Forma diferencial	Forma integral	Comentario
$\nabla \cdot \mathbf{D} = \rho_v$	$\oint_S \mathbf{D} \cdot d\mathbf{S} = \int_v \rho_v dv$	Ley de Gauss
$\nabla \cdot \mathbf{B} = 0$	$\oint_S \mathbf{B} \cdot d\mathbf{S} = 0$	No existencia de monopolos
$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}$	$\oint_L \mathbf{E} \cdot dl = -\frac{\partial}{\partial t} \int_S \mathbf{B} \cdot dS$	Ley de Faraday
$\nabla \times \mathbf{H} = \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t}$	$\oint_L \mathbf{H} \cdot dl = \int_S \left(\mathbf{J} + \frac{\partial \mathbf{D}}{\partial t} \right) \cdot d\mathbf{S}$	Ley de circuitos de Ampere

Table 1: Leyes de Maxwell

0.1 Leyes de maxwell

Se presentan las ecuaciones de Maxwell en la tabla 1. Donde es necesario recordar el operador DEL (**??**)

- El gradiente de un escalar V: ∇V
- La divergencia de un vector A: ∇·A
- La rotacional de un vector A: ∇×A
- El Laplaciano de un escalar V: ∇^2V

Además se tienen ecuaciones auxiliares:

Relación entre la Densidad de Campo Eléctrico y la Intensidad de Campo Eléctrico.

$$\mathbf{D} = \varepsilon \mathbf{E} \tag{1a}$$

Relación entre la Densidad de Campo Magnético y la Intensidad de Campo Magnético.

$$\mathbf{B} = u\mathbf{H} \tag{1b}$$

Densidad de Corriente de conducción.

$$\mathbf{J} = \sigma \mathbf{E} \tag{1c}$$

Densidad de Corriente de convección en función de la densidad de carga volumétrica.

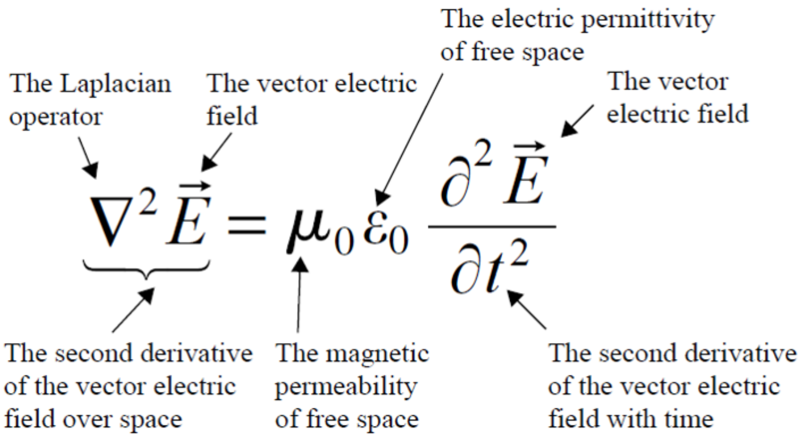
$$\mathbf{J} = \rho_v \mathbf{v} \tag{1d}$$

Hay ligeras modificaciones si son para conductores malos (aislantes):

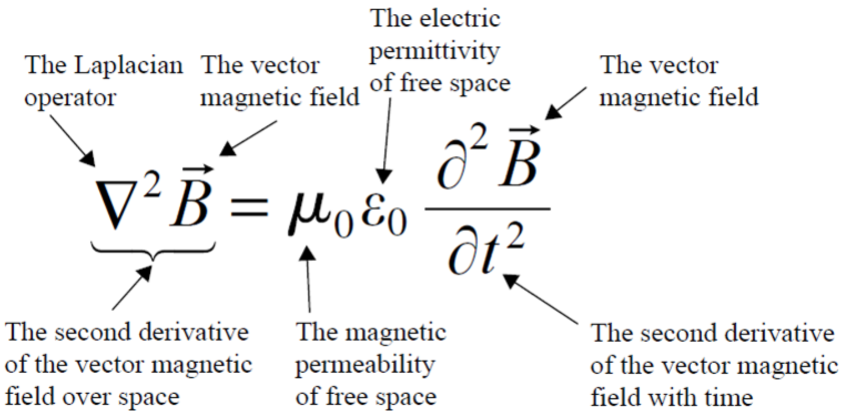
$$\mathbf{D} = \varepsilon \mathbf{E} + P \tag{2a}$$
$$\mathbf{B} = u(\mathbf{H} + M) \tag{2b}$$

Donde P es el campo de polarización y M es el campo de magnetización, cuando el dieléctrico es lineal se tiene:

$$P = \chi_e \varepsilon_0 \mathbf{E} \qquad\qquad M = \chi_m \mathbf{H}$$



(a) Ecuación de onda para campos eléctricos.



(b) Ecuación de onda para campos magnéticos.

Figure 1: Ecuaciones de onda

0.2 Métodos para control de la congestión

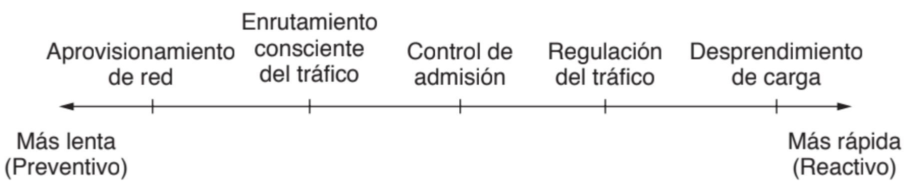
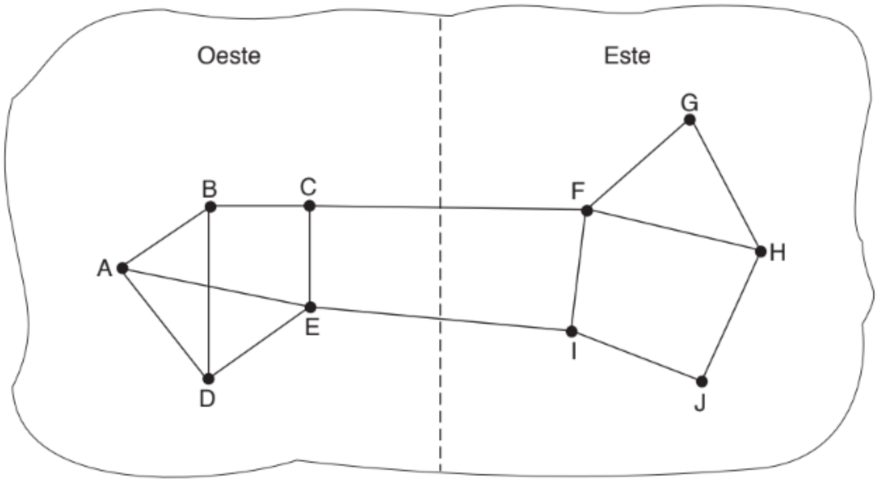


Figure 2: Escalas de tiempo de los métodos para el control de la congestión.

0.2.1 Enrutamiento consiente de tráfico

El objetivo es desviar el tráfico de los puntos más activos hacia otros, esto esta en desuso debido a su oscilación



Sean dos sistemas autónomos como la image, los enlaces entre ambos sistemas son CF y EI, al congestionarse CF se enrutará los paquetes hacia EI para despejar CF. Pero luego de este proceso EI estará congestionado y verá EI como una ruta optima. El sistema de enrutamiento será **oscilatorio**.

0.2.2 Regulación de tráfico

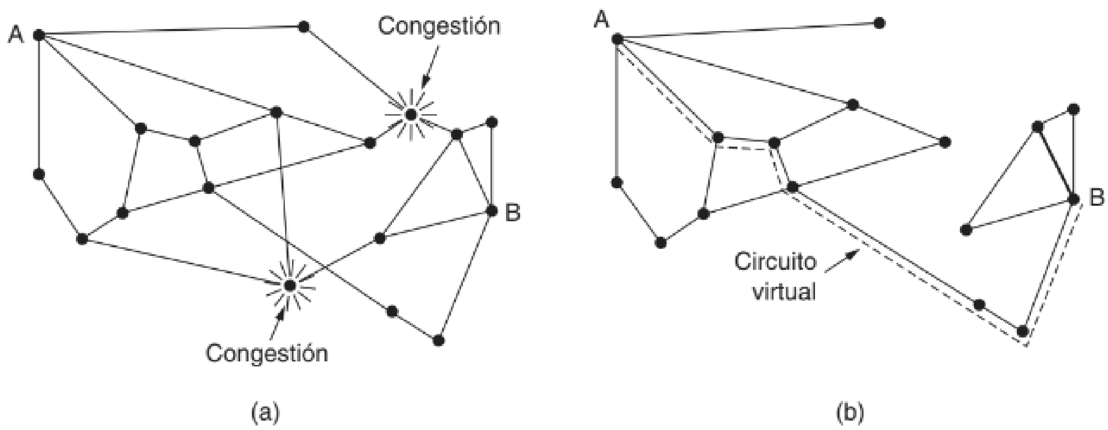


Figure 3: (a) Una red congestionada. (b) La parte de la red que no esta congestionada.

En la figura 3.a se muestra la red con dos puntos de congestión, lo que haremos es eliminar esos puntos de congestión y calculando una nueva ruta virtual evitando los puntos de congestión. El problema puede ser que esta desviación tome un poco más de tiempo. Al existir una congestión, el router congestionado devuelve un trama al azar y envía un **paquete regulador** con un bit de encabezado al emisor para que no generé más paquetes y se reduzca el tráfico. Esto no es equitativo pues los emisor más rápidos tendrán más paquetes congestionados que lo emisores lentos.

0.2.2.1 ECN

ENC o notificación explicita de congestión, funciona similar al **paquete regulador**, en vez de crear un datagrama que vuelva al emisor, se configura dos bits para que cuando lleguen al receptor este se enteré en donde hubo una congestión y este mandará un mensaje al emisor directo para que tome las medidas.

```python
#Importamos librerias
import math
import matplotlib.pyplot as plt
import numpy as np
#Registramos los valores del usuario, de debe ingresar un complejo como: a+bj
number=complex(input("Ingresa el numero complejo: "))#Numero complejo
orden=int(input("Orden de la raiz: "))#Orden de la raiz
k=orden#Limite para la iteracion
if number.real==0:#Si la parte real es 0 se hacen excepciones para la indeterminacion
    if number.imag==0:
        theta=0
    elif number.imag>0:
        theta=math.pi/2
    elif number.imag<0:
        theta=3*math.pi/2
else:
    theta=math.atan(number.imag/number.real)#Thera calculada usando un arctg
valor=[]#Array de respuestas
#Bucle de iteracion
for i in range(0,k,1):
    r1=(abs(number)**(1/orden))*(math.cos((theta+2*i*math.pi)/(orden))+
        1j*math.sin((theta+2*i*math.pi)/(orden)))#Se calcula la rpta para cada valor de k
    valor.append(np.around(r1,4))#Se anade la respuesta al vector de respuestas y se redondea, el numero despues de r1 indica la cantidad de decimales
    print("Respuesta "+str(i))#Se imprime las respuestas
    print("------------------")
    print(valor[i])
    print("++++++++++++++++++")
# Extraemos la parte real
x =[index.real for index in valor]
# Extraemos la parte imaginaria
y =[index.imag for index in valor]
ax =plt.gca()#Obtenemos las propiedades del plot
# plot the complex numbers
plt.scatter(x, y)#Ploteamos los puntos
plt.plot(x,y,color='b',zorder=1)#Unimos los puntos
plt.ylabel('Imaginary')#Rotulo de los ejes
plt.xlabel('Real')
plt.grid()#Activamos la grilla
for i_x, i_y in zip(x, y):#Obtenemos los valores para punto
    plt.text(i_x, i_y+0.02, '({}, j{})'.format(i_x, i_y))
plt.axvline(x=0, c="red")#Ploteamos los ejes del origen
plt.axhline(y=0, c="red")
plt.title("Raices complejas de "+str(number))#Anadimos titulos
plt.show()#Mostramos la grafica
```