Pricing Asset Backed Securities – Anuj Godhani

Part 1 – 2

- Implements the actual asset-backed securities (the liabilities) in addition to the Waterfall mechanism that calculates the cashflows at each time period for each loan. The objective is to create well-designed tranche classes which will seamlessly work with Loan classes.
- Implement metrics on the Waterfall. This includes Internal Rate of Return (IRR), Reduction in Yield (DIRR), and Average Life (AL). The objective and outcome is to be able to calculate and provide useful metrics on the structure

tranche.py under ABS

Tranche()	Class can be found in tranche.py Takes notional, notional percent, rate and subordination level upon initialization It sets flag to sequential. There's a method to set the flag after creating the class object. The following methods are present • IRR • DIRR • AL
StandardTranche()	Class derived from tranche class Upon initialization set current time = 0 current notional balance = notional current interest due = 0 current interest shortfall = 0 current principal due = 0 current principal shortfall = 0 current interest paid = 0 current principal paid = 0 principal collections = 0
increaseTimePeriod()	this method increases time period by 1 and changes the related variables that changes with time • Increase current time period by 1

	• set current interest due to (rate * notional balance of the last period), + any interest shortfall from the last period • set interest shortfall, interest and principal paid to 0 because we haven't paid anything with this method • **the principal due is defined based on flag.** And it also depends on principal collections which we will set in waterfall.
makePrincipalPayment(amount)	this method makes principal payments and returns the amount left after making a payment. It also sets principal shortfall as the difference between principal due and payment made if the notional balance is 0, we don't need to pay anything. Else, current principal paid to min of balance and amount We then reduce the current notional balance by the amount of principal paid
makeInterestPayment(amount)	Makes interest payments and returns the amount left after making the payments. Its works like the makePrincipalPayments function. Sets current interest shortfalls if any
notionalBalance()	returns the amount of notional still owed to the tranche for the current time period
interestDue()	returns the amount of interest due for the current time period. Its calculated when we change the time period
reset()	Resets everything back to the initialization values

securities.py under ABS

StructuredSecurities()	Class can be found in securities.py Initialized with total notional amount manage all tranches in a list set the mode to 'Sequential'. Can be changed with a method maintain a reserved account that is cash left after making payments
addTranche(cls, ntl_per, rate, flag)	this method adds tranches to our list

	takes the class which should be a Tranche class object, rate, subordination level and notional percent as arguments saves the tranche class object and appends the tranche list
changeMode(mode)	This changes the mode. Options are Sequential or Pro-Rata
increaseTimePeriod()	Increases time period for all tranches in our list
makePayments(cash_amount, principal collections)	First, keep track of cash we have Then loop through the tranches and pay interests first. We have already sorted our list based on subordination level If there is any cash left, pay the principal as per the mode. The function takes principal collections as a parameter because our principal payments function requires it. Store the reserved account if any cash left
getWaterfall()	Returns a list of lists with each list containing Interest Due, Interest Paid, Interest Shortfall, Principal Paid, Balance for each tranche for a given timeperiod

Pool.py under loans

poolCSV(filepath)	This function can be found in pool.py. It reads the CSV file and creates a loan pool object

Waterfall.py under ABS

doWaterfall(loanPool, security)	Function Can be found in waterfall.py Returns tranche metrics, loan pool waterfall, securities waterfall and reserve account for each time periods till all loans mature

Part 1 and part 2 have been tested in test1-2.py

Part 3

The last part is to value and rate the ABS. This entails creating a Monte Carlo simulation to simulate thousands of different credit default scenarios, all of which help determine the rating of the structure. The outcome will be a rate, rating, and Weighted Average Life (WAL) for each tranche of our very simple structure.

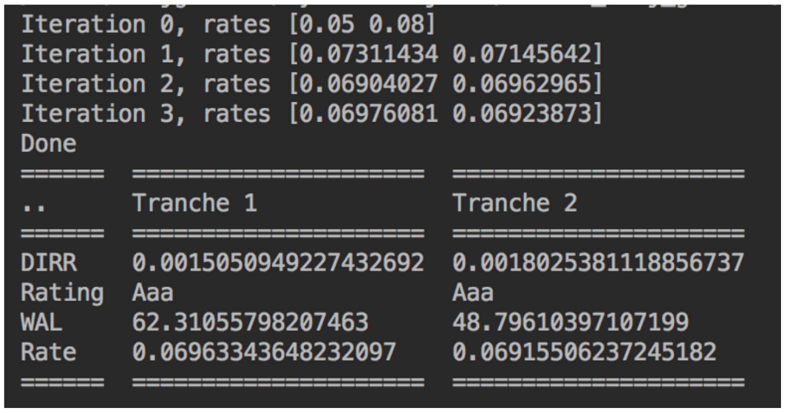
checkDefaults(flag)	Changes have been made to loan_base.py If flag is 0 sets the loan as defaulted.
checkDefaults(n)	Changes have been made to pool.py For a given time period, it assigns defaults to a loan and returns total recovery value for the defaulted loans. **Changes have also been made to waterfall.py**

Montecarlo.py

simulateWaterfall(loan pool, securities, NSIM)	Found in montecarlo.py Runs waterfall NSIM times and returns sum of DIRR and AL from all simulations
makeSecurities(notional, percent, rates, sub-level)	Takes parameters as lists and creates securities. Becomes easier to implement.
calcYields(dirr, wal)	Calculates yield as per the given formula
runSimulationParallel(loanpool, securities, NSIM, Num_processes = 20)	Creates processes and divides the simulations into different processes. Collects the results from output queue. Sums them up and returns average DIRR and AL across all the simulations
runMonte(loanpool, NSIM. Tolerance)	Calls runSimulationParallel takes the result and calculates yield. Based on these yields it calculates new tranche rates. If the new tranche rates differ by more than tolerance, rates are modified and the procedure is repeated.

Part 3 has been tested in test3.py

Final Results



```
Iteration 0, rates [0.05 0.08]
Iteration 1, rates [0.07311434 0.07145642]
Iteration 2, rates [0.06904027 0.06962965]
Iteration 3, rates [0.06976081 0.06923873]
Done
======  =====================  =====================
..      Tranche 1              Tranche 2
======  =====================  =====================
DIRR    0.0015050949227432692  0.0018025381118856737
Rating  Aaa                    Aaa
WAL     62.31055798207463      48.79610397107199
Rate    0.06963343648232097    0.06915506237245182
======  =====================  =====================
```