# Tercer Parcial
# Herramientas computacionales

Universidad Nacional de Colombia

Departamento de Física

Pedro José Leal Mesa [*]

**Resumen**

*En el presente documento se demuestra la solución de 3 Ecuaciones diferenciales parciales por medio del método de diferencias finitas. Además, se encuentra el volumen de una hyper-elipse para n- dimensiones con n de 1 a 9 y comparándolo con los valores teóricos descritos. Todos los cálculos fueron desarrollados en Octave.*

## Índice



[*] plealm@unal.edu.co

# 1. Primer Punto

*Resuelva la siguiente ecuación diferencial parcial, usando diferencias finitas:*

$$\frac{\partial u}{\partial t} + D\frac{\partial^2 u}{\partial t^2} + a\frac{\partial u}{\partial x} = c^2\frac{\partial^2 u}{\partial x^2}$$

*Donde D, c y a son constantes.*

El método de diferencias finitas consiste en una aproximación de los diferenciales por medio de la definición de limites del diferencial. Esta aproximación ha sido comúnmente usada para la resolución de problemas complejos. Para entender más clara mente este método se refiere a las aproximaciones del diferencial de primer orden como:

$$\frac{\partial u(x,t)}{\partial t} = \frac{u(i,j+1) - u(i,j)}{\Delta t}, \tag{1}$$

tal que, como la derivada es temporal y esta se encuentra en la segunda componente, se realiza el método dejando la coordenada espacial sin cambios, pero la coordenada temporal con la aproximación. Para el caso de una derivada espacial de segundo orden:

$$\frac{\partial^2 u(x,t)}{\partial x^2} = \frac{u(i+1,j) - 2u(i,j) + u(i-1,j)}{(\Delta x)^2}. \tag{2}$$

Al ser una derivada espacial se deja constante la parte temporal. Para resolver las ecuaciones diferenciales presentadas en los puntos 1, 2 y 3 se usará el formalismo de la ecuación 1 y 2.

Habiendo explicado el formalismo de las diferencias finitas, este es aplicado a la ecuación de interés encontrando una serie de recurrencia como:

$$u(i,j+1) = \left(\frac{\Delta t^2}{\Delta t + D}\right)\left[\frac{u(i,j)}{\Delta t} + D\frac{2u(i,j) - u(i,j-1)}{\Delta t^2} + a\frac{u(i,j) - u(i+1,j)}{\Delta x} + c^2\frac{u(i+1,j) - 2u(i,j) + u(i-1,j)}{\Delta x^2}\right].$$

Tomando esta serie como base, para poder solucionar esta ecuación diferencial es necesario tener unas condiciones de frontera, las cuales se han tomado como:

$$u(i,1) = cos(x(i)^2 * \pi - 10) \qquad u(i,N) = cos(10x(i)^4/\pi).$$

Finalmente, se definen los valores de los parámetros a usar siendo:

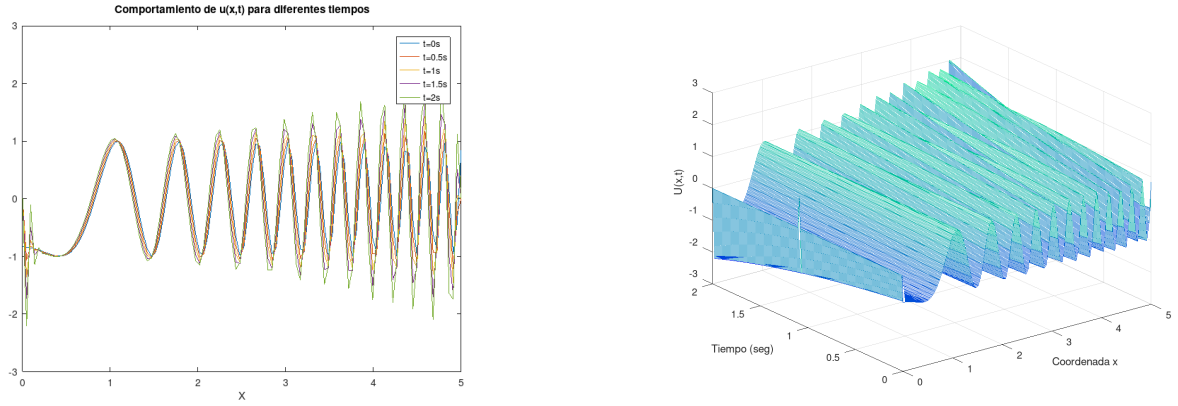
$$D = 5e-5 \qquad a = 9e-6 \qquad c = 4e-3;$$

Con todas estas especificaciones fue creado el siguiente código que resuelve la ecuación diferencial de segundo orden.

```
% Tercer Parcial
% Primer Punto
clear all; close all;clc

L = 5; N = 110;
dt = 1e-2;t = 0:dt:2;
x = linspace(0,L,N);dx = x(2)-x(1);

D = 5e-5;a = 9e-6;c = 4e-3;

```

```
u = zeros(length(x),length(t));

for i = 1:N
  u(i,1) = cos(x(i)^2*pi -10);
  u(i,N) = cos(10*x(i)^4/pi);
endfor

for j = 1:length(t)-1
    for i = 2:length(x)-1
        u(i,j+1) = (dt^2/(dt + D))*(u(i,j)/dt + D*(2*u(i,j) - u(i-1,j))/dt^2 +
        a*(u(i,j) - u(i+1, j))/dx + c^2*(u(i+1,j) -2*u(i,j) + u(i-1, j))/dx^2);
    endfor
endfor


figure(1); clf;
plot(x,u(:,1),x,u(:,ceil(0.5/dt)),x,u(:,ceil(1/dt)),x,u(:,ceil(1.5/dt)), x,u(:,ceil(2/dt)))
legend("t=0s","t=0.5s","t=1s","t=1.5s", "t=2s");
title('Comportamiento de u(x,t) para diferentes tiempos'); xlabel('X');
print -dpng -r100 u_t.png

figure(2);clf;
[tt,xx] = meshgrid(t,x);
mesh(xx,tt,u); colormap winter;
xlabel('Coordenada x')
ylabel('Tiempo (seg)')
zlabel('U(x,t)')
print -dpng solucion.png
```

los resultados de este código, se observan el la imagen a continuación:




((a)) Comportamiento de u(x, y) para diferentes tiempos.

((b)) Superficie solución de u(x, y).

Figura 1: Solución de la función u(x,y).

Aunque en la gráfica de la superficie de u(x, t), pareciera ser estacionaria en el tiempo en la gráfica de la izquierda demuestra el incremento en el tiempo que se espera de la solución de esta ecuación de onda tanto temporal como espacialmente, pero por el orden de los coeficientes la principal contribución a la ecuación diferencial, viene dado por la derivada de primer orden que denota una pendiente. Por lo tanto, para las condiciones asumidas se encuentra la solución oscilatoria esperada.

## 2. Segundo Punto

*Resuelva el siguiente conjunto de ecuaciones diferenciales parciales, usando el método de diferencias finitas:*

$$\frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2} = a_1 u + b_1 w$$
$$\frac{\partial^2 w(x,y)}{\partial x^2} + \frac{\partial^2 w(x,y)}{\partial y^2} = a_2 u + b_2 w$$

$a_1, a_2, b_1$ y $b_2$ *son constantes.*

El sistema de ecuaciones diferenciales presentadas puede ser desacoplado [3] usando un cambio de variables, pero no es necesario para encontrar una solución numérica, aunque colabora para el desarrollo y por tanto se tendrá en cuenta; por ello se realiza el método de diferencias finitas (ecuaciones 1 y 2), realizando el álgebra correspondiente, se llegan a las expresiones recurrentes:

$$u(i,j) = \frac{1}{4}\left(u(i+1,j) + u(i,j+1) + u(i,j-1) - h^2(a_1 u(i,j) + b_1 w(i,j))\right)$$
$$w(i,j) = \frac{1}{4}\left(w(i+1,j) + w(i,j+1) + w(i,j-1) - h^2(a_2 u(i,j) + b_2 w(i,j))\right).$$

De tal forma es evidente que se trata de una solución de una ecuación del tipo possion, pero este tipo de ecuación diferencial solo se diferencia de una de laplace porque se encuentra igualada a una función en vez de 0, por ello tomando los parámetros igual a 0 se llega a las expresiones vistas en clase. Las series de recurrencia son expresadas por medio de funciones dependiente de los parámetros y las variables.

$$f_1(u,w) = h^2(a_1 u(i,j) + b_1 w(i,j))$$
$$f_2(u,w) = h^2(a_2 u(i,j) + b_2 w(i,j)).$$

Las condiciones de contorno para este problema son:

$$u(1,i) = 20y(i)^5 \qquad W(1,i) = cos(12y(i)/\pi - 50)$$
$$u(N,i) = cos(20y(i)/\pi) \qquad W(N,i) = cos(20y(i)/\pi)$$
$$u(i,1) = 10x(i)^2 \qquad W(i,1) = cos(2x(i)/\pi - 2)$$
$$u(i,N) = 2(5x(i)^5 - 3) \qquad W(i,N) = cos(20x(i)/\pi)$$

Los valores de los parámetros para la resolución de la ecuación tipo poisson son:

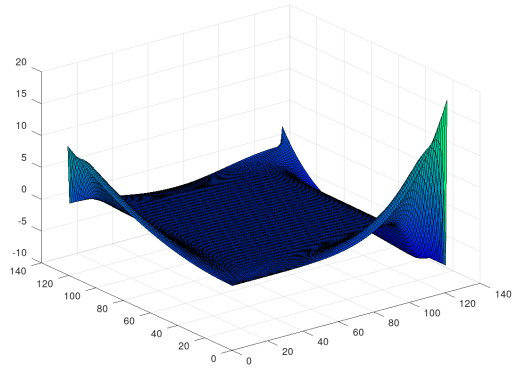
$$a_1 = 7{,}5 \quad b_1 = 1250 \quad a_2 = 2500 \quad b_2 = 450$$

como se observa en el siguiente código:

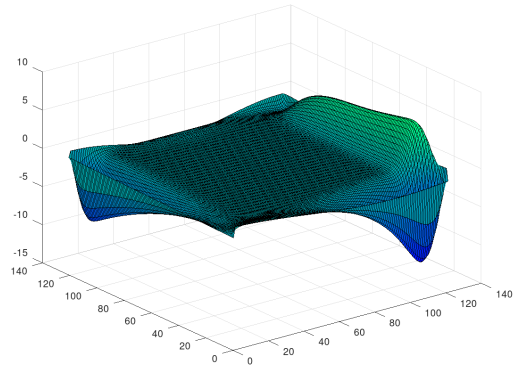
```
% Tercer Parcial
% 2 Punto
% Pedro Jose Leal

clear all; close all; clc; tic
%%%%%%%%%%%%%%%% Condiciones del m todo %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
L = 120; N = L + 2; h = 1/(L+1);
a1 = 7.5; b1 = 1250; a2 = 2500; b2 = 450;
x=0:h:1; y=0:h:1;
% Definici n del las matrices U y W
U(N,N)=zeros; UNew(N,N)=zeros;
W(N,N)=zeros; WNew(N,N)=zeros;
```

```
%%%%%%%%%%%%%%%% Condiciones de frontera %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i=1:N
  %lado izquierdo
  UNew(1,i) = 20*y(i)^5;
  WNew(1,i) = cos(12*y(i)/pi - 50);
  %lado derecho
  UNew(N,i) = cos(20*y(i)/pi);
  WNew(N,i) = cos(20*y(i)/pi);
  %abajo
  UNew(i,1) = 10*x(i)^2;
  WNew(i,1) = cos(2*x(i)/pi - 2);
  %arriba
  UNew(i,N) = 2*(5*x(i)^5 - 3);
  WNew(i,N) = cos(20*x(i)/pi);
endfor

steps = 0; accuracy = 0.1;
%%%%%%%%%%%%%%%% Diferencias Finitas %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
while true
  U=UNew; W=WNew;
  for i=2:N-1;
    for j=2:N-1;
      UNew(i,j)=0.25*(U(i-1,j)+U(i+1,j)+U(i,j-1)+U(i,j+1)-h^2*(a1*U(i,j)+b1*W(i,j)));
      WNew(i,j)=0.25*(W(i-1,j)+W(i+1,j)+W(i,j-1)+W(i,j+1)-h^2*(a2*U(i,j)+b2*W(i,j)));
    endfor
  endfor
  steps=steps+1;
  % Condici n del error para mejor resultado
  error1=0; n=0; k=0; error2=0;
  for i=1:L+1
    for j=1:N-1
      if (UNew(i,j)~=0) && (UNew(i,j)~=U(i,j)) && (WNew(i,j)~=0) && (WNew(i,j)~=W(i,j))
        error1=error1+abs(1-U(i,j)/UNew(i,j));
        error2=error2+abs(1-U(i,j)/UNew(i,j));
        n=n+1; k=k+1;
      endif
    endfor
  endfor
%% Bucle del erro
  if n~=0
    error1=error1/n; error2=error2/n;
  endif
%% Condici n para parar el while
  if(error1<accuracy) && (error2<accuracy)
    break
  endif
endwhile

%%%%%%%%%%%%%%%% Graficaci n de los resultados %%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Soluci n para U
figure(1); clf; surf(U);colormap winter;
print -dpng U_poisson.png;
% Soluci n para W
figure(2); clf; surf(W);colormap winter;
print -dpng W_poisson.png;toc
```

Las gráficas resultado se muestran a continuación:



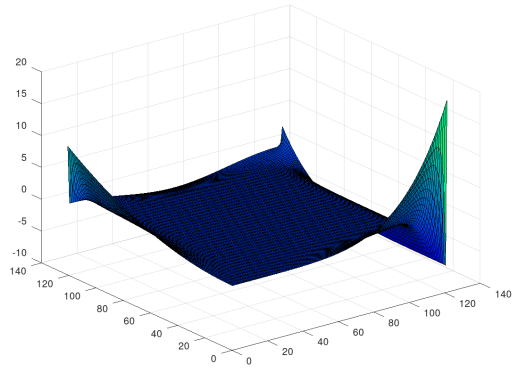

((a)) u(x, y)



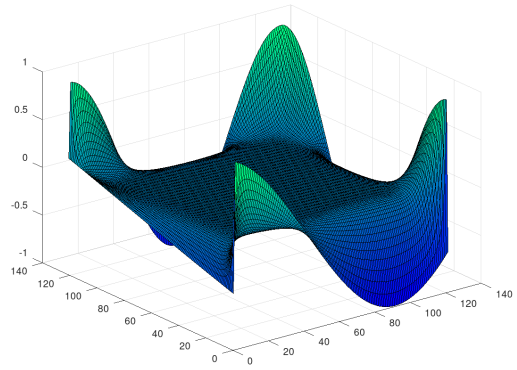

((b)) w(x, y)

Figura 2: Solución de las funciones u(x,y) y w(x,y) para la ecuación de poisson.

En el caso de que se trate como una ecuación de Laplace, es decir $a_1 = a_2 = b_1 = b_2 = 0$, se observa el correspondiente comportamiento




((a)) u(x, y)




((b)) w(x, y)

Figura 3: Solución de las funciones u(x,y) y w(x,y) para la ecuación de laplace.

Se observa que es ligero el cambio en la función u(x,y), esto se debe a las condiciones de frontera adoptadas. Por otra parte la función w(x, y) demuestra un cambio abrupto como era de esperarse dado que se están resolviendo dos diferentes sistemas de ecuaciones. Pero para el entorno central se encuentra una congruencia entre ambos sistemas de ecuaciones lo que hace fiable la implementación del mpetodo de Diferencias Finitas.

# 3. Tercer Punto

*Resuelva la siguiente ecuación diferencial parcial, usando los métodos desarrollados en clase:*

$$i\frac{\partial u}{\partial t} + \frac{\partial^2 u}{\partial y^2} + a_1 w = 0$$
$$i\frac{\partial w}{\partial t} + \frac{\partial^2 w}{\partial y^2} + a_2 w = 0$$

*Donde i es el número complejo, y a 1 , a 2 , son constantes.*

Teniendo presente el análisis encontrado en [2] y las definiciones del método de diferencias finitas (ecuaciones 1 y 2, se pueden re expresar las anteriores expresiones como:

$$u(i,j+1) = -a_1 w(i,j)\frac{\Delta t}{i} - \frac{\Delta t}{i\Delta y^2}(u(i+1,j) - 2u(i,j) + u(i-1,j)) + u(i,j)$$
$$w(i,j+1) = -a_2 u(i,j)\frac{\Delta t}{i} - \frac{\Delta t}{i\Delta y^2}(w(i+1,j) - 2w(i,j) + w(i-1,j)) + w(i,j),$$

Además de tomar como condiciones temporales

$$u(i,1) = cos(y(i)) \qquad w(i,1) = e^{-y^2}.$$

Por ultimo, definiendo el valor de los parámetros definidos:

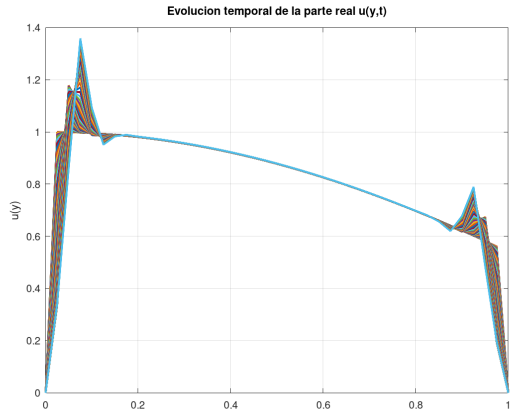
$$a1 = 5 \qquad a2 = 7i.$$

Con todas las condiciones expuestas anteriormente, se propone el siguiente código solución para las partes reales e imaginarias de ambas funciones.

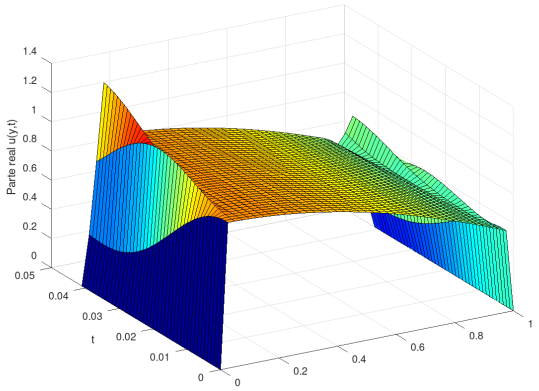
```
% Tercer Parcial
% Punto 3
clear all; close all; clc;
numy=41;numt=41;dy=1/(numy-1);dt=0.001;y=0:dy:1;
iimag = sqrt(-1);
a1 = 5; a2 = 7*iimag;

u=zeros(numy,numt);
w=zeros(numy,numt);

u(1,1)=0; u(numy,1)=0;
w(1,1)=0; w(numy,1)=0;

for i=2:numy-1;
u(i,1)= cos(y(i)); %Condicion temporal
w(i,1)= exp(-y(i)^2); %Condicion temporal
endfor
t(1) = 0;

for j=1:numt
t(j+1)=t(j)+dt;
for i=2:numy-1
u(i,j+1)=u(i,j)-((dt/iimag)*(a1*w(i,j) + (u(i+1,j) - 2*u(i,j) + u(i-1,j))/dy));
w(i,j+1)=w(i,j)-((dt/iimag)*(a2*u(i,j) + (w(i+1,j) - 2*w(i,j) + w(i-1,j))/dy));
endfor
endfor
%%%%%%%%
figure(1);clf;
for k=1:numt
plot(y,real(u(:,k)),'linewidth',2);
grid on; hold on; title('Evolucion temporal de la parte real u(y,t)');
ylabel('y');ylabel('u(y)');
drawnow
endfor
print -dpng ureal.png;
figure(2);clf;
for k=1:numt
```

```
plot(y,real(w(:,k)),'linewidth',2);
grid on; hold on; title('Evolucion temporal de la parte real w(y,t)');
ylabel('y');ylabel('w(y)');
drawnow
endfor
print -dpng wreal.png;
figure(3);clf;
for k=1:numt
plot(y,imag(u(:,k)),'linewidth',2);
grid on; hold on; title('Evolucion temporal de la parte imaginaria u(y,t)');
ylabel('y');ylabel('u(y)');
drawnow
endfor
print -dpng uimg.png;
figure(4);clf;
for k=1:numt
plot(y,imag(w(:,k)),'linewidth',2);
grid on; hold on; title('Evolucion temporal de la parte imaginaria w(y,t)');
ylabel('y');ylabel('w(y)');
drawnow
endfor
print -dpng wimg.png;
figure(5);clf;
[Y,T]=meshgrid(y,t);
colormap jet;
surf(Y,T,real(u'));ylabel('y');ylabel('t');
zlabel('Parte real u(y,t)'); view(330,30);
print -dpng ureal2.png;
figure(6);clf;
colormap jet;
surf(Y,T,real(w'));ylabel('y');ylabel('t');
zlabel('Parte real w(y,t)'); view(330,30);
print -dpng wreal2.png;
figure(7);clf;
colormap winter;
surf(Y,T,imag(u'));ylabel('y');ylabel('t');
zlabel('Parte imaginaria de u(y,t)'); view(330,30);
print -dpng uimg2.png;
figure(8);clf;
colormap winter;
surf(Y,T,imag(w'));ylabel('y');ylabel('t');
zlabel('Parte imaginaria de w(y,t)'); view(330,30);
print -dpng wimg2.png;
```

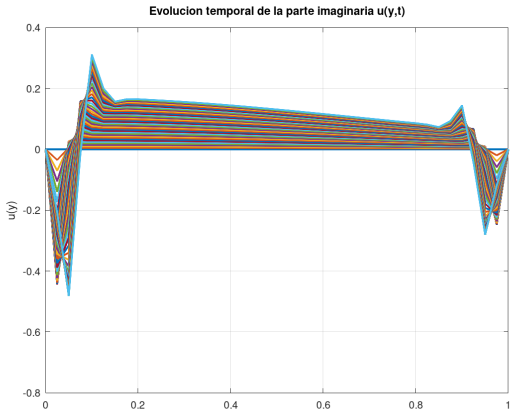
Los resultados del presente código se encuentran en las 8 diferentes imágenes que se encuentran debajo.



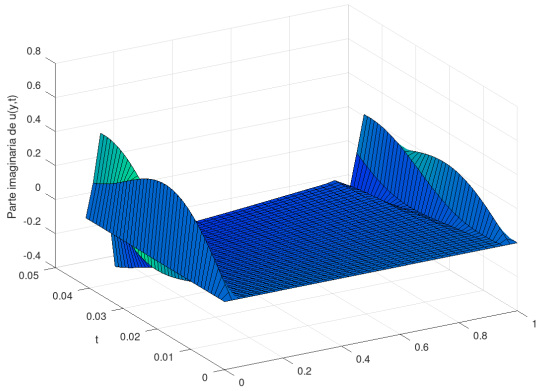

((a)) Parte real de u(y, t).




((b)) Parte real de u(y, t) superficie.



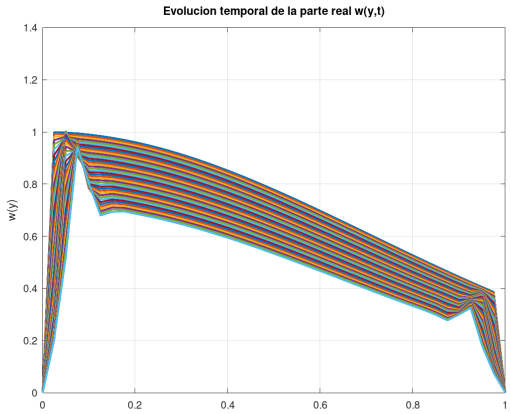

((c)) Parte imaginaria de u(y, t).



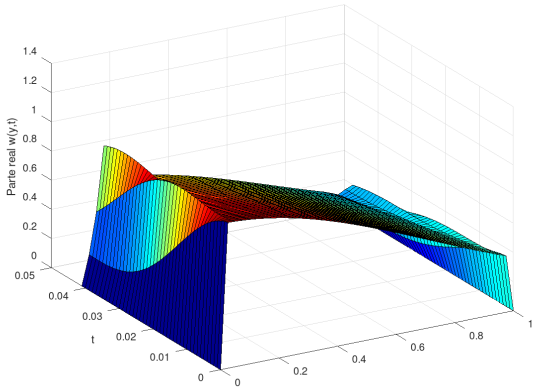

((d)) Parte imaginaria de u(y, t) superficie.

Figura 4: Evolución de u(y,t) de forma real e imaginaria.

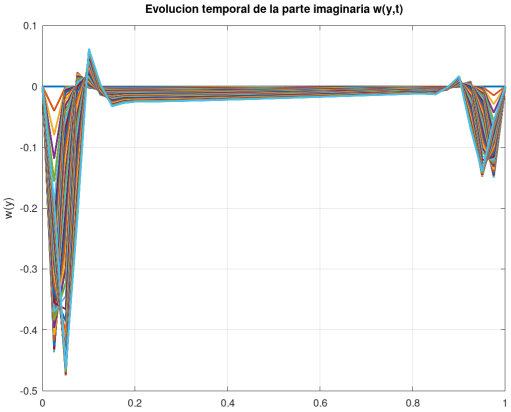
Se puede observar que la parte imaginaria de u(y,t) es bastante significativa y no es posible despreciarla, donde es mas apreciable en la superficie imaginaria. Posteriormente se observa los resultados para w(y,t).



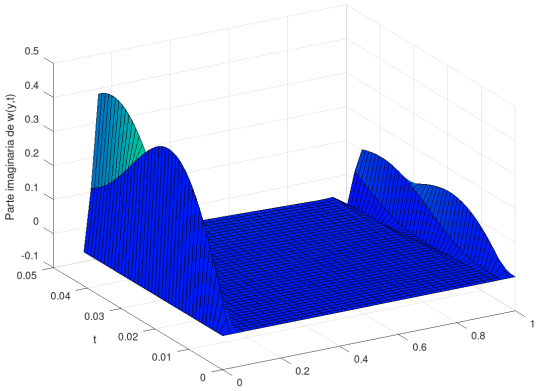

((a)) Parte real de w(y, t).




((b)) Parte real de w(y, t) superficie.




((c)) Parte imaginaria de w(y, t).




((d)) Parte imaginaria de w(y, t) superficie.

Figura 5: Evolución de w(y,t) de forma real e imaginaria.

Se observa que las soluciones poseen comportamientos similares, esto puede ser explicado por sus condiciones temporales puestas al principio, lo que corresponde a una similitudes en la forma de la solución. De igual manera de encuentra importancia en la parte imaginaria. Por ultimo, es importante notar la simetría existente en las soluciones lo que da fiabilidad en estas.

# 4. Cuarto Punto

*Usando el algoritmo MonteCarlo desarrollado en clase, calcule el volumen de un elipsoide en 2,3,4,5,6,7,8,9 dimensiones. Para comparar sus resultados de simulación, use la siguiente expresión analıtica para el volumen de un elipsoide en n-dimensiones:*

$$V_n = a_1 a_2 a_3 \dots a_n \frac{\pi^{\frac{\pi}{2}}}{\Gamma\left(\frac{n}{2}+1\right)}$$

El método MonteCarlo es un proceso estadístico que colabora a realizar aproximaciones de expresiones matemáticas complejas por medio del uso de números pseudo-aleatorios dados por el lenguaje de computación que a diferencia

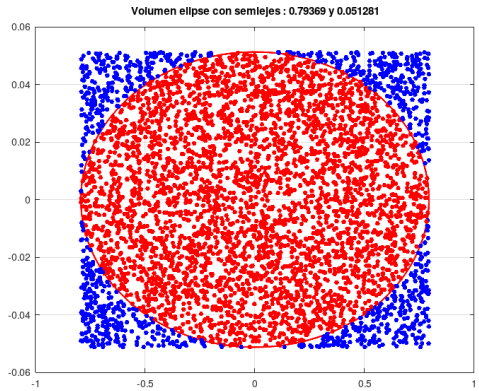
de los métodos numéricos usuales, este evalúa N puntos en un espacio M-dimensional para producir una solución.

Por ello se desarrollo un algoritmo que obtiene el valor del volumen de una hyper-elipsoide de n-dimensiones, con comparación de su valor teórico esperado [1]:

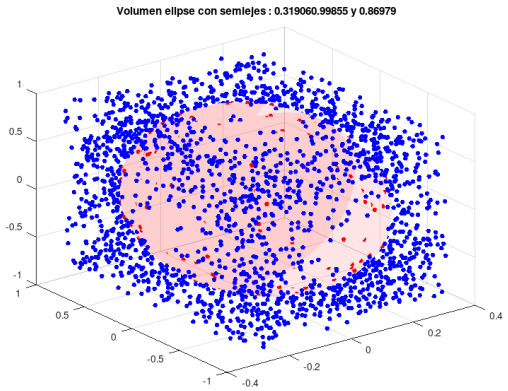
```
% Tercer Parcial
% 4 Punto N merica

clear all; close all; clc;

N = 5e3;
d = input('Dimensi n a analizar: ');
inside = 0;

for i =1:d
  a(1,i) = rand;
endfor

if d == 2
  t = linspace(0,2*pi);
  X = a(1)*cos(t); Y = a(2)*sin(t);
  figure(1); clf;
  plot(X, Y, '-r', 'LineWidth', 2); hold on;
  fn = fullfile("Montecarlo2.mp4");
  w = VideoWriter(fn);
  open(w);
  for jj = 1:N
    x = 2*a(1)*rand - a(1); y = 2*a(2)*rand - a(2);
    R = sqrt((x/a(1))^2 + (y/a(2))^2);
    if R <= 1
      inside += 1;
      titulo = ['Volumen elipse con semiejes : ', num2str(a(1)), ' y ', ...
   num2str(a(2))];
   title(titulo);
      plot(x, y, 'r.', 'Markersize', 15); grid on; drawnow
     else
      plot(x, y, 'b.', 'Markersize', 15); grid on; drawnow
    endif
    writeVideo (w, getframe (gcf));
  endfor
  close(w)
  print -dpng -r100 MonteCarlo2.png ;
endif

if d == 3
  [X, Y, Z] = ellipsoid(0, 0, 0,a(1), a(2), a(3));
  figure(1); clf;
  surf(X, Y, Z,'FaceColor','r', 'FaceAlpha', .1, 'EdgeAlpha', .3);
  hold on;
  fn = fullfile("Montecarlo3.mp4");
  w = VideoWriter(fn);
  open(w);
  for kk = 1:N
    x = 2*a(1)*rand - a(1); y = 2*a(2)*rand - a(2);
    z = 2*a(3)*rand - a(3);
    R = sqrt((x/a(1))^2 + (y/a(2))^2 + (z/a(3))^2);
    if R <= 1
      inside += 1;
      titulo = ['Volumen elipse con semiejes : ', num2str(a(1)) , num2str(a(2)), ' y ', ...
   num2str(a(3))];
   title(titulo);
      plot3(x, y, z, 'r.', 'Markersize', 15); grid on; drawnow
     else
      plot3(x, y, z, 'b.', 'Markersize', 15); grid on; drawnow
    endif
    writeVideo (w, getframe (gcf));
```

```
  endfor
  close(w)
  print -dpng -r100 MonteCarlo3.png ;
endif

if d >=4 || d == 1
for j = 1:N
  x = rand(1, d);
  R = 0;
  for k = 1:d
    R += ((2*a(k)*x(k)-a(k))/(a(k)))^2;
  endfor
  r = sqrt(R);
  if r <= 1
    inside += 1;
  endif
endfor
endif

an = 1;
for ii = 1:d
  an *= a(ii);
endfor
format long
V = 2^d*an*(inside/N);
Vexact = an*pi^(d/2)*(1/(gamma((d/2)+1)));
error = 100*(abs(V- Vexact)/Vexact);
ejes = ['Los semiejes usados fueron :' , num2str(a)];
vol =[ 'Volumen calculado por Montecarlo: ' , num2str(V)];
vol1 = [ 'Volumen te rico: ' , num2str(Vexact)];
vol2 = ['El error porcentual es: ', num2str(error)];

disp('')
disp(ejes)
disp(vol)
disp(vol1)
disp(vol2)
```

Es importante notar que para que el programa funcione debidamente en Octave es importante usar la librería vídeo. Los resultados se muestran en las siguientes figuras para dimensión 2 y 3, ademas para estas figuras se exporto el vídeo del MonteCarlo que será anexado con este documento pdf.




((a)) Dimensión n = 2.




((b)) Dimensión n = 3.

Figura 6: Volumen de una elipse (dimensión 2) y una elipsoide (dimensión 3) usando el método Montecarlo, donde los puntos rojos son aquellos que caen dentro y los azules, los que caen afuera.

Como se observa, para tener mejores resultados es necesario aumentar el numero de puntos N, pero para ello se recomienda optimizar el código ya que con $1e5$ puntos con los que se realizó el análisis tarda alrededor de 2 minuto y medio. Los resultados para cada dimensión se muestran a continuación:

| **Dimensiones** | **Párametros** | **Volumen MonteCarlo** | **Volumen Teorico** | **Error porcentual [%]** |
|---|---|---|---|---|
| 1 | 0.28029 | 0.56058 | 0.56058 | 1.9805e-14 |
| 2 | 0.79369<br>0.051281 | 0.12748 | 0.12787 | 0.30534 |
| 3 | 0.31906<br>0.99855<br>0.86979 | 1.1528 | 1.1608 | 0.68732 |
| 4 | 0.73658<br>0.28602<br>0.43317<br>0.24599 | 0.10761 | 0.11078 | 2.8614 |
| 5 | 0.5103<br>0.11004<br>0.54433<br>0.80574<br>0.66925 | 0.088399 | 0.08676 | 1.8886 |
| 6 | 0.96263<br>0.59303<br>0.36625<br>0.25268<br>0.44914<br>0.28004 | 0.033936 | 0.034338 | 1.171 |
| 7 | 0.44327<br>0.25237<br>0.50708<br>0.64395<br>0.46285<br>0.42107<br>0.99251 | 0.031656 | 0.033385 | 5.1805 |
| 8 | 0.71887<br>0.99549<br>0.59139<br>0.46015<br>0.0013624<br>0.10522<br>0.89553<br>0.70752 | 6.8827e-05 | 7.179e-05 | 4.1272 |
| 9 | 0.40856<br>0.48764<br>0.42502<br>0.15237<br>0.55867<br>0.40803<br>0.33731<br>0.12048<br>0.46632 | 0.00019975 | 0.00018384 | 8.6552 |

Tabla 1: Resultados del uso del programa solución para el punto 4.

Es posible concluir que el error porcentual depende principalmente de los valores tomados por los semi-ejes, aunque posee tambien dependencia sobre la dimensión manejada, por ello para 1 dimensión es increíblemente pequeño. Al ser así depende de dimensión y de parámetros.

# 5. Conclusiones

Los resultados obtenidos fueron resultados de la implementación del método de diferencias finitas para la resolución de ecuaciones diferenciales parciales, este tipo de ecuaciones abundan en la naturaleza y permiten entender desde una perspectiva física-matemática el universo que nos rodea, pero por desgracia la dificultad de resolver estas de manera analítica es bastante alto, por ello este método es una piedra fundamental para cualquier físico. Los códigos desarrollados fueron implementados en Octave encontrado resultados coherentes para los diferentes sistemas de ecuaciones diferenciales que dependen de las condiciones de frontera, condiciones iniciales y los parámetros escogidos.

El método MonteCarlo es un método usado en bastantes ramas de la física, pero es tan útil que incluso se usa como simulación en centro de investigación de altas partículas como CERN (ATLAS), por su increíble utilidad y maleabilidad a diferentes sistemas complejos de matemáticas es otra gran herramienta. La implementación de este método en el ultimo punto fue excelente aunque se podría optimizar el algoritmo para mejorar de esta manera su eficacia, además de definir ciertos semi-ejes que ayuden a entender la dependencia de las dimensiones y los semi-ejes con respecto al volumen de las hiper-elipses.

# Referencias


[1] *1420-1415594291.pdf*. https://oaji.net/articles/2014/1420-1415594291.pdf. (Accessed on 02/05/2022).

[2] *Belmonte_etal.pdf*. https://www.ugr.es/~ecuadif/files/Belmonte_etal.pdf. (Accessed on 02/07/2022).

[3] *spde0104.pdf*. http://eqworld.ipmnet.ru/en/solutions/syspde/spde0104.pdf. (Accessed on 02/06/2022).