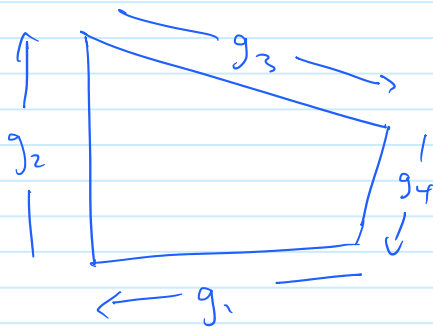
(controls shape).

Here, we control only element diameters, so

use a scalar function.

a) Generate vertices on boundary:

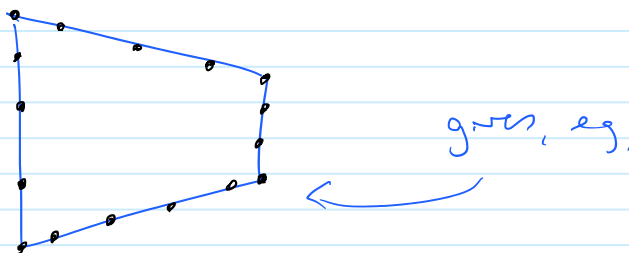
first, let  $g_1(t), \dots, g_M(t)$  parametrize the boundary



On each side, we generate a set of points in a similar manner to 1D case, but using line integrals:

$$\int_0^{t_j^i} \frac{1}{H(g_i(\tau))} \left| \frac{dg_i(\tau)}{d\tau} \right| d\tau = j$$

$$\text{and } x_j^i = g_i(t_j^i)$$

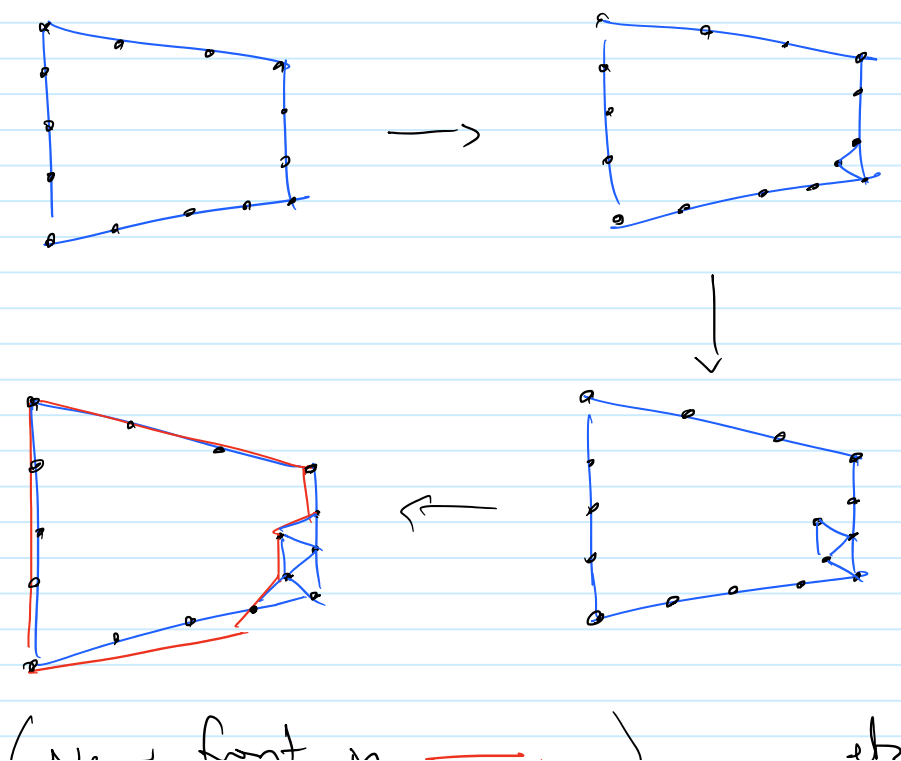


NB: if boundary was curved, at this point linear interpolations of points will be the boundary of our triangulation.

b) The advancing front method keeps track of the front (interior boundary of the incomplete triangulation), and adds triangles along the front until

the front disappears. At each step

- i) pick an edge of front (eg the smallest edge) say  $AB$
- ii) place a point a distance away from front as specified by spacing function call this  $C$
- iii) test if either there are any already existing vertices near the point, if no, replace  $C$  by this.
- iv) test if the triangle  $ABC$  intersects the front, if no replace  $C$  by a point on front
- v) add  $ABC$  to triangulation, remove  $AB$  from front and add  $AC / BC$  (or if  $AC$  or  $BC$  were in front already, remove them instead).



(New front in —) etc.

3. Post processing: sometimes, after the grid has been generated, we can modify it to improve regularity etc. Consider two approaches:

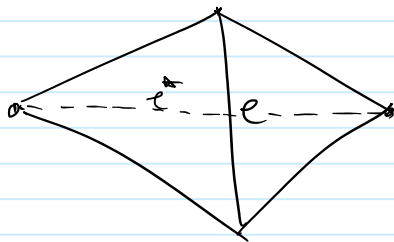
1. Diagonal swapping (already familiar from the Delaunay algorithm)
2. Node displacement.

a) Diagonal swaps: suppose we have some regularity measure of a triangle,

$$\text{eg } S(K) = \left| \frac{|K|}{\sum_{\substack{\text{sum} \\ \text{over} \\ \text{edges of} \\ K}} (e_k)^2} - \frac{\sqrt{3}}{12} \right|$$

which is zero for equilateral triangles.

then for each edge  $e$  in triangulation, form



see if sum of squares of  $S$  is decreased by swapping  $e$  with  $e^*$ .

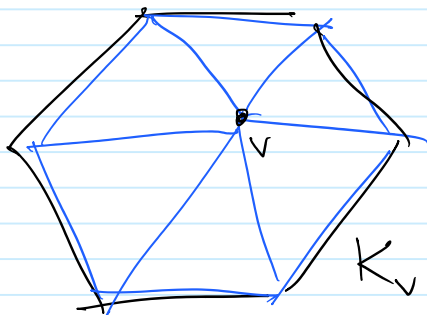
loop through all edges like this until no swaps improve.

NB: as each map strictly reduces  $\sum_K [S(K)]^2$ , we will terminate.

Also, the function  $S(K)$  might also take into account global features of the geometry.

b) Node displacement: previous routines modify connectivity data, not vertex locations. Now consider modifications that leave connectivity invariant and move vertex points.

i) Laplacian regularization: here, we consider the polygon of all elements containing a given internal vertex  $v$ :



we then move  $v$  to the barycentre (centre of mass) of  $K_v$

$$\text{ie } v \rightarrow v' = \frac{1}{|K_v|} \int_K x \, dx$$

Minimizes

$$\sum \int (x_v - x)^2 \, dx \quad \text{if process}$$

$$\sum_v \int_{K_v} (x_v - x)^2 dx \quad \text{if process converges.}$$

ii) Can move points to other locations, eg centre of mass of  $\partial K_v$

$$v \mapsto v' = \frac{1}{|\partial K_v|} \int_{\partial K} x dx$$

or simple centre of mass of vertices of  $\partial K$ :

$$v \mapsto v' = \frac{\sum_{i=1}^n w_i}{n}$$

where  $w_i, i=1, \dots, n$  are these vertices.

(can produce bad results, but simple)

iii) Can modify for spacing function, eg.

$$v \mapsto v' = \frac{\int_{K_v} \mu(x) x dx}{\int_{K_v} \mu(x) dx.}$$

moves to centre of mass with density function  $\mu(x)$ , eg take  $\mu(x) = \frac{1}{H(x)}$

iv) Concave case: centre of mass may be outside element. Can form  $\hat{K}_i$  instead from elements in convex subdomain  $\hat{\Omega}_i$ .