|  |  |
| --- | --- |
| Class Rocket |  |
| Figura 1      - vectores<T> vector\_velocity;  - vectores<T> posicion;  - T a;  - T height;  - Environment<T> ambiente;  - Earth<T> mytierra;  - T dv;  - T v;  - T latitude;  - T longitude;  - vectores<T> difvector\_velocity;  - T D; // arrastre  - T w;//velocidad angular  - T num\_push ;  - T latitudeb;  - double q;  - T Cl;//coeficiente de sustentacion  - T Cd;//coeficientes de arrastre  - T pressuergases;  - T velocityout;//es la velocidad que los gases estan saliendo  - T velocitysoun;  ////////////////datos iniciales del cohete//////////////////  - double azimut=53; //ángulo de la orientación o dirección del movimiento del cohete sobre el norte de la Tierra.  - double theta=20; //ángulo de trayectoria de vuelo  - double phi; //ángulo de latitud  - double lamnda=121 ; //ángulo de longitud  - double dt; //diferencial del tiempo  - double vb; //velocidad base  - double dough=121; //masa del cohete  - double doughcarga=343; //masa de la carga(satélite)  - double areaall=5454; //área total del cohete  - double area\_nozzle=223; //área de la tobera  - double diameter=245; //diámetro del cohete  - double tip\_length=12; //longitud de punta=cono(modelismo)  - double anguloataque=0;  - double A=(diameter\*diameter\*PI)/4;//reference\_surface  + void position();  + void velocity();  + void acceleration();  + void update\_var();//variables necesarias para el movimiento  + void setlatitud\_longitud(T & latitud ,T & longitud, T & altitud);  + void get\_cdycl();  + T get\_velocidad\_ascenso();  ///gets  + T getdough();  + T getheight();  + T getareaall();  + T getarea\_nozzle();  + T push();//empuje  + T dif\_velocity();  + void movimiento(); | |
|  | |

CL y arrastre CD son valores dependientes de la forma geométrica del cuerpo, el ángulo de ataque y el número Mach, mientras que la superficie de referencia S solo depende del diámetro d del cohete

S = \_d2=4 (Fleeman, 2006)

|  |  |
| --- | --- |
| Rocket : drag |  |
| A, D ∈ ℝ | =densidad del aire  A=superficie de referencia  =coeficiente de arrastre  d=diámetro de la nave  V=velocidad del cuerpo  Rocket.cpp línea 184  template<class T>  T Rocket<T>::drag()  {  get\_cdycl();  this->D=1/2\*ambiente.airDensity\*pow(v,2)\*A\*Cd;  return D;  } |
| D |  |



|  |  |
| --- | --- |
| Rocket : lift |  |
| A, D ∈ ℝ  RocketD | =densidad del aire  A=superficie de referencia  =coeficiente de arrastre  d=diámetro de la nave  V=velocidad del cuerpo  Rocket.cpp linea 192  template<class T>  T Rocket<T> :: lift()  {  return 1/2\*ambiente.airDensity\*pow(v,2)\*A\*C1;  } |
| D |  |

|  |  |
| --- | --- |
| Rocket:movimiento |  |
|  | Rocket.cpp linea 112  template<typename T>  void Rocket<T>::movimiento()  {  acceleration();  velocity();  position();  } |
| void |  |

|  |  |
| --- | --- |
| Rocket:acceleration |  |
| a ∈ ℝ  Rocketa | A ángulo de azimut ver figura 1  ángulo de trayectoria de vuelo  T fuerza de impulso  D fuerza de arrastre  m masa del cohete  ángulo de longitud  velocidad angular  fuerza potencial  Rocket.cpp linea 62  template<class T>  void Rocket<T>::acceleration()  {  num\_push=1;  anguloataque=0;  D=drag();  r=2;  mytierra.get\_W(dough,r,phi);  this->a=num\_push\*cos(anguloataque)/dough -D/dough-mytierra.Wr\*cos(theta)/dough + mytierra.Wphi\*sin(theta)\*cos(azimut)/dough  -pow(w,2)\*r\*cos(phi)\*(sin(phi)\*sin(theta)\*cos(azimut)-cos(phi)\*cos(theta));  cout<<"ACELERACION"<< num\_push\*cos(anguloataque)/dough -D/dough-mytierra.Wr;  } |
|  |  |

|  |  |
| --- | --- |
| Rocket:velocity |  |
| Vector , d  *, ,* ∈ ℝ  d=  d=  d= | y d son la velocidad y diferencial de velocidad.  es el ángulo de Trayectoria.  es el ángulo de ataque.  es el modulo del vector de velocidad actual  Rocket.cpp linea 83  template<class T>  void Rocket<T>::velocity()  {  //aquí llamamos al  acceleration();  dv = a\*dt;  this->v=v+dv;  T dvx=v\*cos(theta);  T dvy=v\*cos(90-azimut);  T dvz=v\*cos(azimut);  this->vector\_velocity.setX(dvx+vector\_velocity.getX() );  this->vector\_velocity.setY(dvy+vector\_velocity.getY() );  this-> vector\_velocity.setZ(dvz+vector\_velocity.getZ() );  cout<<"la velocidad es igual"<<vector\_velocity.getX()<<" "<<vector\_velocity.getY()<<" "<<vector\_velocity.getZ()<<" "<<endl;  cout<<"posicion"<<posicion.getX()<<" "<<posicion.getY()<<" "<<posicion.getZ()<<" "<<endl;  } |
|  |  |

|  |  |
| --- | --- |
| Rocket:posicion |  |
| Vector d,  *, ,* ∈ ℝ | son la velocidad y diferencial de velocidad.  posición actual del cohete  Rocket.cpp linea 102  template<typename T>  void Rocket<T>::position()  {  T dx=vector\_velocity.getX()\*dt;  T dy=vector\_velocity.getY()\*dt;  T dz=vector\_velocity.getZ()\*dt;  this->posicion.setX(posicion.getX()+dx);  this->posicion.setY(posicion.getY()+dy);  this->posicion.setZ(posicion.getZ()+dz);  } |
|  |  |

|  |  |
| --- | --- |
| mach | v, |
| float, float |
| M   |  |  |  |  | | --- | --- | --- | --- | |  | v/velocitysoun>1 | M=true | M=false | | v velocidad actual.  vs velocidad del sonido.  Rocket.cpp linea 18  template<class T>  bool mach(T v,T velocitysoun)  {  if(v/velocitysoun>1)  return true;  return false;  } |
| M |  |

|  |  |
| --- | --- |
| Rocket:get\_cdycl |  |
| q  , , ,   |  |  |  | | --- | --- | --- | |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | |  |  |  |  | | son la velocidad y diferencial de velocidad.  posición actual del cohete  tip\_length o Longitud de la punta  d diámetro  M es el resultado al ejecutar la función mach()  Rocket.cpp linea 143  template<class T>  void Rocket<T>::get\_cdycl()  {  q=(1.0/2.0)\*ambiente.getdensitybyLm()\*vb\*vb;  ///presion dinamica q=S;  T divide =tip\_length/diameter;  //ambiente.getspeedsound();  T numMatch=mach(v,ambiente.speedsound);  cout<<"El numero de match"<<numMatch<<endl;  cout<<dough/q\*tip\_length;  //coeficiente de arrastre se divide en 3 coeficientre de onda ,coeficiente base y coeficiente fricción  T cdf=0.53\*(divide)\*pow(dough/q\*tip\_length,0.2);//coeficiente de fricción por la forma del cuerpo  T cdo=0;  T cdb=0.12+0.13\*pow(dough,2);  if(numMatch)  { if(divide<25 && divide>5)///que pasa en los casos cuando no esta en estos parámetros  cdo=(1586+1834/pow(dough,2))\*pow(1/(tan(0.5/divide)) ,1.69);  cdb=0.25/numMatch;  this->Cd=cdf+cdo+cdb;  }  else  this->Cd=cdf+cdb;  this->Cl=0.53\*(sin(2\*anguloataque)\*cos(anguloataque/2))+(divide)\* pow(sin(anguloataque),2);  } |
|  |  |