Bond Pricing Model

This is a C++ implementation of a pricing model for a bond that pays periodic coupons of equal size and at equal intervals.

The price of the bond is calculated as follows,

$$P = \frac{C}{n}\frac{\left[1-\left[\frac{1}{\left(1+\frac{r}{n}\right)^{nT}}\right]\right]}{\frac{r}{n}} + \frac{F}{\left(1+\frac{r}{n}\right)^{nT}}$$

where

C = cash flow per period
n = compounding per period
r = interest rate
F = face value of the bond
T = time to maturity (in years)

Duration, which is the first derivative of the price with respect to interest rate, is calculated as follows,

$$D = \frac{\sum_{t=1}^{m}\frac{C*t}{\left(1+\frac{r}{n}\right)^{t}} + \frac{F*m}{\left(1+\frac{r}{n}\right)^{m}}}{P}$$

Convexity, which is the second derivative of the price with respect to interest rate, is calculated as follows,

$$Convexity = \frac{1}{P\left(1+\frac{r}{n}\right)^{2}}\sum_{t=1}^{m}\left[\frac{C}{\left(1+\frac{r}{n}\right)^{t}}(t^2+t)\right]$$

****
The header file, or the interface, for the class **Bond**, which consists of member data and member function declarations.
****

```
//Bond.h

#ifndef BOND_H
#define BOND_H

class Bond
{
     public:
            Bond();
            Bond(double _C, double _n, double _F, double _r, double _T);
            Bond(const Bond& bond2);
            Bond& operator = (const Bond& bond2);
            virtual ~Bond();

            double NPV() const;               //price of the bond

            double pricingCalc() const;       //calculates the price

            /////////functions that calculate some sensitivities////////

            double duration() const;         //Macaulay duration

     public:
             double C;                       //coupon rate
             double n;                       //number of annual coupons
             double F;                       //face value of the bond
             double r;                       //discount rate
             double T;                       //years until redemption
};

#endif
```

```
***********************************************************************
The implementation file for the class Bond, which defines each
function and constructor. Uses Bond.h.
***********************************************************************

//Bond.cpp

#ifndef BOND_CPP
#define BOND_CPP

#include "Bond.h"

#include<iostream>
#include<math.h>

using namespace std;


Bond::Bond()
{
     C = 0.13;
     n = 1.0;
     F = 1000.0;
     r = 0.07;
     T = 10.0;
}

Bond::Bond(double _C, double _n, double _F, double _r, double _T)
                : C(_C), n(_n), F(_F), r(_r), T(_T)
{
}

Bond::Bond(const Bond& b2)
{
     C = b2.C;
     n = b2.n;
     F = b2.F;
     r = b2.r;
     T = b2.T;
}

Bond& Bond::operator = (const Bond& bond2)
{
     if(this == &bond2)
        return *this;

     C = bond2.C;
     n = bond2.n;
     F = bond2.F;
     r = bond2.r;
     T = bond2.T;

     return *this;
}

Bond::~Bond()
{

}

double Bond::pricingCalc() const
{
     double m = n * T;

     double PMT = ((C*F)/n);
     double PVannuity = (1 - (1 / pow((1 + r/n),m))) / (r/n);
     double PVparvalue = F / pow((1+r/n),m);

     return PMT*PVannuity + PVparvalue;
}

double Bond::NPV() const
{
     return pricingCalc();
}

double Bond::duration() const
{
     double P = pricingCalc();
     double m = n * T;
     double Disc = (r/n);

     return ((((C*F)/n)/(pow(Disc,2))*((1-(1/pow((1+Disc),m)))))+(m*(F-
((C*F/2)/Disc))/(pow((1+Disc),(m+1)))))/P/n;

}

#endif
```