Universidad Galileo
FISICC

Matemática VI

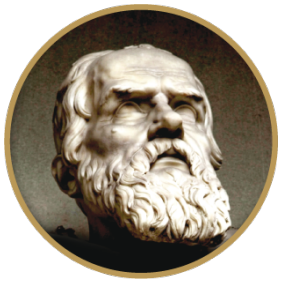
Sección "A"

# Ecuación de Calor

---

*Integrantes:* *Carné:*
Dong Ju Baek 15011370
Carlos Montiel 15000552

Guatemala, octubre del 2017



Galileo UNIVERSIDAD
La Revolución en la Educación

# Índice

# 1 Introducción

El calor fluye de caliente a frío, para una barra delgada el calor fluiría al lado que tenga menos temperatura, el balance tiene la forma,

$$q(x)\Delta y\Delta z\Delta t - q(x+\Delta x)\Delta y\Delta z\Delta t = \Delta x\Delta y\Delta z r C\Delta T$$

dividiendo entre el volumen del elemento y $\Delta t$,

$$\frac{q(x)-q(x+\Delta x)}{\Delta x} = \rho C\frac{\delta T}{\delta t}$$

entonces,

$$\frac{-\partial q}{\partial x} = \rho C\frac{\partial T}{\partial t}$$

Existen varios casos de dimensiones para la ecuación de calor, Fourier sugiere que la razón de cambio del flujo de energía calórica por unidad de área a través de una superficie es proporcional al gradiente de temperatura negativo en la superficie,

$$q = -k\,\nabla u$$

donde $k$ es la conductividad térmica y $u$ es la temperatura. En una dimensión, el gradiente es una derivada espacial ordinaria, y así la ley de Fourier es,

$$q = -k\frac{\partial u}{\partial x}$$

EL cual puede reescribirse como

$$u_t = \frac{k}{c_p\rho}u_{xx}$$

O bien,

$$\frac{\partial U}{\partial t} = k\frac{\partial U^2}{\partial^2 x}$$

Donde esta ecuación de calor pertenece a la familia de ecuaciones parabólicas, para resolver ecuaciones parabólicas se resuelve sustituyendo las derivadas parciales por diferencias finitas, donde existen cambios tanto en el tiempo como en el espacio como concideración para cumplir que la temperatura $U(x,t)$.
Existen dos métodos de solución que son los esquemas explícitos y los implicitos.

# 2 Difusión 1D Crank-Nicolson

El principio máximo es una de las propiedades características básicas de las soluciones de ecuaciones diferenciales parciales de segundo orden de tipos parabólicos (y elípticos). La preservación de esta propiedad para soluciones de problemas discretizados correspondientes es un requisito muy natural en el modelado numérico confiable y significativo de varios fenómenos de la vida real (conducción de calor, contaminación del aire, etc.).

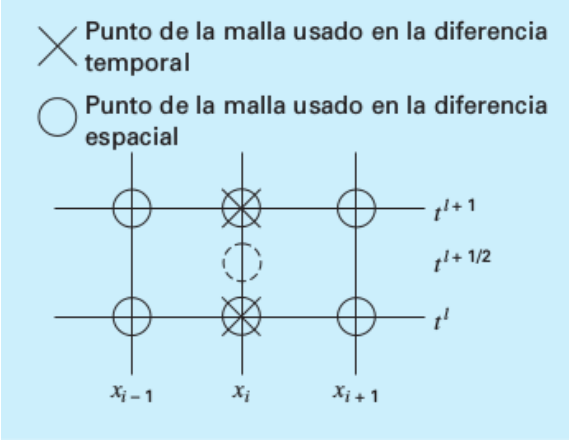
La discretización de Crank-Nicolson para la ecuación de calor es:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{k}{2\Delta x^2}((u_{i-1}^{n+1} - 2u_i^{n+1} + u_{i+1}^{n+1}) + (u_{i-1}^n - 2u_i^n + u_{i+1}^n)$$

Donde es demostrable que,

$$MAX|u_i^{n+1}| <= MAX|u_i^n|$$

por lo tanto, $O(\Delta t^2, \Delta x^2)$
El método de Crank-Nicolson ofrece un esquema el cual se necesita saber ciertas condiciones iniciales para hacer un cálculo, lo que busca es tomar una muestra de temperatura en medio de dos puntos de temperatura en tiempo donde $t = 1/2$ entre punto temporal, en la siguiente plantilla se puede apreciar visualmente.




A este tipo de problema, se le conoce como problema de estado estacionario.
Para resolver este tipo de problema se requiere mucho computo, manualmente se podría volver muy tedioso, por otra parte, de la discretización podemos obtener una ecuación el cual, $r = k\frac{\Delta t}{\Delta x^2}$ entonces,

$$-ru_{i-1}^{n+1} + (1 + 2r)u_i^{n+1} - ru_{i+1}^{n+1} = -ru_{i-1}^n + (1 + 2r)u_i^n - ru_{i+1}^n$$

Tome en cuenta que la plantilla vista con anterioridad es descrita por esa ecuación, esta ecuación genera un sistema de ecuaciones con matrices tridiagonales, el cuál está compuesta de 3 matrices, una matriz que es tridiagonalmente constante (ver ecuación)

# 3 Algoritmo

En el siguiente bloque de código se presenta el algoritmo para la solución del método de Crank-Nicolson escrito en python usando SageMath.

```
import numpy as np
#funcion para animar
list_to_animate = []
def animar(A):
    list_to_animate.append(A)
    return list_to_animate
#definition of numerical parameters
@interact
def _(N=51, dt=5.e-4, L=float(1), nsteps=620, nplot=20, k=1):
    dx=L/(N-1)
    r = k*dt/dx**2
#initialize matrices A, B and b array
A = np.zeros((N-2,N-2))
B = np.zeros((N-2,N-2))
b = np.zeros((N-2))
#define matrices A, B and b array
for i in range(N-2):
    if i==0:
        A[i,:] = [2+2*r if j==0 else (-r) if j==1 else 0 for j in range(N
        B[i,:] = [2-2*r if j==0 else r if j==1 else 0 for j in range(N-2
        b[i] = 0. #boundary condition at i=1
    elif i==N-3:
        A[i,:] = [-r if j==N-4 else 2+2*r if j==N-3 else 0 for j in range
        B[i,:] = [r if j==N-4 else 2-2*r if j==N-3 else 0 for j in range(
        b[i] = 0. #boundary condition at i=N
    else:
        A[i,:] = [-r if j==i-1 or j==i+1 else 2+2*r if j==i else 0 for j
        B[i,:] = [r if j==i-1 or j==i+1 else 2-2*r if j==i else 0 for j i

#initialize grid
x = np.linspace(0,1,N)
#initial condition
u = np.asarray([2*xx if xx<=0.5 else 2*(1-xx) for xx in x])
#evaluate right hand side at t=0
bb = B.dot(u[1:-1]) + b
for j in range(nsteps):
    u[1:-1] = np.linalg.solve(A,bb) #solucion en dominio
    #update en lado derecho
    bb = B.dot(u[1:-1]) + b
    if(j%nplot==0): #plot results every nplot timesteps
        S = list_plot(zip(x.tolist(), u.tolist()),ymax=1)
        animar(S)
animate(list_to_animate)
```

<Visita para ver la explicacion del algoritmo>

# 4 Conclusiones

De la variación de los métodos explícitos e implícitos, el mñetodo explícito no es estable, puede tener algún tipo de oscilación el cual haga una deformación en el tiempo, mientras los métodos implícitos si convergen perfectamente es probable que haya algún tipo de deformación en el tiempo pero se garantiza resultados muy cercanos a la exactitud del flujo del calor.

# 5 Referencias

Chapra. (2007). Métodos numéricos para ingeniería. México: Mc Graw Hill. Wenshen An Introduction to Numerical Computation, World Scientific Publishing Company (December 28, 2015).