These are the methods needed for the constructor to work properly

void setRo(double altitude)

IF altitudeToDensityMap.find(altitude) != altitudeToDensityMap.end()

ro <- altitudeToDensityMap[altitude]

ELSE

ro <- interpolation(altitude, altitudeToDensityMap)

void setGravity(double altitude)

IF altitudeToGravityMap.find(altitude) != altitudeToGravityMap.end()

gravity <- altitudeToGravityMap[altitude]

ELSE

gravity <- interpolation(altitude, altitudeToGravityMap)

void setSpeedOfSound(double altitude)

IF altitudeToSoundMap.find(altitude) != altitudeToSoundMap.end()

speedOfSound <- altitudeToSoundMap[altitude]

ELSE

speedOfSound <- interpolation(altitude, altitudeToSoundMap)

void setMach()

mach <- v / speedOfSound

void setC()

IF machToCMap.find(mach) != machToCMap.end()

c <- machToCMap[mach]

ELSE

c <- interpolation(mach, machToCMap)

void setDrag()

dragForce <- 0.5 \* c \* ro \* v \* v \* area

void setAcceleration()

acceleration <- dragForce / mass

void setDdx()

ddx <- -sin(radians) \* acceleration

void setDdy()

ddy <- -gravity -cos(radians) \* acceleration

double interpolation(double inputMiddle, map<double, double> m)

it <- m.begin()

WHILE it != m.end()

IF it.key > inputMiddle

inputEnd <- it.key

outputEnd <- it.value

it--

inputBegin <- it.key

outputBegin <- it.value

BREAK

it++

IF it == m.end()

it--;

return it->second;

RETURN outputBegin + (inputMiddle - inputBegin) \* (outputEnd - outputBegin) / (inputEnd - inputBegin)