CVA Crib Sheet

CVA is a hard Big Data and a hard Big Compute problem.

It's not as if CVA is new, it's been around for more than a decade. What changed things was the recent financial crisis when things changed. We went from seeing almost no credit risk, to seeing credit risk everywhere; now we have CVA (Counterparty Valuation Adjustment), DVA (Debt Valuation Adjustment), FVA (Funding Valuation Adjustment), and LVA (Liquidity Valuation Adjustment); indeed we are losing count of XVA (fill in the "X") adjustments. In short, banking and finance now sees credit risk under every rock and stone in sight—lift up any rock and you will find credit risk lying underneath so it seems!

For Assignment B you will be implementing CVA only, and using only a very simple model. The framework you build will be simple, but it will have the data and computational challenges inherent in doing CVA. But, have no doubt, your solution could be reworked to do full CVA in a richer and more realistic fashion and for that reason it is a very valuable skill.

Now, on to the technical aspects of simplified CVA.

Time	0	t_1	t_2	t_3	t_4	.....	$t_n=T$
Default probability		q_1	q_2	q_3	q_4	.....	q_n
PV of expected net exposure		v_1	v_2	v_3	v_4	.....	v_n

$$\text{CVA} = (1-R)\sum_{i=1}^{n} q_i v_i \qquad \text{where } R \text{ is the recovery rate}$$

The formula for DVA, incidentally, is very similar except that the default probability is that of the bank rather than the counterparty.

For this assignment we are not modeling the probability of default from one time step to the next along a simulation path in any sophisticated way. Rather we're using a "hazard rate" as a way of calculating default probabilities using these (approximate) formulas:

Probability of default at time step i: $P(t_i) = e^{-h*t_i}$

Probability of default between time step i and i+1: $q_i = P(t_i) - P(t_{i+1})$

All of this, of course, leads us to the steps needed to do Assignment B, bank-wide (or enterprise-wide) CVA:

1) Generate the synthetic bank data. You will need this to test your program.
2) On startup, your program will read the simulation & model parameters and store them in data structures (for example a C/C++ structs).
3) Your program will also read the counterparties, deals and whatever other data you have used to model a simple bank. You will also need to pull in the current spot EUR/USD as the starting point for your simulation.
4) Transfer what data you need to the GPU (device memory). Once it's there, leave it there and do as much work on it before bringing it back to the host machine.
5) Your program will now enter the Monte-Carlo simulation phase of the problem.
6) You will value all the deals in all future states-of-the-world (in this case it's very simple as you must model the evolution of just one asset, EUR/USD) along a simulation path. Choose a suitable number of time steps, say monthly, over the 5 year horizon. (Tip: make NUM_TIMESTEPS another parameter.)
7) For each counterparty at each time step you will work out the net exposure. Because each counterparty is it's own netting set (netting sets are important because it makes CVA *additive*) in this assignment, you simply take all the mark-to-market values of the deals held by a counterparty and net them of against one another to get a net exposure (+ve if the counterparty owes the bank in the case of default of the counterparty, -ve if the bank owes the counterparty in the case of it's own default). Note that you will not need a hazard rate (that drives default probability) for the bank because in this assignment we are not doing DVA.
8) From the exposures you can calculate the CVA values along a simulation path, and discount them back to time zero for each

counterparty. As this point you have a vector of CVA values of length NUM_COUNTERPARTIES for a *single path*.

9) Now repeat steps 6 through 8 for all the other simulation paths.
10) At this point you have the CVA values for all counterparties over all simulation paths. Now you need to average them over the number of simulation paths.
11) Finally, you reduce the CVA numbers over all counterparties to arrive at an aggregate CVA for the whole bank. Don't forget to multiply the final answer by (1 – R) to account for the recovery rate.

Remember that parameters and data can be hard-wired in your program, say, in a header file; or you can pull them in through a parameters file. Note that you will get extra credit for using a parameters file as it makes for a much more flexible solution.