Homework 5

- This homework consists of three questions. Your code will be graded by a Python script which compares your result against the baseline result. It is therefore important that your Python stack is identical to the baseline so that you avoid losing marks even though your code is correct.
- Submit a python file, `CreditModeling.py` and a short report describing your implementation approach for each question and your result in Question 3. Archive your python file and short analysis report into one file.
- Be sure to include your name as part of the archive filename, and at the top of your report.

1. Question 1

Implement the Bates Characteristic Function and modify the calibration code provided in class to calibrate the Bates model. Your code, implemented in the file `BatesCalibration.py` should use the `readTDAmeritradeData` function to read in the S&P500 Option Chain data `SPX_08082013.csv`. Demonstrate that your code works by running it with at least 32 option contracts. Here is the implementation of the `BatesCOS` pricing function and its characteristic function.

```
def BatesCOS(S,K,T,r,sigma,lmbda,meanV,v0,rho,a_prime,b_prime,lmbda_prime,otype,N=256):
    c1 = r*T+(1-exp(-lmbda*T))*(meanV-v0)/(2.0*lmbda)-0.5*meanV*T
    c2 = 1.0/(8.0*lmbda**3)*(sigma*T*lmbda*exp(-lmbda*T)*(v0-meanV)*(8.0*lmbda*rho-4.0*sigma)+
    lmbda*rho*sigma*(1-exp(-lmbda*T))*(16.0*meanV-8.0*v0)+
    2.0*meanV*lmbda*T*(-4.0*lmbda*rho*sigma+sigma**2+4.0*lmbda**2)+
    sigma**2*((meanV-2.0*v0)*exp(-2.0*lmbda*T)+
    meanV*(6.0*exp(-lmbda*T)-7.0)+2.0*v0)+8.0*lmbda**2*(v0-meanV)*(1-exp(-lmbda*T)))
    a = c1-12.0*sqrt(abs(c2))
    b = c1+12.0*sqrt(abs(c2))
    x = log(float(S)/K)
    k = np.arange(0,N)
    if (otype == 'C'):
       U = 2.0/(b-a)*(xi(k,a,b,0,b) - psi(k,a,b,0,b))
    else:
       U = 2.0/(b-a)*(-xi(k,a,b,a,0) + psi(k,a,b,a,0))
    unit = [1.0] * N
    unit[0] = 0.5
    ret = 0
# Note that BatesCF is independent of the strike
    BCF = BatesCF(k*pi/(b-a),T,r,sigma,lmbda,meanV,v0,rho,a_prime,b_prime,lmbda_prime)
    for i in range(N):
      ret += unit[i]*BCF[i]*exp(1j*float(k[i])*pi*(x-a)/(b-a))*U[i]
    return K*exp(-r*T)*ret.real
```

```
def BatesCF(u,T,r,sigma,lmbda,meanV,v0,rho,a,b,lmbda_prime):
    ret = HestonCF(u,T,r,sigma,lmbda,meanV,v0,rho)
    ret *= exp(lmbda_prime*T*(-a*u*1j + (exp(u*1j*log(1.0+a)+0.5*b*b*u*1j*(u*1j-1.0))-1.0)))
    return ret
```

2. Question 2

Write a function

`def defaultTimes(hazardRates, correlation, samples):`

which returns a $M \times N$ matrix of simulated default time arrays using M samples for each N securities, where:
'hazardRates' is an N-element array containing time independent hazard rates applicable to the N entities. Hazard rates should be greater than or equal to zero.
'correlation' is an $N \times N$ correlation matrix, or a constant correlation[1] to use. Correlations should fall in the range of $[-1, 1]$.
'samples' is a two-dimensional $M \times N$ array of random uniform ($U[0, 1]$) samples or a count M of N-dimensional sample.
~~'r' is the risk free rate used for discounting. vectors to generate.~~

Your code should verify that input values are valid and throw an exception if this is not the case.

3. Question 3

Write a function

```
def portfolioLosses(notionals, maturities,
        recoveryRates, hazardRates, correlation, samples, r):
```

which returns the expected loss of the portfolio where:

- 'notionals' is an N-element array containing the notional exposures to the N securities.
- ' maturities' is an N-element array of the maturity of each of the N securities. This value should be greater than zero.
- 'recoveryRates' is an N-element array containing time independent recovery rates in the range $[0, 1]$, as applied at the default times.
- 'hazardRates' is an N-element array containing time independent hazard rates applicable to the N entities. Hazard rates should be greater than or equal to zero.
- 'correlation' is an $N \times N$ correlation matrix, or a constant correlation to use. Correlations should fall in the range of $[-1, 1]$.
- 'samples' is a two-dimensional $M \times N$ array of random uniform ($U[0, 1]$) samples or a count M of N-dimensional sample.

[1] A constant correlation matrix with parameter ρ on N entities is a correlation matrix with every off-diagonal element equal to ρ and 1 one on the main diagonal. Such matrices are used as a way of dealing with the essentially unknowable correlations between underlying normal variables of copula models. The relationship between tranche price and implied (constant) correlation is analogous to the relation between option prices and implied volatilities.

- 'r' is the risk free rate used for discounting.

Your code should use `defaultTimes` from Question 2 and handle all degenerate input values by throwing exceptions.

Use your code to calculate the net present value of the expected loss of a portfolio over a three year period with six corporate bonds, each expiring in three years, with recovery rates $[0.1, 0.3, 0.15, 0.4, 0.4, 0.4]$, hazard rates $[0.3, 0.03, 0.05, 0.1, 0.0010, 0.9]$. You may assume that the notional of each bond is \$1MM, $\rho = 0.4$ and the risk free rate $r = 0.01$. Use $M = 10,000$ simulation paths.