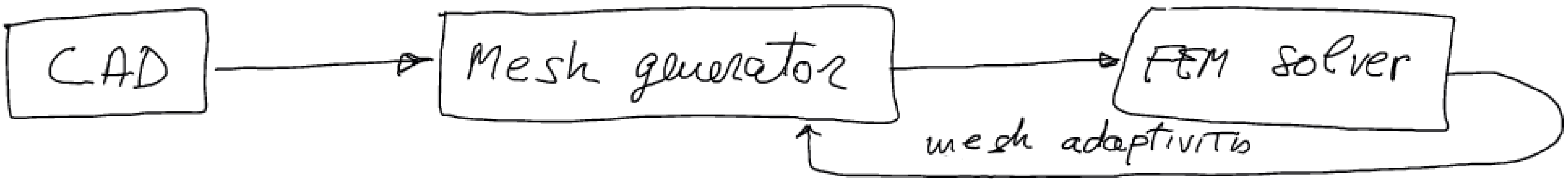
Triangolazione ed elementi costanti e lineari

Lo scopo è quello di definire spazi finito-dimensionali che siano sottospazi di H^1 (oppure H^1_0).

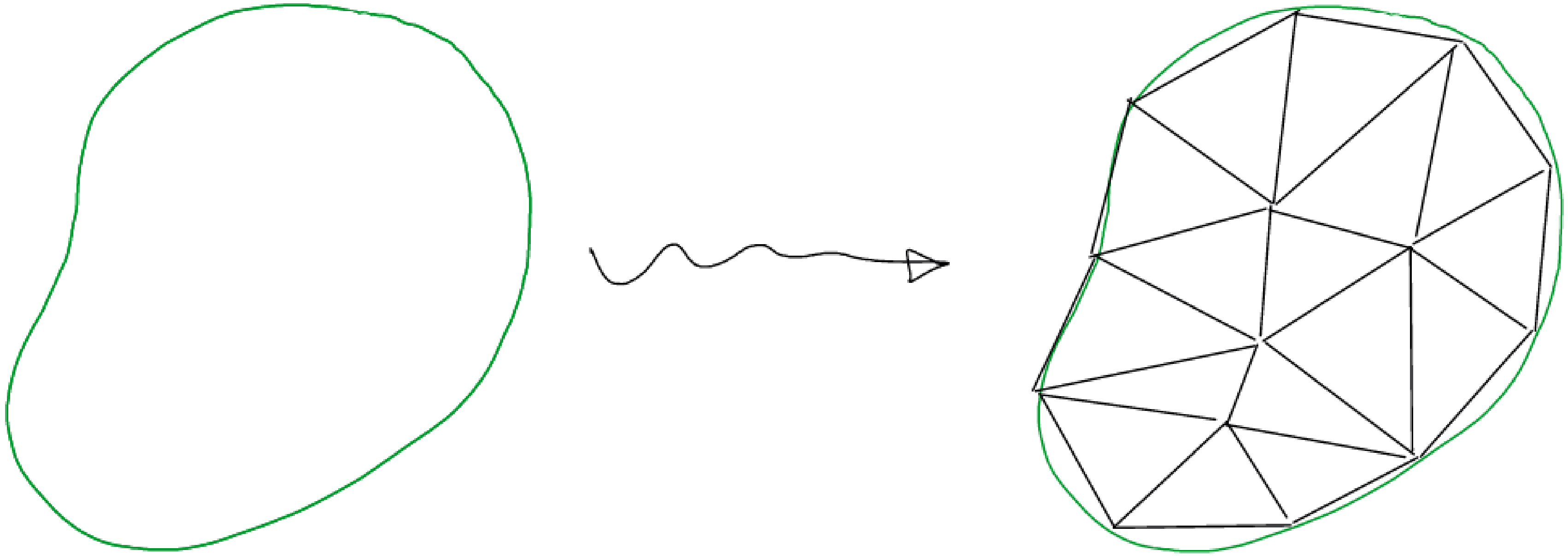
In particolare:

- definiremo il concetto di triangolazione $\mathcal{T}_h$ di Ω e le qualità che una triangolazione deve avere
- utilizzeremo codice disponibile (MATLAB oppure on-line) per costruire triangolazioni
- visualizzeremo l'interpolata P0 oppure P1 su $\mathcal{T}_h$

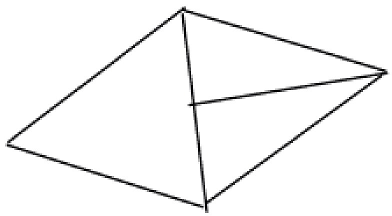
Premessa: la generazione della mesh è una parte complessa nelle applicazioni reali (geometria 3D da CAD) e indipendente dal solutore FEM



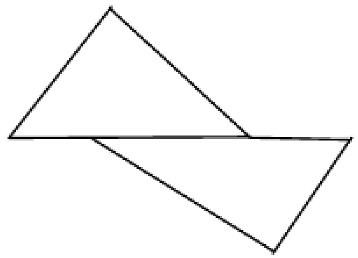
Noi ci limiteremo ad una descrizione rapida e all'utilizzo di codice "black box" nel caso bidimensionale $\Omega \subset \mathbb{R}^2$



1. due triangoli ("elementi") adiacenti condividono un intero lato oppure un vertice di $\mathcal{T}_h$, cioè sono esclusi i casi: (ADMISSIBILITY)



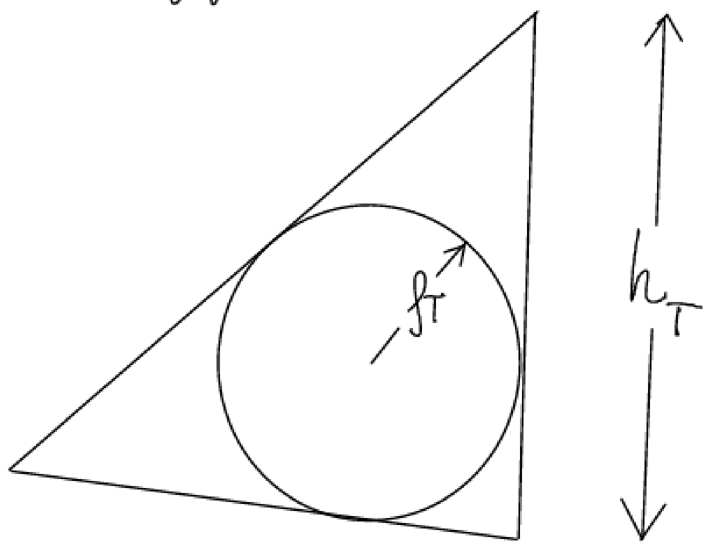
oppure



(2) (SHAPE REGULARITY): non vogliamo triangoli troppo "acuti". Precisamente, se consideriamo una famiglia di triangolazioni $\{\mathcal{T}_h\}_{h>0}$, e definiamo

h_T = diametro dell'elemento $T \subset \mathcal{T}_h$

ρ_T = raggio del cerchio inscritto in T

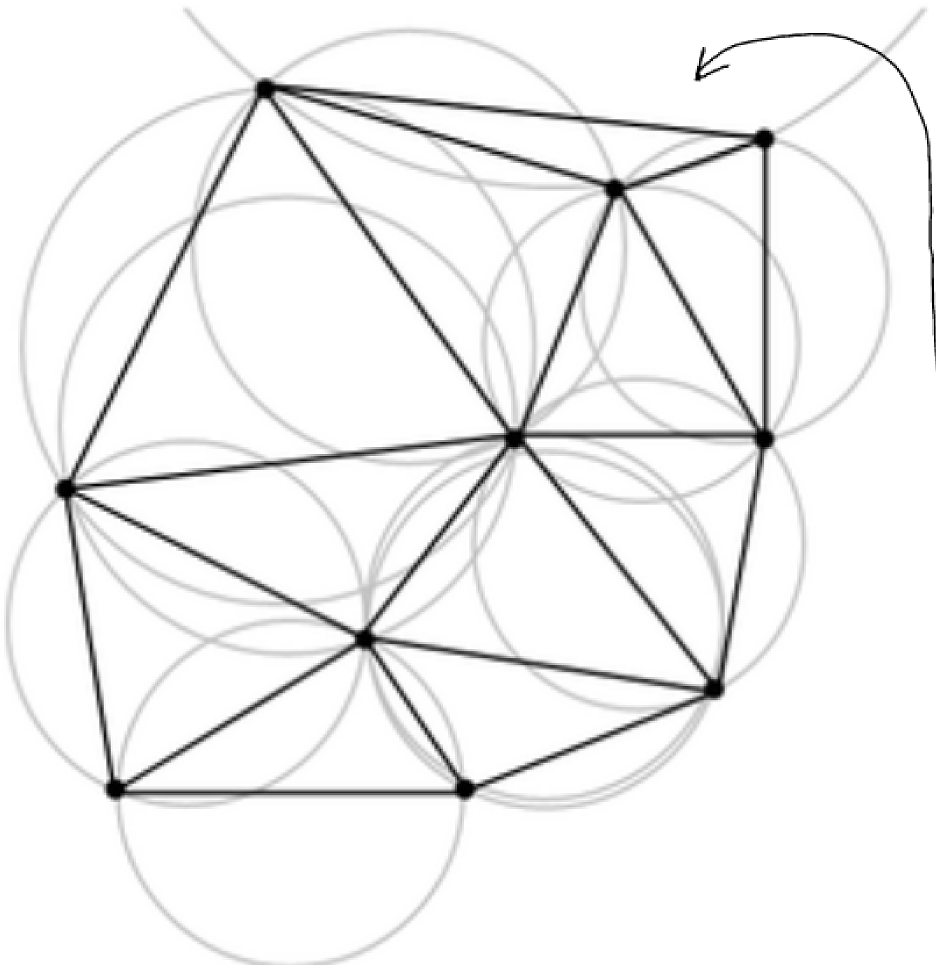


allora $$\sup_h \sup_{T \in \mathcal{T}_h} \frac{h_T}{\rho_T} \leq C_{shape} < +\infty$$

oss: il parametro h della famiglia $\mathcal{T}_h$ solitamente è $h = \max_{T \in \mathcal{T}_h} h_T$

Per garantire la shape-regularity di una triangolazione usualmente si chiede la proprietà di Delaunay:

Il cerchio circoscritto ad ogni triangolo non contiene vertici di altri triangoli. Ad esempio:

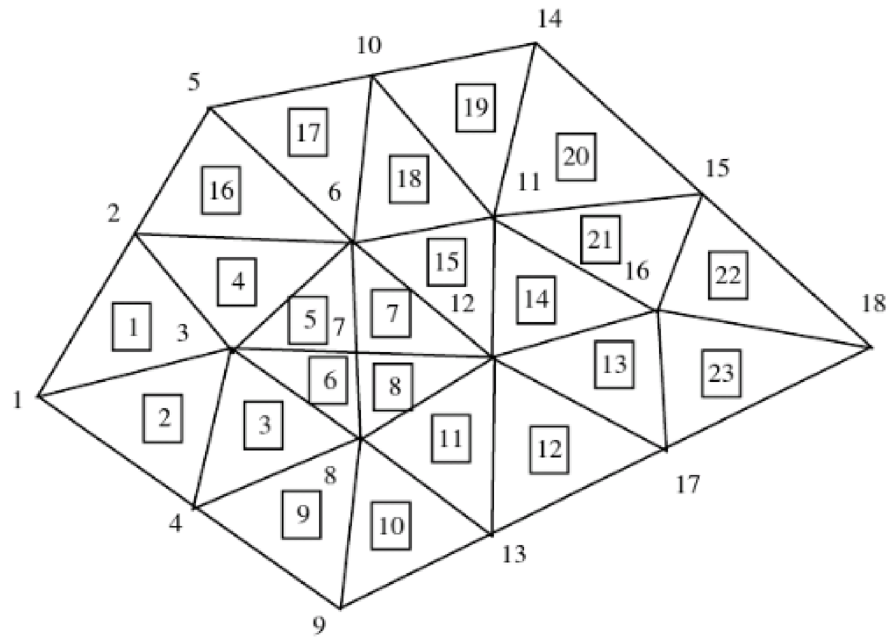


Inoltre, esistono algoritmi veloci per trovare triangolazioni di Delaunay dato un insieme iniziale di vertici (in matlab: delaunay(x, y))

osservazione: non sono impediti i triangoli "acuti" ma, dati i vertici, una triangolazione di Delaunay è in questo senso una triangolazione ottimale

Come si descrive una triangolazione? la struttura dati essenziali è formata dalle coordinate dei vertici e dalla matrice di incidenza.

$$p = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \\ \vdots & \end{bmatrix};$$



$$t = \begin{bmatrix} 1 & 3 & 2 \\ 1 & 4 & 3 \\ 4 & 8 & 3 \\ & \vdots & \end{bmatrix};$$

sulla riga i-sima ci sono gli indici dei vertici del triangolo di indice i (ordine possibilmente antiorario

Alcuni codici per triangolare domini 2D

- il PDEToolbox di Matlab contiene (anche) un triangolatore che consente di raffinare in modo adattivo, consente di descrivere (in un file) o disegnare (via interfaccia grafica) domini complessi

  [p, e, t] = initmesh(geom_file, parametri ....)

  [p, e, t] = refinemesh(....)

- Triangle (by J.R. Shewchuk), versatile and well written C triangulator

- distmesh_2d + delaunay : il primo (by Per-Olof Persson e Gilbert Strang) posiziona i vertici e successivamente chiama il secondo (comando nativo in MATLAB) per "triangolare".

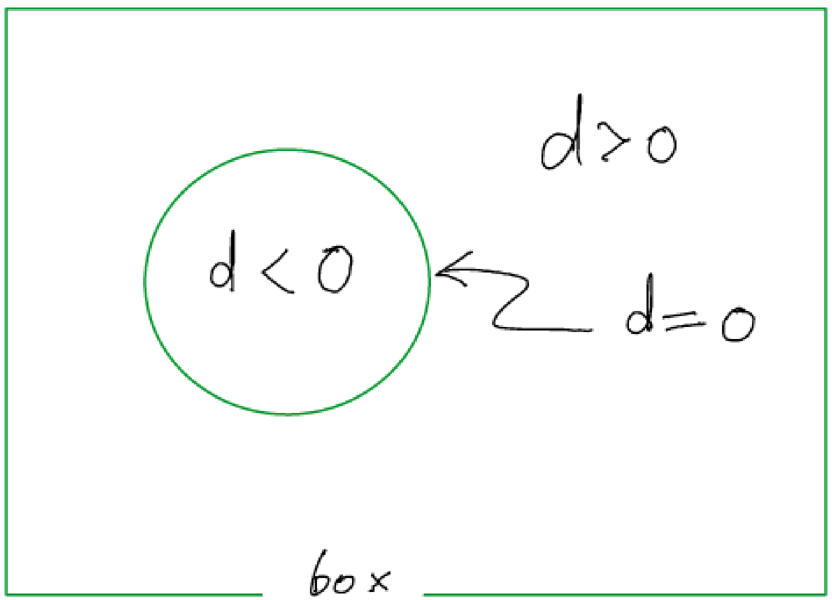
Noi utilizzeremo l'ultima soluzione:

$[p,t] = \text{distmesh\_2d}\,(df,\ hf,\ h_0,\ box,\ n_{it},\ [\,])$

dove

$df =$ signed dist. function (handle)

$[d_1; d_2; \dots] = df\,([x_1, y_1;\ x_2, y_2,\ x_3, y_3;\ \dots])$



es:

$$df = \sqrt{x^2+y^2} - 1$$

cioè in matlab:

`df = @(p) (sum(p.^2, 2)).^(1/2) -1`

Osservazione: unione, differenza, intersezione di domini si possono facilmente scrivere in termini della df!

hf = scaled meshsize function (handle)
ha la stessa sintassi di df e restituisce uno scalare che è proporzionale alla mesh size.
Se si vuole una mesh "quasi-uniforme" hf è la function (handle) costante:

hf = @(p) ones(size(p,2), 1)

h_0 = scalare per indicare il diametro massimo della mesh (diametro dell'elemento più grande)

n_{it} = iterazioni dell'algoritmo che ottimizza la posizione dei vertici della triangolazione

l'ultimo argomento è un vettore di punti che rappresentano vertici vincolati della triangolazione; non useremo questa funzione

Rappresentare graficamente $\mathcal{T}_h$ e funzioni costanti a tratti (P0) oppure lineari a tratti (P1) continue

Matlab dispone dei due comandi

trimesh $(t, p(:,1), p(:,2), \dots)$ disegna mesh o funzione

trisurf $(\dots)$ disegna grafico funzione

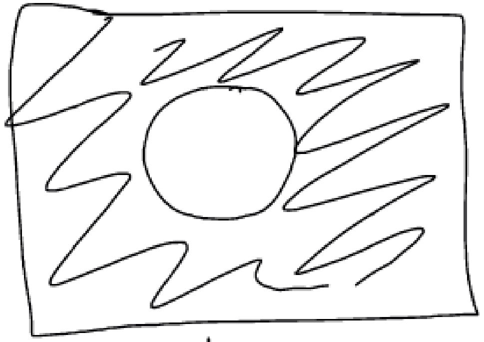
patch $\left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}, \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix}, \dots \right)$ disegna un solo triang.

se input è matrice allora disegna più triangoli (uno per col.)

trimesh e trisurf consentono di fare il grafico di funzioni P1 continue, ma per le costanti a tratti si può utilizzare patch

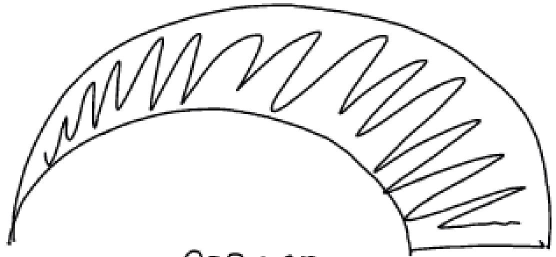
Esercizi proposti

1. triangolare i seguenti domini:



buco

luna



corno

con mesh uniformi oppure mesh che "raffinano" alcune parti del dominio (ad esempio i vertici della luna)

2. individuare i vertici di bordo nei domini precedenti ed evidenziarli sulla mesh

3. rappresentare sul primo dominio ("buco") l'interpolata costante e lineare della funzione $u = x^2 + y^2$