Guia general de computación gráfica con Python

Carlos Andres Lopez

Guia general de computación gráfica con Python

Carlos Andres Lopez

8 de agosto de 2017

Índice general

Introduccion

Generalidades

Esta guía tiene como objeto recopilar información asociada al curso de computación gráfica de la Universidad Tecnológica de Pereira, la cual tiene como prerequisitos asignaturas de programación básica, programación orientada a objetos, algebra lineal, cálculo diferencial e integral, por lo que de ser necesario hará referencia a estos temas sin profundizar en ellos.

No se pretende enseñar a programar o explicar métodos matemáticos mas alla de los necesarios implicitos en los temas que se abordan, se asume que el lector ya conoce aspectos básicos de las disciplinas mencionadas.

Por último cabe mencionar el empleo del lenguaje python como herramienta de programación, la cual se ha seleccionado gracias a su versatilidad, su alta curva de aprendizaje, la cantidad de material bibliográfico y librerías disponibles, entre ellas pygame imprenscindible en las aplicaciones aqui construidas.

Contenido de este libro

La información aqui expuesta se ha dividido en cuatro partes: ecuaciones, transfomraciones lineales, algorimos gráficos y construcciones geométricas.

La primera sección aborda los temas asociados a las ecuaciones de la recta y el plano, el segundo capitulo se centra en las transformaciones lineales básicas, a cntinuación se analizarán los algoritmos incrementales y finalmente las construcciones geométricas como lo son los planos isometricos y sus vistas.

Códigos en python

A los temas expuestos se asociara códigos implementados en python, las funciones, clases y sus métodos siempre estarán documentados acorde con convención PEP 257 (Python Enhancement Proposals) [1], con las siguientes reglas

básicas a saber:

- La cadena de documentacion comienza y finaliza con comillas para comentario multilinea (”’).
- La primera linea debe ser una cadena explicativa del subprograma terminada con punto. Debe ser una sola linea y a continuacion debe haber un espacio en blanco.
- Se debe adicionar cadena con la descripcion del subprograma, los parámetros requeridos, que hace la subrutina y en caso de retornar un valor o estructura describirlos, mencionar excepciones que puede generar y cualquier otro detalle que se considere importante.
- Se recomienda una linea en blanco antes de las comillas de cierre de comentario.

Capítulo 1

Ecuaciones de la recta y el plano

1.1. Pendiente de una recta

Definimos pendiente de una recta a la cantidad de unidades de desplazamiento vertical de una recta en relacion con el numero de unidades de desplazamiento horizontal, es decir que la pendiente es la razon de cambio vertical hacia arriba o hacia abajo respecto a la razon de cambio horizontal a izquierda o derecha.

$$m = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1}, x_1 \neq x_2 \tag{1.1}$$

La pendiente de una recta vertical no está definida.

De lo anterior se desprende la ecuacion punto-pendiente de una recta:

$$y_2 - y_1 = m(x_2 - x_1) \tag{1.2}$$

La ecuacion punto-pendiente se utiliza cuando se tiene una gráfica y desea conocerse la ecuación.

Ejemplo 1.1.1. Hallar la ecuación de recta de la figura 2.1.
Tenemos p=(-4,-7) y q=(1,3):

$$m = \frac{3 - (-7)}{1 - (-4)} = \frac{10}{5} = 2$$

Usando la ecuación punto-pendiente:

$$\begin{aligned} (y - 3) &= 2(x - 1) \\ y &= 2x - 2 + 3 \\ y &= 2x + 1 \end{aligned}$$

Figura 1.1: Recta definida por $\overline{pq}$

Si en cambio se tiene una ecuación y se busca generar la gráfica se debe utilizar la ecuación pendiente-intersección:

$$y = mx + b \tag{1.3}$$

Donde se trata de una recta con pendiente m y un punto de intersección en (0,b).

Desde la programación tenemos entonces dos clases de funciones a construir, aquellas que permitan encontrar la ecuación y aquellas que permitan graficar la ecuación. Si bien es cierto que la computación grafica se debe ocupar solo del segundo aspecto, con frecuencia se hace útil disponer de funciones que nos permitan trabajar con las ecuaciones.

Emplearemos la ecución de punto-pendiente para implementar una función que nos retorne dado un punto y la pendiente nos retorne la ecuación. Para explicar mejor la forma en como debemos abordar el problema, tomaremos un ejemplo puntual.

Ejemplo 1.1.2. Hallar la ecuación de la recta que pasa por el punto (1,2) y cuya pendiente es 3.

$$\begin{aligned} y - y_1 &= m(x - x1) \\ y - 2 &= 3(x - 1) \\ y &= 3x - 3 + 2 \\ y &= 3x - 1 \end{aligned}$$

Observamos que de forma general la ecuacion es una cadena de la forma y=mx + b donde b=-mx + y, es decir que podemos construir una funcion que nos calcule b y otra que nos retorne la ecuación.

Código Fuente 1.1.1. Función que retorna b.

```
def b(p,m):
    '''Retorna constante b.

    b=-mx + y

    Parametros:
    p -- Lista con valores x,y del punto
    m -- Pendiente de la recta
```

```
    ,,,
    res=-(m*p[0])+p[1]
    return res
```

Código Fuente 1.1.2. Función que retorna la ecuación.

```
def Ecuacion(p,m):
    '''Retorna ecuacion de la recta.

    y=mx+b

    Parametros:
    p -- Lista con valores x,y del punto
    m -- Pendiente de la recta

    ,,,
    x=m*p[0]
    c_mx=str(m)+'x'
    nb=b(p,m)
    c_b='+' + str(nb)
    if nb < 0:
        c_b=str(nb)
    ecuacion='y='+ c_mx + c_b
    return ecuacion
```

Al emplear la funcion Ecuacion:

```
print Ecuacion([1,2],3)
>>y=3x-1
```

Tambien es posible construir una función que dados los parametros de pendiente y constante de desplazamiento(b) nos calcule los valores de y, de ahi en adelante es posible agregar elementos de parametrización tales como dominio inicial y final de la función e incremento.

Código Fuente 1.1.3. Función que retorna valores de la recta.

```
def Recta(m, b, xmin=0, xmax=10, inc=1):
    '''Retorna valores de y para la ecuacion de la recta.

    y=mx+b

    Parametros:
    m -- Pendiente de la recta
    b -- Constante de desplazamiento
    xmin -- Valor minimo de x
    xmax -- Valor maximo de x
    inc -- Incremento o razon de cambio

    ,,,
    y=[]
    if xmin>xmax:
        aux=xmin
        xmin=xmax
```

```
        xmax=aux
    i=xmin
    while i <= xmax:
        val=(m*i)+b
        y.append(val)
        i+=inc
    return y
```

Al emplear la funcion Recta para la ecuación y=3x-1 y valores entre [-1,10]:

```
print Recta(3,-1,-1,10)
>>[-4, -1, 2, 5, 8, 11, 14, 17, 20, 23, 26]
print Recta(3,-1)
>>[-1, 2, 5, 8, 11, 14, 17, 20, 23, 26, 29]
print Recta(3,-1,-1.0,2.0,0.5)
>>[-4.0, -2.5, -1.0, 0.5, 2.0, 3.5, 5.0]
```

La pendiente de una recta es útil para establecer si dos rectas son paralelas o perpendiculares. Dos rectas distintas son paralelas si sus pendientes son iguales y dos rectas distintas son perpendiculares si se cumple que:

$$m_1 = -\frac{1}{m_2} \tag{1.4}$$

La función que calcula la pendiente de una recta se muestra a continuación.

Código Fuente 1.1.4. Función que retorna penidente m.

```
def Pendiente(p1,p2):
    '''Retorna la pendiene de una recta.

    m=(y2-y1)/(x2-x1)

    Parametros:
    p1 -- Lista de valores x,y del punto 1
    p2 -- Lista de valores x,y del punto 2

    Excepciones:
    Error de division por cero si x2-x1=0

    '''
    num=p2[1]-p1[1]
    den=p2[0]-p1[0]
    try:
        val=num/den
        return val
    except ZeroDivisionError:
        return float('Inf')
```

Al emplear la función pendiente dados dos puntos: a) p=[10,20] y q=[10,-1], b) p=[1,2] y q=[3,1].

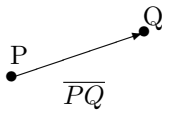
```
m=Pendiente([10,20],[10,-1])
```

```
print m-1
>>inf
m=Pendiente([1.0,2.0],[3.0,1.0])
print m
>>-0.5
```

Nota: *Se debe recordar que en Python valores flotantes generan valores flotantes y valores enteros generan valores enteros.*

1.2. Vectores

Definiremos vector como una magnitud que posee además dirección y sentido. De esta forma la recta $\overline{PQ}$ es el segmento orientado con punto inical P y punto final Q y se denota su longitud $\|\overline{PQ}\|$ que llamaremos magnitud del vector o norma del vector.



La representacion canónica de un vector es un conjunto de segmentos orientados equivalentes cuyo punto inicial se encuentra en el origen. La representación canónica de un vector dados sus puntos extremosn P y Q en el plano es:

Sea $P = (P_x, P_y)$ y $Q = (Q_x, Q_y)$

$$\vec{v} = \overline{PQ} = (Q_x - P_x, Q_y - P_y) \tag{1.5}$$

Si $\vec{v} = (v_1, v_2)$ entonces $v_1 = Q_x - P_x$ y $v_2 = Q_y - P_y$.

Un vector v en el plano (R^2) puede describirse mediante coordenadas (v_1, v_2), estas son denominadas componentes del vector. Si el vector a representar se encuentra en el espacio (R^3) sus componentes pueden se expresan mediante (v_1, v_2, v_3). De forma general un vector en el espacio R^n se expresa mediante $(v_1, \ldots, v_n)$.

La magnitud de un vector en el espacio R^n esta definido por la norma de un vector:

$$\|\vec{v}\| = \sqrt{v_1^2 + \ \ldots \ + v_n^2} \tag{1.6}$$

La magnitud de un vector es equivalente a la distancia entre los puntos que definen el vector.

Ejemplo 1.2.1. Dados los puntos A=(1,4) y B=(3,2) expresar el vector en su forma canónica y hallar la distancia entre A y B.

$$\vec{v} = (3-1, 2-4) = (2, -2)\ forma\ canonica$$
$$d(A,B) = \|\vec{v}\| = \sqrt{(2)^2 + (-2)^2} = \sqrt{8}$$

Código Fuente 1.2.1. Función que retorna vector canónico.

```
import math

def Vector(p1,p2):
    '''Retorna el vector canonico.

    v=(p2[0]-p1[0], ... , p2[n]-p1[n])

    Parametros:
    p1 --- Punto inicial.
    p2 --- Punto final.

    '''
    r=len(p1)
    v=[]
    for i in range(r):
        val=p2[i]-p1[i]
        v.append(val)
    return v
```

Al emplear la función Vector:

```
print Vector([1,4],[3,2])
>>[2, -2]
print Vector([1,3,3],[2,-1,5])
>>[1, -4, 2]
```

Código Fuente 1.2.2. Función que retorna la norma del vector.

```
def Norma(v):
    '''Retorna la norma de un vector.

    v=raiz((v1*v1) + ... + (vn*vn))

    Parametros:
    v ---  vector
    '''
    r=len(v)
    suma=0
    for i in range(r):
        suma+=v[i]**2
    res=math.sqrt(suma)
    return res
```

Al emplear la función Norma:

```
print Norma([2.0,-2.0])
>>2.82842712475
print Norma(Vector([1.0,4.0],[3.0,2.0]))
>>2.82842712475
print Norma([1.0, -4.0, 2.0])
>>4.58257569496
```

Las operaciones básicas entre vectores son las siguientes:

- Suma de $\vec{u}$ y $\vec{v}$: $\vec{u}+\vec{v}=(u_1+v_1,\ldots,u_n+v_n)$.
- Producto por escalar k y $\vec{v}$: $k\vec{v}=(kv_1,\ldots,kv_n)$.
- Opuesto de $\vec{v}$: $\vec{v}=-\vec{v}=(-1)\vec{v}=(-v_1,\ldots,-v_n)$.
- Diferencia de $\vec{u}$ y $\vec{v}$: $\vec{u}-\vec{v}=\vec{u}+(-\vec{v})=(u_1-v_1,\ldots,u_n-v_n)$.

Las propiedades de la suma son:

- $\vec{u}+\vec{v}=\vec{v}+\vec{u}$
- $(\vec{u}+\vec{v})+\vec{w}=\vec{u}+(\vec{v}+\vec{w})$
- $\vec{u}+0=\vec{u}$ (Existencia del vector nulo).
- $\vec{u}+(-\vec{u})=0$

Las propiedades del producto por escalar son:

- $c(d\vec{u})=(cd)\vec{u}$
- $(c+d)\vec{u}=c\vec{u}+d\vec{u}$
- $c(\vec{u}+\vec{v})=c\vec{u}+c\vec{v}$
- $1(\vec{u})=\vec{u}\ ,\ 0(\vec{u})=0$

Si se tiene un vector $\vec{v}$ y un escalar c entonces:

$$\|c\vec{v}\|=|c|\|\vec{v}\| \tag{1.7}$$

donde |c| es valor absoluto de c y se le llama longitud escalar de un vector.

Si se tiene un vector $\vec{v}$ no nulo, es posible encontrar un vector unitario $\vec{u}$ en la misma deirección de $\vec{v}$ y con magnitud $\|\vec{u}\|=1$:

$$\vec{u}=\frac{\vec{v}}{\|\vec{v}\|} \tag{1.8}$$

Los vectores unitarios (1,0) y (0,1) se conocen como vectores unitarios canónicos del plano y de denotan por:

$$i = (1, 0)\ y\ j = (0, 1) \tag{1.9}$$

Estos vectores sirven para representar cualquier vector:

$$v = (v_1, v_2) = v_1(1, 0) + v_2(0, 1) = v_1 i + v_2 j \tag{1.10}$$

Esta expresión es conocida como la combinación lineal de i y j.

Código Fuente 1.2.3. Función que retorna vector unitario de $\vec{v}$.

```
def Unitario(v):
    '''Retorna el vector unitario de v.

    u=v/Nomra(v)

    Parametros:
    v --- vector

    Excepciones:
    Error de division por cero si Norma=0

    ,,,
    u=[]
    r=len(v)
    n=Norma(v)
    try:
        for i in range(r):
        val=v[i]/n
        u.append(val)
        return u
    except ZeroDivisionError:
        return float('Inf')
```

Al emplear la función Unitario:

```
print Unitario([4.0,3.0])
>>[0.8, 0.6]
```

El **producto punto o producto escalar** entre dos vectores $\vec{u}$ y $\vec{v}$ da como resultado un escalar:

$$\vec{u}.\vec{v} = (u_1.v_1) + \ldots + (u_n.v_n) \tag{1.11}$$

Es posible calcular el ángulo entre dos vectores si se conoce la norma de los vectores y el producto punto entre ellos:

$$cos\alpha = \frac{|\vec{u}.\vec{v}|}{\|\vec{u}\|.\|\vec{v}\|} \tag{1.12}$$

Ejemplo 1.2.2. Hallar el ángulo entre $\vec{u} = (4, -1)$ y $\vec{v} = (3, 3)$.

$$\begin{aligned}
\vec{u}.\vec{v} &= (4)(3) + (-1)(3) = 12 - 3 = 9 \\
\|\vec{u}\| &= \sqrt{4^2 + (-1)^2} = \sqrt{16 + 1} = \sqrt{17} \\
\|\vec{v}\| &= \sqrt{3^2 + 3^2} = \sqrt{9 + 9} = \sqrt{18} \\
cos\alpha &= \frac{|9|}{\sqrt{17}.\sqrt{18}} = 0{,}5145 \\
\cos\alpha^{-1}(0{,}5145) &= 59{,}036^\circ \\
\alpha &= 59{,}036^\circ
\end{aligned}$$

Las propiedades del producto punto son:

- $\vec{u}.\vec{v} = \vec{v}.\vec{u}$
- $\vec{w}.(\vec{u}.\vec{v}) = \vec{w}.\vec{u} + \vec{w}.\vec{v}$
- $(k.\vec{u}).\vec{v} = k.(\vec{u}.\vec{v}) = \vec{u}.(k.\vec{v})$ donde k escalar.

Código Fuente 1.2.4. Función que retorna producto punto de dos vectores.

```
def Punto(u,v, k=1):
    '''Producto punto entre vectores u y v.

    u.v=(u1.v1) + ... + (un-vn), si k=1
    k(u.v)=k(u1.v1) + ... + k(un-vn)

    Parametros:
    u --- vector
    v --- vector
    k --- escalar por defecto 1

    Excepciones:
    Error de desbordamiento si los vectores de longitud distinta

    '''

    lon_u=len(u)
    try:
        suma=0
        for i in range(lon_u):
            suma+=k*(u[i]*v[i])
        return suma
    except Exception, ex:
        print ex
```

Al emplear la función Punto:

```
u=[4,-1]
v=[3,3]
print Punto(u,v)
>>9
```

```
print Punto(u,v,2)
>>18
```

El cuadrado de la norma de un vector es igual al producto punto del vector por si mismo:

$$\|\vec{v}\|^2 = \vec{v}.\vec{v} \tag{1.13}$$

Dos vectores son paralelos si son proporcionales entre ellos:

$$\vec{u} \parallel \vec{v} \Leftrightarrow \vec{u} = \lambda\vec{v} \tag{1.14}$$

Dos vectores son perpendiculares si su producto punto es cero:

$$\vec{u} \perp \vec{v} \Leftrightarrow \vec{u}.\vec{v} = 0 \tag{1.15}$$

La proyección ortogonal de un vector $\vec{u}$ sobre el vector $\vec{v}$ es:

$$proy_v u = \left(\frac{\vec{u}.\vec{v}}{\|\vec{v}\|^2}\right)\vec{v} \tag{1.16}$$

1.3. Ecuacion de la recta

Una forma de describir una recta L es afirar que todos los puntos entre P=(x,y) y $Q = (x_1, y_1)$ que forman el segmento $\overline{PQ}$ son paralelos al vector $\vec{v}$, de esta forma:

$$\overline{PQ} = \lambda\vec{v} \tag{1.17}$$

donde λ es un escalar.

De esta forma, una manera de expresar el segmento $\overline{PQ}$ es el vector $\vec{PQ}$, según expresion (1.5) :

$$\overline{PQ} = \vec{PQ} = (x - x_0, y - y_0) \tag{1.18}$$

De la expresion (1.17) y (1.18) tenemos:

$$\begin{gathered}\vec{PQ} = \lambda\vec{v} \\ (x - x_0, y - y_0) = \lambda(v_1, v_2)\end{gathered} \tag{1.19}$$

De lo anterior se desprende la ecuacion **vectorial de la recta**:

$$(x, y) = (x_0, y_0) + \lambda(v_1, v_2) \tag{1.20}$$

Si se despejan las variables x e y, la **ecuación paramétrica de la recta**:

$$\begin{aligned} x &= x_0 + \lambda v_1 \\ y &= y_0 + \lambda v_2 \end{aligned} \tag{1.21}$$

Si eliminamos el factor de proporcionalidad λ aparece la **ecuación simetrica de la recta**:

$$\begin{aligned} \lambda &= \frac{x - x_0}{v_1} \\ \lambda &= \frac{y - y_0}{v_2} \\ \frac{x - x_0}{v_1} &= \frac{y - y_0}{v_2} \end{aligned} \tag{1.22}$$

Ejemplo 1.3.1. Dados los puntos A=(-1,2) y B=(2,4) encontrar: (a) ecuación vectorial de la recta, (b) ecuación paramétrica de la recta, y (c) ecuación simetrica de la recta.

Hallamos el vector $\vec{v}$ que nos da la dirección del segmento $\overline{AB}$:

$$\vec{v} = (2 - (-1), 4 - 2) = (3, 2)$$

(a) Ecuación vectorial de la recta. Tomamos el vector $\vec{v}$ y cualquier punto (A o B):

$$(x, y) = (-1, 2) + \lambda(3, 2)$$

(b) Ecuación paramétrica de la recta.

$$\begin{aligned} x &= -1 + 3\lambda \\ y &= 2 + 2\lambda \end{aligned}$$

(c) Ecuación simetrica de la recta.

$$\frac{x + 1}{3} = \frac{y - 2}{2}$$

Código Fuente 1.3.1. Función que retorna vector dados dos puntos.

```
def Vector(p,q):
    '''Retorna el vector dados puntos p y q.

    v=(q0-p0, ... , qn-pn)

    Parametros:
    p --- punto P
    q --- punto Q
```

```
    '''
    lon_p=len(p)
    try:
        v=[]
        for i in range(lon_p):
            v.append(q[i]-p[i])
        return v
    except Exception, ex:
        print ex
```

Al emplear la función Punto:

```
A=[-1,2]
B=[2,4]
print vector(A,B)
>>[3, 2]
```

El siguiente subprograma retorna el valor evaluado en un punto empleando la ecuacion paramétrica o si se desea la ecuación paramétrica dado un punto, un vector y de forma opcional, el parametro de variación. Se adiciona un parámetro adicional que permite espcificar si mostrar o no la ecuación empleada.

Código Fuente 1.3.2. Función que retorna ecuación paramétrica de la recta.

```
def Ec_Parametrica(p,v,t=1,ver=0):
    '''Retorna ecuacion parametrica dado punto y vector.

    x=x0-t.vx
    y=y0-t.vy

    Parametros:
    p --- punto P de la recta
    v --- vector
    t --- parametro por defecto 1
    ver --- ver ecuacion 0:no, 1 si

    '''
    if ver==0:
        x=p[0]+(t*v[0])
        y=p[1]+(t*v[1])
        return [x,y]
    else:
        s_x='x=' + str(p[0]) + '+(' + str(v[0]) + ')t'
        s_y='y=' + str(p[1]) + '+(' + str(v[1]) + ')t'
        com='para t=' + str(t)
        return [s_x,s_y,com]
```

Al emplear la función Ec Parametrica:

```
P=(-1,2)
v=(3,2)
print Ec_Parametrica(P,v)
>>[2, 4]
```

```
print Ec_Parametrica(P,v,2)
>>[5, 6]
print 'Ecuacion parametrica: '
for campo in Ec_Parametrica(P,v,2,1):
    print campo
>>Ecuacion parametrica:
>>x=-1+(3)t
>>y=2+(2)t
>>para t=2
```

Una vez se tiene la ecuación continua de la recta, es posible llegar a la **ecaución general de la recta o ecuación cartesiana** :

$$Ax + By + C = 0 \tag{1.23}$$

Partamos de la ecuacón continua:

$$\begin{aligned} \frac{x - x_0}{v_1} &= \frac{y - y_0}{v_2} \\ (x - x_0)v_2 - (y - y_0)v_1 &= 0 \\ xv_2 - yv_1 - x_0v_2 + y_0v_1 &= 0 \end{aligned}$$

En donde las constantes A y B:

$$\begin{aligned} A &= v_2 \\ B &= -v_1 \end{aligned} \tag{1.24}$$

Decimos entonces que para una recta es posible asociar un vector que determina su dirección llamado vector director de la recta:

$$\vec{v} = (-B, A) \tag{1.25}$$

Se puede conocer la dirección de una recta expresada en su forma general tomando el vector director, el ángulo de una recta, comprar dos rectas para establecer si son perpendiculares, paralelas o calcular el angulo formado entre ellas.

Dadas las rectas L: Ax+By+C=0 y M:A'x+B'y+C=0 estas serán paralelas cuando:

$$\vec{u} = \lambda\vec{v}$$

Como $\vec{u} = (-B, A)$ y $\vec{v} = (-B', A')$ entonces:

$$\begin{aligned} (-B, A) &= \lambda(-B', A') \\ -B &= \lambda(-B') \\ A &= \lambda A' \end{aligned}$$

Al despejar λ:

$$\frac{A}{A'} = \frac{B}{B'} \tag{1.26}$$

Si lo anterior no se cumple, las rectas se cortaran en algun punto.

Y serán perpendiculares si:

$$\vec{u}.\vec{v} = 0$$

Entonces:

$$\begin{gathered}(-B, A).(-B', A') = 0 \\ (-B)(-B') + (A)(A') = 0 \\ BB' + AA' = 0\end{gathered} \tag{1.27}$$

Si se toma la ecuación de la recta y eliminamos la constante C:

$$\begin{gathered}Ax + By + C = 0 \\ Ax + By = -C \\ \frac{Ax}{-C} + \frac{By}{-C} = \frac{-C}{-C} \\ \frac{x}{\frac{-C}{A}} + \frac{y}{\frac{-C}{B}} = 1\end{gathered}$$

Si definimos las constantes $a = \frac{-C}{A}$ y $b = \frac{-C}{B}$ aparece la **ecuacion canónica de la recta** :

$$\frac{x}{a} + \frac{y}{b} = 1 \tag{1.28}$$

Esta ecuación tiene la particularidad de permitir conocer el corte con los ejes x e y de la recta.

Ejemplo 1.3.2. Hallar la ecuación cartesiana de la recta dado el vector $\vec{v} = (3, 2)$ y el punto $P = (-1, 2)$ luego encontrar el punto de corte con los ejes cartesianos.

$$\begin{gathered}\vec{v} = (3, 2) \\ A = 2 \ y \ B = -3 \\ C = -(-1)(2) + (2)(3) = 2 + 6 = 8 \\ 2x - 3y + 8 = 0\end{gathered}$$

Ahora hallamos los puntos de corte.

$$a = \frac{-C}{A} = \frac{-8}{2} = -4$$
$$b = \frac{-C}{B} = \frac{-8}{-3} = \frac{8}{3}$$

Punto de corte con el eje x=-4, punto de corte con el eje $y = \frac{8}{3}$.

Código Fuente 1.3.3. Función que retorna Ecuacion cartesiana de la recta.

```
def EcRecta_Cartesiana(p,v):
    '''Retorna la ecuacion cartesiana de la recta

    Ax + By + C = 0
    A=v2
    B=-v1
    C=-xov2 + yov1

    Parametro:
    p --- punto P de la recta
    v --- vector director

    '''
    c=-(p[0]*v[1])+(p[1]*v[0])
    ec=str(v[1]) + 'x + ' + str(-v[0]) + 'y + ' + str(c) +' = 0'
    return ec
```

Al emplear la función para la ecuación cartesiana de la recta:

```
p=[-1,2]
v=[3,2]
print EcRecta_Cartesiana(p,v)
>>2x + -3y + 8 = 0
```

1.4. Ecuacion del plano

En el plano es conveniente el uso de vectores para encontrar la ecuación de una recta, es decir que si se tienen los punto $P = (x_1, y_1, z_1)$ y $Q = (x, y, z)$ podemos encontrar un vector en la dirección del segmento $\overline{PQ}$:

$$\vec{PQ} = (x - x_1, y - y_1, z - z_1) \tag{1.29}$$

Este es el vector director de la recta.

La ecuación general de la recta esta definida como:

$$Ax + By + Cz + D = 0 \tag{1.30}$$

De las ecuaciones (1.29) y (1.30) se tiene:

$$\begin{aligned} x &= x - x_1 \\ y &= y - y_1 \\ z &= z - z_1 \end{aligned} \tag{1.31}$$

1.5. Puntos de corte

1.6. Ejercicios propuestos

1. Implemente algoritmo que dados los puntos A_1 y A_2 y la recta definida por estos y los puntos B_1 y B_2 y la recta definida por estos muestre si son perpendiculares.

2. Implemente algoritmo que dados los puntos A_1 y A_2 y la recta definida por estos y los puntos B_1 y B_2 y la recta definida por estos muestre si son paralelos.

3. Implemente algoritmo que dados los vectores u,v en el espacio R^n retorne el vector suma.

4. Implemente algoritmo que dados el vector v en el espacio R^n y el escalar k retorne el producto por escalar.

5. Implemente algoritmo que dado el vector v en el espacio R^n retorne el vector opuesto.

6. Implemente algoritmo que retorne el coseno director de un vector en el espacio.

7. Implemente algoritmo que retorne la proyección del vector $\vec{u}$ sobre $\vec{v}$.