PyFi Methodology

This document is intended to provide an explanation of the methodology used in the PyFi package. This will hopefully make review easier than trying to read code. This document isn't meant to derive or justify any of the formulas or methods - this information can be found elsewhere.

1 Functions

1.1 Present/Future Value of Cash Flows

- **pv(.)** - returns the present value of a series of cash flows.

$$pv = \sum_{t=1}^{n} CF_t * (1 + (apr * dt))^{-t*dt} \tag{1}$$

  A few things should be noted about this formula:

  - CF is the cash flow list that's passed to the function. The code assumes the first cash flow happens at $t = 1$, so CF_t would be `cash_flows[t-1]`.
  - apr is the nominal annual interest rate.
  - n is the total number of cash flow periods. This is equivalent to the length of `cash_flows`.
  - dt is the time, in years, between cash flow periods. Note that $dt * n$ is the total number of years that the cash flows cover.

- **fv(.)** - returns the future value by taking the pv, then bringing it forward in time.

$$fv = pv * (1 + (apr * dt))^{n} \tag{2}$$

1.2 Numerical Approaches

- **irr(.)** - returns the internal rate of return of a series of cash flows. The IRR is the apr that satisfies

$$0 = pv = \sum_{t=1}^{n} CF_t * (1 + (apr * dt))^{-t*dt} \tag{3}$$

  This is achieved numerically by performing the following steps:

  1. Guess an initial apr, and adjust by some factor (initially .1) in one direction until the pv switches signs.

2. When the pv switches signs, divide the adjusting factor by 10 then adjust in the direction that brings the pv closer to 0.
3. Repeat step 2 until $pv = 0$ or the apr is found to 10 decimal places.

Note that at least 1 cash flow must be of another sign than the others, or else $apr = \infty$. All cash flows having the same sign will throw an exception.

1.3 Cash Flow Characteristics

- **macD(.)** - returns the Macaulay duration of a series of cash flows. This is the weighted average maturity of the cash flows.

$$pv_t = CF_t * (1 + (apr * dt))^{-t*dt} \tag{4}$$

$$MacD = \sum_{t=1}^{n} \frac{(t * dt) * pv_t}{pv_t} = \frac{\sum_{t=1}^{n}(t * dt) * pv_t}{pv} \tag{5}$$

- **modD(.)** - returns the Modified duration of a series of cash flows. This is the percentage change in pv for a percentage *point* change in yield, or apr. Technically, it is given by

$$ModD = -\frac{1}{pv} * \frac{\partial pv}{\partial(apr * dt)} \tag{6}$$

However, PyFi uses the following convenient relationship to find it:

$$ModD = \frac{MacD}{1 + (apr * dt)} \tag{7}$$

- **convexity(.)** - returns the convexity of a series of cash flows. This is defined as

$$Convexity = \frac{1}{pv} * \frac{\partial^2 pv}{\partial(apr * dt)^2} \tag{8}$$

For PyFi's purposes, with periodic compounding, this formula becomes:

$$Convexity = \frac{\sum_{t=1}^{n} t * (t + 1) * dt^2 * w_t}{(1 + (apr * dt))^2} \tag{9}$$

where

$$w_t = \frac{pv_t}{pv} \tag{10}$$

is the weight of each particular payment in the total pv. Note that in the code, the values t and $t + 1$ in (9) are instead $t + 1$ and $t + 2$, due to the list indexing beginning at 0.

1.4 Amortization

The `Amortize` class requires the user to enter 4 of the 5 necessary variables for the amortization of a loan, and the program calculates the remaining variable. Except for the interest rate calculation, all of the formulas are rearrangements of

$$PV = PMT * \frac{1 - (\frac{1}{1+i})^n}{i} + FV \left(\frac{1}{1+i}\right)^n \tag{11}$$

Note that the multiplier for PMT is the factor for an annuity-immediate of n periods at rate i.

- PV is the present value. In this code, this is the variable `principal`. Formula (11) shows how this value is calculated.
- PMT is the periodic payment. This value is fixed, with the last period sometimes being an exception in order to meet the desired future value. This is calculated as follows:

$$PMT = \frac{PV - \frac{FV}{(1+i)^n}}{\frac{1-\frac{1}{(1+i)^n}}{i}} \tag{12}$$

- FV is the future value, often set to 0.

$$FV = \frac{PV - PMT * \frac{1-(\frac{1}{1+i})^n}{i}}{\left(\frac{1}{1+i}\right)^n} \tag{13}$$

- n is the number of payments, or the number of periods.

$$n = \frac{\ln(\frac{i*PV-PMT}{i*FV-PMT})}{\ln(\frac{1}{1+i})} \tag{14}$$

- i is the interest rate. This is the rate as it applies to each period - **not** the apr. This is calculated using the **irr(.)** function.

2 Option Pricing

2.1 Binomial Trees

PyFi currently utilizes three main types of binomial models - CRR, Jarrow-Rudd, and Tian - for three types of options - European, American, and Bermudan. It also supports Binary options, by replacing the payoff formulas appropriately.

The binomial models are developed by matching the moments of the model to the moments of the stock movement given that it is lognormal. The result is

a system of two equations with three unknowns: p^*, u, and d.

$$p^*u + (1-p^*)d = e^{(r-q)\Delta t} \tag{15}$$

$$p^*u^2 + (1-p^*)d^2 = e^{2(r-q)\Delta t+\sigma^2\Delta t} \tag{16}$$

Each of the three models use a different third condition to solve the above.

See http://www.mimuw.edu.pl/~apalczew/CFP/_lecture1.pdf for an easy and intuitive explanation of this system.

The binomial tree algorithm used by PyFi is as follows:

1. Find u, d, and p^*
2. Generate the stock price tree
3. Using the final nodes of the stock price tree, determine the option values at expiry
4. At each previous node, calculate the discounted risk-netutral expected value as
$$V = e^{-(r-q)\Delta t}(p^*V_u + (1-p^*)V_d) \tag{17}$$
where V_u and V_d are the option values at one up and down step from the node, respectively.
5. If early exercise is allowed, compare the option value with the exercise value and take the max.
6. Repeat steps 4-5 until the initial node price is calculated.

2.1.1 CRR

The Cox-Ross-Rubinstein model uses the following third equation:

$$u = d^{-1}$$

which results in the following solutions for the unknowns:

$$p^* = \frac{e^{(r-q)\Delta t} - d}{u - d} \tag{18}$$

$$u = e^{\sigma\sqrt{\Delta t}} \tag{19}$$

$$d = e^{-\sigma\sqrt{\Delta t}} = u^{-1} \tag{20}$$

Note that in the CRR model, $S_{ud} = S_{du} = S_{init}$ throughout the tree, making it symmetric.

2.1.2 Jarrow-Rudd (JR)

The Jarrow-Rudd model is often called the "equal probability" model, since it uses

$$p^* = \frac{1}{2}$$

as the third equation to solve the system. The complete model is:

$$p^* = \frac{1}{2} \tag{21}$$

$$u = e^{(r-q-\frac{1}{2}\sigma^2)\Delta t+\sigma\sqrt{\Delta t}} \tag{22}$$

$$d = e^{(r-q-\frac{1}{2}\sigma^2)\Delta t-\sigma\sqrt{\Delta t}} \tag{23}$$

Note that since $ud \neq 1$, the tree is not symmetric.

2.1.3 Tian

The Tian model matches the third moment of the movement with the lognormal model, so the third equation is

$$p^*u^3 + (1-p^*)d^3 = \left(e^{(r-q)\Delta t}\right)^3 \left(e^{\sigma^2\Delta t}\right)^3 \tag{24}$$

and allowing $X = e^{\sigma^2\Delta t}$, we arrive at the following model:

$$p^* = \frac{e^{(r-q)\Delta t} - d}{u - d} \tag{25}$$

$$u = \frac{1}{2}e^{(r-q)\Delta t}X\left(X + 1 + \sqrt{X^2 + 2X - 3}\right) \tag{26}$$

$$d = \frac{1}{2}e^{(r-q)\Delta t}X\left(X + 1 - \sqrt{X^2 + 2X - 3}\right) \tag{27}$$

2.1.4 Greeks

The "Greeks" for the options can be approximated in a binomial environment by converting the partial derivative associated with that Greek into a discrete approximation. Below are the formulas used for the three Greeks calculated by PyFi for each binomial option:

$$\Delta = \frac{\partial V}{\partial S} = \frac{V_u - V_d}{S_u - S_d} \tag{28}$$

$$\Gamma = \frac{\partial^2 V}{\partial S^2} = \frac{\frac{V_{uu}-V_{ud}}{S_{uu}-S_{ud}} - \frac{V_{du}-V_{dd}}{S_{du}-S_{dd}}}{S_{uu} - S_{dd}} \tag{29}$$

$$\Theta = \frac{\partial V}{\partial dt} = \frac{V - V_{ud}}{2 * dt} \tag{30}$$