ECUACIÓN DE ONDA EN 2D

MÉTODOS NUMÉRICOS

AUTORES:

DÍAZ FALCÓN, ROBERTO

HERNÁNDEZ GONZÁLEZ, ÓSCAR DAVID

MARTÍN PÉREZ, ALBERTO

ÍNDICE

CONCEPTO DE ONDA

Una **onda** es una perturbación de alguna propiedad de un medio, que se propaga a través del espacio transportando energía. El medio perturbado puede ser de naturaleza diversa como aire, agua, un trozo de metal o el vacío, y las propiedades que sufren la perturbación pueden ser también variadas, por ejemplo, densidad, presión, campo eléctrico o campo magnético.

La propiedad del medio en la que se observa la particularidad se expresa como una función tanto de la posición como del tiempo $\psi(\vec{r},t)$. Matemáticamente se dice que dicha función es una onda si verifica la ecuación de ondas:

$$\nabla^2\psi(\vec{r},t)=\frac{1}{v^2}\cdot\frac{\partial^2\psi}{\partial t^2}(\vec{r},t)$$

donde *v* es la velocidad de propagación de la onda. Por ejemplo, ciertas perturbaciones de la presión de un medio, llamadas sonido, verifican la ecuación anterior, aunque algunas ecuaciones no lineales también tienen soluciones ondulatorias.

En la siguiente figura podemos observar la propagación de ondas en el agua.

CLASIFICACIÓN DE LAS ONDAS

Las ondas se clasifican atendiendo a diferentes aspectos:

1. En función del medio en el que se propagan:

    - ***Ondas mecánicas**:* las ondas mecánicas necesitan un medio elástico (sólido, líquido o gaseoso) para propagarse. Las partículas del medio oscilan alrededor de un punto fijo, por lo que no existe transporte neto de materia a través del

medio. Como en el caso de una alfombra o un látigo cuyo extremo se sacude, la alfombra no se desplaza, sin embargo una onda se propaga a través de ella. Dentro de las ondas mecánicas tenemos:

1. Ondas elásticas
2. Ondas sonoras
3. Ondas de gravedad

- ***Ondas electromagnéticas:*** las ondas electromagnéticas se propagan por el espacio sin necesidad de un medio pudiendo, por tanto, propagarse en el vacío. Esto es debido a que las ondas electromagnéticas son producidas por las oscilaciones de un campo eléctrico en relación con un campo magnético asociado.

- ***Ondas gravitacionales:*** las ondas gravitacionales son perturbaciones que alteran la geometría misma del espacio-tiempo y aunque es común representarlas viajando en el vacío, técnicamente no puede afirmarse que se desplacen por ningún espacio sino que en sí mismas son alteraciones del espacio-tiempo.

2. En función de su propagación o frente de onda:

- ***Ondas unidimensionales:*** las ondas unidimensionales son aquellas que se propagan a lo largo de una sola dirección del espacio, como las ondas en los muelles o en las cuerdas. Si la onda se propaga en una dirección única, sus frentes de onda son planos y paralelos.
- ***Ondas bidimensionales o superficiales:*** son ondas que se propagan en dos direcciones. Pueden propagarse, en cualquiera de las direcciones de una superficie, por ello, se denominan también ondas superficiales. Un ejemplo son las ondas que se producen en la superficie de un lago cuando se deja caer una piedra sobre él.
- ***Ondas tridimensionales o esféricas:*** son ondas que se propagan en tres direcciones. Las ondas tridimensionales se conocen también como ondas esféricas, porque sus frentes de ondas son esferas concéntricas que salen de la fuente de perturbación expandiéndose en todas direcciones. El sonido es una onda tridimensional. Son ondas tridimensionales las ondas sonoras (mecánicas) y las ondas electromagnéticas.

3. En función de la dirección de la perturbación:

    - ***Ondas longitudinales:*** el movimiento de las partículas que transportan la onda es paralelo a la dirección de propagación de la onda. Por ejemplo, un muelle que se comprime da lugar a una onda longitudinal.
    - ***Ondas transversales:*** las partículas se mueven perpendicularmente a la dirección de propagación de la onda.

4. En función de su periodicidad:

    - ***Ondas periódicas:*** la perturbación local que las origina se produce en ciclos repetitivos. Por ejemplo, una onda senoidal.
    - ***Ondas no periódicas:*** la perturbación que las origina se da aisladamente o, en el caso de que se repita, las perturbaciones sucesivas tienen características diferentes. Las ondas aisladas se denominan también pulsos.

FENÓMENOS ONDULATORIOS

Son los efectos y propiedades exhibidas por las entidades físicas que se propagan en forma de onda:

- **Difracción** - Ocurre cuando una onda al topar con el borde de un obstáculo deja de ir en línea recta para rodearlo.
- **Efecto Doppler** - Efecto debido al movimiento relativo entre la fuente emisora de las ondas y el receptor de las mismas.
- **Interferencia** - Ocurre cuando dos ondas se combinan al encontrase en el mismo punto del espacio.
- **Reflexión** - Ocurre cuando una onda, al encontrarse con un nuevo medio que no puede atravesar, cambia de dirección.
- **Refracción** - Ocurre cuando una onda cambia de dirección al entrar en un nuevo medio en el que viaja a distinta velocidad.
- **Onda de choque** - Ocurre cuando varias ondas que viajan en un medio se superponen formando un cono.

ECUACIÓN DE ONDA

La **ecuación de onda** es una importante ecuación en derivadas parciales que describe una variedad de ondas, como las ondas sonoras, las ondas de luz y las ondas de agua. Es

importante en varios campos como la acústica, el electromagnetismo y la dinámica de fluidos.

Introducción

La ecuación de onda es el ejemplo prototipo de una ecuación diferencial parcial hiperbólica. En su forma más elemental, la ecuación de onda hace referencia a un escalar *u* que satisface:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \cdot \nabla^2 u$$

donde *c* es una constante equivalente a la velocidad de propagación de la onda y ∇^2 es el laplaciano, como ya describimos antes.

Por ejemplo, para una onda sonora en el aire a 20 ºC, la velocidad de propagación es aproximadamente de 343 m/s, mientras que para una cuerda vibrante, la velocidad puede variar mucho dependiendo de la densidad lineal de la cuerda y su tensión. Para un resorte de espiral puede ser tan lento como un metro por segundo.

Un modelo más realista para simular el comportamiento de las ondas considera que la velocidad de propagación de la onda puede variar con su frecuencia, este fenómeno es conocido como dispersión. En este caso, *c* debe remplazarse por la velocidad de fase:

$$v_p = \frac{w}{k}$$

Otra corrección común a esta ecuación es considerar que la velocidad puede también depender de la amplitud de la onda, lo que nos lleva a una ecuación de onda no-lineal:

$$\frac{\partial^2 u}{\partial t^2} = c^2(u) \cdot \nabla^2 u$$

También habría que considerar que una onda puede ser transmitida en un portador en movimiento (por ejemplo la propagación del sonido en el flujo de un gas). En tal caso el escalar *u* contendrá un Número Mach (el cual es positivo para la onda que se mueva a los largo del flujo y negativo para la onda reflejada).

La ecuación de onda elástica en tres dimensiones describe la propagación de onda en un medio elástico homogéneo isotrópico. La mayoría de los materiales sólidos son elásticos, por lo que esa ecuación describe tales fenómenos como ondas sísmicas en la Tierra y ondas de ultrasonido usadas para determinar defectos en los materiales. Mientras sea lineal, esta ecuación tiene una forma más compleja que las ecuaciones dadas arriba, porque debe tomar en cuenta los movimientos longitudinal y transversal:

$$\rho \cdot \frac{\partial^2 u}{\partial t^2} = f + (\lambda + 2\mu)\nabla(\nabla \cdot u) - \mu\nabla \times (\nabla \times u)$$

Donde:

- λ y μ son los módulos supuestos del Lamé que describen las propiedades elásticas del medio.
- ρ es densidad,
- f es la función de entrada (fuerza de impulso),
- u es el desplazamiento.

Hay que notar que en esta ecuación, la fuerza y el desplazamiento son cantidades vectoriales. Esta ecuación es conocida a veces coma la ecuación de onda vectorial.

Hay variaciones de la ecuación de onda que también pueden ser encontradas en mecánica cuántica y relatividad general.

DESARROLLO MATEMÁTICO DE LA ECUACIÓN DE ONDA EN DOS DIMENSIONES

Si la ecuación de onda en su forma más elemental, que citamos anteriormente, la desarrollamos para el caso de dos dimensiones tenemos:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \cdot \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right)$$

Supongamos un sistema de ejes cartesianos (0, x, y, z). Consideramos contenido en el plano xy un dominio rectangular Ω definido sobre un material elástico de dimensiones $l_x \times l_y$.

Desarrollando esta ecuación para cada instante de tiempo obtendremos un sistema de ecuaciones, cuya resolución requerirá ecuaciones adicionales. Al tratarse de una ecuación en derivadas parciales de segundo grado, requerirá de dos ecuaciones adicionales: la condición inicial (para $t = 0$) y la derivada de la condición inicial. Asimismo, será necesario el planteamiento de las condiciones de contorno, que supondremos nulas. Así, el planeamiento completo del problema queda:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \cdot \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \begin{cases} x_{\min} < x < x_{\max} \\ y_{\min} < y < y_{\max} \\ 0 < t \end{cases}$$

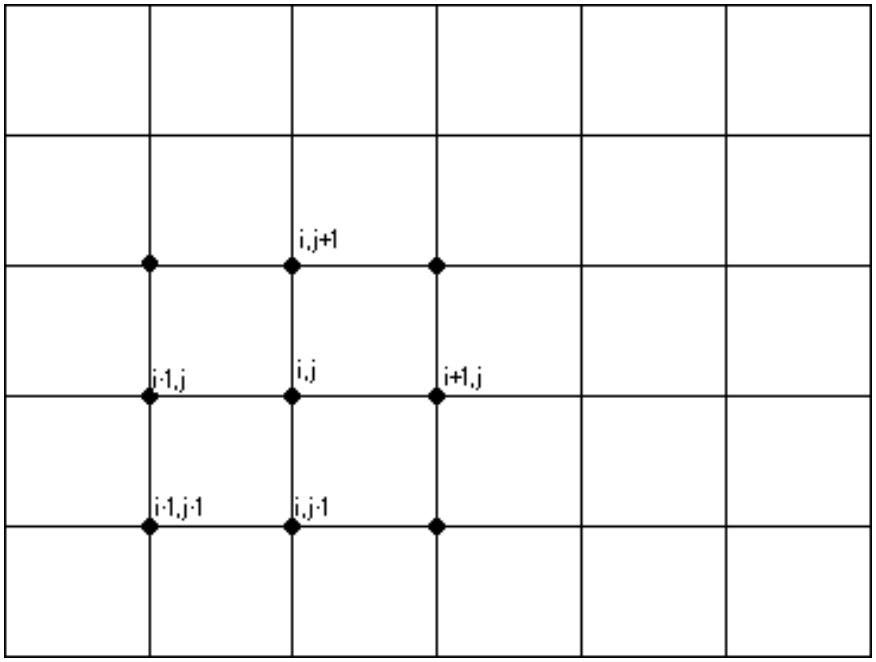
- *condiciones de contorno:*

$$u(x_{\min}, y_{\min}, t) = u(x_{\min}, y_{\max}, t) = u(x_{\max}, y_{\min}, t) = u(x_{\max}, y_{\max}, t) = 0$$
$$\{0 < t\}$$

- *condiciones iniciales*

$$\begin{aligned} u(x, y, 0) &= f(x, y) \\ u'(x, y, 0) &= g(x, y) \end{aligned} \qquad \begin{cases} x_{\min} < x < x_{\max} \\ y_{\min} < y < y_{\max} \end{cases}$$

Si ahora, desarrollamos cada término de la ecuación de onda para una abscisa i, una ordenada j y en el tiempo k, en función de los puntos circundantes, tal y como se muestra en la figura siguiente, obtendríamos las siguientes relaciones:



$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{i,j,k} = \frac{u_{i-1j}^k - 2\cdot u_{ij}^k + u_{i+1j}^k}{\Delta x^2} + O(\Delta x^2)$$

$$\left.\frac{\partial^2 u}{\partial y^2}\right|_{i,j,k} = \frac{u_{ij-1}^k - 2\cdot u_{ij}^k + u_{ij+1}^k}{\Delta y^2} + O(\Delta y^2)$$

$$\left.\frac{\partial^2 u}{\partial t^2}\right|_{i,j,k} = \frac{u_{ij}^{k-1} - 2\cdot u_{ij}^k + u_{ij}^{k+1}}{\Delta t^2} + O(\Delta t^2)$$

Sustituyendo estas relaciones en la ecuación de onda tenemos:

$$\frac{u_{ij}^{k-1} - 2\cdot u_{ij}^k + u_{ij}^{k+1}}{\Delta t^2} = c^2 \cdot \left(\frac{u_{i-1j}^k - 2\cdot u_{ij}^k + u_{i+1j}^k}{\Delta x^2} + \frac{u_{ij-1}^k - 2\cdot u_{ij}^k + u_{ij+1}^k}{\Delta y^2} \right)$$

Despejando de la ecuación anterior el término u_{ij}^{k+1} tenemos que:

$$u_{ij}^{k+1} = c^2 \cdot \left(\frac{u_{i-1j}^k - 2\cdot u_{ij}^k + u_{i+1j}^k}{\Delta x^2} + \frac{u_{ij-1}^k - 2\cdot u_{ij}^k + u_{ij+1}^k}{\Delta y^2} \right) \cdot \Delta t^2 + 2\cdot u_{ij}^k - u_{ij}^{k-1}$$

En esta ecuación podemos observar que el valor de la función en el instante $k+1$ depende del valor de la función los instantes k y $k-1$.

A la hora de construir los algoritmos hemos considerado que el instante $k+1$es coincidente con el tiempo t, con lo que nos quedaría:

$$k+1 = t \Rightarrow k = t-1 \Rightarrow k-1 = t-2$$

Así, por ejemplo, para $k=1$la ecuación quedaría:

$$u_{ij}^2 = c^2 \cdot \left(\frac{u_{i-1j}^1 - 2\cdot u_{ij}^1 + u_{i+1j}^1}{\Delta x^2} + \frac{u_{ij-1}^1 - 2\cdot u_{ij}^1 + u_{ij+1}^1}{\Delta y^2} \right) \cdot \Delta t^2 + 2\cdot u_{ij}^1 - u_{ij}^0$$

Por otra parte, si definimos una función $g\ (x_i, y_j)$ como la derivada de la función u entre dos instantes de tiempo sucesivos:

$$\frac{u_{ij}^{k} - u_{ij}^{k-1}}{\Delta t} = g(x_i, y_j)$$

Y despejando el u_{ij}^{k} de la ecuación anterior obtenemos:

$$u_{ij}^{k} = u_{ij}^{k-1} + g(x_i, y_j) \cdot \Delta t$$

Esta ecuación nos permitirá obtener la matriz u_{ij}^{1} a partir de u_{ij}^{0}, la cual es conocida. Con estos dos valores estaremos en disposición de obtener u_{ij}^{2}, tal y como hemos indicado en la última ecuación de la página anterior.

RESOLUCIÓN DEL TRABAJO

Función ecuacion_ondas_2d

[U,X,Y,T] = ECUACION_ONDAS_2D(xmin,xmax,dx,ymin,ymax,dy,tmax,dt,c,g,U0)

Como ya se nos dice en la propia subrutina, ésta lo que hace es calcular la ecuación de ondas 2D U(x,y,t) para los límites x,y especificados con sus incrementos desde t = 0 hasta tmax con pasos de dt.

En la función arriba mostrada, U0 es una matriz acorde a las dimensiones de los ejes x,y que describe el valor de U para t_0 mientras que g es una función g(x,y) que puede definirse con INLINE y que describe dU/dt para U_1. g se utiliza para calcular U_1 a partir de U_0.

Para este fin, definimos la función objetivo en matlab:

```
function [U,X,Y,T] =
ecuacion_ondas_2d(xmin,xmax,dx,ymin,ymax,dy,tmax,dt,c,g,U0)
```

Nos aseguramos de que se cumplen las precondiciones, tales como que la menor de las x sea, evidentemente, menor que el valor máximo que ésta adquiere, etc. Utilizamos ahora la función assert (la cual se explicará a continuación), con la que conseguimos que en caso de que no se cumplan estas precondiciones, se nos muestre un mensaje que indica el error:

```
assert(xmin<xmax,'xmin tiene que ser menor que xmax');
assert(ymin<ymax,'ymin tiene que ser menor que ymax');
assert(dx>0,'dx tiene que ser positivo');
assert(dy>0,'dy tiene que ser positivo');
assert(dt>0,'dt tiene que ser positivo');
assert(tmax>0,'tmax tiene que ser positivo');
```

Ahora, lo que hacemos es crear tres vectores (X, Y, T), que parten del valor inicial (determinado por nosotros como valor mínimo o impuestos, como el tiempo) hasta el valor final (máximo), con un intervalo definido también por nosotros. De esta forma se crea un vector que no requiere que se indique su longitud con anterioridad:

```
X = xmin:dx:xmax;
Y = ymin:dy:ymax;
T = 0:dt:tmax;
```

No obstante, sí que nos interesa comprobar que los vectores van a tener más de un elemento, también con la función assert. Evidentemente, en el caso de los vectores, su longitud debe ser mayor de 1 (si no sería un punto) y en el de la matriz, sus dimensiones deben ser concordantes con la longitud de los vectores que hemos asociado a ella:

```
assert(length(X)>1,'intervalo en x muy pequeño o dx muy grande');
assert(length(Y)>1,'intervalo en y muy pequeño o dy muy grande');
assert(length(T)>1,'tmax muy pequeño o dt muy grande');
assert(size(U0,1) == length(X),'el numero de filas de U0 tiene que
                    ser igual a la dimensión de X');
assert(size(U0,2) == length(Y),'el numero de columnas de U0 tiene que
                    ser igual a la dimensión de Y');
```

Una vez hecho esto, evaluamos la función g mediante la función mostrada a continuación, la cual se ya se explicará y calculamos la matriz en el instante posterior al inicial (U1). Como puede verse, trasponemos la matriz G, ya que en nuestra matriz una fila se corresponde con la variable X y en la representación debe ser con la columna (lo mismo ocurre con la variable Y):

```
G = evalua_funcion(g,X,Y);
G = G';
U1 = U0 + G*dt;
```

Preparamos la función con los valores iniciales y los valores de los bordes, haciéndolos cero (condición inicial). Hay que remarcar el hecho de que hemos puesto que las dimensiones de la matriz sean las longitudes de los respectivos vectores (X, Y, T), más 2 en los casos de X e Y, debido a que partiremos de los valores en los bordes de la malla:

```
U = zeros(length(X)+2,length(Y)+2,length(T));
```

Ahora le asignaremos a la matriz los valores correspondientes al instante inicial y al inmediatamente posterior:

```
U(2:length(X)+1,2:length(Y)+1,1) = U0;
U(2:length(X)+1,2:length(Y)+1,2) = U1;
```

Con estos datos conocidos, pasamos a calcular los valores de U para t2 en adelante:

```
for t  = 3:length(T)
```

(Aquí lo que hacemos es calcular cada Uij, partiendo de las ecuaciones de derivadas parciales expuestas en la teoría, para ese t. Si en las ecuaciones obtenidas en la teoría asumimos k+1=t):

```
    for i = 2:length(X)+1
        for j = 2:length(Y)+1
            parcialx = (U(i-1,j,t-1) - 2*U(i,j,t-1) +U(i+1,j,t-
                       1))/(dx^2);
            parcialy = (U(i,j-1,t-1) - 2*U(i,j,t-1) +U(i,j+1,t-
                       1))/(dy^2);
            U(i,j,t) = c^2*(parcialx+parcialy)*(dt^2) + 2*U(i,j,t-1)
                       - U(i,j,t-2);
        end
    end
end
```

Finalmente, quitamos el borde auxiliar de la superficie (el que añadimos en la matriz al poner que sus dimensiones eran las de los vectores X e Y más 2:

```
    U = U(2:length(X)+1,2:length(Y)+1,:);
return;
```

Función assert

La función assert lo que hace es que en función de un valor introducido en la misma (a), que no tiene que ser un único valor, sino que puede ser una expresión sencilla (nosotros usaremos una comparación del tipo < o >, por ejemplo). Así la función necesita de dos variables:

```
function assert(a,mensaje)
```

Ahora comprobamos que el número de argumentos introducido no sea menor de 2. En caso de que sí lo sea (lo cual nos indica que falta la variable mensaje), se le atribuye uno general:

```
if nargin < 2
    mensaje = sprintf(' condición %s falsa',inputname(1));
end
```

Ahora le otorgamos a mensaje una cadena de caracteres que nos indique en caso de error que éste tuvo lugar en esta función:

```
mensaje = sprintf('assert: %s',mensaje);
```

Finalmente, en caso de que la condición a analizar no se cumpla, se verá en pantalla el mensaje asociado a ese error, además de que también se parará todo el proceso:

```
if not(a)
   fprintf('\n\n');
   fprintf(mensaje);
   fprintf('\n\n');
   error(mensaje);
end

return;
```

Función evalua_funcion

Esta función lo que hace es devolver en una matriz los valores de la función objetivo F para los valores correspondientes a los vectores (X,Y). Así, nos queda el siguiente desarrollo:

```
function Z = evalua_funcion(F,X,Y)
```

Inicializamos a cero la matriz Z, cuyas dimensiones serán las longitudes de los vectores X e Y dados. Luego, lo que hacemos es ir introduciendo los valores en la matriz:

```
    Z = zeros(length(X),length(Y));
    for i = 1:length(X)
        for j = 1:length(Y)
            Z(i,j) = F(X(i),Y(j));
        end
    end
```

(Como puede verse, trasponemos la matriz por el motivo explicado anteriormente)

```
    Z = Z';
return;
```

Función video_ondas

Con esta función lo que haremos será crear el video que luego se visualizará en matlab.

De nuevo, miraremos el número de argumentos que tiene la función. En caso de ser menor de 5, tomaremos el paso como 1 (valor que puede cambiarse a cualquier otro, en función de la calidad que busquemos en el vídeo). Si además, el número fuera menor de 6, tomaremos como color el *interp* (valor que hemos tomado como estándar, en caso de que no se indique otro):

```
function video_ondas_2d(U,X,Y,T,paso,color)
    if nargin < 5
        paso = 1;
    end
    if nargin < 6
        color = 'interp';
    end
```

Asimismo, se define el tipo de sombreado que se puede usar. Hemos puesto el *phong* como estándar, pero también podrían usarse otros formatos, como el *flat* o el *Gouraud*:

```
    sombreado = 'phong';
```

Pasamos a definir la variable *zratio*, la cual usaremos más adelante. Esta variable la usaremos para mantener una relación fija entre los ejes que facilite la visualización de las representaciones gráficas, de forma que los ejes X e Y de la gráfica sean *zratio*veces mayores que el eje relacionado con la variable Z (la cual hará referencia al valor contenido en cada una de las celdas de la matriz U; es decir, que Z en el instante t=1, para X=2 e Y=4 valdrá: U(2,4,1)=9=Z). En nuestro caso hemos tomado un valor de 5:

```
    zratio = 5;
```

Calculamos los límites para los ejes de la gráfica a representar. Esto lo hacemos para conseguir que todas las gráficas mantengan la misma proporción; ya que, si por ejemplo tenemos un instante en el que el máximo es 10 y en el inmediatamente siguiente es 20, en el vídeo final se apreciaría el cambio de escala, con lo que no se obtendría una buena visualización del proceso.

Como no sabemos cuál es el máximo valor de Z que se alcanza, buscaremos en toda la matriz el valor mayor y el menor. Inicialmente hemos puesto, como puede verse, un valor máximo de Z muy pequeño y un valor mínimo muy grande:

```
    maxz = -100000e+060;
    minz = 10000e+060;
```

Ahora lo que haremos es buscar cuál va a ser el valor máximo y mínimo de z que habrá entre todos los almacenados en la matriz de trabajo U. Para ello, iremos desde la primera hasta la última (siendo ésta la longitud del vector T, asociado al tiempo):

```
    for k = 1:length(T)
```

Seleccionaremos el máximo de los máximos de la matriz para ese instante. Si este *máximo* obtenido es mayor que el valor preestablecido *maxz* haremos que sea el nuevo *maxz*, y así sucesivamente para todos los instantes:

```
        maximo = max(max(U(:,:,k)));
```

```
        if maximo > maxz
            maxz = maximo;
        end
```

Lo mismo haremos para el mínimo. Buscaremos el menor de los mínimos de la matriz y repetiremos el proceso para cada uno de los instantes, seleccionando al más pequeño de todos:

```
        minimo = min(min(U(:,:,k)));
        if minimo < minz
            minz = minimo;
        end
    end
```

Los valores iniciales máximos y mínimos ya los damos preestablecidos, pues vendrán dados por la longitud de los vectores X e Y (que a su vez nos indican las dimensiones de la matriz U):

```
    minx = X(1);
    maxx = X(length(X));
    miny = Y(1);
    maxy = Y(length(Y));
```

Lo primero que haremos será representar la gráfica para T=0. Para hacerlo utilizaremos la función *surf*. Representaremos en función de las variables X, Y y la que hemos denominado Z (Uij). No obstante, agregaremos varios parámetros más, como son el tipo de color que hemos definido, el sombreado, etc.:

```
        surf(X,Y,U(:,:,1)','FaceColor',color,...
            'EdgeColor','none',...
            'FaceLighting',sombreado);
```

Ahora definiremos las propiedades de la gráfica, mediante la función *set*. Entre ellas, escalaremos los ejes tanto de X, como de Y, como de Z, mediante la propiedad de modo *DataAspectRatio*. Con esto conseguimos hacer que en la representación, los ejes representen los intervalos entre los cuales tienen valores asociados. Por ejemplo, en el caso del eje Z, representaremos un eje con una longitud igual a la diferencia del valor máximo que alcanza y del mínimo (además, como ya se dijo anteriormente, multiplicaremos por 5 esta diferencia, de forma que al producirse la representación los tres ejes queden escalados de la forma (X,Y,Z)=(5,5,1)):

```
        set(gca,'DataAspectRatio',[(maxx-minx) (maxy-miny) zratio*(maxz-
                                   minz)]);
```

Para que se llevara a cabo una correcta representación, se definieron varios parámetros, como son: el ángulo del punto de vista respecto de los ejes, la procedencia de la luz, los límites de los ejes, los títulos de cada uno de los mismos, así como el título de la gráfica:

```
        view(-50,30);
        camlight left;
        axis([minx maxx miny maxy minz maxz]);
        xlabel('x');
        ylabel('y');
        zlabel('z');
        title('Ecuacion de ondas: t = 0');
```

Añadimos el siguiente mensaje que se observa una vez se ejecuta la subrutina, de forma que espera a la confirmación por parte del usuario para comenzar el vídeo:

```
    input('presiona tecla para ver video');
```

Debemos determinar el número de *frames* que se van a mostrar en el vídeo, para ello hacemos que vaya dibujando la gráfica para cada instante desde el T=1 hasta el T final de la misma forma que ya se explicó para el T=0, con un incremento de tiempo igual al paso definido anteriormente. A cada una de estas imágenes le pondremos el título correspondiente al instante al que se refiere y las iremos almacenando en un vector M:

```
    i =1;
    for k = 1:paso:length(T)
        surf(X,Y,U(:,:,k)','FaceColor',color,...
            'EdgeColor','none',...
            'FaceLighting',sombreado);
        set(gca,'DataAspectRatio',[(maxx-minx) (maxy-miny) zratio*(maxz-
                                    minz)]);
        view(-50,30);
        camlight left;
        axis([minx maxx miny maxy minz maxz]);
        xlabel('x');
        ylabel('y');
        zlabel('z');
        title(sprintf('Ecuacion de ondas: t = %3.2f',T(k)));
        M(i) = getframe;
        i = i+1;
    end
```

Finalmente, lo que haremos será generar la película a partir del vector M, reproduciéndola una sola vez:

```
    movie(M,1);
return;
```

Función ecuacion_calor_2d

Esta función es similar a la ecuacion_ondas_2d explicada anteriormente. Por este motivo sólo haremos pequeños comentarios importantes, haciendo hincapié en la diferencia existe entre ambas funciones.

[U,X,Y,T] = ECUACION_CALOR_2D(xmin,xmax,dx,ymin,ymax,dy,tmax,dt,alfa,U0)

Esta función calcula la ecuación del calor 2D U(x,y,t) para los límites x,y especificados con sus incrementos desde t = 0 hasta tmax con pasos de dt. U0 es una matriz acorde a las dimensiones de los ejes x,y que describe el valor de U para t_0.

```
function [U,X,Y,T] =
ecuacion_calor_2d(xmin,xmax,dx,ymin,ymax,dy,tmax,dt,alfa,U0)
```

Nos aseguramos de que se cumplen las precondiciones:

```
    assert(xmin<xmax,'xmin tiene que ser menor que xmax');
    assert(ymin<ymax,'ymin tiene que ser menor que ymax');
    assert(dx>0,'dx tiene que ser positivo');
    assert(dy>0,'dy tiene que ser positivo');
```

```
    assert(dt>0,'dt tiene que ser positivo');
    assert(tmax>0,'tmax tiene que ser positivo');

    X = xmin:dx:xmax;
    Y = ymin:dy:ymax;
    T = 0:dt:tmax;
```

Comprobamos dimensiones de vectores y matrices:

```
    assert(length(X)>1,'intervalo en x muy pequeño o dx muy grande');
    assert(length(Y)>1,'intervalo en y muy pequeño o dy muy grande');
    assert(length(T)>1,'tmax muy pequeño o dt muy grande');
    assert(size(U0,1) == length(X),'el numero de filas de U0 tiene que
                        ser igual a la dimensión de X');
    assert(size(U0,2) == length(Y),'el numero de columnas de U0 tiene
                        que ser igual a la dimensión de Y');
```

Preparamos la función con los valores iniciales y haciendo los valores de los bordes a cero:

```
    U = zeros(length(X)+2,length(Y)+2,length(T));
    U(2:length(X)+1,2:length(Y)+1,1) = U0;
```

Calculamos los valores de U para t2 en adelante:

```
    for t  = 2:length(T)
```

(Calculamos cada Uij para ese t)

```
        for i = 2:length(X)+1
            for j = 2:length(Y)+1
                parcialx = (U(i-1,j,t-1) - 2*U(i,j,t-1) +U(i+1,j,t-
                            1))/(dx^2);
                parcialy = (U(i,j-1,t-1) - 2*U(i,j,t-1) +U(i,j+1,t-
                            1))/(dy^2);
```

Y he aquí la diferencia entre ambas funciones. En este caso, la ecuación que se utiliza para determinar los valores de cada una de las celdas de la matriz es ligeramente diferente, tal como se podrá comprobar en la teoría:

```
                U(i,j,t) = alfa*(parcialx+parcialy)*dt + U(i,j,t-1);
            end
        end
    end
```

Finalmente, quitamos el borde auxiliar de la superficie:

```
    U = U(2:length(X)+1,2:length(Y)+1,:);
return;
```

Ecuación de ondas: Ejemplo 1

En este primer ejemplo, veremos el movimiento que se genera cuando tiramos una piedra a un contenedor con agua en calma total.

```
function [U_,X_,Y_,T_] = ejemplo1
    fprintf('Ejemplo 1 de ecuación de ondas\n');
    fprintf('Simula tirar una piedra en el agua\n');
```

Para ello, definiremos los valores máximos y mínimos de todas las variables, así como sus incrementos. Como puede observarse, en el caso del tiempo no hace falta poner el mínimo, pues siempre será 0:

```
    xmin = -25; xmax = 25; dx = 1;
    ymin = -25; ymax = 25; dy = 1;
    tmax = 100;  dt = 0.1;
```

Ahora pediremos que se introduzca el valor de la constante c (que nos vendrá dada por el enunciado). Hemos tomado que tenga valores comprendidos entre 0 y 7, ya que en los casos en los que supera este valor daba problemas:

```
    c = -1;
    while c <= 0 || c > 7
        fprintf('Introduzca un valor para c entre 0 y 7\n');
        c = input('>');
    end
```

Una vez hecho esto, comienzan los cálculos destinados a obtener el vídeo:

```
    fprintf('\nCalculando U(x,y,t)\n');
```

Definimos la función g como:

```
    g = inline('-exp(-((x)^2+(y+5)^2)/4)');
```

Inicializamos la matriz U0, poniéndola toda a 0. Sus dimensiones vendrán dadas por la diferencia entre los valores máximos y mínimos de X e Y divididos por su respectivo incremento:

```
    U0 = zeros(length([xmin:dx:xmax]),length([ymin:dy:ymax]));
```

Ahora invocamos a las funciones ecuacion_ondas_2d y a video_ondas_2d para que se calculen todos los valores necesarios y se obtenga el vídeo solicitado:

```
    [U,X,Y,T] =
ecuacion_ondas_2d(xmin,xmax,dx,ymin,ymax,dy,tmax,dt,c,g,U0);

    video_ondas_2d(U,X,Y,T,2,'cyan');
```

Hemos de tener en cuenta que podría resultar interesante que nos devolviera el valor final de la matriz U, pero esto sólo se hará en el caso de que se pida expresamente, ya que el archivo así generado ocupa 20 Mb y el ordenador, en varias ocasiones, se nos bloqueó:

```
    if nargout > 0
        U_ = U;
    X_ = X;
    Y_ = Y;
    T_ = T;
    end
return
```

Ecuación de ondas: Ejemplo 2

En este segundo ejemplo, veremos el movimiento de una onda teniendo en cuenta otras condiciones iniciales. El resto se mantendrá muy similar al caso anterior.

```
function [U_,X_,Y_,T_] = ejemplo2
    fprintf('Ejemplo 2 de ecuación de ondas\n');
```

Para ello, definiremos los valores máximos y mínimos de todas las variables, así como sus incrementos. Como puede observarse, en el caso del tiempo no hace falta poner el mínimo, pues siempre será 0:

```
    xmin = -22; xmax = 22; dx = 1;
    ymin = -22; ymax = 22; dy = 1;
    tmax = 100;  dt = 0.1;
```

Ahora pediremos que se introduzca el valor de la constante c (que nos vendrá dada por el enunciado). Hemos tomado que tenga valores comprendidos entre 0 y 7, ya que en los casos en los que supera este valor daba problemas:

```
    c = -1;
    while c <= 0 || c > 7
        fprintf('Introduzca un valor para c entre 0 y 7\n');
        c = input('>');
    end
```

Una vez hecho esto, comienzan los cálculos destinados a obtener el vídeo:

```
    fprintf('\nCalculando U(x,y,t)\n');
```

Definimos la función g como:

```
    g = inline('0*x + 0*y');
```

Inicializamos la matriz U0, poniéndola toda a 1. Sus dimensiones vendrán dadas por la diferencia entre los valores máximos y mínimos de X e Y divididos por su respectivo incremento:

```
    U0 = ones(length([xmin:dx:xmax]),length([ymin:dy:ymax]));
```

Ahora invocamos a las funciones ecuacion_ondas_2d y a video_ondas_2d para que se calculen todos los valores necesarios y se obtenga el vídeo solicitado:

```
    [U,X,Y,T] =
ecuacion_ondas_2d(xmin,xmax,dx,ymin,ymax,dy,tmax,dt,c,g,U0);

    video_ondas_2d(U,X,Y,T,2,'green');
```

Hemos de tener en cuenta que podría resultar interesante que nos devolviera el valor final de la matriz U, pero esto sólo se hará en el caso de que se pida expresamente, ya que el archivo así generado ocupa 20 Mb y el ordenador, en varias ocasiones, se nos bloqueó:

```
    if nargout > 0
        U_ = U;
    X_ = X;
    Y_ = Y;
    T_ = T;
    end
return
```

Ejemplo calor

En este tercer y último ejemplo, veremos la evolución de una placa rectangular calentada en un punto.

```
function [U_,X_,Y_,T_] = ejemplo_calor1
    fprintf('Ejemplo 1 de ecuación del calor\n');
    fprintf('Simula tirar una piedra en el agua\n');
```

Para ello, definiremos los valores máximos y mínimos de todas las variables, así como sus incrementos. Como puede observarse, en el caso del tiempo no hace falta poner el mínimo, pues siempre será 0 (al igual que en el ejemplo 1):

```
    xmin = -15; xmax = 15; dx = 1;
    ymin = -15; ymax = 15; dy = 1;
    tmax = 100;  dt = 0.1;
```

Ahora pediremos que se introduzca el valor de la constante alfa (que nos vendrá dada por el enunciado). Hemos tomado que tenga valores comprendidos entre 0 y 10, ya que en los casos en los que supera este valor daba problemas:

```
    alfa = -1;
    while alfa <= 0 || alfa > 10
        fprintf('Introduzca un valor para alfa entre 0 y 1\n');
        alfa = input('>');
    end
```

Una vez hecho esto, comienzan los cálculos destinados a obtener el vídeo:

```
    fprintf('\nCalculando U(x,y,t)\n');
```

Definimos la función g. Con ella determinaremos el valor de la matriz U en el instante inicial, por lo que partimos de que las condiciones iniciales no son iguales para todos los puntos de la misma en dicho momento:

```
    g = inline('20*exp(-((x)^2+(y+5)^2)/20) -5');
    U0 = evalua_funcion(g,[xmin:dx:xmax],[ymin:dy:ymax]);
```

Ahora invocamos a las funciones ecuacion_ondas_2d y a video_ondas_2d para que se calculen todos los valores necesarios y se obtenga el vídeo solicitado:

```
    [U,X,Y,T] =
ecuacion_calor_2d(xmin,xmax,dx,ymin,ymax,dy,tmax,dt,alfa,U0);

    video_ondas_2d(U,X,Y,T,2);
```

Hemos de tener en cuenta que podría resultar interesante que nos devolviera el valor final de la matriz U, pero esto sólo se hará en el caso de que se pida expresamente, ya que el archivo así generado ocupa 20 Mb y el ordenador, en varias ocasiones, se nos bloqueó:

```
    if nargout > 0
        U_ = U;
    X_ = X;
    Y_ = Y;
    T_ = T;
    end
return
```

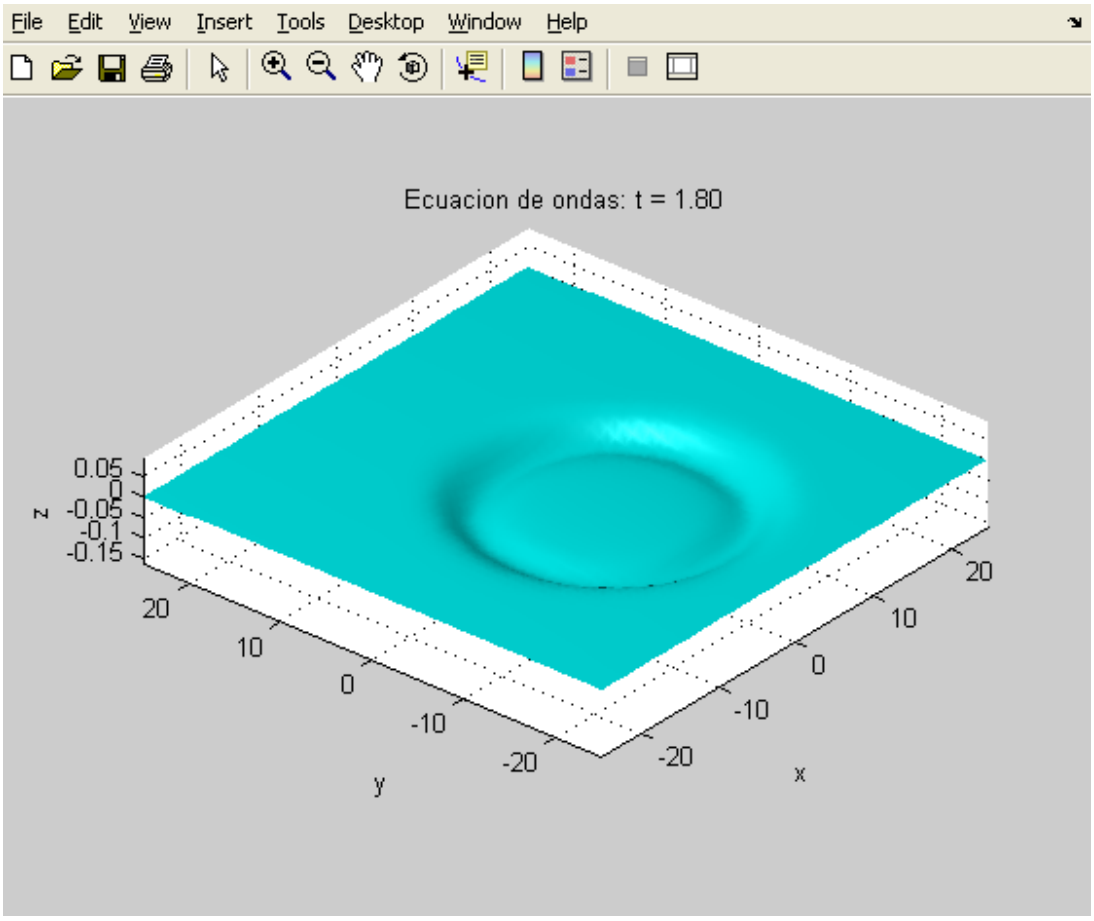
Ejecuciones de los ejemplos

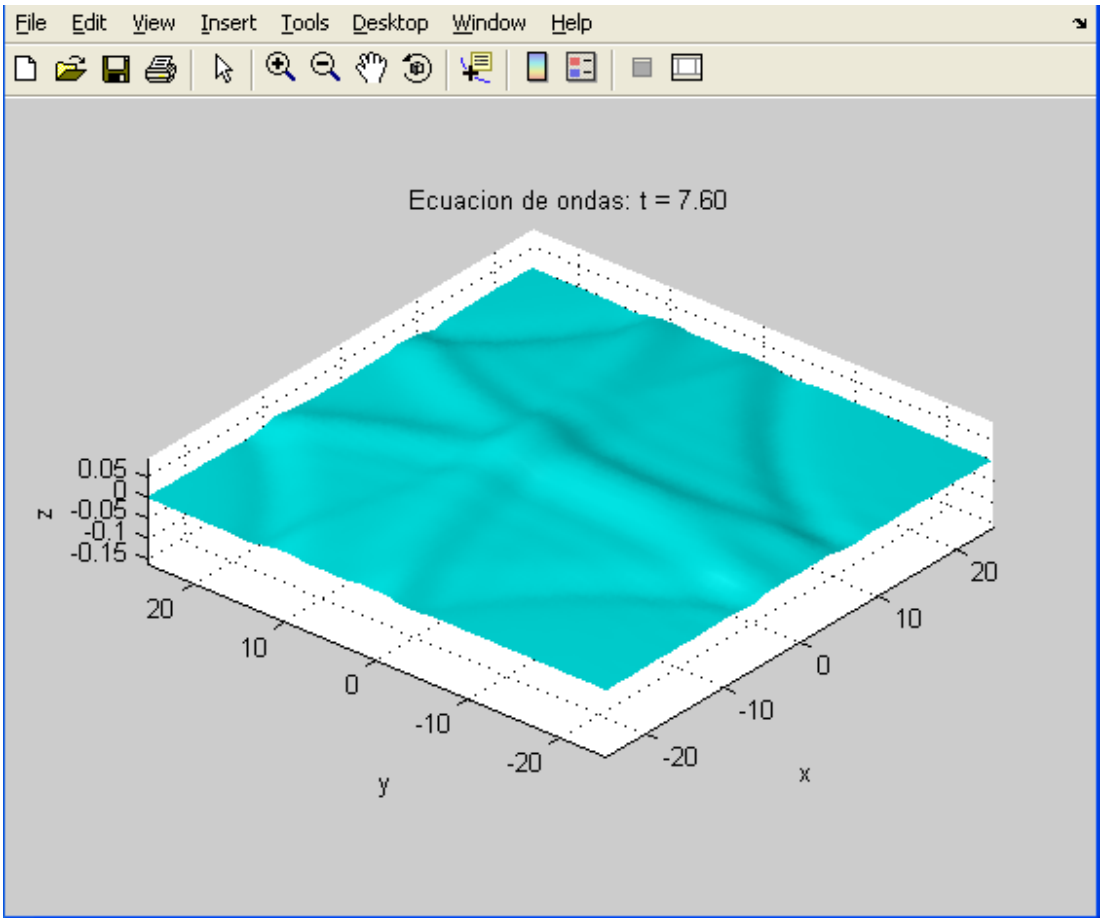
En esta sección mostraremos los resultados obtenidos al ejecutar los tres ejemplos desarrollados.

EJEMPLO 1

```
>> ejemplo1
Ejemplo 1 de ecuación de ondas
Simula tirar una piedra en el agua
Introduzca un valor para c entre 0 y 7
>7

Calculando U(x,y,t)
presiona tecla para ver video
```

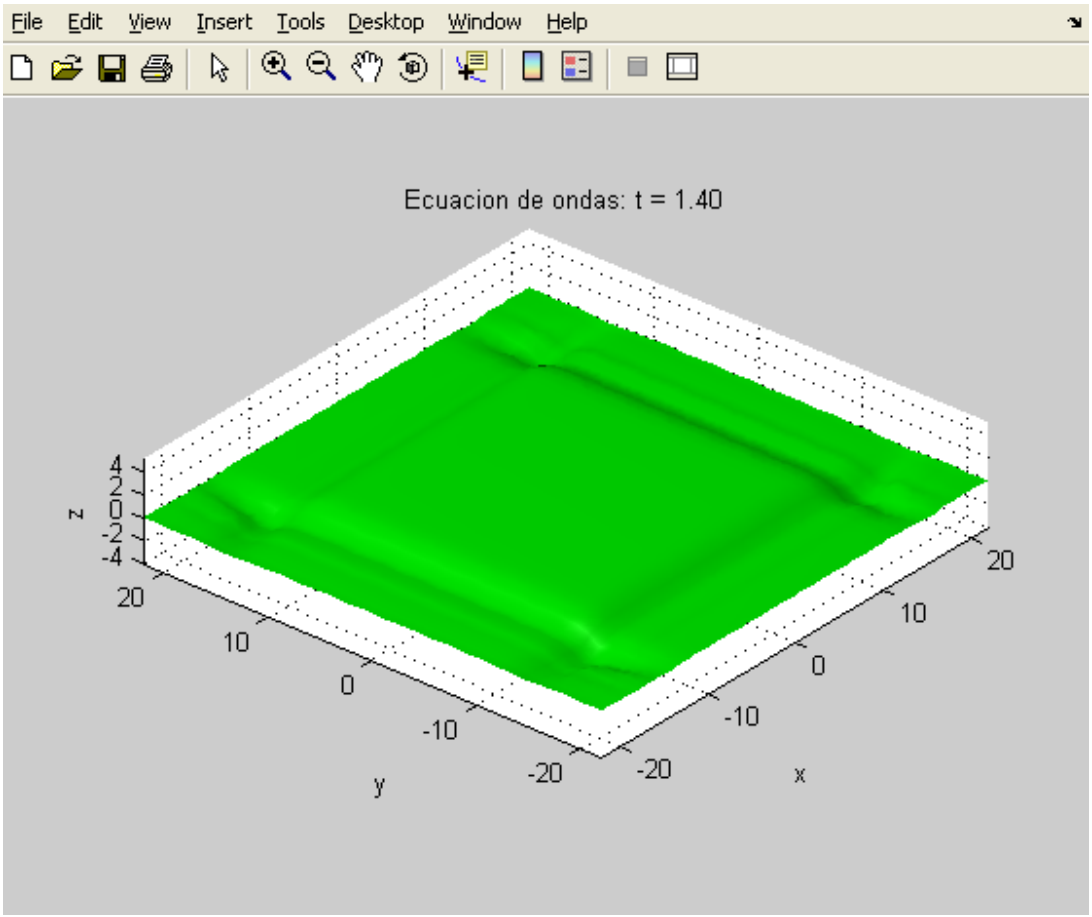


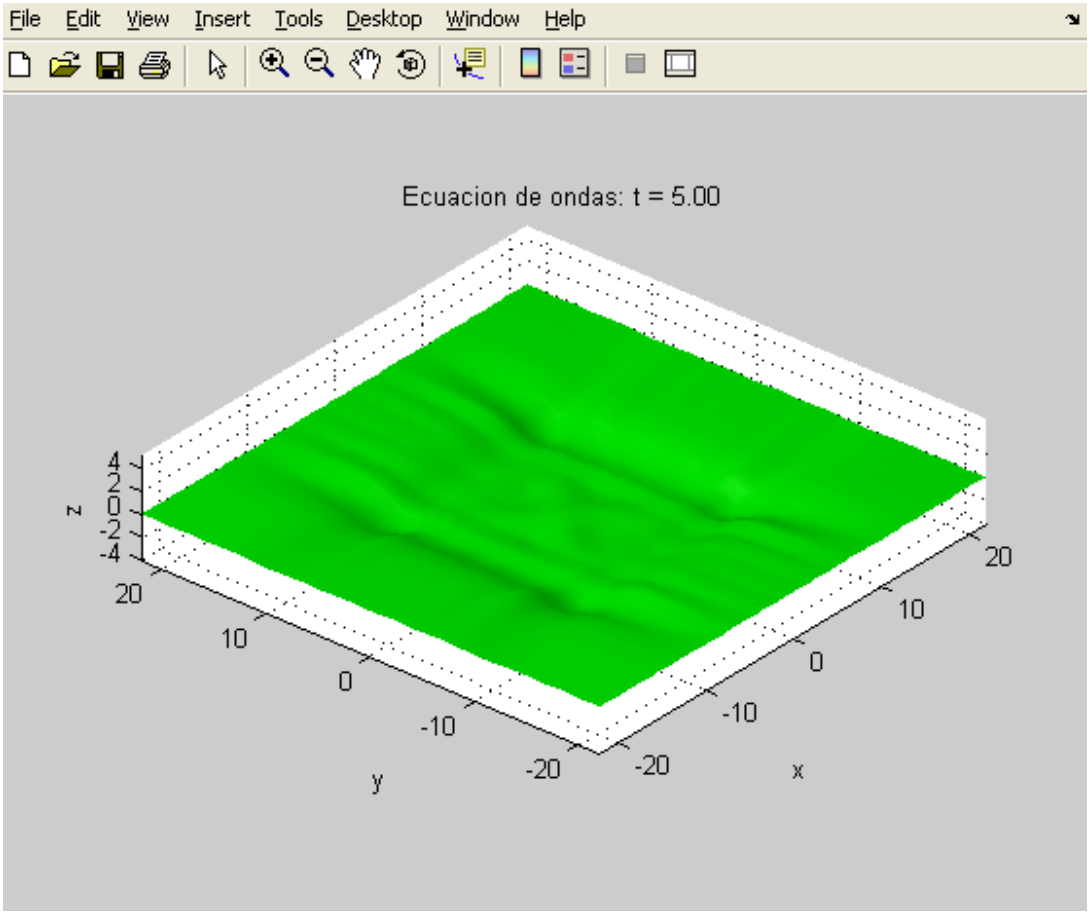

File Edit View Insert Tools Desktop Window Help
Ecuacion de ondas: t = 7.60
z
0.05
0
-0.05
-0.1
-0.15
20
10
0
-10
-20
y
20
10
0
-10
-20
x

EJEMPLO 2

>> ejemplo2
Ejemplo 2 de ecuación de ondas
Introduzca un valor para c entre 0 y 7
>7

Calculando U(x,y,t)
presiona tecla para ver video

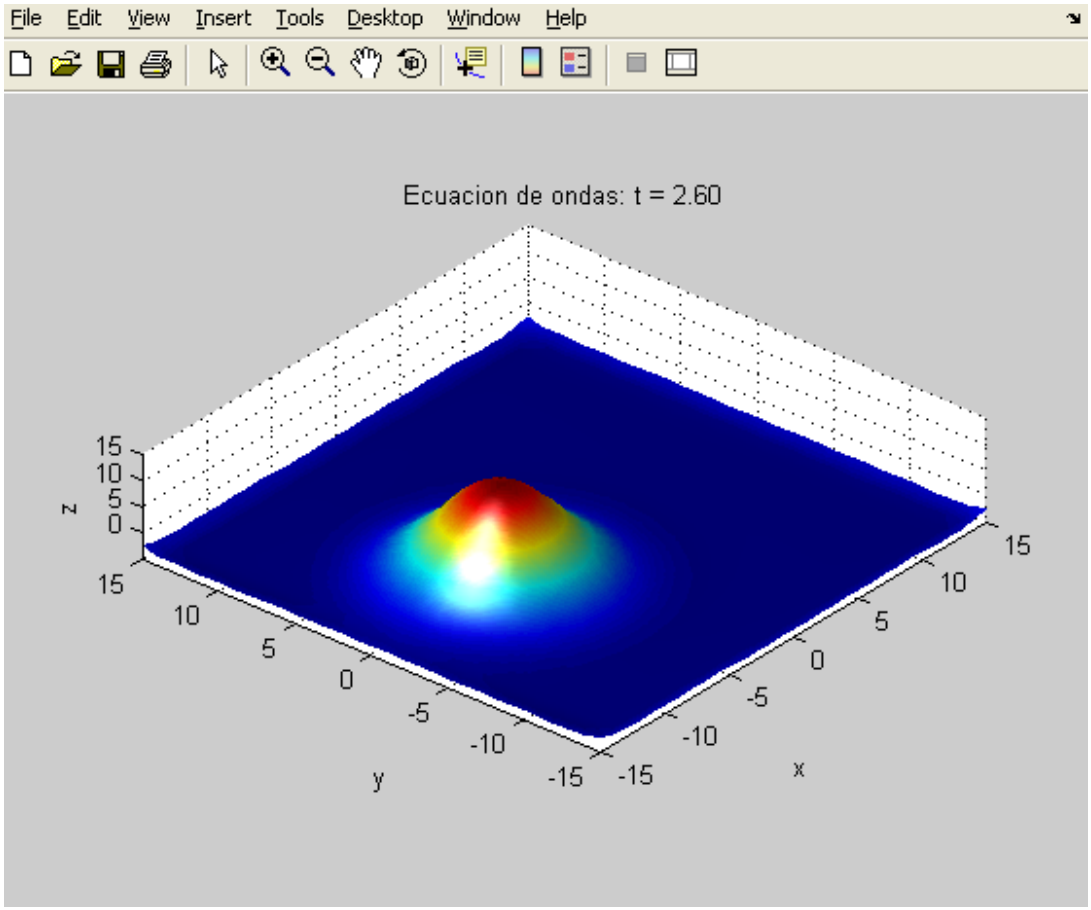


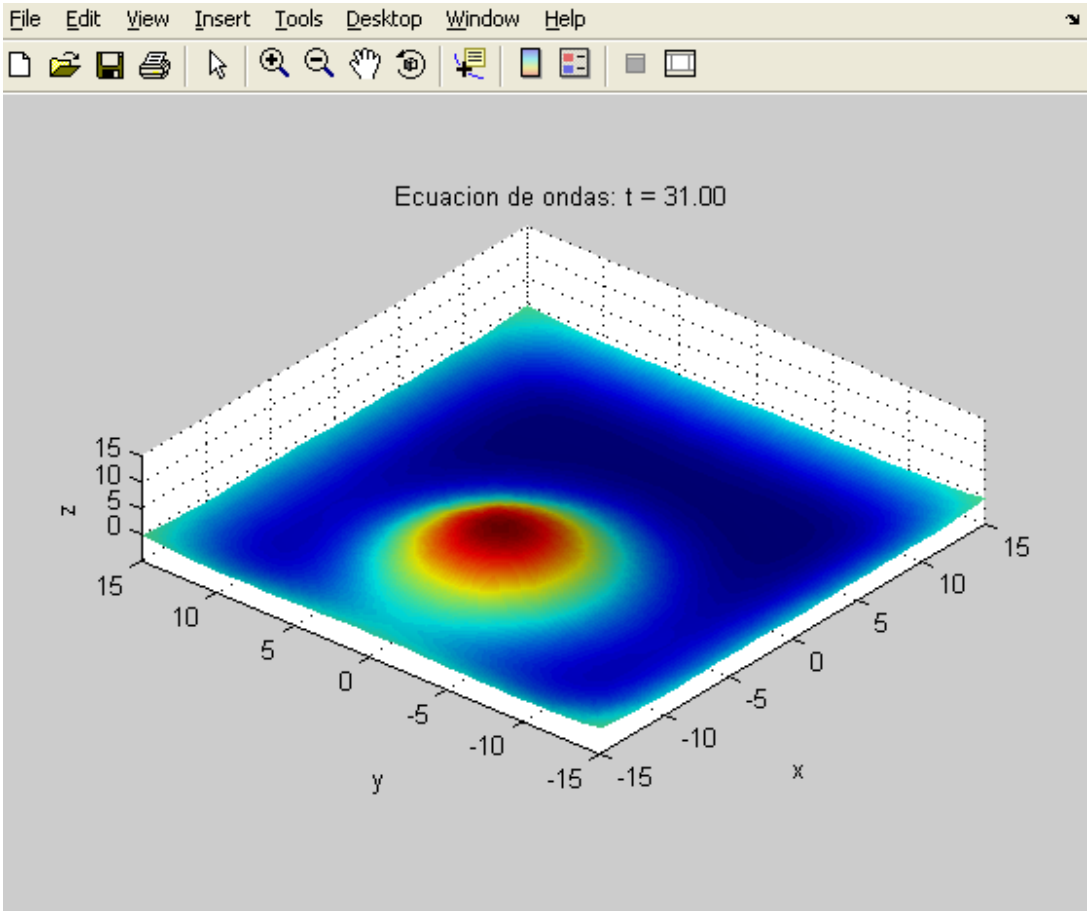

File Edit View Insert Tools Desktop Window Help
Ecuacion de ondas: t = 5.00
z
4
2
0
-2
-4
20
10
0
-10
-20
y
-20
-10
0
10
20
x

EJEMPLO 3 (EXTRA)

```
>> ejemplo_calor1
Ejemplo 1 de ecuación del calor
Simula tirar una piedra en el agua
Introduzca un valor para alfa entre 0 y 1
>0.16

Calculando U(x,y,t)
presiona tecla para ver video
```




File Edit View Insert Tools Desktop Window Help
Ecuacion de ondas: t = 31.00
z
15
10
5
0
y
15
10
5
0
-5
-10
-15
x
-15
-10
-5
0
5
10
15