Implementación de la Cuerda Vibrante con Python
Segundo borrrador

Juan Pablo Romero Bernal*
Andrés Armando Quintero

Correo-e: jromerobernal@gmail.com

21 de Noviembre de 2006

Resumen. El presente documento constituye un informe preliminar acerca de la implementación del fenómeno de la cuerda vibrante usando el lenguaje Python junto con el módulo VPython. Se presenta un esbozo de las capacidades de la aplicación, así como algunas de sus limitaciones y proyecciones a futuro. Adicionalmente se presentan de manera detallada los procedimientos realizados tanto desde el punto de vista técnico como metodológico y al final algunas conclusiones acerca de este trabajo.

Palabras clave: Series de Fourier, Python, VPython, Modo normal

---

1. Introducción

El fenómeno de la cuerda vibrante tiene varias particularidades, respecto al aprendizaje en Ingeniería, aunque sea un problema muy sencillo y al igual que muchos otros problemas relaciona la física con la matemática de una manera muy especial. En particular, pensamos que sería una buena introducción desde el punto de vista práctico hacia el conocimiento de las Series de Fourier y sus componentes básicos (frecuencia, periódo, etc,.), para luego retomar un curso mucho más teórico. Por qué puede ser una introducción a las Series de Fourier ?. En primer lugar, es un problema que involucra la solución de una ecuación diferencial parcial, muy conocida (ecuación de onda unidimensional) que es:

$$\frac{\partial^2 u}{\partial t^2} = a^2 \frac{\partial^2 u}{\partial x^2} \tag{1}$$

en donde el procedimiento para su solución es dominado en su mayoría por estudiantes de 5 Semestre de Ingeniería y el cual conduce de forma casi automática al estudio de las Series de Fourier, debido a que una de las funciones solución[1] de (1) está dada en términos de una de ellas; de hecho:

$$u(x,t) = \sum_{n=1}^{\infty} \operatorname{sen}\left(\frac{n\,\pi\,x}{l}\right)\left[E_n \cos\left(\frac{n\,\pi\,c\,t}{l}\right) + F_n \operatorname{sen}\left(\frac{n\,\pi\,c\,t}{l}\right)\right] \tag{2}$$

que corresponde a una Serie de Fourier[2]. A partir de este punto, se puede iniciar el estudio de los temas relacionados con la expansión de funciones en series de Fourier, contando con un fundamento práctico y físico, que creemos le será de utilidad al estudiante. Sin embargo, ese conocimiento no puede quedarse en le esfera teórica. Al contrario debe ser probado y analizado, para que el proceso de apropiación de las ideas sea mucho más claro y atrayente. Por ello, pensando en eso y teniendo como antecedente teórico lo anteriormente descrito, se ha desarrollado una aplicación informática, que permite en primera instancia observar el fenómeno físico y apreciar el papel de las series de Fourier. En las secciones siguientes se hará una descripción generalizada de las funcionalidades de la herramienta.

∗. Integrantes del proyecto de investigación en Series de Fourier dirigido por la profesora Isabel Amaya y el profesor Rafael Garzón.

1. Esto debido a que hay otras funciones que solucionan la ecuación diferencial, por ejemplo véase [1], pág 102, solución de D'Alembert.

2. Una explicación detallada de la solución de la ecuación diferencial la encuentra usted en [2].

2. GENERALIDADES

La aplicación permite simular el movimiento de la cuerda, para una cantidad de estímulos limitados, dado que el establecimiento de un algoritmo que permita realizar una simulación bajo cualquier deformación inicial aún se encuentra en desarrollo y requiere de mucho más estudio para que su implementación permita obtener resultados con coherencia teórica. Sin embargo los cuatro estímulos que soporta la aplicación actualmente son los siguientes (dando lugar a una serie de estímulos derivados):

1. Deformación Tringular

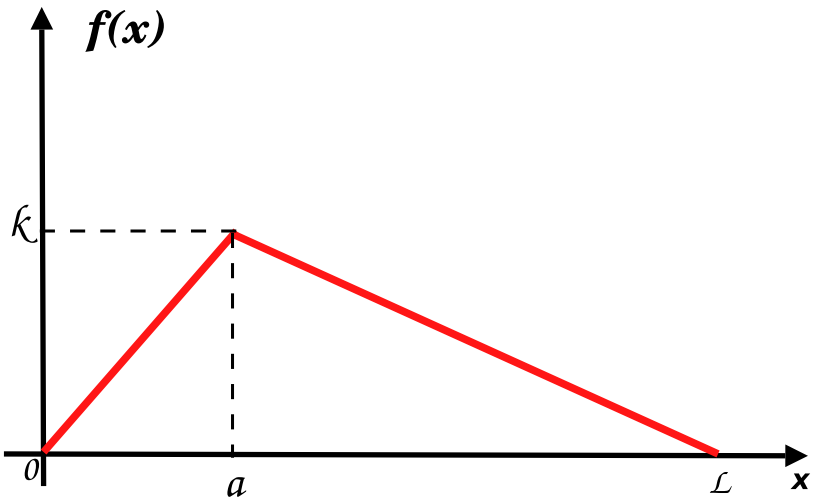

Figura 1. Deformación triangular como condición inicial de la cuerda vibrante

$$u(x,0) = f(x) = \begin{cases} \frac{k}{a}\,x & \text{para } 0 < x < a \\ \\ \frac{k}{l-a}\,(l-x) & \text{para } a < x < l \end{cases} \quad \text{y} \quad \left.\frac{\partial u(x,t)}{\partial t}\right|_{t=0} = 0$$

donde $f(x)$ es la función que define la deformación inicial en el intervalo de $[0, l]$ en el instante $t = 0$ y con velocidad inicial cero. En otras palabras, $f(x)$ define las condiciones iniciales para resolver la ecuación (1), con las condiciones de frontera de extremos fijos:

$$u(0,t) = 0 \quad \text{y} \quad u(l,t) = 0 \tag{3}$$

que serán aplicables a las demás deformaciones. Así la función que determina la posición de la cuerda en cualquier tiempo t es:[3]

$$u(x,t) = \frac{2\,k\,l^2}{\pi^2\,a\,(l-a)} \sum_{n=1}^{\infty} \frac{1}{n^2} \operatorname{sen}\left(\frac{n\,\pi\,a}{l}\right) \operatorname{sen}\left(\frac{n\,\pi\,x}{l}\right) \cos\left(\frac{n\,\pi\,c\,t}{l}\right)$$

3. Los procedimientos de cálculo para llegar a la solución de la ecuación diferencial, se adjuntarán a este documento cuando su versión definitiva sea publicada.

con:

$$E_n = \frac{2\,k\,l^2}{\pi^2\,a\,(l-a)\,n^2}\,\mathrm{sen}\left(\frac{n\,\pi\,a}{l}\right)$$

$$F_n = 0$$

2. Deformación ZigZag:

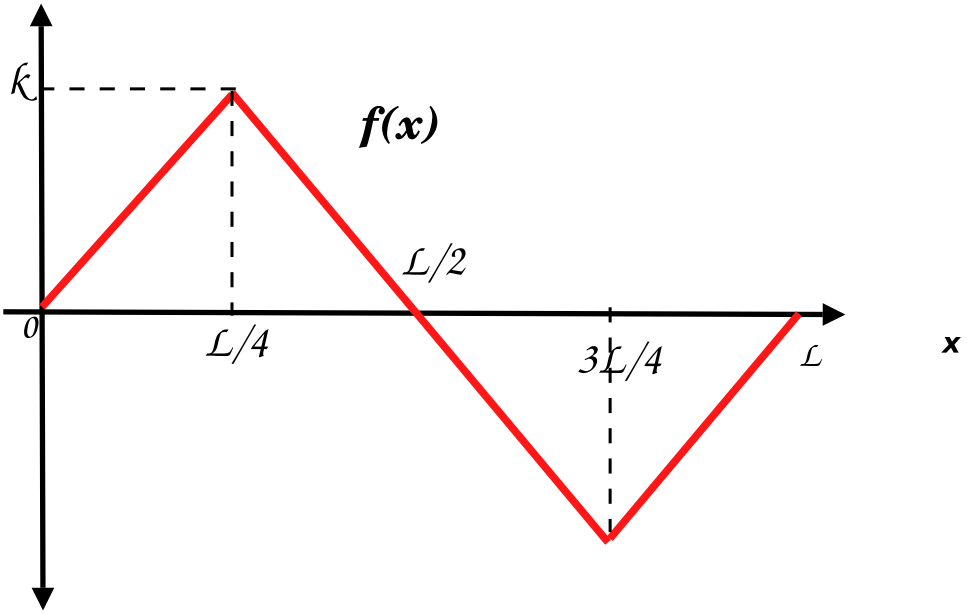

Figura 2. Deformación de zigzag como condición inicial de la cuerda vibrante

$$u(x,0) = f(x) = \begin{cases} \frac{4\,k}{l}\,x & \text{para } 0 < x < \frac{l}{4} \\ \frac{-4\,k}{l} + 2\,k & \text{para } \frac{l}{4} < x < \frac{3\,l}{4} \\ \frac{4\,k}{l}\,x - 4\;k & \text{para } \frac{3\,l}{4} < x < l \end{cases} \qquad \text{y} \qquad \left.\frac{\partial u(x,t)}{\partial t}\right|_{t=0} = 0$$

Para esta deformación, la función que determina la posición de la cuerda en cualquier instante t es:

$$u(x,t) = \frac{16\,k}{\pi^2\,n^2}\sum_{n=1}^{\infty}\mathrm{sen}\left(\frac{n\,\pi\,x}{l}\right)\left[\mathrm{sen}\left(\frac{\pi\,n}{4}\right) - \mathrm{sen}\left(\frac{3\,\pi\,n}{4}\right)\right]\cos\left(\frac{c\,n\,\pi\,t}{l}\right)$$

$$E_n = \frac{16\,k}{\pi^2\,n^2}\left[\mathrm{sen}\left(\frac{\pi\,n}{4}\right) - \mathrm{sen}\left(\frac{3\,\pi\,n}{4}\right)\right]$$

$$F_n = 0$$

3. Deformación Trapezoidal:

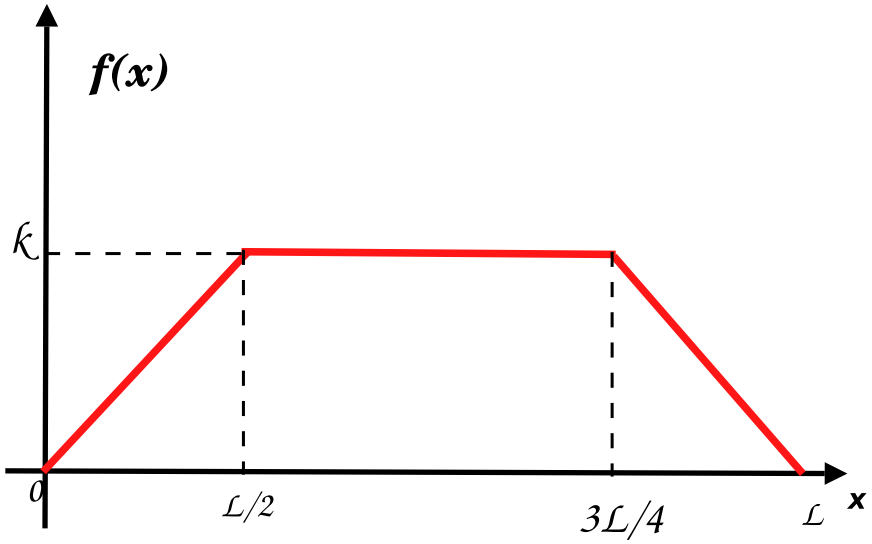

Figura 3. Deformación de trapecio como condición inicial de la cuerda vibrante

$$u(x,0)=f(x)=\begin{cases}\frac{4\,k}{l}\,x & \text{para } 0<x<\frac{l}{4}\\ k & \text{para } \frac{l}{4}<z<\frac{3\,l}{4}\\ \frac{-4\,k}{l}\,x+4\,k & \text{para } \frac{3\,l}{4}<x<l\end{cases} \quad \text{y} \quad \left.\frac{\partial u(x,t)}{\partial t}\right|_{t=0}=0$$

donde:

$$u(x,t)=\frac{8\,k}{\pi^2\,n^2}\sum_{n=1}^{\infty}\text{sen}\left(\frac{n\,\pi\,x}{l}\right)\cos\left(\frac{c\,n\,\pi\,t}{l}\right)\left[\text{sen}\left(\frac{3\,\pi\,n}{4}\right)+\text{sen}\left(\frac{\pi\,n}{4}\right)\right]$$

$$E_n=\frac{8\,k}{\pi^2\,n^2}\sum_{n=1}^{\infty}\left[\text{sen}\left(\frac{3\,\pi\,n}{4}\right)+\text{sen}\left(\frac{\pi\,n}{4}\right)\right]$$

$$F_n=0$$

Para cada una de estas deformaciones el usuario proporciona los valores de las variables relacionadas, por ejemplo corrimiento en el eje x en la deformación triangular.

3. Proceso de simulación

Cuando el usuario, ha elegido la deformación que quiere observar, junto con los parámetros necesarios, los cuales puede asignar de forma interactiva, se inicia el proceso de simulación en donde interviene varios factores: la cuerda, el espectro de frecuencia discreta, los modos normales, la aproximación en serie de Fourier al movimiento de la cuerda y la superposición de los modos normales o términos de la serie. Estos factores son claves en el proceso debido a que son los que el usuario pueden usar como elementos fundamentales para la comprensión de la teoría relacionada. A continuación una toma de pantalla de la aplicación:

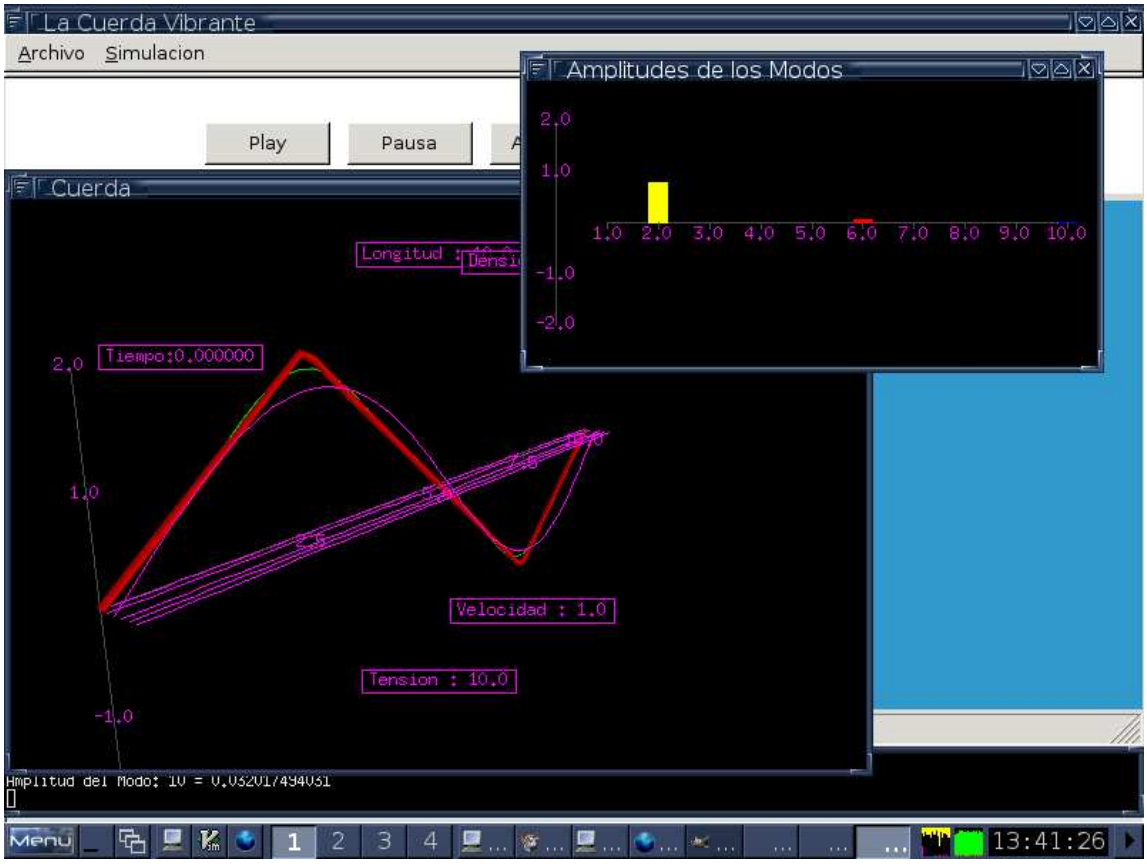

Figura 4. Captura de una simulación con deformación de zigzag

En la imagen se puede apreciar la cuerda vibrante (color rojo), la suma de los modos normales (color verde), que en este caso es hasta el sexto término, pues el modo seleccionado en la ventana Amplitud de Modos (color rojo) y los modos 1,2,3,4, y 5, en color violeta. Cuando el usuario da click en el botón Play, la simulación se iniciará y podrá observar la vibración de la cuerda junto con los modos normales, además de poder seleccionar en cualquier momento el modo que desee.

4. Implementación

La implementación de la aplicación se ha elaborado, pensando en que pueda ser extensible y que sea fácil su modificación para otro tipo de deformaciones. Por ejemplo, para cada deformación se crea una clase que define la función solución ($u(x,t)$) de forma genérica (dentro del módulo ecu), por ejemplo:

```
from visual import *
from visual.graph import *

class EcDefTriangular:
    def __init__(self, cuerda, amplitud, a):
        self.cuerda=cuerda
        self.k=amplitud
        self.l=cuerda.l
        self.a=a
        self.c1=1
        self.c2=1
        self.c1=2*self.k*self.l**2/(pi**2*self.a*(self.l-self.a))
        self.c2=pi*self.a/self.l
```

```
    def En(self, n):
        return self.c1/n**2*sin(self.c2*n)


    def Fn(self, n):
        return 0
```

implementa la deformación triangular (compárese con la función descrita más arriba). Adicionalmente se debe implementar otra clase sobre el módulo DIALOG, que define los aspectos básicos de la simulación, el cálculo del buffer (función **getBuffer**) que almacena los datos y mejora el rendimiento de la aplicación y las funciones asociadas para la captura de la deformación por parte del usuario (función **tratarMouse**). Aquí va un ejemplo para la deformación triangular:

```
from ecu import *
from util import *
from ventanas import *
from operator import mod
from visual.graph import *

class VDTriangular(Ventana):
    def __init__(self, main, timeInvoker, cuerda, psX, psT, numMods):
        self.cuerda=cuerda
        self.psX=psX
        self.psT=psT
        self.numMods=numMods
        Ventana.__init__(self, main, timeInvoker, Plano3D('Deformacion Triangular', 600,
400, (self.cuerda.l/2, 0., 0.)))
        self.plano.setEjeX(0., self.cuerda.l, 5)
        self.plano.setEjeY(-2., 2., 5)
        self.plano.setEjeZ(0., 0., 0)
        self.plano.setXYMalla(21, 9)
        self.plano.setRejilla(0.5, 0.5, 0.5)

        self.curva=curve(x=[0, self.cuerda.l], y=[0, 0], radius=0.05)

    def getBuffer(self):
        b=BufferMuestreo(EcuCuerda(EcDefTriangular(self.cuerda, self.curva.y[1],
self.curva.x[1])), self.psX, self.psT, self.numMods)
        b.calcular()
        return b

    def tratarMouse(self):
        if self.scene.mouse.clicked:
            pos=self.scene.mouse.project(normal=(0, 0, 1))
            print "Punto Elegido X: " + str(pos.x)
            print "Punto Elegido Y: " + str(pos.y)
            if pos.x>0.1 and pos.x<self.cuerda.l and abs(pos.y) <= 2:
                pos=self.plano.getPuntoRej(pos)
                self.curva.x=[0, pos[0], self.cuerda.l]
                self.curva.y=[0, pos[1], 0]

    def tratarTeclado(self):
        return
```

Los aspectos restantes están escritos de forma genérica y son aplicables a cualquier deformación sin necesidad de reescribir ninguna parte de la aplicación. Es de aclrara que se ha usado para la implementación el lenguaje Python junto con el módulo VPython (Visual Python), que ha permitido un desarrollo rápido y sencillo. De hecho se puede obervar que la implementación de la función $u(x, t)$ es casi como escribir seudo-código.

5. Proyecciones

A futuro esperamos implementar algunas otras funcionalidades extras, además de presentar nuevos perfiles para la aplicación, entre las que se encuentran:

- Desarrollo de una algortimo que permita simular en forma eficiente y coherente cualquier deformación de la cuerda.
- Integrar un módulo que permita el desarrollo en series de Fourier de una gran cantidad de funciones, en principio soportando alguna familia de funciones, que posteriormente de manera progresiva irá creciendo.
- Implementación de otrs fenómenos físicos relacionados como la membrana vibrante.

6. Pruebas

Para las pruebas de la aplicación se ha usado Maxima[4] como soporte para la elaboración de cálculos y demás operaciones. Básicamente se han generado listas de datos y se han comparado con las emitidas por el aplicativo, respecto a amplitudes de los modos, aproximación por suma de los términos de la serie de Fourier. A continuación el código elaborado para hacer las pruebas:

```
triangular(t,k,l,a,p):=block(
        [f:2*k*l^2/(%pi^2*a*(l-a)),
         g:1/n^2,
         h:sin(n*%pi*x/l)*sin(n*%pi*a/l)*cos(n*%pi*c*t/l),
         En:f*g*sin(n*%pi*a/l)],
         display2d:false,
         Total:sum(f*g*h,n,1,50),
         sumas:makelist(sum(f*g*h,n,1,i),i,1,p),
         modos:makelist(ev(f*g*h),n,1,p),
         espectros:makelist(abs(ev((f*g*sin(n*%pi*a/l))/2)),n,1,p))$

zigzag(t,k,l,p):=block(
        [pr:16*k/(%pi^2*n^2),
         h:sin(n*%pi/4)-sin(3*n*%pi/4),
         g:sin(n*%pi*x/l)*cos(c*n*%pi*t/l)],
         display2d:false,
         Total:sum(pr*h*g,n,1,50),
         sumas:makelist(sum(pr*h*g,n,1,i),i,1,p),
         modos:makelist(ev(pr*h*g),n,1,p),
         espectrosz:makelist(abs(ev(pr*h/2)),n,1,p))$

trapecio(t,k,l,p):=block(
        [pr:8*k/(%pi^2*n^2),
         h:sin(n*%pi/4)+sin(3*n*%pi/4),
         g:sin(n*%pi*x/l)*cos(c*n*%pi*t/l)],
         display2d:false,
```

4. http://maxima.sourceforge.net

```
Total:sum(pr*h*g,n,1,50),
sumas:makelist(sum(pr*h*g,n,1,i),i,1,p),
modos:makelist(ev(pr*h*g),n,1,p),
espectrosz:makelist(abs(ev(pr*h/2)),n,1,p))$
```

Se han creado tres funciones que representan los tres estímulos y éstas generan las listas anteriomente mencionadas. Por ejemplo, fue de mucha utilidad utilizar MAXIMA para generación de gráficos:

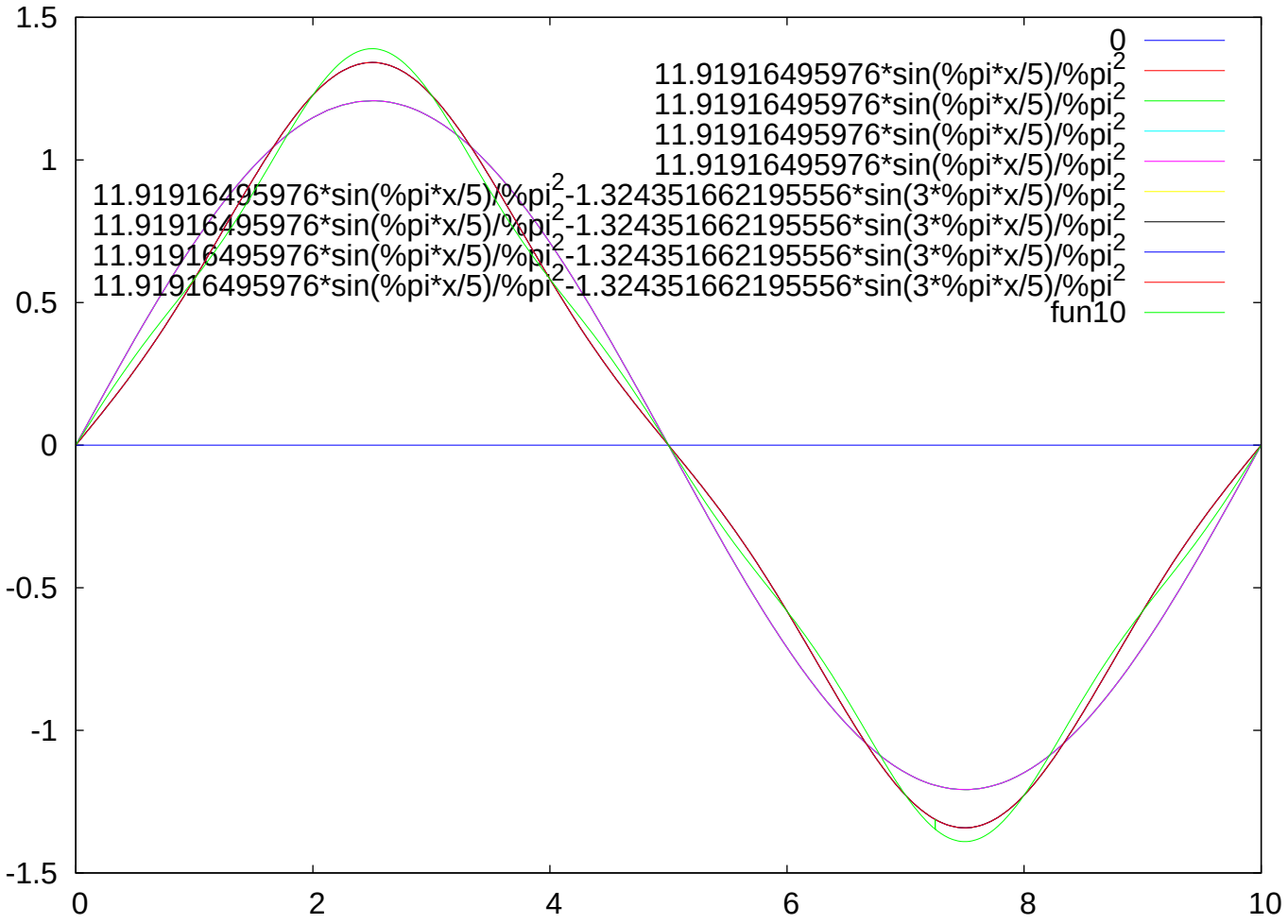

Figura 5. Aproximación generada por MAXIMA

Lo anterior nos sirvió de guia para corroborar los resultados generados por la aplicación.

7. BIBLIOGRAFÍA

[1] KREIZIG, ERWIN., *Matemáticas Avanzadas para Ingeniería*, Ed. Limusa
[2] HWEIP, HSU., *Análisis de Fourier*, Ed. Addison-Wesley
[3] FRENCH, A.P., *Vibraciones y Ondas*, Ed. Reverté
[4] LUTZ, MARK., *Programming Python*, Ed. O'Reilly
[5] RAPPIN, NOEL & DUNN, ROBIN, *wxPython in Action*, Ed. Manning.