Instituto Politécnico Nacional
Escuela Superior de Ingeniería Textil

Academia de Ciencias Básicas

Laboratorio de Ciencias Básicas

Unidad de Aprendizaje:
Cálculo Vectorial

“Librería en lenguaje C++ para la Unidad de Aprendizaje de Cálculo Vectorial”
(lcv.h)

Profesor:
Rodolfo Villalobos Martínez

Octubre de 2023

Es una librería denominada lcb.h en su versión 1.0 desarrollada para la unidad de aprendizaje de Calculo Vectorial que se encuentra dentro del repositorio Github con la dirección https://github.com/rod00001/lcv.git

Esta escrita en lenguaje c++. Es una versión simplificada y de implementación sencilla. Puede ser compilada y ejecutada desde un celular o computadora portátil que cuente con la aplicación para compilar el código.

Se pretende que los alumnos puedan comprobar o encontrar resultados a los problemas planteados de forma sencilla y rápida.

El código se encuentra bajo la licencia “Sin Licencia” que se presenta a continuación:

Se trata de software libre y sin trabas que se ha liberado en el dominio público.

Cualquier persona es libre de copiar, modificar, publicar, usar, compilar, vender o distribuir este software, ya sea en forma de código fuente o como un binario compilado, para cualquier propósito, comercial o no comercial, y por cualquier medio.

En las jurisdicciones que reconocen las leyes de derechos de autor, el autor o los autores de este software dedican todos y cada uno de los derechos de autor sobre el software al dominio público. Hacemos esta dedicación en beneficio del público en general y en detrimento de nuestros herederos y sucesores. Pretendemos que esta dedicación sea un acto manifiesto de renuncia a perpetuidad de todos los derechos presentes y futuros de este software en virtud de la ley de derechos de autor.

EL SOFTWARE SE PROPORCIONA "TAL CUAL", SIN GARANTÍA DE NINGÚN TIPO, EXPRESA O IMPLÍCITA, INCLUIDAS, ENTRE OTRAS, LAS GARANTÍAS DE COMERCIABILIDAD, IDONEIDAD PARA UN PROPÓSITO PARTICULAR Y NO INFRACCIÓN. EN NINGÚN CASO LOS AUTORES SERÁN RESPONSABLES DE NINGUNA RECLAMACIÓN, DAÑOS U OTRA RESPONSABILIDAD, YA SEA EN UNA ACCIÓN CONTRACTUAL, EXTRACONTRACTUAL O DE OTRO TIPO, QUE SURJA DE, FUERA DE O EN RELACIÓN CON EL SOFTWARE O EL USO U OTRAS TRANSACCIONES EN EL SOFTWARE.

Contenido.

13. //pasa de coordenadas cilíndricas a rectangulares, double cilarect(double r=0,double teta=0,double z=0);

14. //pasa de coordenadas esféricas a rectangulares, double esfarect(double r=0,double teta=0,double phy=0);

15. //pasa de coordenadas rectangulares a esféricas, double rectaesf(double x=0,double y=0,double z=0);

16. //pasa de coordenadas esféricas a rectangulares en grados, double esfarectengra(double r=0,double teta=0,double phy=0);

17. //pasa de coordenadas rectangulares a esféricas en grados, double rectaesfengra(double x=0,double y=0,double z=0);

18. //pasa de coordenadas esféricas a cilíndricas, double esfacil(double p=0,double teta=0,double phy=0);

19. //pasa de coordenadas cilíndricas a esféricas, double cilaesf(double r=0,double teta=0,double z=0);

20. //pasa de coordenadas esféricas a cilíndricas en grados, double esfacilengra(double p=0,double teta=0,double phy=0);

21. //pasa de coordenadas cilindricas a esfericas en grados, double cilaesfengra(double r=0,double teta=0,double z=0);

22. //producto punto entre dos vectores, double upuntov(double u1=0,double u2=0,double u3=0,double v1=0,double v2=0,double v3=0);
23. //suma de dos vectores u+v, double umasv(double u1=0,double u2=0,double u3=0,double v1=0,double v2=0,double v3=0);

24. //resta de dos vectores u-v, double umenosv(double u1=0,double u2=0,double u3=0,double v1=0,double v2=0,double v3=0);

25. //vector por escalar k*u, double kporu(double k=0, double u1=0,double u2=0,double u3=0);

26. //determina si son ortogonales dos vectores, double vortou (double u1=0,double u2=0,double u3=0,double v1=0,double v2=0,double v3=0);

27. //Calcula el determinante de una matriz A de tamaño 3*3, double detA3(double u1=0,double u2=0,double u3=0,double v1=0,double v2=0,double v3=0,double w1=0,double w2=0,double w3=0);

28. //Calcula la logitud de un vector, double longu(double u1=0,double u2=0,double u3=0);

29. //Muestra el vector dado normalizado, double normu(double u1=0,double u2=0,double u3=0);

30. //Producto cruz entre u y v, double ucruzv(double u1=0,double u2=0,double u3=0,double v1=0,double v2=0,double v3=0);

31. //Devuelve el triple producto punto de los tres vectores, double tripleprod(double u1=0,double u2=0,double u3=0,double v1=0,double v2=0,double v3=0,double w1=0,double w2=0,double w3=0);

32. //Guarda un archivo frecta.csv con los puntos de la recta en el intervalo y usando el paso dado en 2D, int frecta (double a, double m,double to,double tf,double paso) ;

33. //Guarda un archivo frectapuntovector.csv con los puntos de la recta en el intervalo y usando el paso dado 3n 3D, int frectapuntovector (double u1=0,double u2=0,double u3=0,double v1=0,double v2=0,double v3=0,double to=0,double tf=10,double paso=0.1) ;

Librería lcv.h

```
#include<iostream>
#include<cmath>
#include<fstream>

#define PI 3.14159265359
#define E 2.718281828

using namespace std;

double rad=0;
double grad=0;
double x=0, y=0, z=0, r=0, teta=0, phy=0;
double dist=0;
double m=0;
double w1=0, w2=0, w3=0, w=0, A=0, w4=0, w5=0, w6=0;

//pasar de grados a radianes
double gradarad(double grad){
   return grad*PI/180;
}

//para pasar de radianes a grados
double radagrad(double rad){
   return rad*180/PI;
}

//para pasar de coordenadas polares a rectangulares ingresando teta en radianes
double polarectenrad(double r=0,double teta=0){
   x=r*cos(teta);
   y=r*sin(teta);
   cout<<"("<<x<<","<<y<<")"<<endl;
   return 0;
}

//pasar de coordenadas polares a rectangulares en grados
double polarectengrad(double r=0,double teta=0){
   teta=gradarad(teta);
   x=r*cos(teta);
   y=r*sin(teta);
   cout<<"("<<x<<","<<y<<")"<<endl;
   return 0;
}

//pasar de coordenadas rectangulares a polares en radianes
double rectapolenrad(double x=0,double y=0){
   r=sqrt((x*x)+(y*y));
```

```
   teta=atan(y/x);
   cout<<"("<<r<<","<<teta<<")"<<endl;
   return 0;
}

//pasar de coordenadas rectangulares a polares en grados
double rectapolengrad(double x=0,double y=0){
   r=sqrt((x*x)+(y*y));
   teta=atan(y/x);
   teta=radagrad(teta);
   cout<<"("<<r<<","<<teta<<")"<<endl;
   return 0;
}

//encontrar el punto medio entre dos puntos
double puntomedio(double x1=0,double y1=0,double z1=0,double x2=0,double y2=0,double
z2=0){
   x=(x1+x2)/2;
   y=(x1+x2)/2;
   z=(x1+x2)/2;
   cout<<"("<<x<<","<<y<<","<<z<<")"<<endl;
   return 0;
}

//regresa la pendiente
double pendiente(double x1=0,double y1=0,double x2=0,double y2=0){
   m=(y1-y2)/(x1-x2);
   teta=atan(m);
   teta=radagrad(teta);
   cout<<"pendiente: "<<m<<" angulo con respecto x: "<<teta<<"grados"<<endl;
   return m;
}

//regresa la distancia entre dos puntos
double distpuntos(double x1=0,double y1=0,double z1=0,double x2=0,double y2=0,double z2=0){
   r=sqrt(((x1-x2)*(x1-x2))+((y1-y2)*(y1-y2))+((z1-z2)*(z1-z2)));
   cout<<r<<endl;
   return r;
}

//pasa de coordenadas rectangulares a cilindricas en grados
double rectacilengra(double x=0,double y=0,double z=0){
   r=sqrt((x*x)+(y*y));
   teta=atan(y/x);
   teta=radagrad(teta);
   z=z;
   cout<<"("<<r<<","<<teta<<","<<z<<")"<<endl;
   return 0;
```

```
}

//pasa de coordenadas cilíndricas a rectangulares en grados
double cilarectengra(double r=0,double teta=0,double z=0){
   teta=gradarad(teta);
   x=r*cos(teta);
   y=r*sin(teta);
   z=z;
   cout<<"("<<x<<","<<y<<","<<z<<")"<<endl;
   return 0;
}

//pasa de coordenadas rectangulares a cilíndricas
double rectacil(double x=0,double y=0,double z=0){
   r=sqrt((x*x)+(y*y));
   teta=atan(y/x);
   z=z;
   cout<<"("<<r<<","<<teta<<","<<z<<")"<<endl;
   return 0;
}

//pasa de coordenadas cilíndricas a rectangulares
double cilarect(double r=0,double teta=0,double z=0){
   x=r*cos(teta);
   y=r*sin(teta);
   z=z;
   cout<<"("<<x<<","<<y<<","<<z<<")"<<endl;
   return 0;
}

//pasa de coordenadas esféricas a rectangulares
double esfarect(double r=0,double teta=0,double phy=0){
   x=r*sin(phy)*cos(teta);
   y=r*sin(phy)*sin(teta);
   z=r*cos(phy);
   cout<<"("<<x<<","<<y<<","<<z<<")"<<endl;
   return 0;
}

//pasa de coordenadas rectangulares a esféricas
double rectaesf(double x=0,double y=0,double z=0){
   r=sqrt((x*x)+(y*y)+(z*z));
   teta=atan(y/x);
   phy=acos(z/(sqrt((x*x)+(y*y)+(z*z))));
   cout<<"("<<r<<","<<teta<<","<<phy<<")"<<endl;
   return 0;
}
```

```
//pasa de coordenadas esféricas a rectangulares en grados
double esfarectengra(double r=0,double teta=0,double phy=0){
   teta=gradarad(teta);
   x=r*sin(phy)*cos(teta);
   y=r*sin(phy)*sin(teta);
   z=r*cos(phy);
   cout<<"("<<x<<","<<y<<","<<z<<")"<<endl;
   return 0;
}

//pasa de coordenadas rectangulares a esféricas en grados
double rectaesfengra(double x=0,double y=0,double z=0){
   r=sqrt((x*x)+(y*y)+(z*z));
   teta=atan(y/x);
   phy=acos(z/(sqrt((x*x)+(y*y)+(z*z))));
   teta=radagrad(teta);
   cout<<"("<<r<<","<<teta<<","<<phy<<")"<<endl;
   return 0;
}

//pasa de coordenadas esféricas a cilíndricas
double esfacil(double p=0,double teta=0,double phy=0){
   r=p*sin(phy);
   teta=teta;
   z=p*cos(phy);
   cout<<"("<<r<<","<<teta<<","<<z<<")"<<endl;
   return 0;
}

//pasa de coordenadas cilíndricas a esféricas
double cilaesf(double r=0,double teta=0,double z=0){
   double p=0;
   p=sqrt((r*r)+(z*z));
   teta=teta;
   phy=acos(z/sqrt((r*r)+(z*z)));
   cout<<"("<<p<<","<<teta<<","<<phy<<")"<<endl;
   return 0;
}

//pasa de coordenadas esféricas a cilíndricas en grados
double esfacilengra(double p=0,double teta=0,double phy=0){
   phy=gradarad(phy);
   r=p*sin(phy);
   teta=teta;
   z=p*cos(phy);
   cout<<"("<<r<<","<<teta<<","<<z<<")"<<endl;
   return 0;
}
```

```
//pasa de coordenadas cilíndricas a esféricas en grados
double cilaesfengra(double r=0,double teta=0,double z=0){
   double p=0;
   p=sqrt((r*r)+(z*z));
   teta=teta;
   phy=acos(z/sqrt((r*r)+(z*z)));
   phy=radagrad(phy);
   cout<<"("<<p<<","<<teta<<","<<phy<<")"<<endl;
   return 0;
}

//producto punto entre dos vectores
double upuntov(double u1=0,double u2=0,double u3=0,double v1=0,double v2=0,double v3=0){
   //cout<<"u punto v= "<<(u1*v1)+(u2*v2)+(u3*v3)<<endl;
   return ((u1*v1)+(u2*v2)+(u3*v3));
}

//suma de dos vectores u+v
double umasv(double u1=0,double u2=0,double u3=0,double v1=0,double v2=0,double v3=0){
   w1=u1+v1;
   w2=u2+v2;
   w3=u3+v3;
   cout<<"u+v= "<<"<"<<w1<<","<<w2<<","<<w3<<">"<<endl;
   return 0;
}

//resta de dos vectores u-v
double umenosv(double u1=0,double u2=0,double u3=0,double v1=0,double v2=0,double v3=0){
   w1=u1-v1;
   w2=u2-v2;
   w3=u3-v3;
   cout<<"u-v= "<<"<"<<w1<<","<<w2<<","<<w3<<">"<<endl;
   return 0;
}

//vector por escalar k*u
double kporu(double k=0, double u1=0,double u2=0,double u3=0){
   w1=k*u1;
   w2=k*u2;
   w3=k*u3;
   cout<<"k*u= "<<"<"<<w1<<","<<w2<<","<<w3<<">"<<endl;
   return 0;
}

//determina si son ortogonales dos vectores
double vortou (double u1=0,double u2=0,double u3=0,double v1=0,double v2=0,double v3=0){
   w=upuntov(u1,u2,u3,v1,v2,v3);
```

```
   if (w != 0)
   {
      cout<<"No son ortogonales."<<endl;
   }else{cout<<"Son ortogonales"<<endl;}
   return 0;
}

//Calcula el determinante de una matriz A de tamaño 3*3
double detA3(double u1=0,double u2=0,double u3=0,double v1=0,double v2=0,double
v3=0,double w1=0,double w2=0,double w3=0){
   w=u1*((v2*w3)-(w2*v3))-u2*((v1*w3)-(w1*v3))+u3*((v1*w2)-(w1*v2));
   cout<<"Det de A: "<<w<<endl;
   return w;
}

//Calcula la longitud de un vector
double longu(double u1=0,double u2=0,double u3=0){
   w=sqrt((u1*u1)+(u2*u2)+(u3*u3));
   cout<<"longitud de u= "<<w<<endl;
   return w;
}

//Muestra el vector dado normalizado
double normu(double u1=0,double u2=0,double u3=0){
   w=longu(u1,u2,u3);
   cout<<"u normalizado= "<<"<"<<u1/w<<","<<u2/w<<","<<u3/w<<">"<<endl;
   return 0;
}

//Producto cruz entre u y v
double ucruzv(double u1=0,double u2=0,double u3=0,double v1=0,double v2=0,double v3=0){
   w1=(u2*v3)-(v2*u3);
   w2=-1*((u1*v3)-(v1*u3));
   w3=(u1*v2)-(v1*u2);
   cout<<"u cruz v= "<<"<"<<w1<<","<<w2<<","<<w3<<">"<<endl;
   return 0;
}

//Devuelve el triple producto punto de los tres vectores
double tripleprod(double u1=0,double u2=0,double u3=0,double v1=0,double v2=0,double
v3=0,double w1=0,double w2=0,double w3=0){
   w4=(v2*w3)-(w2*v3);
   w5=-1*((v1*w3)-(w1*v3));
   w6=(v1*w2)-(w1*v2);
   w=upuntov(u1,u2,u3,w4,w5,w6);
   cout<<"Triple producto= "<<w<<endl;
   return w;
}
```

```
//Guarda un archivo frecta.csv con los puntos de la recta en el intervalo y usando el paso dado en
2D
int frecta (double a, double m,double to,double tf,double paso) {
   double ft=0;
    ofstream myfile;
    myfile.open ("frecta.csv");
    cout<<"Funcion recta= "<<endl;

   for (float t = to; t < tf; t=t+paso)
   {
      ft=a+(m*t);
      myfile<<t<<","<<ft<<endl;
      cout<<t<<","<<ft<<endl;
   }

    myfile.close();
  return 0;
}

//Guarda un archivo frectapuntovector.csv con los puntos de la recta en el intervalo y usando el
paso dado 3n 3D
int frectapuntovector (double u1=0,double u2=0,double u3=0,double v1=0,double v2=0,double
v3=0,double to=0,double tf=10,double paso=0.1) {

    ofstream myfile;
    myfile.open ("frectapuntovector.csv");

   cout<<"Recta punto vector= "<<endl;
   for (float t = to; t < tf; t=t+paso)
   {
      w1=u1+(v1*t);
      w2=u2+(v2*t);
      w3=u3+(v3*t);
      myfile<<t<<","<<w1<<","<<w2<<","<<w3<<endl;
      cout<<t<<","<<w1<<","<<w2<<","<<w3<<endl;
   }

    myfile.close();
  return 0;
}
```