Queens College, CUNY, Department of Computer Science

Computational Finance

CSCI 365 / 765

Spring 2018

Instructor: Dr. Sateesh Mane

due date: indeterminate

This is an addendum to Homework 2

2 Homework: Addendum to Bond class

In Homework 2, we wrote a simple `Bond` class. However, the software design has some weak points. In this assignment, we shall modify the `Bond` class to remedy some of those defects. Recall the class signature:

```
class Bond
{
public:
  Bond(double T, double F, double c=0, int freq=2);
  ~Bond();

  // public methods
  int set_coupons(std::vector<double> & c);
  int FV_duration(double t0, double y,
                  double &B,
                  double &Macaulay_duration,
                  double &modified_duration) const;

  int yield(double B_target, double tol, int max_iter, double t0,
            double & y, int & num_iter) const;

  double FairValue(double t0, double y) const;
  double maturity() const { return T_maturity; }

private:
  // data
  double Face;
  double T_maturity;
  int cpn_freq;
  int num_coupons;
  std::vector<double> coupons;
};
```

2.9 `num_coupons`

- The most glaring bad feature is the data member `num_coupons`.
- As one student correctly pointed out, why is the value `num_coupons=0` not allowed?
- **This is an important question.**
- *The answer is that the name* `num_coupons` *is a mistake, it is not really the number of coupons.*
- Recall that a zero coupon bond pays no coupons.
- Nevertheless, to calculate the bond price of a zero coupon bond (given the yield), we still require a parameter n. Recall the formula for a newly issued zero coupon bond with a semiannual frequency (so $n = 2T$)

$$B_{\text{zero coupon}} = \frac{F}{(1 + \frac{1}{2}y)^n}\,. \tag{2.9.1}$$

- Hence `num_coupons` is really the ***number of coupon periods.***
  1. The correct relationship is really

$$\text{number\_coupon\_periods} = \text{maturity} \times \text{coupon\_frequency}\,. \tag{2.9.2}$$

  2. Obviously the number of coupon periods cannot be zero.
  3. That is the real answer to the student's question.
- **Replace "`num_coupons`" by "`num_coupon_periods`" (global string replacement).**

2.10 Negative coupon values

- The class method `set_coupons()` does not allow negative coupon values.
- **However, I forgot to include the same check in the class constructor.**
  1. Consider a bond created with inputs $T = 5$, $F = 100$, $\mathbf{c = -1}$, $f = 2$.
  2. The bond is `Bond bond(T,F,c,f)` or `Bond bond(5,100,-1,2)`.
  3. The code in the constructor will set the coupons to -1, which is bad.

     ```
     if (num_coupons > 0) {
       coupons.resize(num_coupons, c);   // *** BAD CODE ***
     }
     ```

- **Edit the code in the constructor to do the following:**

  ```
  if (num_coupons > 0) {
    if (c < 0.0) c = 0.0;
    coupons.resize(num_coupons, c);   // no negative coupons
  }
  ```

- Note that because of "call by value" we can change the value of c inside the constructor.

2.11 Set flat coupons

- The constructor sets the coupons to equal values ("flat coupons").
- There is a class method `set_coupons()` to set variable rate coupons.
- *But what if we want to modify the coupons to a new value, all equal?*
- We have to create a vector with all equal values and call `set_coupons()`.
- This is obviously not complicated to do, but still . . .
- **Let us add a method "`void set_flat_coupons(double c)`" to the class.**
- It is really just for convenience.

```
void Bond::set_flat_coupons(double c)
{
  if (c < 0.0) c = 0.0;
  std::fill (coupons.begin(), coupons.end(), c);
}
```

2.12 New class signature

```
class Bond
{
public:
  Bond(double T, double F, double c=0, int freq=2);
  ~Bond();

  // public methods
  void set_flat_coupons(double c);
  int set_coupons(std::vector<double> & c);
  int FV_duration(double t0, double y,
                  double &B,
                  double &Macaulay_duration,
                  double &modified_duration) const;

  int yield(double B_target, double tol, int max_iter, double t0,
            double & y, int & num_iter) const;

  double FairValue(double t0, double y) const;
  double maturity() const { return T_maturity; }

private:
  // data
  double Face;
  double T_maturity;
  int cpn_freq;
  int num_coupon_periods;
  std::vector<double> coupons;
};
```