

# Práctica 2.

# Mallado/Meshing

Métodos Numéricos para la Computación

Grado en Ingeniería Informática  
Escuela de Ingeniería Informática  
Universidad de Las Palmas de Gran Canaria

Curso 2015/2016

# Objetivos de la Práctica

Adquirir habituación, practica y experiencia en el uso de mallado de superficies y objetos, tanto 2D como 3D

Utilizar herramientas educativas como Distmesh para ilustrar las situaciones que suelen ser tratadas en herramientas más profesionales.

Adquirir habilidad en el uso de modelos de objetos 3D para fines numéricos o de Visualización.

# Tarea 1

- Descargar la herramienta Distmesh en una carpeta y descomprimirla
- Situar en MATLAB en esta carpeta. Observar la existencia de ficheros .m, .cpp, .mexw32 y .mexw64 → explicación del profesor.
- Ejecutar meshdemo2d para 2D
- Ejecutar meshdemon2d para 3D
- Ejecutar diversos demos 2D y 3D según las ordenes contenidos en el documento tutorial de Distmesh y trasladadas parcialmente a este documento.
- Capturar las ventanas que muestran las demos e incluirlas en una memoria con una explicación que recoja el material proporcionado y del trabajo personal del alumno.

# Distmesh 2D

```
function [p,t]=distmesh2d(fd,fh,h0,bbox,pfix,varargin)
```

This meshing function produces the following outputs:

- The node positions  $\mathbf{p}$ . This  $N$ -by-2 array contains the  $x, y$  coordinates for each of the  $N$  nodes.
- The triangle indices  $\mathbf{t}$ . The row associated with each triangle has 3 integer entries to specify node numbers in that triangle.

The input arguments are as follows:

- The geometry is given as a distance function  $\mathbf{fd}$ . This function returns the signed distance from each node location  $\mathbf{p}$  to the closest boundary.
- The (relative) desired edge length function  $h(x, y)$  is given as a function  $\mathbf{fh}$ , which returns  $h$  for all input points.
- The parameter  $\mathbf{h0}$  is the distance between points in the initial distribution  $p_0$ . For uniform meshes ( $h(x, y) = \text{constant}$ ), the element size in the final mesh will usually be a little larger than this input.
- The bounding box for the region is an array  $\mathbf{bbox}=[x_{\min}, y_{\min}; x_{\max}, y_{\max}]$ .
- The fixed node positions are given as an array  $\mathbf{pfix}$  with two columns.
- Additional parameters to the functions  $\mathbf{fd}$  and  $\mathbf{fh}$  can be given in the last arguments  $\mathbf{varargin}$  (type `help varargin` in MATLAB for more information).

# Distmesh 2D. Input

- `fd()` es una función de distancia al contorno de la superficie. El mallado se realiza en la zona que  $fd(x,y) < 0$ .
- `fh()` es una función que define el tamaño variable de los triángulos.
- `h0` es el tamaño de los triángulos sin “esfuerzos de deformación” en la situación inicial.
- `bbox` es el Bounding Box o rectángulo que incluye la función. útil para definir los puntos iniciales.
- `pp` es una colección de puntos fijos que serán incluidos en la triangulación, además de los que incluya aleatoriamente el programa.
- `Varairgin` son argumentos adicionales para las funciones `fd()` y `fh()`.

# Distmesh. Output

- P el conjunto de puntos de la triangulación que incluye los puntos fijos definidos por el usuario en pp, así como los puntos variables introducidos inicialmente de forma aleatoria y posteriormente movidos de sitio hasta alcanzar una situación de equilibrio.
- T el conjunto de triangulos

# Distmesh. Distribuciones

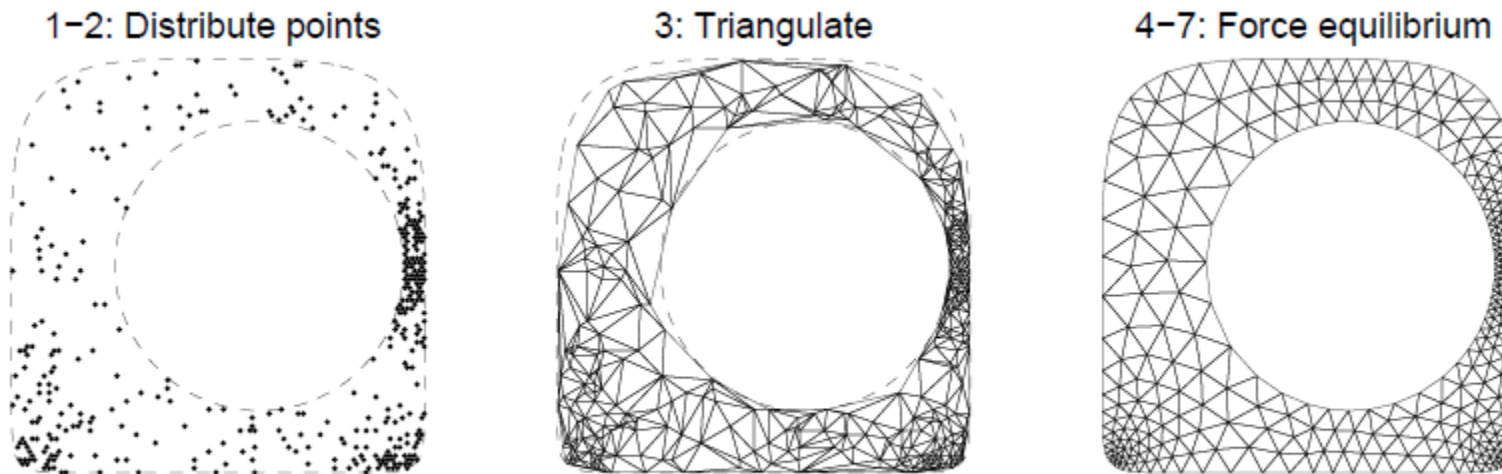


FIG. 3.2. *The generation of a non-uniform triangular mesh.*

Distribución inicial de puntos aleatorios. Triangulación inicial y triangulación final con triángulos de calidad y variable.

# Distmesh. Composición

Composición de sub-domios a partir de unión, intersección y diferencia de otros

**Union :**

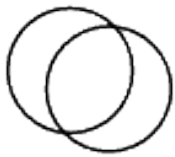
$$d_{A \cup B}(x, y) = \min(d_A(x, y), d_B(x, y))$$

**Difference :**

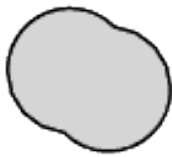
$$d_{A \setminus B}(x, y) = \max(d_A(x, y), -d_B(x, y))$$

**Intersection :**

$$d_{A \cap B}(x, y) = \max(d_A(x, y), d_B(x, y))$$



Overlapping  
Circles



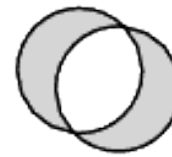
Union



Intersection

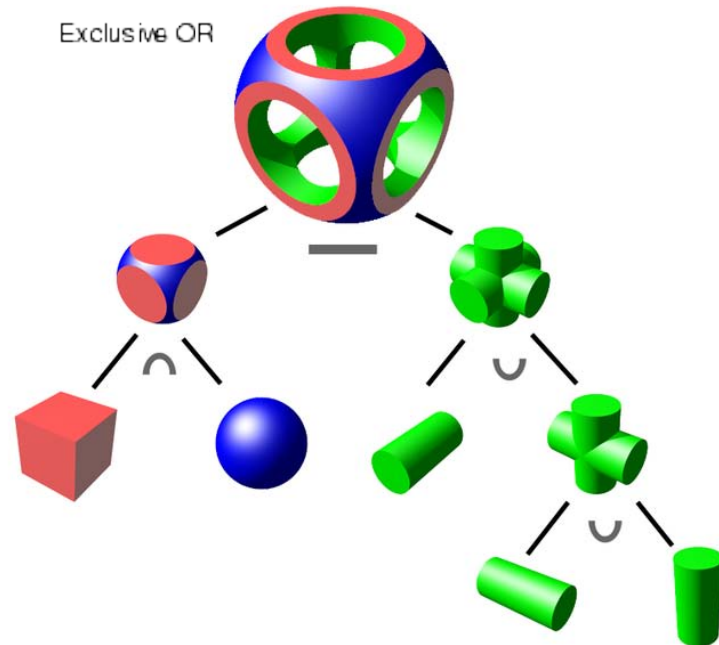


Subtraction



Exclusive OR

Operaciones muy comunes en programas de Diseño para Ingeniería Mecánica, por ejemplo SolidWorks





function d=dcircle(p,xc,yc,r) d=sqrt((p(:,1)-xc).^2+(p(:,2)-yc).^2)-r;	<i>% Circle</i>
function d=drectangle(p,x1,x2,y1,y2) d=-min(min(min(-y1+p(:,2),y2-p(:,2)), ... -x1+p(:,1)),x2-p(:,1)));	<i>% Rectangle</i>
function d=dunion(d1,d2) d=min(d1,d2);	<i>% Union</i>
function d=ddiff(d1,d2) d=max(d1,-d2);	<i>% Difference</i>
function d=dintersect(d1,d2) d=max(d1,d2);	<i>% Intersection</i>
function p=pshift(p,x0,y0) p(:,1)=p(:,1)-x0; p(:,2)=p(:,2)-y0;	<i>% Shift points</i>
function p=protate(p,phi) A=[cos(phi),-sin(phi);sin(phi),cos(phi)]; p=p*A;	<i>% Rotate points around origin</i>
function d=dmatrix(p,xx,yy,dd,varargin) d=interp2(xx,yy,dd,p(:,1),p(:,2),'*linear');	<i>% Interpolate d(x,y) in meshgrid matrix</i>
function h=hmatrix(p,xx,yy,dd,hh,varargin) h=interp2(xx,yy,hh,p(:,1),p(:,2),'*linear');	<i>% Interpolate h(x,y) in meshgrid matrix</i>
function h=huniform(p,varargin) h=ones(size(p,1),1);	<i>% Uniform h(x,y) distribution</i>

# Ejemplos:

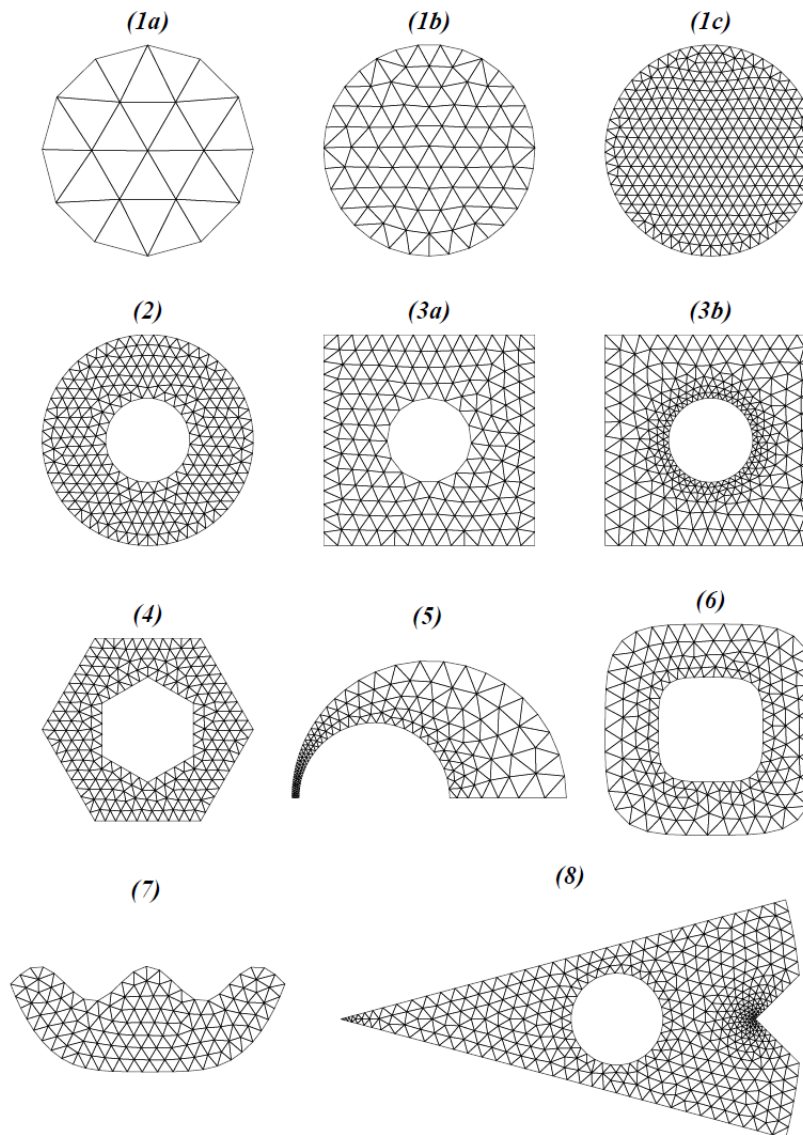


FIG. 5.1. Example meshes, numbered as in the text. The color shows the distance function  $d(x, y)$ , from blue at the boundary to red inside the region. Examples (3b), (5), (6), and (8) have varying size functions  $h(x, y)$ . Examples (6) and (7) use Newton's method (4.6) to construct the distance function.

# Ejemplo 1

(1) *Unit Circle.* We will work directly with  $d = \sqrt{x^2 + y^2} - 1$ , which can be specified as an inline function. For a uniform mesh,  $h(x, y)$  returns a vector of 1's. The circle has bounding box  $-1 \leq x \leq 1$ ,  $-1 \leq y \leq 1$ , with no fixed points. A mesh with element size approximately  $h_0 = 0.2$  is generated with two lines of code:

```
>> fd=inline('sqrt(sum(p.^2,2))-1','p');  
>> [p,t]=distmesh2d(fd,@huniform,0.2,[-1,-1;1,1],[]);
```

The plots (1a), (1b), and (1c) show the resulting meshes for  $h_0 = 0.4$ ,  $h_0 = 0.2$ , and  $h_0 = 0.1$ . Inline functions are defined without creating a separate file. The first argument is the function itself, and the remaining arguments name the parameters to the function (**help inline** brings more information). Please note the comment near the end of the paper about the relatively slow performance of inline functions.

Another possibility is to discretize  $d(x, y)$  on a Cartesian grid, and interpolate at other points using the **dmatrix** function:

```
>> [xx,yy]=meshgrid(-1.1:0.1:1.1,-1.1:0.1:1.1);    % Generate grid  
>> dd=sqrt(xx.^2+yy.^2)-1;                          % d(x,y) at grid points  
>> [p,t]=distmesh2d(@dmatrix,@huniform,0.2,[-1,-1;1,1],[],xx,yy,dd);
```

# Ejemplo 2

(2) *Unit Circle with Hole.* Removing a circle of radius 0.4 from the unit circle gives the distance function  $d(x, y) = |0.7 - \sqrt{x^2 + y^2}| - 0.3$ :

```
>> fd=inline('-0.3+abs(0.7-sqrt(sum(p.^2,2)))');  
>> [p,t]=distmesh2d(fd,@huniform,0.1,[-1,-1;1,1],[]);
```

Equivalently,  $d(x, y)$  is the distance to the difference of two circles:

```
>> fd=inline('ddiff(dcircle(p,0,0,1),dcircle(p,0,0,0.4))','p');
```

# Ejemplo 3

(3) *Square with Hole.* We can replace the outer circle with a square, keeping the circular hole. Since our distance function `drectangle` is incorrect at the corners, we fix those four nodes (or write a distance function involving square roots):

```
>> fd=inline('ddiff(drectangle(p,-1,1,-1,1),dcircle(p,0,0,0.4))','p');  
>> pfix=[-1,-1;-1,1;1,-1;1,1];  
>> [p,t]=distmesh2d(fd,@huniform,0.15,[-1,-1;1,1],pfix);
```

A non-uniform  $h(x,y)$  gives a finer resolution close to the circle (mesh (3b)):

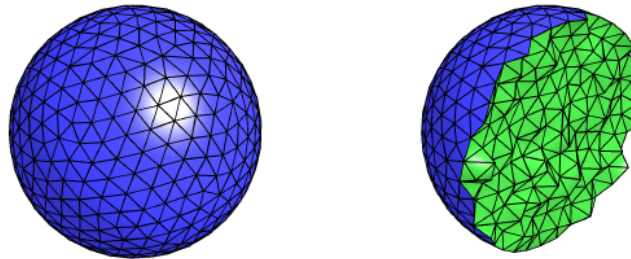
```
>> fh=inline('min(4*sqrt(sum(p.^2,2))-1,2)','p');  
>> [p,t]=distmesh2d(fd,fh,0.05,[-1,-1;1,1],pfix);
```

# Distmesh 3D. Ejemplo 9

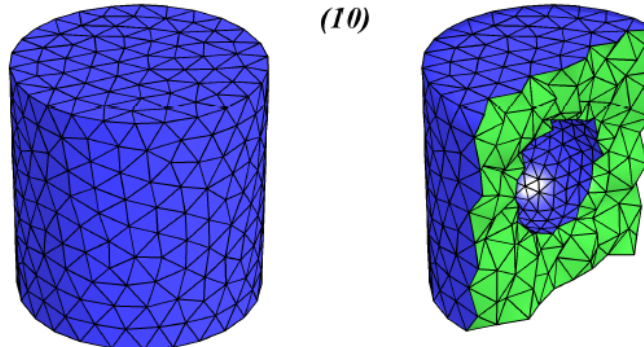
(9) *Unit Ball*. The ball in 3-D uses nearly the same code as the circle:

```
>> fd=inline('sqrt(sum(p.^2,2))-1','p');  
>> [p,t]=distmeshnd(fd,@huniform,0.15,[-1,-1,-1;1,1,1],[]);
```

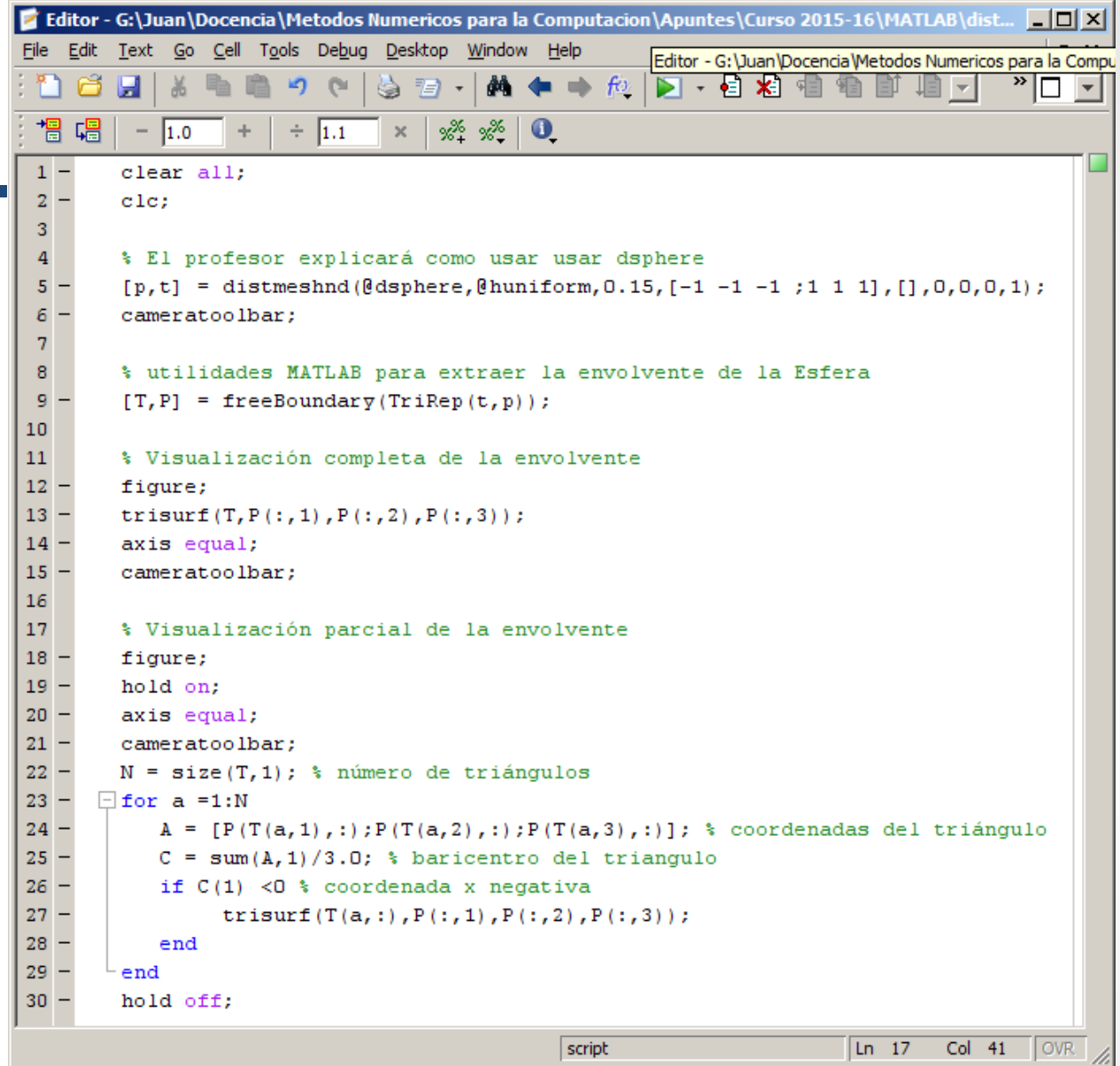
(9)



(10)



# Tarea 2.

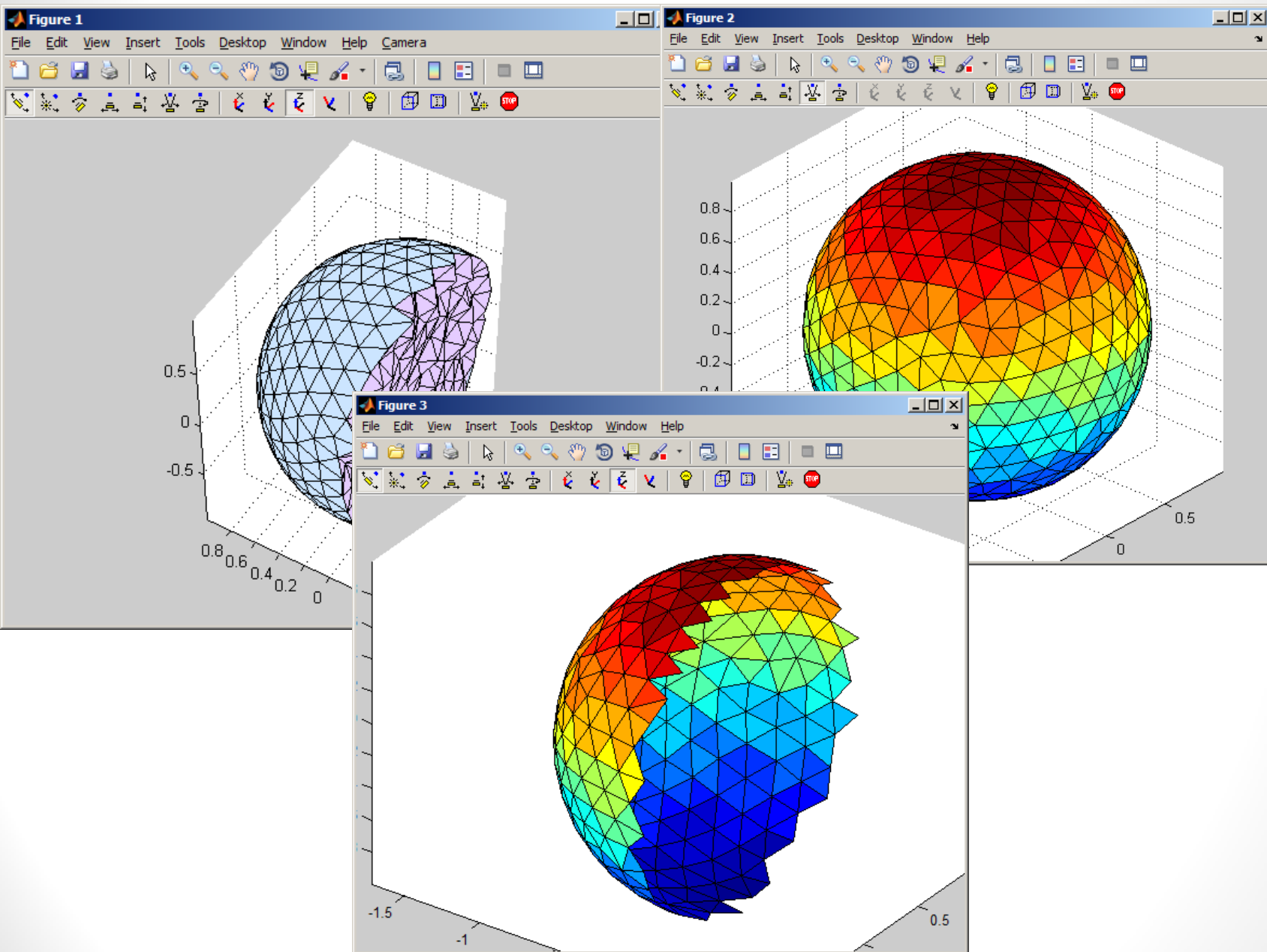


The image shows a MATLAB Editor window with a script for generating a sphere mesh and visualizing its surface. The script includes comments in Spanish explaining the steps: using `dsphere` for mesh generation, `freeBoundary` for extracting the surface, and `trisurf` for visualization. It also includes a loop to visualize individual triangles and their barycenters.

```
1 - clear all;
2 - clc;
3
4 - % El profesor explicará como usar dsphere
5 - [p,t] = distmeshnd(@dsphere,@huniform,0.15,[-1 -1 -1 ;1 1 1],[[],0,0,0,1]);
6 - cameratoolbar;
7
8 - % utilidades MATLAB para extraer la envoltente de la Esfera
9 - [T,P] = freeBoundary(TriRep(t,p));
10
11 - % Visualización completa de la envoltente
12 - figure;
13 - trisurf(T,P(:,1),P(:,2),P(:,3));
14 - axis equal;
15 - cameratoolbar;
16
17 - % Visualización parcial de la envoltente
18 - figure;
19 - hold on;
20 - axis equal;
21 - cameratoolbar;
22 - N = size(T,1); % número de triángulos
23 - for a =1:N
24 -     A = [P(T(a,1),:);P(T(a,2),:);P(T(a,3),:)]; % coordenadas del triángulo
25 -     C = sum(A,1)/3.0; % baricentro del triangulo
26 -     if C(1) <0 % coordenada x negativa
27 -         trisurf(T(a,:),P(:,1),P(:,2),P(:,3));
28 -     end
29 - end
30 - hold off;
```

- Ejecutar el programa de mallado de la esfera con `distmesh`, generar la envoltente superficial y visualizar una parte seccionada.







# Tarea 3

- Descargar el toolbox de lectura de ficheros PLY disponible en el Campus Virtual en una carpeta.
- Descomprimirlo y situar el punto de trabajo de MATLAB en el mismo.
- Copiar/Descargar diversos modelos de mallado de objetos proporcionados en el Campus Virtual y en algún repositorio. Por ejemplo teapot, diferentes resoluciones de bunny y alguno del repositorio citado en el Tema.
- Realizar las operaciones de lectura y visualización, modificando los mapas de colores.
- Capturar pantallas y elaborar una memoria sobre las actividades realizadas y del trabajo personal del alumno., incluyendo una descripción del formato PLY

# Que debe entregar el alumno?

- Cada alumno entregará en el Campus Virtual una memoria en PDF o Word en la que estará contenida una descripción del trabajo realizado, incluyendo descripción, el listado MATLAB de la actividad realizada y la captura de pantalla de las gráficas o imágenes generadas. Para autentificar las imágenes cuando sea posible el alumno incluirá su nombre en cada imagen mediante la función title().
- En principio la tarea quedará abierta para su entrega hasta cierta fecha que se indicará.
- Se puede trabajar en grupo en el Laboratorio, pero la memoria elaborada y entregado será individual.