

```
>>vx=-Y./(X.^2+Y.^2);
>>vy=X./(X.^2+Y.^2);           % calculamos vectores
>>h=quiver(X,Y,vx,vy);
>>axis square
```

Observa que cerca del (0,0) los vectores se hacen muy grandes. Para evitar que esto distorsione el dibujo hemos optado por no dibujar los vectores correspondientes a puntos muy cercanos al origen. Con este fin hemos utilizado una variable (en realidad una matriz) lógica `ind` que toma valor uno únicamente si el punto correspondiente (X,Y) está cerca del origen.

Una vez salvada esta dificultad, hemos procedido a dibujar el campo de velocidades resultante.

El comando `quiver` devuelve en realidad dos punteros, uno a las líneas y otro a la *cabeza* del vector. Sus valores opcionales son similares a los ya vistos en secciones anteriores.

Ejercicio 10.7 Siguiendo con las instrucciones desplegadas arriba, observa qué sucede si se ejecuta

```
set(h(1),'linewidth',1,'color','r','linestyle',':')
set(h(2),'color','k')
```

¿Podrías eliminar la punta de los vectores?

Ejercicio 10.8 Utilizando el comando `quiver3`, dibuja el campo de velocidades que a cada punto le asigna el vector (velocidad) dado por

$$\left(-\frac{x}{\sqrt{x^2 + y^2 + z^2}}, -\frac{y}{\sqrt{x^2 + y^2 + z^2}}, -\frac{z}{\sqrt{x^2 + y^2 + z^2}} \right)$$

(Ayuda: el comando `meshgrid` es también el apropiado para construir mallas en 3D)

10.4. Dibujos sobre dominios mallados en triángulos

Los comandos que hemos estudiado a lo largo de la sección precedente están pensados para dibujar superficies definidas esencialmente sobre una cuadrícula. Aunque en muchos casos esto es suficiente, en otras muchas ocasiones se trabaja con funciones definidas sobre conjuntos, o **dominios** en la terminología habitual, mucho más generales.

Una forma muy simple de trabajar con estos dominios es dividirlos en triángulos y construir la superficie solapando diferentes planos definidos sobre cada triángulo⁹. Los resultados que se obtienen son bastante satisfactorios, puesto que los triángulos son más flexibles que los cuadriláteros a la hora de adaptarse a los dominios.

Esta división en triángulos de un dominio se denomina **triangulación** o **mallado** del dominio. Se dice que un mallado se hace más fino si el tamaño de los triángulos disminuye.

Una triangulación es **conforme** si la intersección de dos lados cualesquiera del mallado es o bien vacía (los triángulos no se tocan), o un vértice o un lado entero. Es decir no

⁹Se utiliza el conocido resultado de que tres puntos no alineados definen un único plano. Es fácil ver también que la superficie así construida es continua

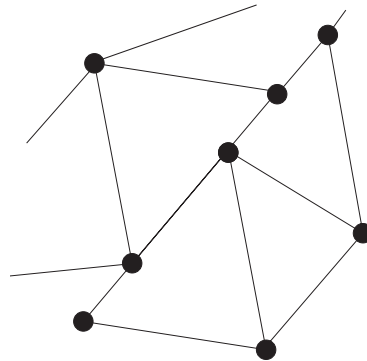


Figura 10.18: Triangulación no conforme.

se admite que un lado de un triángulo pueda ser parte de dos lados de dos triángulos diferentes (véase la Figura 10.18). Por otro lado, una familia de triangulaciones se dice **regular** si los triángulos *no se aplanan*, es decir, los ángulos de los triángulos no se hacen muy pequeños.

La siguiente cuestión es cómo almacenar la información de una triangulación. Si optáramos por guardar cada triángulo, con sus correspondientes vértices, guardaríamos información redundante. Por ejemplo, un vértice compartido por 6 triángulos sería almacenado 6 veces.

En lugar de ello se opta por una estructura más elaborada pero más económica. Se empieza almacenando las dos coordenadas de los vértices en dos vectores que denotaremos por x e y . En segundo lugar, *apuntamos* los vértices que corresponden a cada triángulo. Esto se hace mediante una matriz, que llamaremos en lo que sigue t , de tres columnas y número de filas igual al número de triángulos. Para saber los vértices que corresponden al triángulo i basta leer la correspondiente fila de t y sus coordenadas serán

$$[x(t(i,1)),y(t(i,1))], \quad [x(t(i,2)),y(t(i,2))], \quad [x(t(i,3)),y(t(i,3))]$$

A modo de ejemplo, el mallado expuesto en la Figura 10.19 se guarda en las siguientes variables

- Triangulos

8	3	16
7	2	15
9	5	21
11	6	20
10	1	14
12	4	17
16	9	21
4	8	17
1	7	14
15	11	20

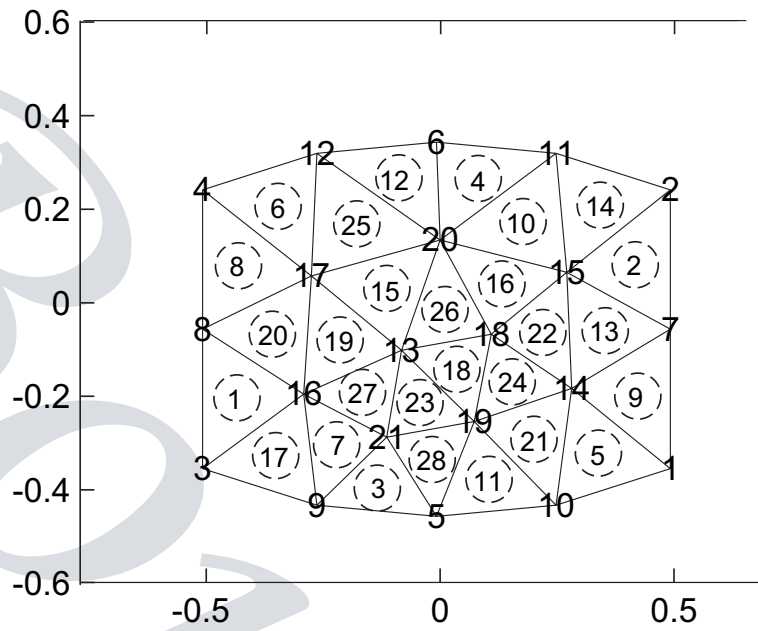


Figura 10.19: Ejemplo de triangulación. En el gráfico la numeración de los triángulos se rodea con un círculo.

5	10	19
6	12	20
14	7	15
2	11	15
17	13	20
18	15	20
3	9	16
18	13	19
16	13	17
8	16	17
10	14	19
14	15	18
19	13	21
14	18	19
12	17	20
13	18	20
13	16	21
5	19	21

■ coordenadas:

x y

0.5033	-0.3584
0.5033	0.2378
-0.4967	-0.3584
-0.4967	0.2378
0.0033	-0.4603
0.0033	0.3397
0.5033	-0.0603
-0.4967	-0.0603
-0.2523	-0.4363
0.2590	-0.4363
0.2590	0.3157
-0.2523	0.3157
-0.0716	-0.1050
0.2924	-0.1866
0.2820	0.0614
-0.2806	-0.1991
-0.2641	0.0537
0.1204	-0.0707
0.0838	-0.2577
0.0113	0.1306
-0.1031	-0.2912

La información anterior es suficiente para construir la malla, la triangulación del dominio. Si además se desea construir una superficie definida sobre ese dominio, basta añadir un vector adicional z de forma que $z(j)$ sea el valor de la función en el nodo j .

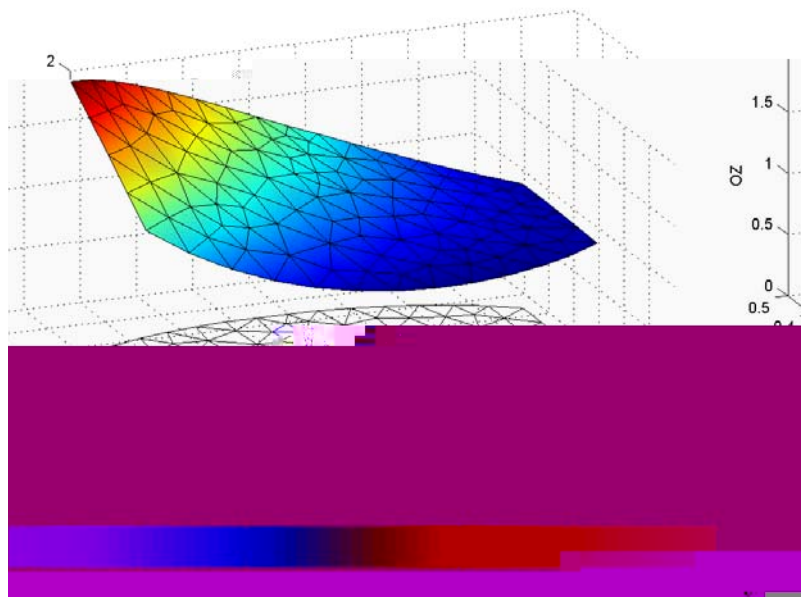


Figura 10.20: Un dominio mallado en triángulos.

Ésta es una forma ya estándar de definir y trabajar con una triangulación que también sigue Matlab¹⁰ con los comandos `trimesh` y `trisurf`.

El primero despliega la malla triangular especificada por `t`, (la matriz conteniendo los triángulos), `x` e `y` (que dan las coordenadas de los vértices),

```
trimesh(t,x,y)
```

Se puede especificar la coordenada z de los vértices (la altura),

```
trimesh(t,x,y,z)
```

con lo que se dibuja la malla 3D correspondiente.

El comando `trisurf` es similar, pero *colorea* las caras. Hablando con propiedad, estos comandos al igual que `mesh` y `surf`, representan superficies. Las opciones para manipular el aspecto final de la superficie son iguales que las de `surf`, incluyendo las ya vistas `facecolor`, `facealpha`, `meshalpha`, `edgecolor`,....

Nota. Un tema nada trivial es la construcción de un mallado sobre un dominio (poligonal) dado. Existen multitud de algoritmos que tratan este problema. En principio se plantea la construcción de una malla gruesa, con pocos triángulos y de área considerable, con los triángulos lo más regulares posibles (sin deformar, *alargar*, en demasía los triángulos).

Posteriormente, se trata de *refinar* la malla, es decir, dividir los triángulos en triángulos más pequeños hasta que se alcance una precisión adecuada.

Esta idea se esconde detrás de aplicaciones como la interpolación (aproximación de una función) y especialmente el método de elementos finitos, probablemente del método¹¹ más utilizado en la resolución de problemas de contorno para ecuaciones en derivadas parciales.

Si se desea información de cómo se puede inicializar un malla en Matlab, así como sobre el algoritmo utilizado se puede consultar el comando `initmesh` (incluido el tema de triangulaciones de Delaunay). Para el refinamiento puedes consultar `refinemesh`.

Existe otra posibilidad más visual, y por tanto más amigable para empezar a trabajar. Teclea

```
>> pdetool
```

Se cargará un entorno gráfico para la resolución de ecuaciones en derivadas parciales. No entraremos en este tema por ser demasiado técnico. En lugar de ello, nos centraremos en la definición de mallados. Una vez dibujado el dominio, y mallado, se puede exportar la malla a la ventana de comandos en las variables `t` y `p`. La primera contiene en las tres primeras filas los triángulos que componen la triangulación mientras que las coordenadas de los vértices están guardadas en las dos filas de `p`. □

¹⁰La toolbox `pdetool` dedicada a la resolución de ecuaciones en derivadas parciales sigue una variante algo más complicada que la expuesta arriba.

¹¹Propiamente hablando es una familia de métodos.