UNIVERSIDAD AUTÓNOMA DE NUEVO LÉON

FACULTAD DE CIENCIAS FÍSICO MATEMÁTICAS

FÍSICA COMPUTACIONAL

PRODUCTO INTEGRADOR DE APRENDIZAJE

INTEGRANTES DEL EQUIPO

Gerardo Ruvalcaba Hijar 1870180

Rene Rogelio Corrales Millán 1941587

José Abraham Morales Vidales 1941505

NOMBRE DEL MAESTRO:

TLAHUICE FLORES ALFREDO

05 de diciembre de 2020

Ecuaciones diferenciales parciales hiperbólicas

Introducción

Las ecuaciones diferenciales parciales hiperbólicas (EDPH) son muy importantes en física, pues describen el comportamiento de las ondas elásticas, acústicas y electromagnéticas. También son de suma importancia en la dinámica de fluidos, porque las EDPH describen el flujo de fluidos sin viscosidad, es de esta rama de la física de donde provienen los avances en los esquemas numéricos de las EDPH. Los métodos numéricos para resolver EDP hiperbólicas se basan en la forma de primer orden, incluso las soluciones para las ecuaciones de orden superior se basan en las siguientes 3 formas de primer orden:

- EDPH lineal conservativa:

$$\frac{\partial}{\partial t}u(x,t) + \frac{\partial}{\partial x}f(x,t) = s(x,t) \tag{1}$$

  Donde *f* es una función lineal de *u* y *s* es un término fuente.

- EDPH lineal no conservativa:

$$\frac{\partial}{\partial t}u(x,t) + a(x)\frac{\partial}{\partial x}u(x,t) = s(x,t) \tag{2}$$

- EDPH no lineal conservativa:

$$\frac{\partial}{\partial t}u(x,t) + \frac{\partial}{\partial x}f(u(x,t)) = s(x,t) \tag{3}$$

  Donde *f* es un función no lineal de *u*.

<u>Problema para analizar: ecuación de onda</u>

La ecuación de onda en una dimensión se puede deducir de la Ley de Hooke:

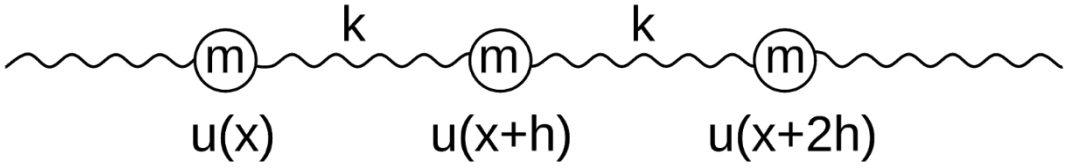


Figura 1. La ecuación de la onda se puede determinar con la ley de Hooke utilizando un esquema en el que se tiene un conjunto de masas unidas por resortes.

u(x) mide la distancia desde la masa *m* al punto de equilibrio. La fuerza ejercida sobre la masa que se en $x+h$ es:

$$F_{Newton} = ma(t) = m\frac{\partial^2}{\partial t^2}u(x+h,t) \tag{4}$$

$$F_{Hooke} = F_{x+2h} - F_x = k[u(x+2h,t) - u(x+h,t)] - k[u(x+h,t) - u(x,t)] \quad (5)$$

Igualando las expresiones:

$$\frac{\partial^2}{\partial t^2}u(x+h,t) = \frac{k}{m}[u(x+2h,t) - u(x+h,t)] - k[u(x+h,t) - u(x,t)] \quad (6)$$

Supongamos que temenos *N* masas distribuidas en una longitud *L=Nh*, con una masa total *M=Nm*, y una constante de resorte total *K=k/N*:

$$\frac{\partial^2}{\partial t^2}u(x+h,t) = \frac{KL^2}{M}\frac{u(x+2h,t) - 2u(x+h,t) + u(x,t)}{h^2} \quad (7)$$

Tomando el límite $N \to \infty,\ h \to 0$, el lado derecho de la ecuación anterior es simplemente la segunda derivada con respect a *x*:

$$\frac{\partial^2 u(x,t)}{\partial t^2} = \frac{KL^2}{M}\frac{\partial^2 u(x,t)}{\partial x^2} \quad (8)$$

Analizando las unidades de la constante $\frac{KL^2}{M}$:

$$\frac{(N/m)m^2}{kg} = \frac{kg \cdot \frac{m}{s^2}(m)}{kg} = m^2/s^2$$

Entonces la constante es la velocidad de propagación al cuadrado

El modelo de ecuación diferencial hiperbólica de segundo orden es la ecuación de onda:

$$\frac{\partial^2 u}{\partial t^2} = \alpha\frac{\partial^2 u}{\partial x^2} \quad (9)$$

Donde $\alpha = v^2$ y *s = 0*. Si el valor de *u* y de su primera derivada están especificadas arbitrariamente en la línea *t = 0*, entonces existe una solución para *u* en cualquier tiempo *t*. La solución para esta ecuación debe describir a una onda.

Objetivos

Encontrar la solución numérica para la ecuación de onda en una y dos dimensión, para condiciones iniciales y en la frontera cualquiera. Graficar los datos obtenidos y compararlos con la solución analítica, además de realizar una animación del movimiento ondulatorio.

Hipótesis

Los datos obtenidos numéricamente en un intervalo de tiempo tendrán un comportamiento ondulatorio. Se espera que sean iguales a la solución analítica.

Metodología

Se realizaron programas en FORTRAN que encuentran la solución de la ecuación de onda en una y dos dimensiones, las cuales son EDP hiperbólica por medio de los métodos explícito e implícito (sólo para 1-D). Se tabularon los datos obtenidos en un archivo de texto. Los mismos códigos de FORTRAN hacen y corren un script de Gnuplot que genera un gif animado que muestra el movimiento ondulatorio de la cuerda o membrana.

Método explícito

La solución para la ecuación de onda existe en un plano general espacio-tiempo, entonces el valor de *u* debe ser determinado en todos los puntos del plano:

Tabla 1.

t_i \ x_j	$j = 1$	$j = 2$	$j = 3$	…	$j = n$
$i = 0$	$u(x_1, t_0)$	$u(x_2, t_0)$	$u(x_3, t_0)$	…	$u(x_n, t_0)$
$i = 1$	$u(x_1, t_1)$	$u(x_2, t_1)$	$u(x_3, t_1)$	…	$u(x_n, t_1)$
…	…	…	…	…	…
$i = n$	$u(x_1, t_n)$	$u(x_2, t_n)$	$u(x_3, t_n)$	…	$u(x_n, t_n)$

En el método explícito se reemplaza las segundas derivadas con la aproximación por diferencias finitas centrales:

$$\frac{\partial^2 u}{\partial x^2} = \frac{1}{(\Delta x)^2}\left(u_{i,j-1} - 2u_{i,j} + u_{i,j+1}\right) \tag{10}$$

$$\frac{\partial^2 u}{\partial t^2} = \frac{1}{(\Delta t)^2}\left(u_{i-1,j} - 2u_{i,j} + u_{i+1,j}\right) \tag{11}$$

Sustituyendo (5) y (6) en la ecuación (4):

$$\frac{1}{(\Delta t)^2}\left(u_{i-1,j} - 2u_{i,j} + u_{i+1,j}\right) = \frac{\alpha}{(\Delta x)^2}\left(u_{i,j-1} - 2u_{i,j} + u_{i,j+1}\right) \tag{12}$$

De la ecuación (7) la variable desconocida es la función de posición *u* en el tiempo futuro t_{i+1} en la posición x_j, despejando esta incógnita se obtiene la expresión que se debe programar:

$$u_{i+1,j} = \frac{\alpha(\Delta t)^2}{(\Delta x)^2}\left(u_{i,j-1} - 2u_{i,j} + u_{i,j+1}\right) + 2u_{i,j} - u_{i-1,j} \tag{13}$$

Método Implícito

Tenemos la ecuación de onda:

$$\frac{\partial^2 u}{\partial t^2} = \alpha \frac{\partial^2 u}{\partial x^2}$$

En el método implícito se utiliza un esquema complicado para aproximar la derivada:

$$\frac{1}{(\Delta t)^2}\left(u_i^{k-1} - 2u_i^k + u_i^{k+1}\right) = \frac{\alpha}{(\Delta x)^2}\begin{bmatrix} \frac{1}{4}\left(u_{i-1}^{k-1} - 2u_i^{k-1} + u_{i+1}^{k-1}\right) \\ +\frac{1}{2}\left(u_{i-1}^{k} - 2u_i^{k} + u_{i+1}^{k}\right) \\ +\frac{1}{4}\left(u_{i-1}^{k+1} - 2u_i^{k+1} + u_{i+1}^{k+1}\right) \end{bmatrix} \tag{14}$$

Donde la variable *k* representa los instantes de tiempo e *i* los instantes de la posición *x*. Agrupamos algunas constantes en una variable *r*.

$$r^2 = \frac{\alpha(\Delta t)^2}{(\Delta x)^2} \tag{15}$$

Se agrupan las incógnitas del lado izquierdo con sus respectivos cofactores, formando un sistema de ecuaciones

$$-\frac{r^2}{4}u_{i-1}^{k+1} + \left(1 + \frac{r^2}{2}\right)u_i^{k+1} - \frac{r^2}{4}u_{i+1}^{k+1} = r^2\begin{bmatrix} \frac{1}{4}\left(u_{i-1}^{k-1} - 2u_i^{k-1} + u_{i+1}^{k-1}\right) \\ +\frac{1}{2}\left(u_{i-1}^{k} - 2u_i^{k} + u_{i+1}^{k}\right) \end{bmatrix} + 2u_i^k - u_i^{k-1} \tag{16}$$

En la primer iteración, *k* = 0, se deben utilizar las condiciones iniciales

$$u(x,0) = -\sin \pi x$$

$$\frac{\partial u}{\partial t} = \frac{1}{2\Delta t} = \left(u_i^{k+1} - u_i^{k-1}\right) = 0$$

$$u_i^{k+1} = u_i^{k-1},\ u_{i-1}^{k+1} = u_{i-1}^{k-1},\ u_{i+1}^{k+1} = u_{i+1}^{k-1} \tag{17}$$

Sustituyendo las ecuaciones (17) en (16):

$$-\frac{r^2}{4}u_{i-1}^{k+1}+\left(1+\frac{r^2}{2}\right)u_i^{k+1}-\frac{r^2}{4}u_{i+1}^{k+1}=r^2\left[\begin{array}{l}\frac{1}{4}\left(u_{i-1}^{k+1}-2u_i^{k+1}+u_{i+1}^{k+1}\right)\\+\frac{1}{2}\left(u_{i-1}^{k}-2u_i^{k}+u_{i+1}^{k}\right)\end{array}\right]+2u_i^k-u_i^{k+1}$$

$$\boxed{-\frac{r^2}{2}u_{i-1}^{k+1}+(2+r^2)u_i^{k+1}-\frac{r^2}{2}u_{i+1}^{k+1}=r^2\left[\frac{1}{2}\left(u_{i-1}^{k}-2u_i^{k}+u_{i+1}^{k}\right)\right]+2u_i^k} \quad (18)$$

Véase que tanto (16) como (18) son sistemas de ecuaciones lineales. El sistema (18) se utilizará en el primer paso, fuera del ciclo, y el sistema (16) se calculará dentro de un ciclo para todos los instantes de tiempo que se deseen.

Código en FORTRAN

En la primera parte del código se declaran los intervalos de *x* y *t*, la longitud de la cuerda, el tiempo final y la constante α.

```
TF=4    !Tiempo final !********
L=1.5   !Longitud de la cuerda
N1=15   !intervalos x
N2=20   !intervalos t !*******

H=L/N1   !delta de x
HT=TF/N2  !delta de t
ALPHA=1  !constante 1/velocida

R=ALPHA*HT/H !Constante
```

Después se declaran las condiciones iniciales y condiciones en la frontera

```
DO WHILE(X.LE.L)
   U(1,I) = F_IN(X) !Función de
   WRITE(*,*)"I=",I,"u=",U(1,I)
   X=X+H
   I=I+1
END DO
!N1=I
!Condiciones en la frontera
T=0
DO I=1,N2+1
   U(I,1)=F_FRON(T)
   U(I,N1+1)=0
   WRITE(*,*) U(I,N1+1)
   T=T+HT
END DO
```

```
Función de condiciones iniciales
   FUNCTION F_IN(X)
   REAL*8 F_IN,X,PI
      PI=ACOS(-1.0)
      !F_IN=-SIN(PI*X)
      !F_IN=COS(PI*X)
      !F_IN=0.0
      !F_IN=SIN(2*PI*0.02*X)
      IF(X.LE.1) THEN
           F_IN=0.05*X
      ELSE IF(X.GT.1) THEN
           F_IN=-0.1*X+0.15
      END IF
   END FUNCTION

Función de condiciones en la frontera
   FUNCTION F_FRON(T)
   REAL*8 F_FRON,T,PI
   PI=ACOS(-1.0)
   !IF(T.LT.1) THEN
   !F_FRON=SIN(T*PI)
   !ELSE
   F_FRON=0.0
   !F_FRON=SIN(2*PI*(-0.2*T))
   !END IF
   END FUNCTION
```

Después se hace el primer paso, calculando los cofactores del sistema (18). También se determina el vector de soluciones (TI).

```
X=2*H
I=2
DO WHILE(X.LE.L+H/2)
   DO J=I-1,I+1
      IF(I.NE.2.OR.J.NE.1)THEN
      IF(J.EQ.I-1.OR.J.EQ.I+1)THEN
          A(I,J)=-R**2/2
      ELSE
          A(I,J)=(2.0+R**2)
      END IF
      END IF
   END DO
   !Se calcula el término independiente
   TI(I)=(R**2/2)*(U(1,I-1)-2*U(1,I)+U(1,I+1))+2*U(1,I)
   WRITE(5,*)TI(I)
   A(I,N1+1)=TI(I)
   X=X+H
   I=I+1
END DO
```

Después resolvemos los siguientes pasos. En esta parte del código se calculan los cofactores del sistema de ecuaciones (16), así como el vector de soluciones o término independiente:

```
!SIGUIENTES MATRICES DE COFACTORES
K=2
T=2*HT
DO WHILE(T.LE.TF)
   DO I=2,N1
      TI(I)=0
      DO J=2,N1+1
         A(I,J)=0
      END DO
   END DO
   X=2*H
   I=2
   DO WHILE(X.LE.L+H/2)
      DO J=I-1,I+1
         IF(I.NE.2.OR.J.NE.1)THEN
         IF(J.EQ.I-1.OR.J.EQ.I+1)THEN
            A(I,J)=-R**2/4
         ELSE
            A(I,J)=(1.0+R**2/2)
         END IF
         END IF
      END DO
      !Término independiente
      T1=0.25*(U(K-1,I-1)-2*U(K-1,I)+U(K-1,I+1))
      T2=(R**2)*(T1+.5*(U(K,I-1)-2*U(K,I)+U(K,I+1)))
      TI(I)=T2+2*U(K,I)-U(K-1,I)
      A(I,N1+1)=TI(I)
      X=X+H
      I=I+1


       END DO
       WRITE(5,*)" "
       DO I=1,N1
          WRITE(5,*)(A(I,J),J=1,N1+1)
       END DO
       CALL GAUSS(A,N1+1)
       !CALL LU(A,N1,TI)
       DO I=2,N1+1
          U(K+1,I)=A(I,N1+1)
       END DO
       T=T+HT
       K=K+1
    END DO
```

Después se hacen los scripts en gnuplot para graficar y hacer las animaciones

```
!Script de gnuplot para graficar
OPEN(7,FILE="implicito.plt")
WRITE(7,*)'set title "Metodo implicito" font ",12"'
write(7,*)"set grid"
write(7,*)"set zeroaxis"
write(7,*)'set xrange[0:',l,']'
!write(7,*)'set yrange[-1.2:0]'
write(7,*)'set xlabel "x" font ",12" '
write(7,*)'set ylabel "u(x,t)" font ",12" '
write(7,*)'plot "implicito.txt" using 1:2 w lp'
do i=3,N2+1
   write(7,*)'replot "implicito.txt" using 1:',i,' w lp'
end do
close(7)
```

```
CALL SYSTEM('implicito.plt')

!Script de gnuplot para hacer la animación de la cuerda
OPEN(9,FILE="implicit_animation.plt")
WRITE(9,*)'stats "fotograma.dat" name "A"'
WRITE(9,*)'set xrange [A_min_x:',L,']'
WRITE(9,*)'set yrange [A_min_y:A_max_y]'
WRITE(9,*)'set border 3'
WRITE(9,*)'set xlabel "X" font ",12"'
WRITE(9,*)'set ylabel "U(x,t)" font ",12"'
WRITE(9,*)'set term gif animate delay 20'
WRITE(9,*)'set output "cuerda implicit.gif"'
z='do for[i=0:int(A_blocks-1)]{plot "fotograma.dat" index i w
WRITE(9,*)z
CLOSE(9)
CALL SYSTEM('implicit_animation.plt')
```

Resultados

Ejemplo 1

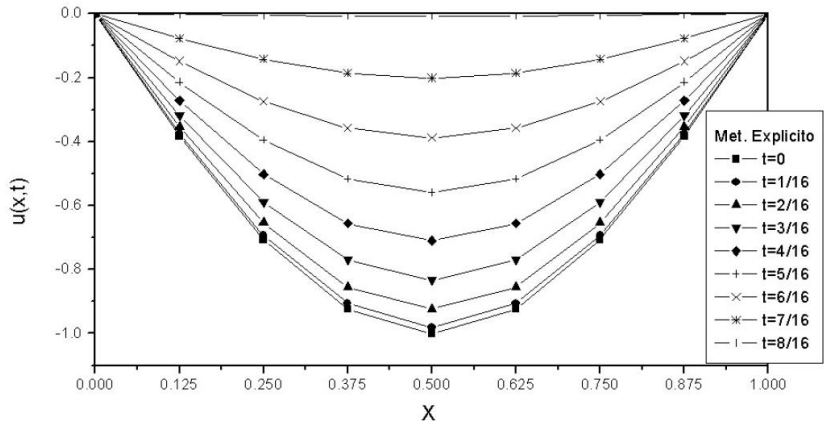
Graficar el movimiento ondulatorio de una cuerda de *1 m* de longitud una constante $\alpha = 1$. Y las siguientes condiciones iniciales y en la frontera:

$$u(x,0) = -\sin \pi x\,, \qquad u_t = 0$$

$$u(0,t) = 0,\ \ u(l,t) = 0$$

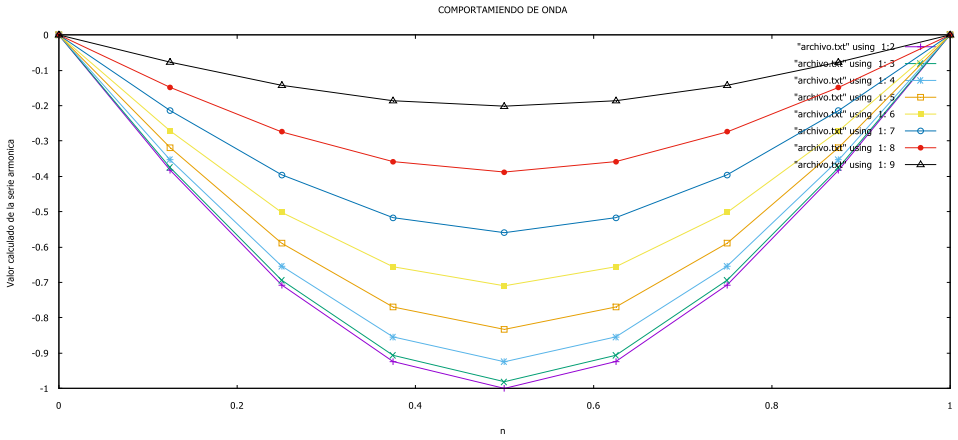
En el artículo de donde se obtuvo este problema nos da los siguientes resultados para el método explícito:

I	x	t = 0	t = 1/16	t = 1/8	t = 3/16	t = 1/4	t = 5/16	T= 3/8	t = 7/16	t = 1/2
1	0.000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
2	0.125	-0.38268	-0.37540	-0.35383	-0.31879	-0.27162	-0.21411	-0.14846	-0.07715	-0.00290
3	0.250	-0.70711	-0.69365	-0.65379	-0.58905	-0.50189	-0.39563	-0.27431	-0.14255	-0.00537
4	0.375	-0.92388	-0.90630	-0.85422	-0.76964	-0.65576	-0.51692	-0.35841	-0.18625	-0.00701
5	0.500	-1.00000	-0.98097	-0.92460	-0.83305	-0.70978	-0.55951	-0.38794	-0.20160	-0.00759
6	0.625	-0.92388	-0.90630	-0.85422	-0.76964	-0.65576	-0.51692	-0.35841	-0.18625	-0.00701
7	0.750	-0.70711	-0.69365	-0.65379	-0.58905	-0.50189	-0.39563	-0.27431	-0.14255	-0.00537
8	0.875	-0.38268	-0.37540	-0.35383	-0.31879	-0.27162	-0.21411	-0.14846	-0.07715	-0.00290
9	1.000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000



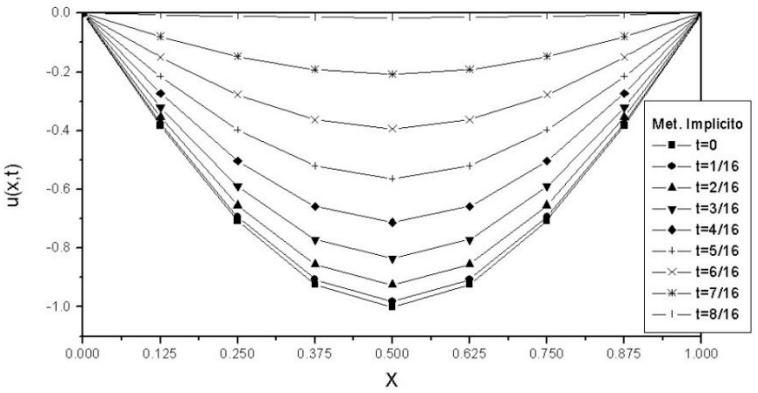
Los resultados que obtuvimos con nuestro programa del método explícito son los siguientes:

```
0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
0.12500 -0.38268 -0.37540 -0.35383 -0.31879 -0.27162 -0.21411 -0.14846 -0.07715 -0.00290
0.25000 -0.70711 -0.69365 -0.65379 -0.58905 -0.50189 -0.39563 -0.27431 -0.14255 -0.00537
0.37500 -0.92388 -0.90630 -0.85422 -0.76964 -0.65576 -0.51692 -0.35841 -0.18625 -0.00701
0.50000 -1.00000 -0.98097 -0.92460 -0.83305 -0.70978 -0.55951 -0.38794 -0.20160 -0.00759
0.62500 -0.92388 -0.90630 -0.85422 -0.76964 -0.65576 -0.51692 -0.35841 -0.18625 -0.00701
0.75000 -0.70711 -0.69365 -0.65379 -0.58905 -0.50189 -0.39563 -0.27431 -0.14255 -0.00537
0.87500 -0.38268 -0.37540 -0.35383 -0.31879 -0.27162 -0.21411 -0.14846 -0.07715 -0.00290
1.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
```



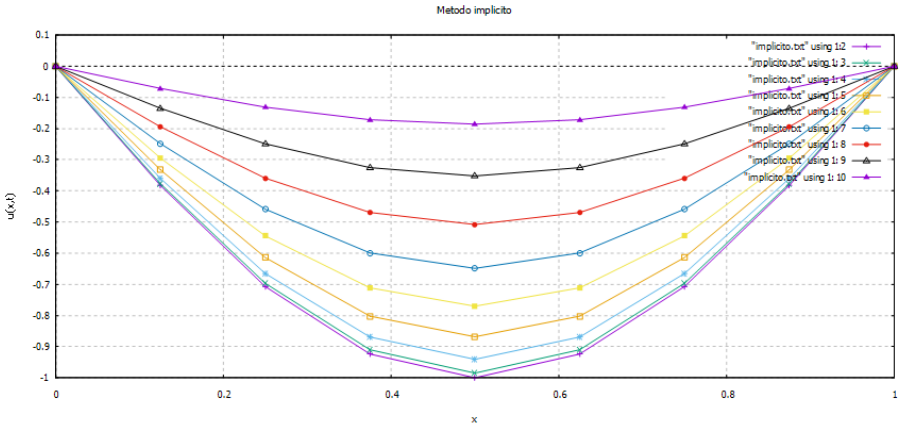
Los resultados correctos, utilizando el método implícito son los siguientes:

l	x	t = 0	t = 1/16	t = 1/8	t = 3/16	t = 1/4	t = 5/16	t= 3/8	t = 7/16	t = 1/2
1	0.000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
2	0.125	-0.38268	-0.37547	-0.35410	-0.31938	-0.27262	-0.21558	-0.15041	-0.07958	-0.00574
3	0.250	-0.70711	-0.69378	-0.65429	-0.59014	-0.50373	-0.39834	-0.27793	-0.14704	-0.01060
4	0.375	-0.92388	-0.90646	-0.85487	-0.77105	-0.65816	-0.52046	-0.36313	-0.19211	-0.01385
5	0.500	-1.00000	-0.98115	-0.92531	-0.83458	-0.71239	-0.56334	-0.39305	-0.20794	-0.01500
6	0.625	-0.92388	-0.90646	-0.85487	-0.77105	-0.65816	-0.52046	-0.36313	-0.19211	-0.01385
7	0.750	-0.70711	-0.69378	-0.65429	-0.59014	-0.50373	-0.39834	-0.27793	-0.14704	-0.01060
8	0.875	-0.38268	-0.37547	-0.35410	-0.31938	-0.27262	-0.21558	-0.15041	-0.07958	-0.00574
9	1.000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

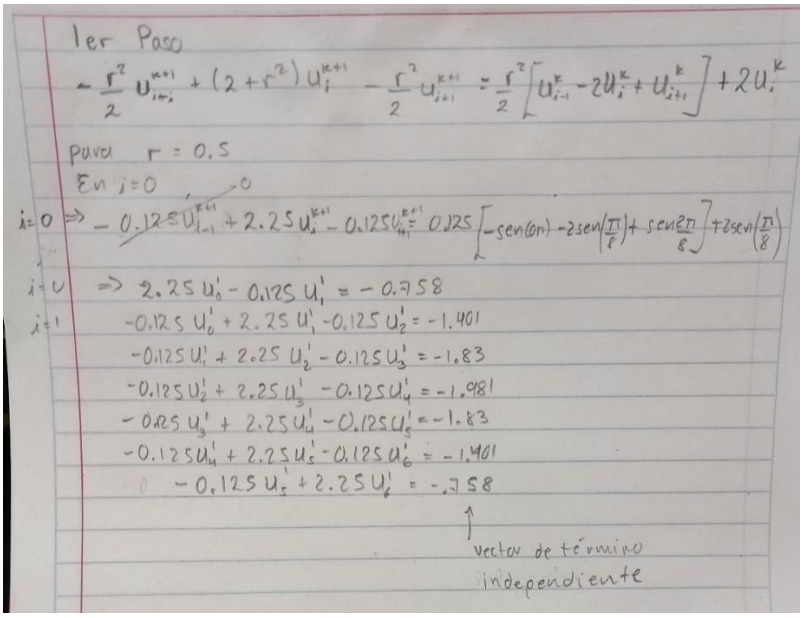


Usando el código para el método implícito:

```
0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
0.12500 -0.38268 -0.37547 -0.35410 -0.31938 -0.27262 -0.21558 -0.15041 -0.07958 -0.00574
0.25000 -0.70711 -0.69378 -0.65429 -0.59014 -0.50373 -0.39834 -0.27793 -0.14704 -0.01060
0.37500 -0.92388 -0.90646 -0.85487 -0.77105 -0.65816 -0.52046 -0.36313 -0.19211 -0.01385
0.50000 -1.00000 -0.98115 -0.92531 -0.83458 -0.71239 -0.56334 -0.39305 -0.20794 -0.01499
0.62500 -0.92388 -0.90646 -0.85487 -0.77105 -0.65816 -0.52046 -0.36313 -0.19211 -0.01385
0.75000 -0.70711 -0.69378 -0.65429 -0.59014 -0.50373 -0.39834 -0.27793 -0.14704 -0.01060
0.87500 -0.38268 -0.37547 -0.35410 -0.31938 -0.27262 -0.21558 -0.15041 -0.07958 -0.00574
1.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
```



Primer paso a mano:



Ejemplo 2

Condiciones iniciales y de frontera. Una cuerda con una longitud de 1.5*ft* y un coeficiente fg/W = 9651 lb²/s² es estirada entre los apoyos A y B. La cuerda es desplazada de su posición de equilibrio jalando en el punto correspondiente a los dos tercios de su longitud, adquiriendo la configuración de la figura 12.

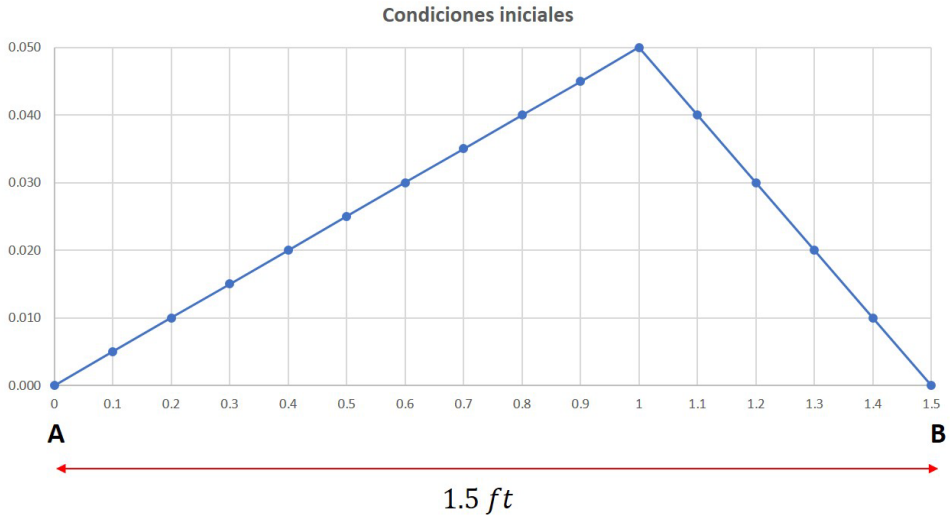
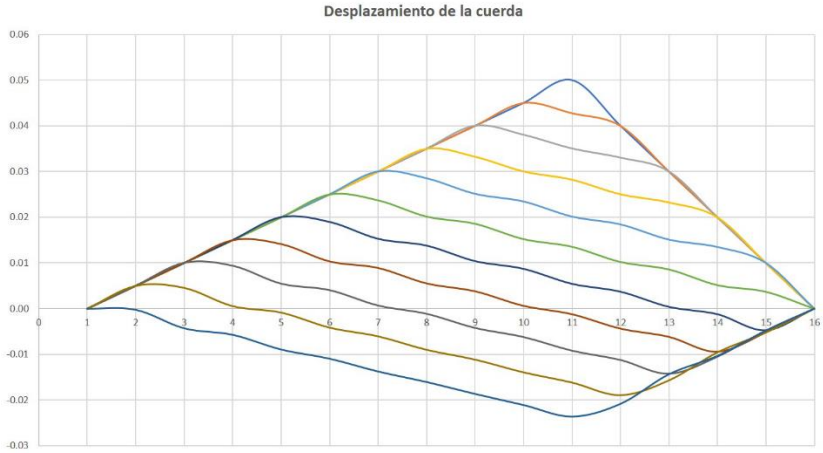


Figura 4. Condiciones iniciales del ejemplo.

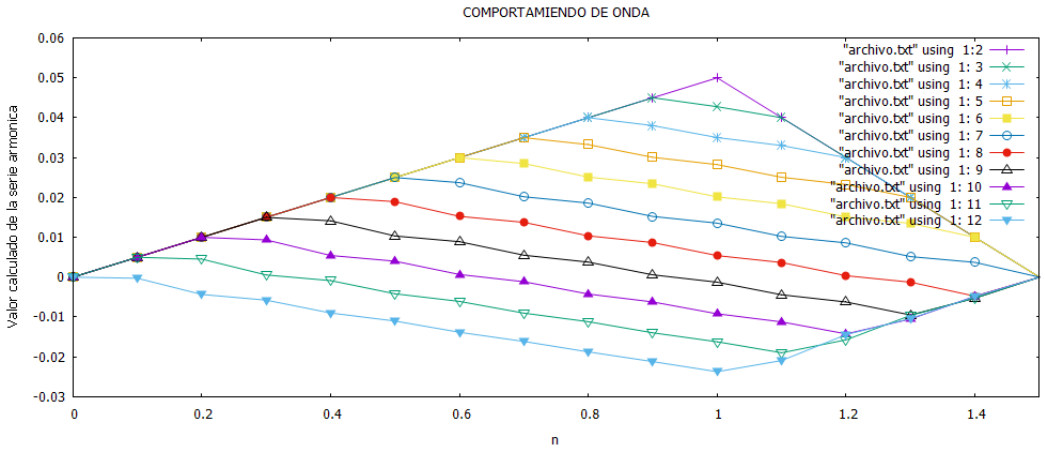
$$u(0,x) = \begin{Bmatrix} 0.05x & 0 \le x \le 1.0 \\ -0.1x + 0.15 & 1 < x \le 1.5 \end{Bmatrix} \tag{19}$$

Este ejemplo se tomó del artículo de Cortés y González, en el que ofrecen los siguientes resultados como los correctos:

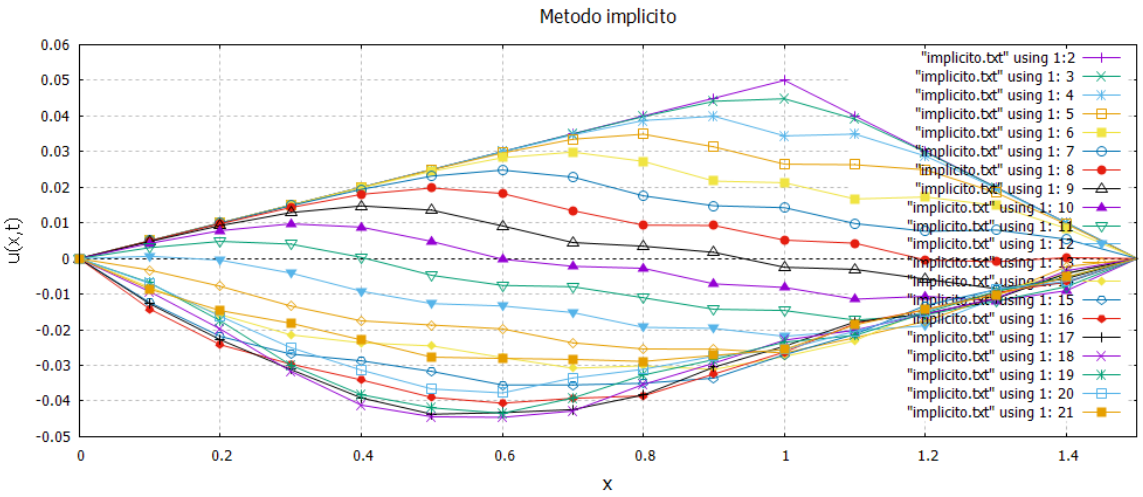
Instante (ti) \ Posición (Xj)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	0.00000	0.00500	0.01000	0.01500	0.02000	0.02500	0.03000	0.03500	0.04000	0.04500	0.05000	0.04000	0.03000	0.02000	0.01000	0.00000
1	0.00000	0.00500	0.01000	0.01500	0.02000	0.02500	0.03000	0.03500	0.04000	0.04500	0.04276	0.04000	0.03000	0.02000	0.01000	0.00000
2	0.00000	0.00500	0.01000	0.01500	0.02000	0.02500	0.03000	0.03500	0.04000	0.03801	0.03502	0.03301	0.03000	0.02000	0.01000	0.00000
3	0.00000	0.00500	0.01000	0.01500	0.02000	0.02500	0.03000	0.03500	0.03326	0.03005	0.02823	0.02505	0.02326	0.02000	0.01000	0.00000
4	0.00000	0.00500	0.01000	0.01500	0.02000	0.02500	0.03000	0.02849	0.02510	0.02343	0.02014	0.01843	0.01510	0.01349	0.01000	0.00000
5	0.00000	0.00500	0.01000	0.01500	0.02000	0.02500	0.02372	0.02017	0.01860	0.01524	0.01357	0.01024	0.00860	0.00517	0.00372	0.00000
6	0.00000	0.00500	0.01000	0.01500	0.02000	0.01894	0.01525	0.01376	0.01037	0.00868	0.00541	0.00368	0.00037	-0.00124	-0.00475	0.00000
7	0.00000	0.00500	0.01000	0.01500	0.01415	0.01034	0.00890	0.00552	0.00378	0.00059	-0.00125	-0.00441	-0.00622	-0.00948	-0.00525	0.00000
8	0.00000	0.00500	0.01000	0.00936	0.00544	0.00403	0.00068	-0.00113	-0.00421	-0.00620	-0.00918	-0.01120	-0.01421	-0.01049	-0.00477	0.00000
9	0.00000	0.00500	0.00455	0.00056	-0.00085	-0.00415	-0.00606	-0.00901	-0.01115	-0.01395	-0.01618	-0.01895	-0.01571	-0.00957	-0.00521	0.00000
10	0.00000	-0.00026	-0.00432	-0.00574	-0.00897	-0.01099	-0.01380	-0.01611	-0.01872	-0.02116	-0.02370	-0.02090	-0.01440	-0.01036	-0.00483	0.00000
11	0.00000	-0.00918	-0.01065	-0.01379	-0.01592	-0.01859	-0.02106	-0.02350	-0.02612	-0.02847	-0.02606	-0.01928	-0.01547	-0.00972	-0.00513	0.00000
12	0.00000	-0.01066	-0.01859	-0.02086	-0.02339	-0.02600	-0.02830	-0.03106	-0.03326	-0.03118	-0.02420	-0.02053	-0.01466	-0.01020	-0.00491	0.00000
13	0.00000	-0.00950	-0.02107	-0.02819	-0.03093	-0.03310	-0.03598	-0.03807	-0.03628	-0.02917	-0.02554	-0.01966	-0.01520	-0.00988	-0.00505	0.00000
14	0.00000	-0.01034	-0.01925	-0.03130	-0.03792	-0.04089	-0.04290	-0.04134	-0.03416	-0.03051	-0.02470	-0.02017	-0.01491	-0.01004	-0.00498	0.00000
15	0.00000	-0.00980	-0.02046	-0.02918	-0.04138	-0.04776	-0.04637	-0.03919	-0.03545	-0.02978	-0.02509	-0.01998	-0.01499	-0.01002	-0.00499	0.00000
16	0.00000	-0.01009	-0.01979	-0.03042	-0.03922	-0.04713	-0.04425	-0.04036	-0.03487	-0.03000	-0.02507	-0.01991	-0.01508	-0.00994	-0.00503	0.00000
17	0.00000	-0.01000	-0.02002	-0.02989	-0.03620	-0.03608	-0.04116	-0.03998	-0.03489	-0.03016	-0.02482	-0.02016	-0.01487	-0.01009	-0.00495	0.00000
18	0.00000	-0.00993	-0.02011	-0.02592	-0.02699	-0.03004	-0.03204	-0.03582	-0.03526	-0.02974	-0.02523	-0.01981	-0.01515	-0.00990	-0.00505	0.00000
19	0.00000	-0.01010	-0.01599	-0.01737	-0.01970	-0.02298	-0.02465	-0.02747	-0.03084	-0.03030	-0.02475	-0.02020	-0.01486	-0.01010	-0.00495	0.00000
20	0.00000	-0.00620	-0.00752	-0.00973	-0.01333	-0.01436	-0.01837	-0.01965	-0.02264	-0.02603	-0.02522	-0.01983	-0.01512	-0.00992	-0.00504	0.00000



Utilizando el programa del método explícito se obtuvieron los siguientes resultados:



Con el otro programa del método implícito se obtiene:



Ecuaciones diferenciales parciales hiperbolicas con dos vaiables espaciales

Por el subtítulo anterior es de esperarse que esta ecuacion sea similar a la ecuacion presentada anteriormente pero con una segunda derivada parcial agregada, y efectivamente, aunque hay varios tipos de ecuaciones que tienen estas caracteristicas nosotros nos enfocaremos en resolver un un tipo en paticular que se presenta de la siguiente manera:

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

Para escoger que metodo utilizar para resolver una ecuacion de este tipo se necesita saber las necesidades de presicion en las mediciones y tambien las características de problema y en que tipo de sistema se aplicara ya que la efectividad de los metodos varia según cada caso.Nuestro sistema a resolver sera el de una membrana que es sometida a vibraciones con condiciones de frontera definidas, debido a que se trata de una membrana rectangular podemos utilizar un el metodo explicito utilizado anteriormente, es decir el metodo de diferencias finitas centrales ya que este es bastante preciso para resolver problemas con estas caracteristicas.

Análogamente al caso unidimensional se reemplaza las segundas derivadas con la aproximación por diferencias finitas centrales:

$$\frac{\partial^2 u}{\partial x^2} = \frac{1}{(\Delta x)^2}\left(u_{i,j-1,k} - 2u_{i,j,k} + u_{i,j+1,k}\right) \tag{20}$$

$$\frac{\partial^2 u}{\partial x^2} = \frac{1}{(\Delta y)^2}\left(u_{i,j,k-1} - 2u_{i,j,k} + u_{i,j,k+1}\right) \tag{21}$$

$$\frac{\partial^2 u}{\partial t^2} = \frac{1}{(\Delta t)^2}\left(u_{i-1,j,k} - 2u_{i,j,k} + u_{i+1,j,k}\right) \tag{22}$$

Sustituyendo (21) y (22) en la ecuación (20):

$$\frac{1}{(\Delta t)^2}\left(u_{i-1,j,k} - 2u_{i,j,k} + u_{i+1,j,k}\right) = \begin{matrix}\frac{\alpha}{(\Delta x)^2}\left(u_{i,j-1,k} - 2u_{i,j,k} + u_{i,j+1,k}\right) \\ + \frac{\alpha}{(\Delta y)^2}\left(u_{i,j,k-1} - 2u_{i,j,k} + u_{i,j,k+1}\right)\end{matrix} \tag{23}$$

Para simplificar la expresión hacemos $\lambda = \frac{\alpha(\Delta t)^2}{(\Delta x)^2}$ y $\beta = \frac{\alpha(\Delta t)^2}{(\Delta y)^2}$

De la ecuación (23) la variable desconocida es la función de posición *u* en el tiempo futuro t_{i+1} en la posición x_j, despejando esta incógnita se obtiene la expresión que se debe programar:

$$\begin{matrix}u_{i+1,j,k} = 2(1-\lambda-\beta)u_{i,j,k} + \lambda\left(u_{i,j-1,k} - 2u_{i,j,k} + u_{i,j+1,k}\right) \\ + \beta\left(u_{i,j,k-1} - 2u_{i,j,k} + u_{i,j,k+1}\right) - u_{i-1,j,k}\end{matrix} \tag{24}$$

El término final de esta expresión nos lleva a una situación particular al momento de calcular los resultados de la ecuación para un instante t=2 ya que se requiere de elementos en un tiempo i=-1 con el cual no contamos, por ello para el caso en donde calculamos los valores correspondientes $u(t_i,x,y)$ en i=1 utilizamos una expansión de Taylor para su aproximación, de ahí tenemos que para i=1

$$u_{i+1,j,k} = (2-\lambda-\beta)u_{i,j,k} + \frac{\lambda}{2}\left(u_{i,j-1,k} + u_{i,j+1,k}\right) + \frac{\beta}{2}\left(u_{i,j,k-1} + u_{i,j,k+1}\right) \tag{25}$$

Considerando $\Delta y = \Delta x$ =h, el criterio de estabilidad para el metoto es el siguiente:

$$1 \leq \Delta t \leq \sqrt{\frac{44h^2}{89\alpha}}$$

Teniendo todo esto en cuenta y con la experiencia del caso unidimensional, en el que implementamos el método en un arreglo de dos dimensiones, una dimensión representando los intervalos temporales y la otra a los intervalos de la variable espacial, donde el contenido eran las coordenadas de las respectivas coordenadas (x,t) , es de esperarse que el lector intuya que este método para dos dimensiones involucre un arreglo pero esta vez tridimensional, y efectivamente ese es el caso, se crea un arreglo tridimensional donde uno de los ejes representa la dimensión espacial y los otros dos ejes a las variables espaciales y el contenido del arreglo son las posiciones en *u* para sus correspondientes coordenadas (t,x,y).

Para t_i con i=1 hasta nt

Y_k \ x_j	$j = 1$	$j = 2$	$j = 3$	…	$j = nx$
$k = 1$	$u(t_1, x_1, y_1)$	$u(t_1, x_2, y_1)$	$u(t_1, x_3, y_1)$	…	$u(t_1, x_{nx}, y_1)$
$k = 2$	$u(t_1, x_1, y_2)$	$u(t_1, x_2, y_2)$	$u(t_1, x_3, y_2)$	…	$u(t_1, x_{nx}, y_2)$
…	…	…	…	…	…
$k = ny$	$u(t_1, x_1, y_{ny})$	$u(t_1, x_2, y_{ny})$	$u(t_1, x_3, y_{ny})$	…	$u(t_1, x_{nx}, y_{ny})$

.

Y_k \ x_j	$j = 1$	$j = 2$	$j = 3$	…	$j = nx$
$k = 1$	$u(t_{nt}, x_1, y_1)$	$u(t_{nt}, x_2, y_1)$	$u(t_{nt}, x_3, y_1)$	…	$u(t_{nt}, x_{nx}, y_1)$
$k = 2$	$u(t_{nt}, x_1, y_2)$	$u(t_{nt}, x_2, y_2)$	$u(t_{nt}, x_3, y_2)$	…	$u(_{nt}, x_{nx}, y_2)$
…	…	…	…	…	…
$k = ny$	$u(t_{nt}, x_1, y_{ny})$	$u(t_{nt}, x_2, y_{ny})$	$u(t_{nt}, x_3, y_{ny})$	…	$u(t_{nt}, x_{nx}, y_{ny})$

Al igual que en el caso unidimensional el metodo trabaja sobre los valores de frontera asi como con las condiciones inciales.

$Y(0,x,y) = f(x,y)$, $Y(t,0,y) = g(t,y)$

$Y(t,A,y) = s(t,y)$, $Y(t,x,0) = G(t,x)$, $Y(t,x,L) = S(t,x)$

Las cuales ocuparan los valores asignados en las posiciones *(1,Xj,Yk)* En nuestro arreglo tridimensional.

CÓDIGO

El código está compuesto básicamente en 3 pasos, los cuales son:

1-Generación de condiciones iniciales.

2-Llenado del arreglo para el número de intervalos requeridos.

2-Impresión de datos

1-Condiciones inciales

Primero se le pide al usuario ingresar las dimensiones en X y Y de la membrana, el número de intervalos, así como, el valor de los incrementos en las variables espaciales Δx y Δy, que en los casos a abordar serán iguales, para que posteriormente el programa asigne el valor del

incremento temporal Δt y de esta menera cumplir con las condiciones de convergencia del metodo donde la variable ht=Δt. Con los datos anteriores podemos calcular las dimensiones en X (ny) y en Y (ny) de nuestro arreglo tridimensional.

```
character*100::z
real*8,dimension(1000,200,200)::mat
write(*,*)"ingrese el numero de intervalos de tiempo"
read(*,*)intervalos


!para poder evealuar necesitamos las ocndiciones inicales
!suponiendo que tenemos una cuerda de longitudd
para llenar las condiciones iniciales de las cuales partiremos
prociedemos a iniciar las primeras condiciones de la matriz
write(*,*)"Programa encargado de resolver EDP hiperbolicas"
write(*,*)" de segundo orden "
or=

write(*,*) "ingrese el ancho en x"
read(*,*) A
write(*,*) "ingrese el largo en y"
read(*,*) L
write(*,*)"ingrese el valor de los incrementos en x y y"
read(*,*)h
ht=h/100.0
pendiente=(ht/h)**or

PRINT*,PENDIENTE
el numero de intervalos en lso que se partira la cuerda es grande
dependiendo del tamaño de la misma para tener mañor presision

r= A/h+1
nx=int(r)
r= L/h+1
ny=int(r)
```

Con la informacion obtenida se llena la matriz de ceros para que fortran no nos de calculos anormales en los siguientes pasos y continuamos con las condiciones de frontera que en este caso son designadas por varios ciclos for los cuales no tienen la mayor complejidad ya que en para los sistemas de nuestro interes las condiciones de frontera son 0. Es decir, nuestro sistema es una membrana rectangular cuyo perimetro esta restringido en el plano XY. Esto se puede expresar como:

$Y(t, A, y) = 0$ $Y(t, 0, y) = 0$ $Y(t, x, 0) = 0$ $Y(t, x, L) = 0$

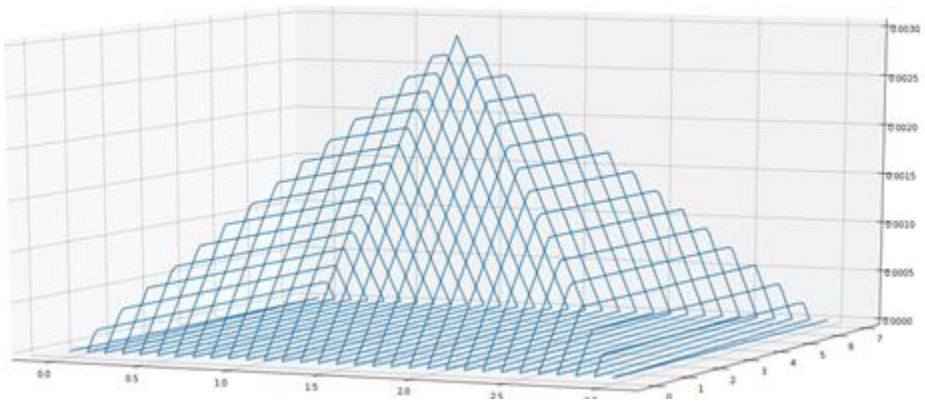
```
vamos a hacer I como el contador de T, J oara X y u para Y
do i=1,intervalos
do j=1,nx
do u=1,ny
mat(i,j,u)=0.0
end do
end do
end do

empezamos nuestras condiciones inicales----------------------
!condiciones para los valores de forntera de y
do i=1,intervalos
do j=1,nx
mat(i,j,1)=0.0
mat(i,j,ny)=0.0
end do
end do
!condiciones para los calores de frontera de x
do i=1,intervalos
do u=1,ny
mat(i,1,u)=0.0
mat(i,nx,u)=0.0
end do
end do
!condiciones para los calores de frontera para t=0
do j=1,nx
do u=1,ny
mat(1,j,u)=0.0
mat(1,j,u)=0.0
end do
end do
```

Una vez teniendo esto se establecen las condiciones correspondientes para t=0, creamos una subrutina llamada “Piramide” que como su nombre lo indica, crea una piramide, esta piramide tiene que tener una altura muy equeña comparada con las dimensiones de la membrana(no necesariamente por que el metodo valla a fallar, si no por que no es propio tener una piramide pronunciada para un sistema fisico como este) . La subrutina trabaja con las dimensiones de la base A y L como parametros de entrada y la colocacion de el pico de la piramide corX,corY,corZ , estas ultimas son establecidas en relacion a las otras anteriore, de la siguiente manera:

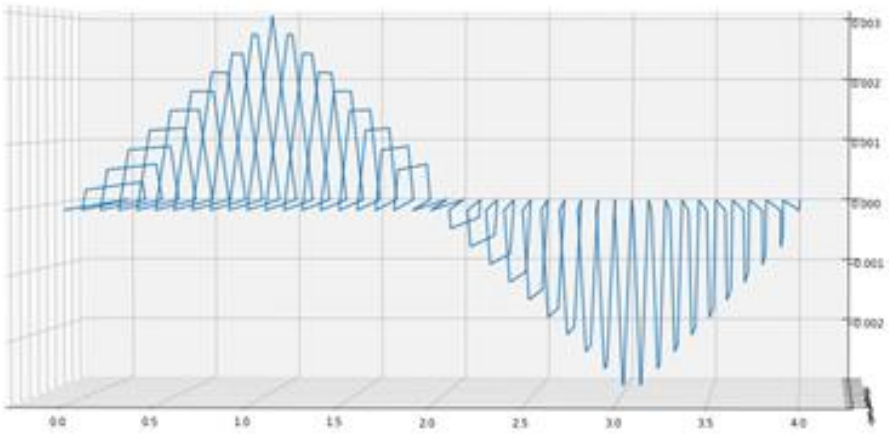
```
corX=A/2
corY=L/2
corZ=0.003
max=corZ

call Piramide(mat,nx,ny,A,L,corX,corY,corZ,h)
```

La subrutina funciona estableciendo ecuaciones para los 4 planos que conforman una piramide a partir de las coordenadas conocidas de sus esquinas y de la punta de la piramide, haciendo uso de la ecuacion para planos en 3d que se puede consultar en cualquier libro de calculo de varias variables aX+bY+cZ+d=0.De ahí se utilizan unos condicionales para saber a que plano corresponen ciertas coordenadas sobre la base de la membrana.Esta subrutina nos permite calcular basicamente cualquier tipo de piramide cuyas coordenads en X y Y esten dentro de los limites de las dimensiones de nuestra membrana.

```
subroutine Piramide(mat,nx,ny,A,L,corX,corY,corZ,h,matr)
real*8,dimension(1000,200,200)::mat
integer::i,j,u,nx,ny
real*8::a1,b1,c1,d1,a2,a3,a4,b2,b3,b4,c2,c3,c4,d2,d3,d4,Y3,h
real*8::corX,corY,corZ,A,L,x,y
a1=0.0
b1=corZ*A
c1=corY*A*(-1)
d1=(a1*corX+b1*corY+c1*corZ)*(-1)
a2=corZ*L*(-1)
b2=0.0
c2=(corX-A)*L
d2=(a2*corX+b2*corY+c2*corZ)*(-1)
a3=0.0
b3=corZ*A
c3=(-1)*A*(corY-L)
d3=(a3*corX+b3*corY+c3*corZ)*(-1)
a4=(corZ*L)*(-1)
b4=0.0
c4=corX*L
d4=(a4*corx+b4*corY+c4*corZ)*(-1)
y=h
do u=2,ny-1
x=h
do j=2,nx-1
Y3=(y-L)*(A-corX)/(L-corY)+A
if ((corX*y/corY).le.x.and.(((corX-A)/corY)*y+A).ge.x)then
mat(1,j,u)=(a1*x+b1*y+d1)*(-1)/c1
else if ((corX*y/corY).ge.x.and.((y-L)*corX/(corY-L)).ge.x)then
mat(1,j,u)=(a4*x+b4*y+d4)*(-1)/c4
else if (((y-L)*corX/(corY-L)).le.x.and.Y3.ge.x)then
mat(1,j,u)=(a3*x+b3*y+d3)*(-1)/c3
```



Una vez teniendo esta util subrutina se puede manipular el codigo para crear otras condiciones iniciales basadas tambien en piramides dentro de la membrana como la siguiente:



Solo por poner un ejemplo.

1- <u>Llenado de el arreglo para el numero de intervalos requeridos.</u>

En este paso solo hacemos uso de las ecuaciones planteadas anteriormente es decir

$$u_{i+1,j,k} = 2(1-\lambda-\beta)u_{i,j,k} + \lambda\left(u_{i,j-1,k} - 2u_{i,j,k} + u_{i,j+1,k}\right) + \beta\left(u_{i,j,k-1} - 2u_{i,j,k} + u_{i,j,k+1}\right) - u_{i-1,j,k}$$

La cual se mete dentro de una funcion llamada Fmult la cual tiene como parametros la el arreglo tridimencional “mat”,λ y β.

```
function Fmult(i,j,u,mat,c,c2)
    real*8,dimension(1000,200,200):: mat
    real*8::c,r,c2
    integer::i,j,u
    r=2.0*(1-c-c2)*mat(i,j,u)+c*(mat(i,j+1,u)+mat(i,j-1,u))
    Fmult=r+c2*(mat(i,j,u+1)+mat(i,j,u-1))-mat(i-1,j,u)
    end
```

Donde c2=β y c=λ. Los parametros i,j,u hacen referencia los indices de la matriz, ya que es llamada desde una serie de ciclos for en donde se hacen los calculos ya la vez se utiliza para la condicion i=1 la ecuación.

```
c2=2000.0*pendiente
!---------------------------------------
!---------------------------------------

k=1
do i=1,intervalos-1
do j=2,nx-1

do u=2,ny-1

if (i.eq.1)then
r=c*(mat(i,j+1,u)+mat(i,j-1,u))/2.0
r2=c2*(mat(i,j,u-1)+mat(i,j,u+1))/2.0
mat(i+1,j,u)=r+(2-c-c2)*mat(i,j-1,u)+r2
else
mat(i+1,j,u)=Fmult(i,j,u,mat,c,c2)
if (mat(i+1,j,u).gt.max)then
max=mat(i+1,j,u)
!go to 45
end if
end if


end do
end do
k=k+1
end do
```

Finalmente, los resultados son escritos en diferentes archivos de textos, con distintos formatos, según los usos que se les tenga que dar a cada 1.

```
open(18,file="archivo.txt")
x=0
do j=1,nx
y=0
do u=1,ny
write(18,*)x,y,mat(1,j,u)
y=y+h
end do
x=x+h
end do
close(18)
!-----------------------------
```

Imprime las coordendas para algun Ti en especifico

```
open(16,file="coor.xyz")

t=0
do i=1,intervalos
write(16,*)"  ",s
x=0.0
do j=1,nx
y=0.0
do u=1,ny


write(16,14)k,x,y,mat(i,j,u),22,k

y=y+h
k=k+1
if (k.gt.s)then
k=1
end if
end do
x=x+h
end do
t=t+ht
end do

close(16)
```

Imprime las coordendas en un formato para la herramienta “Pymol”

```
open(85,file="fotograma.txt")
do i=1,intervalos

x=0.0
do j=1,nx
y=0.0
do u=1,ny


write(85,*)x,y,mat(i,j,u)

y=y+h
end do
x=x+h
end do
write(85,*)"
write(85,*)"

end do
close(85)
```

Impresión para ser graficada por gnuplot

Tambien crea un archivo plt para graficar los resultados y uno mas para la creacion de un gif animado con gnuplot.

```
!-----------------------------------------------------
open(30,file="plotCoordenadas.plt")
write(30,*)"set terminal gif animate delay 2"
write(30,*)"set output ""cuerda.gif"""
write(30,*)"#set terminal x11"

write(30,*)"set ylabel ""Y"""
write(30,*)"set xlabel ""X"""
write(30,*)"set zlabel ""Z"""
write(30,*)"set xrange [0:",A,"]"
write(30,*)"set yrange [0:",L,"]"
write(30,*)"set zrange [",(-1.7*max),":",(1.7*max),"]"
write(30,*)"set title ""COMPORTAMIENDO DE ONDA"""
write(30,*)"set style data points"
write(30,*)"set surface"
write(30,*)"stats ""fotograma.txt"" name ""A"""


write(30,*)"file1=""fotograma.txt"""
z="for [i=0:int(A_blocks-1)]{splot file1 index i w p}"
write(30,*)"do ",z

close(30)
```

Archivo plt para gif.

```
open(43,file="ploteo.plt")
write(43,*)"#set terminal x11"
write(43,*)"set ylabel ""Y"""
write(43,*)"set xlabel ""X"""
write(43,*)"set zlabel ""Z"""
write(43,*)"set xrange [0:",A,"]"
write(43,*)"set yrange [0:",L,"]"
write(43,*)"set zrange [",(-1.7*max),":",(1.7*max),"]"
write(43,*)"set title ""COMPORTAMIENDO DE ONDA"""
write(43,*)"set style data points"
write(43,*)"set surface"
write(43,*)"set termoption dashed"

hivo gif de salida
write(43,*)"set isosamples 51"
write(43,*)"stats ""fotograma.txt"" name ""A"""


write(43,*)"file1=""fotograma.txt"""
z="for [i=0:int(A_blocks-1)]{splot file1 index i w p}"

write(43,*)"do ",z

close(43)
```

Script de archivo plt para graficar con gnuplot.

Resultados

Debido a que no encontramos ningun ejercicio similar para poder comparar nuestro programa, hicimos el analisis de su funcionamiento de acuerdo a las graficas genereadas,las cuales coinciden con el tipo de comportamiento que deberia observarse de un fenomeno de este tipo,los casos que se muestran son algunos son pocos. El usuario puede modificar con bastante libertad las condiciones iniciales asi como modificar las constantes de velocidad de propagacion de la onda tanto para x como para y, teniendo en cuenta siempre los criterios de convergencia y de ser posible un entendimiento general del metodo para su correcto aprovechamiento.

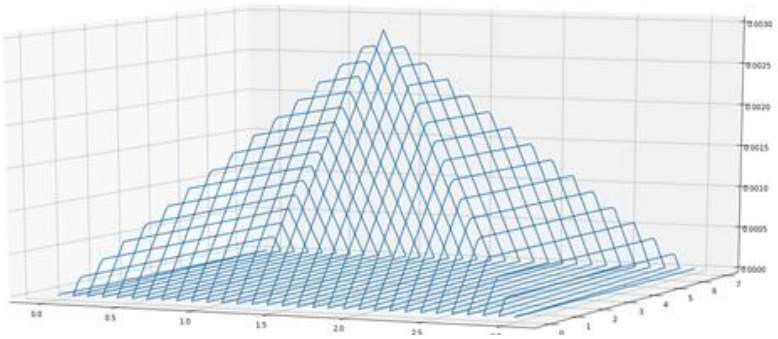
Los resultados siguientes muestran membranas con diferentes dimensiones y modos de vibracion que comparten las siguientes caracteristicas en comun:

α=2000, Δx=Δy=0.1 , Δt=0.001

Ejemplo 1. Membrana rectangular con un modo de vibración.

A=3 L=7

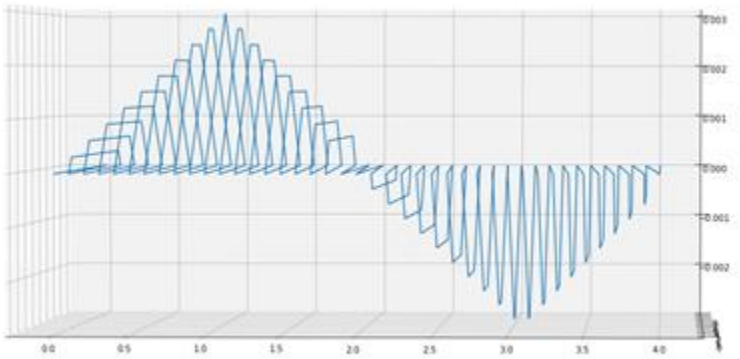
Condiciones iniciales: pirámide con pico en (A/2,L/2,0.003)



Ejemplo 2. Membrana rectangular con dos modos de vibración.

A=4 L=8

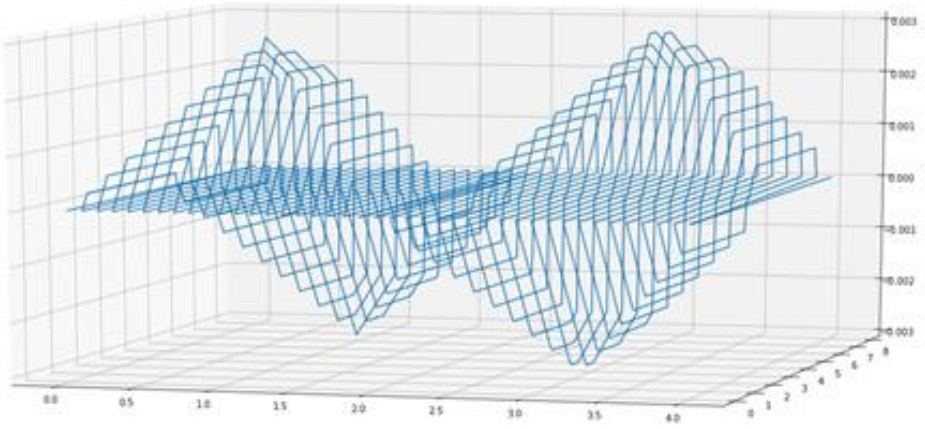
Condiciones iniciales: pirámides con picos en (A/4,L/2,0.003) y (3A/4,L/2,-0.003)



Ejemplo 3. Membrana rectangular con 4 modos de vibración.

A=4 L=8

Condiciones iniciales: 4 pirámides con picos en (A/4,L/4,0.003) , (3A/4,L/4,-0.003), (3A/4,3L/4,0.003) y (A/4,3L/4,-0.003).



Utilizando diferentes condiciones iniciales

Hasta ahora hemos analizado la vibración de una membrana utilizando condiciones iniciales en las cuales simulamos que la membrana está siendo agarrada de unos puntos específicos. Aunque los resultados sean bastante satisfactorios; debido a que, podemos observar los distintos modos de vibración utilizando pirámides como condiciones iniciales, es pertinente que se haga uso de las funciones trigonométricas para las condiciones iniciales, con el fin de comparar los resultados numéricos con los analíticos.

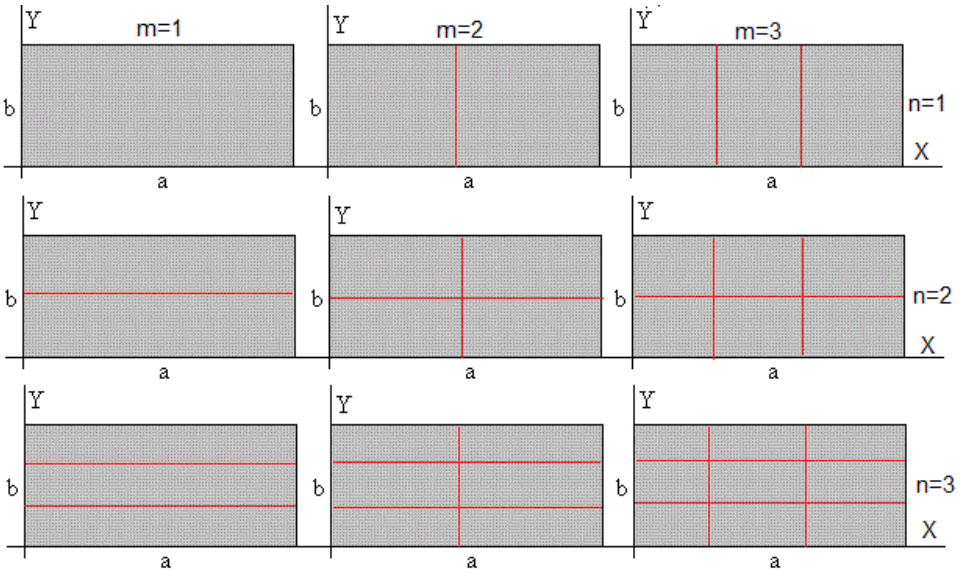
Angel Franco García en su artículo *Modos de vibración de una membrana rectangular* publicado en 2016, menciona que la solución a la ecuación de onda en dos dimensiones está dada de la siguiente manera:

$$u(x,y,t) = \sin\left(\frac{m\pi}{a}x\right)\sin\left(\frac{n\pi}{b}y\right)(A_{mn}\cos(\omega_{mn}\mathrm{t}) + B_{mn}\sin(\omega_{mn}\mathrm{t})) \qquad (25)$$

Esta solución corresponde al modo de vibración m,n. Donde $a = ancho(x), b = largo(y)$. $A_{mn}, B_{mn}: constante$. La frecuencia ω_{mn} la vamos a poder expresar de la siguiente forma:

$$\omega_{mn}^2 = c^2\left(\frac{m^2\pi^2}{a^2} + \frac{n^2\pi^2}{b^2}\right)$$

Los modos de vibración m,n de la membrana rectangular los podemos visualizar de la siguiente manera:



Crédito a Ángel Franco García

Entonces lo que se puede hacer es utilizar la ecuación (25) para obtener condiciones iniciales. Obsérvese que, al principio, es decir en $t = 0$, la ecuación (25) quedaría:

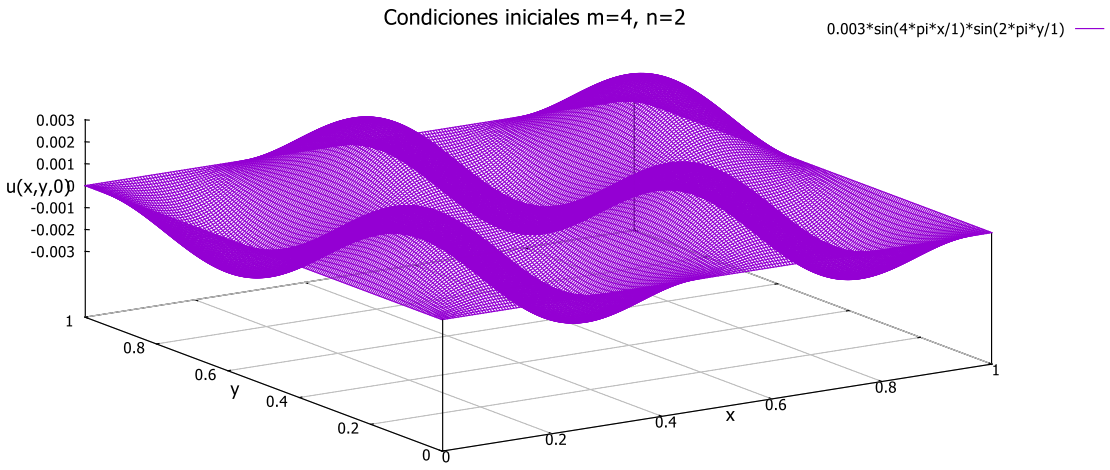
$$u(x, y, 0) = A_{mn} \sin\left(\frac{m\pi}{a}x\right) \sin\left(\frac{n\pi}{b}y\right) \tag{26}$$

Entonces la ecuación (26) será la ecuación para utilizar como condiciones iniciales dentro de dos ciclos anidados (uno desde $x = 0$ a $x = a$, y el otro desde $y = 0$ a $y = b$). En este caso la constante A_{mn} sería la amplitud de la "primer onda" de la membrana. Nótese que, utilizando esta ecuación seremos capaces de controlar con mayor facilidad la forma y cantidad de modos normales de vibración de la membrana.

```
function Fsum(x,y,t,NODX,NODY,w,A,L)
real*8::x,y,t,pi,w,A,L,Fsum
integer::NODX,NODY
pi=acos(-1.0)
Fsum=0.003*(sin(NODX*pi*x/A)*sin(NODY*pi*y/L))*cos(w*t)
END
```

En la imagen podemos ver la parte del código que se modificó (agregando la ecuación 26). Ahora se guardan las condiciones iniciales en una función Fsum, en vez de hacer las pirámides. Obsérvese que *NODX* = *m*, *NODY* = *n*.

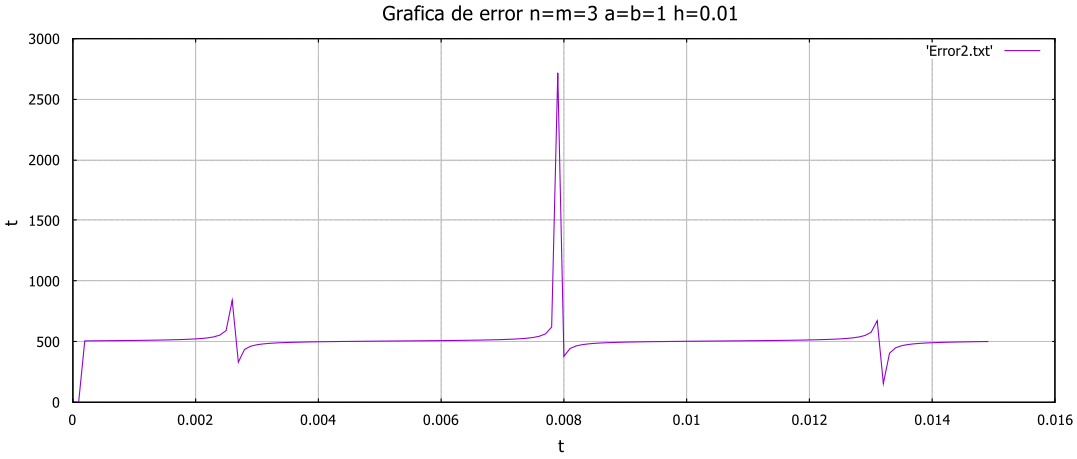
Para $a = b = 1$ y $m = 4, n = 2$. Tendríamos las condiciones iniciales siguientes



Error entre analítico y numérico

Para n=m=3 y un delta x de 0.1 se obtiene la siguiente gráfica de error:

Con los mismos valores para n y m pero con un valor de delta x igual a 0.01



Conclusión

Se logró resolver la ecuación de la onda por medio de ambos métodos, explícito e implícito. Además de lograr con éxito la visualización del comportamiento ondulatorio de la cuerda en una dimensión por medio de gif animados. Se logró con éxito resolver de manera numérica la ecuación de onda de dos dimensiones utilizando esquemas explícitos, y se pudo comparar los resultados analíticos con los numéricos. Como se pudo observar en las gráficas de error, es pertinente aumentar disminuir el delta x con el fin de aumentar la precisión de los cálculos, aunque esto conlleva a un aumento en el tiempo de carga del programa. En general se cumplieron los objetivos propuestos al inicio de este proyecto y los programas de resultado serán de gran utilidad para proyectos futuros.

Referencias

Cortés, J., González, M., & Pinilla, V. (2019). Solución numérica de ecuaciones diferenciales parciales. *Plataforma educativa para análisis numérico*.

Nakamura, S. (1992). *Métodos Numéricos Aplicados con Software.* Atlacomulco: Pearson.

Resolución de la ecuación de difusión en 2-D y 3-D utilizando diferencias finitas generalizadas. Consistencia y Estabilidad.

Fowler, M. (2020). A Vibrating Membrane. *Chemistry. LibreTexts.* Recuperado el 04 de diciembre de 2020 de: https://chem.libretexts.org/Courses/University_of_California_Davis/UCD_Chem_110A%3A_Physical_Chemistry__I/UCD_Chem_110A%3A_Physical_Chemistry_I_(Larsen)/Text/02%3A_The_Classical_Wave_Equation/2.05%3A_A_Vibrating_Membrane

González, P. Solución Numérica de Ecuaciones en Derivadas Parciales. *Universidad de Granada*. Recuperado el 05 de diciembre de 2020 de: https://www.ugr.es/~prodelas/ftp/ETSICCP/Resoluci%F3nNum%E9ricaEDPs.pdf

García, A. (2016). *Modos de vibración de una membrana rectangular*.