Finding NIMo

Jonathan Sandberg
Kinetix Trading Solutions
Princeton, New Jersey
jonathan.sandberg@kinetixtt.com

Summary
The Board of Governors of the Federal Reserve System, via the Comprehensive Capital Analysis and Review (CCAR) program, has opened up a $100+ Bn annual market in optimizing the Net Interest Margins at big banks in expected case scenarios. This note describes a novel Control Theory implementation of Net Interest Margin Optimization (NIMo) most relevant to banks with 1+ trillion dollar balance sheets and subject to CCAR regulatory reporting. We are working with Cray Inc. to host this computation on a large-scale, parallel, competitive, contemporary hardware platform transitioning from AVX2 on Haswell to AVX 3.2 on Skylake.

1. Introduction

For the first time, the trifecta of CCAR, "free" Floating Point arithmetic, and parallel Numerical Optimization makes it feasible to automate the Firm capital allocation plan implementation at the Accrual Portfolio security level. Control loop feedback can come from anywhere in the capital allocation process, from regional branches exceeding, missing, or delaying their new business quotas to market shocks such as CNY devaluation or the Swiss Central Bank abandoning the EUR peg.

All large U.S. Banks have been running CCAR development and U.S. Federal Reserve reporting programs in the past several years. These CCAR projects are cleaning up the centralized global balance sheet data (positions, indicatives, market data) inside the banks so that the worst-case liquidity scenarios reported to the Fed make sense. The novel idea presented in this note is to make CCAR sunk costs produce operational efficiency and more revenue in banks with large balance sheets. We propose simulating the Firm's full Balance Sheet, in the expected case rather than the worst case, to direct automated global capital allocation implementation.

"Free" Floating Point has been the label recently applied to contemporary commodity processor codes executing dozens of floating-point instructions per core every clock, on average. Typically you need vector code issued from your compiler to get the "free" FLOPS. Historically, this has only been sporadically accomplished in Wall Street shops. This fact is not lost on Intel with their new Code Modernization effort. Despite the recent worries about Skylake delays demonstrating the end of Moore's Law, there may still be some wave to ride from the perspective of certain types of FP performance. The Golden Age of Floating Point computing could continue to increase FLOPS supply through 2020. Another 8x in throughput from vector issue on a commodity processor or through on chip GPUs could be on the drawing board. One of the things you can do with these "free" FLOPS is make CCAR Full Balance sheet simulation faster on a small commodity microprocessor core

footprint. Assume for a moment that the Accrual Portfolio security's balance, rate of return, and default models can be improved to pull down the expected case approximation error. For example, you can use standard Mortgage Backed Security prepayment model techniques applied to pools of deposits and credit card accounts. Then there are enough "free" FLOPs to simulate a full multi-trillion dollar balance sheet security-by-security for a stochastic market model (e.g., LMM) Monte Carlo derivation of the expected Net Interest Margin or Net Interest Revenue. That is, of course, provided your Firm is on-board with the Intel Code Modernization idea.

Net Interest Margin (NIM) is the difference between the rate of return of Liabilities and Assets on the Balance Sheet. NIM Optimization is a natural problem to attack given a fast balance sheet simulator. Why is that?

Figure 1: St Louis Fed – Net Interest Margin for all U.S. Banks

NIM is reported in basis points or hundredths of a percent. Figure 1 shows the St. Louis Fed data for Net Interest Margin for all U.S. Banks. It is currently around a 30 year low just above 300 bps. So the bank's core business of funding commercial and consumer loans with deposits is generating a historically small return.

Current Rank	Prev. Rank	Bank	Assets USD m	Loc ccy + Or -	Capital USD	BS Date
1	1	Industrial & Commercial Bank of China Limited, China	3,124,474	7.84%	58,035.91	31.12.13
2	2	China Construction Bank Corporation, China	2,537,402	9.95%	41,292.05	31.12.13
3	3	BNP Paribas SA, France	2,474,078	-5.61%	36,849.92	31.12.13
4	4	Agricultural Bank of China Limited, China	2,405,091	9.95%	53,643.29	31.12.13
5	5	Bank of China Limited, China	2,291,492	9.41%	46,140.19	31.12.13
6	6	Deutsche Bank AG, Germany	2,214,678	-20.32%	3,587.14	31.12.13
7	7	Barclays Bank PLC, UK	2,173,936	-11.82%	3,977.48	31.12.13
8	8	Crédit Agricole SA, France	2,112,250	-4.98%	10,314.73	31.12.13
9	9	Japan Post Bank Co Ltd., Japan	1,961,701	1.34	33,903.79	31.03.14
10	11	JPMorgan Chase Bank National Association, USA	1,945,467	2.57%	1,785.00	31.12.13
11	10	The Bank of Tokyo-Mitsubishi UFJ Ltd, Japan	1,760,014	7.32%	16,583.39	31.03.14
12	12	Société Générale, France	1,697,721	-1.25%	1,371.63	31.12.13
13	13	The Royal Bank of Scotland plc, UK	1,688,912	-20.58%	10,943.86	31.12.13
14	14	BPCE, France	1,544,145	-2.09%	22,251.24	31.12.13
15	15	Banco Santander SA, Spain	1,533,312	-12.13%	7,788.62	31.12.13
16	16	Sumitomo Mitsui Banking Corporation, Japan	1,518,269	3.58%	18,776.46	31.03.13
17	17	Mizuho Bank Ltd , Japan	1,437,609	77.82%	13,600.89	31.03.14
18	18	Bank of America NA, USA	1,433,716	-2.74%	3,020.00	31.12.13
19	19	Lloyds TSB Bank Plc, UK	1,427,395	-9.50%	2,606.39	31.12.13
20	20	Wells Fargo Bank NA, USA	1,373,600	8.49%	519	31.12.13

Figure 2: Top 20 Global Banks by Assets (Acuity)

What is a basis point of NIM worth in dollars? That depends on the size of your balance sheet. Figure 2 shows the Top 20 Global banks by Assets. In the Balance Sheet the Assets must be offset by the Liabilities therefore the NIM rate multiplied by the Assets tells you the Net Interest Revenue. A basis point is worth more or less $150MM to 250MM to the banks listed in Figure 2, right?

How much variability is there in the NIM reported by the banks historically? The graph shown in Figure 3 shows there is a wide dispersion in the historically reported NIM figures. For example, in 2005 there was about 100bps of NIM dispersion between banks of various sizes. NIM levels appear to converge during market crisis such at 2002 and 2008-9 and then disperse, again.

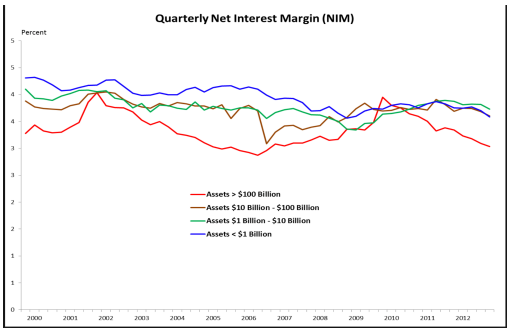

Figure 3: Historical Net Interest Margin (FDIC, Gruenberg, 2013)

The idea in this note is to provide an alternative to banks currently steering a rocket with a joystick. We propose to implement a control loop to numerically optimize NIM/NIR/Full Bank Balance Sheet/Asset & Liability at the security level in the Accrual Portfolio. We are still gathering hard data but we plausibly expect to automate Bank NIM Growth on the order of 10 bps per annum. Moreover there appears to be sufficient FP capacity to numerically optimize NIM using all the target market's assets (e.g., start with U.S. Fed data). The applications of this technology are abundant.

2. Previous Work

The most relevant literature for Net Interest Optimization concerns the elements of the trifecta: U.S. Fed. CCAR Balance Sheet Simulation, "Free" FP in Finance, and Optimization and Control Theory.

U.S. Fed CCAR Balance Sheet Simulation

This work arose from the detailed code optimization of the CCAR Balance Sheet simulation for interest rate forecasting on a client bank's accrual portfolio. We found that if the balance and rate of return models were simply multivariable regression fits (as a computational form) then the entire five-year balance sheet simulation code could be optimized to run in about one second of elapsed time on a single Haswell core. You simply need to strip out the IO and vectorize the inner loop code in an existing large distributed simulator for a $1+ trillion Balance Sheet. It takes significant effort to implement such an optimization, so why would you do that? Well, if you can simulate the entire balance sheet security-by-security with that sort of frequency you can optimize it with a Cray XC, for example. Hanweck and Ryu 2005 discuss the dynamics of NIM to credit, interest rate, and term structure shocks.

There is extensive documentation of CCAR from the Board of Governors of the Federal Reserve System. McKinsey has a good discussion of U.S. Bank Liquidity and funding. Both McKinsey and van Deventer at Kamakura have detailed recent discussions of Transfer Pricing systems used for running balance sheets.

The literature for Accrual Portfolio balance, rate of return, and default models is plentiful from the worst-case risk management perspective but less abundant from the average case modeling perspective. You effectively want prepayment model literature for credit cards, mortgages, and demand deposits like you see in Hayre or Reel. It is plausible that banks have their own coarse granularity balance models for deposit and credit card pools in their current capital allocation process. How else do they analyze the trade offs in their current capital allocation planning? Given the multiple years of CCAR development and the effort required to clean the accrual portfolio data, there is no reasonable scenario where any bank currently optimizes NIM security-by-security through the entire Accrual Portfolio.

	Q1 2011	Q1 2012	Q2 2012	Q3 2012	Q4 2012	Q1 2013	Q2 2013	Q3 2013	Q4 2013	Q1 2014	Q2 201
U.S. Bancorp	3.69%	3.60%	3.58%	3.59%	3.55%	3.48%	3.43%	3.43%	3.40%	3.35%	3.27%
Wells Fargo	4.05%	3.91%	3.91%	3.66%	3.56%	3.48%	3.46%	3.38%	3.26%	3.20%	3.15%
Citigroup	2.88%	2.90%	2.81%	2.86%	2.93%	2.88%	2.85%	2.81%	2.88%	2.90%	2.87%
Bank of Americ	2.66%	2.50%	2.20%	2.31%	2.34%	2.36%	2.35%	2.33%	2.44%	2.29%	2.22%
JPMorgan	2.89%	2.61%	2.47%	2.43%	2.40%	2.37%	2.20%	2.18%	2.20%	2.20%	2.19%

Figure 4: Historical US Bank NIM (Forbes)

Moreover, look at Figure 4 showing the clustering behavior of the U.S. Bank NIM historically. Given the reserves and settlement payments stemming from the Credit Crisis, London Whale, Libor fixing, and other issues there is no player here who is obviously optimizing NIM at the expense of their competitors. The Bank of England's chief economist, Andy Haldane, said in 2009 that*: "there is not a scrap of evidence of economies of scale or scope in banking -- of bigger or broader being better."* The good news is NIMo can fix that by making large banks more revenue as well as automating a key part of retail banking to make it safer and more transparent.

Free FP in Finance

Many Finance computations fall into a class of computations designated as "simple expression evaluation." That means, informally, that the quantities being computed are derived from polynomial evaluations. Common Fixed Income security valuation methodologies including: Black Scholes, Garman-Kolhagen, vanilla CDS and swaps, US Treasuries, and most Corporate bonds amount to simple expression evaluation. The typical Taylor Theorem approach to risk evaluation extends the range of simple expression evaluation to first and second order market risk and P&L explanatories. Throw in a Monte Carlo simulation on top of the simple expression evaluation valuation and risk, and you get a computational characterization of the various popular portfolio level Firm Risk methodologies, including VaR and CVA. They are all various forms of involved, but nevertheless simple, expression evaluation. If you don't have to confront much embedded option exercise, principle component

decomposition, or a Crank –Nicolson pde solver in the inner loop of your program, the execution profile may look effectively like simple expression evaluation.

Expression evaluation is interesting to the code optimization process because it implies a certain type of execution profile on contemporary microprocessors, GPUs, and FPGAs. On commodity microprocessors, for example, expression evaluation means you can write code that will typically only suffer compulsory cache misses in the multilevel memory hierarchy. You only wait, on average, a small number (1+) of microprocessor core cycles for each of the program's memory accesses in the inner loop. The superscalar and superpipelined Fused Multiply Add (FMA) units can be automatically scheduled by the compiler with few bubbles/stalls from straightforward user code. For example, Hager recently figured how to get this vectorized code execution from C++ templates in the Blaze library.

Given what we know about simple expression evaluation, programmers with optimizing compilers can generate code for commodity processors that saturate the FMA FP execution units in each of the cores of the microprocessor. The microprocessor can execute simple expression evaluation FLOPS at close to peak speed. Depending on the clock frequency, throughput of 32 sp FLOPs per 4GHz clock per core are common. At one level, this is one of the thrusts of Intel's Code Modernization drive. The idea is to teach more programmers how to generate optimized/competitive performing FP code. The standard relevant references are Hennessey & Paterson, Muller et.al., Higham, Goldberg, and Kahan.

2015 is the Golden Age of Floating Point. Cheap commodity microprocessors are delivering 2x improved vectorized FP performance in every other generation of relatively inexpensive flagship chips. Colwell's keynote presentation at HotChips 25 outlines the recent history and sets some near term expectations (circa 2013). We have raced through SSE, AVX, to AVX2 FP x86 instruction set extensions in the current Haswell chips in past few years. We are expecting AVX 3.2 in Skylake to provide another doubling of raw FP execution performance per core. AMD is talking about pulling the GPU on chip. The absurdity of all this is that, our current "free" FP is about to get even less expensive in the next couple generations of microprocessors. Intel is pushing the Code Modernization program, more or less just the way Colwell predicted in his 2013 Hot Chips talk. Several researchers are actively investigating high performance FP in Finance relevant to Risk computations, volatility model calibration, and Monte Carlo optimizations. See Albanese et.al., Dixon et.al., or Giles.

Optimization and Control Theory

Mittlemann maintains a website named Benchmarks for Optimization Software that, among other things, lists contemporary execution times for benchmark serial and parallel Linear Programming codes available commercially. Our preliminary estimates start out at about half a million variables for a full accrual portfolio. We

expect to use sparse Simplex to implement NIR optimization initially. Standard commercial vendors include CPLEX, Mosek, and Gurobi. Klein and Neira describe a distributed memory implementation of Nelder-Mead Simplex that could be applicable if our required variable count increases.

3. Balance Sheet Control Theory

The computational runtime of a NIMo run is divided between 1) a full balance sheet Monte Carlo simulation of the runoff and new investment portfolios out to the simulation horizon; and 2) selecting the optimal timing and size of the new investments via NLP/LP. Assuming there are just on the order of 10K new investments, the full balance sheet MC simulation will dominate the overall NIMo runtime 100:1. Estimate assuming less than 10mm runoff positions; daily simulation periods; 10K paths in the Monte Carlo; we estimate somewhat optimistically $\sim$2000 seconds on 1000 cores assuming reasonable parallel scaling and tuned code. Note the NLP/LP optimization execution runtime does not depend on the number of securities in the Accrual Portfolio. This is important because the expected asymptotic runtime complexity starts out as cubic. The NLP/LP runtime depends on the cardinality of the set of parameterized new investments contemplated in the discretionary capital plan. The Control Theory loop surrounding NIMo looks for attenuation in the implementation of the capital plan; computes explanatories between theoretical/model and actual/realized values; and amends the capital plan accordingly for the next simulation periods.

There are a couple obvious applications of a Control Theory framework for Capital allocation. First, use a given capital allocation plan specification and let NIMo determine the explanatories for the realization of the given capital allocation plan. For example, a bank could begin to accumulate historical performance information on all its retail braches and compensate in the capital allocation planning process for any historically documented variation at the security level. For example, if upstate New York mortgages are consistently running light of the planned levels we can use that information allocating capital going forward to reduce variance from the theoretical optimal NIM.

The second application is to use NIMo to amend a capital allocation plan after a large surprise market event invalidates some of the assumptions made in the previous capital allocation plan.

4. One Period Model

We ran a one period LP feasibility model with an accrual portfolio consisting of a mortgage pool and a cards pool. Figure 5 shows the sensitivity of the toy model to the approximation error in the runoff portfolio models; the optimal selection of new investments; and the variations to the capital plan implementation. This one period model should be extended to a multi-period model to better simulate the capital plan implementation.

Runoff portfolio consists of 5 units of Cards and 5.5 units of Mortgages. Cards are returning 295 bps annually and the Mortgages are returning 375 bps. The single optimization period is one year. On the liability side we are paying 50 bps for 10 units of deposits. 0.5 units are funded through short-term debt. The remaining terms on the run off contracts are all longer than the one-year optimization horizon. The default charges are set at 100 bps of the outstanding notional and are realized at the conclusion of the simulation period. The new investments include Mortgages at 370 bps and Cards at 280 bps as well as short-term debt at 66 bps. The optimization model will let you go long or short any quantity of short-term debt at 66 bps. The assumed 100 bps default charge is realized at the end of the simulation period for the new investments printed during the simulation. The new investment liabilities include Deposit A at 51.5 bps and Deposit B at 51 bps. Deposit A has a 98% Implementation Factor and Deposit B is 99%.

The capital plan implementation controls include timing and implementation factors. The Timing represents the fraction of the simulation period the max balance in the targeted new investment was achieved. We assume the latency in achieving the max new investment balance is 20% of the simulation period. The Implementation Factor represents the magnitude of the desired balance achieved in the simulation time period. We assume Cards hit 97% of the target amount and Mortgages hit 95% of the desired target amount. The Timing and Implementation Factor values are realized at the conclusion of the simulation period in the objective function.

The optimization equations implement the following in Simplex LP for a time ending te NIR objective function. te NIR is simply the balance times the rate for the Runoff portfolio plus the sum of the new investment balance times the return rate with adjustments corresponding to the Timing and Implementation Factor.
We have ts Cash units available to fund new investment in the period. This cash can be leveraged by the Max Leverage Rate during the period to obtain the Max Leverage Cash. The te NIM Asset Revenue and Liability Revenue is derived from the NIR by dividing by the terminal te Asset Sum or te Liability Sum, respectively. The Deposit, Cards, and Mortgage Limits cap the amount of new investment that can be printed during the period to achieve the Max Leverage Rate.

Figure 5 first lists the Runoff model sensitivity for 10 bps (geometric) moves relative to the base NIR. We are approximately 2.5 times more sensitive to perturbations in the return rate than the balance. The default and balance perturbation effects are about the same. You really want a controlled approximation error on the return Rate model. New Investment model sensitivities run about 12x smaller than the Runoff sensitivities. That makes sense given the relative levels of assets between Runoff and New Investments.

Finally the Capital Plan implementation sensitivity shows you want to take less leverage to protect your NIM level. Obviously max NIR and max NIM are very

different optimization objective functions. The perturbations move from 10 bps to 10% in the capital plan sensitivity in this toy model. Perhaps it is obvious, the NIM is most sensitive to perturbations in the Mortgage Limit and the Mortgage Timing.

	Scenario	te NIR	Delta to Base	te NIM	Delta to Base
	Base	0.312433522	0	286.814144	0
Runoff Model Sensitivity	Runoff Asset Rate +10 bps	0.312796928	0.000363405	287.146025	0.331881718
	Runoff Liability Rate + 10 bps	0.312380998	-5.25248E-05	286.777582	-0.036562115
	Runoff Asset Balance +10 bps	0.31251467	8.11471E-05	286.945018	0.130874489
	Runoff Liability Balance + 10 bps	0.312428076	-5.44628E-06	286.809319	-0.004824492
	Runoff Asset Default +10 bps	0.312713843	0.00028032	287.00345	0.189305803
	Runoff Liability Default + 10 bps	NA		NA	
New Invest Model Sensitivity	New Asset Rate + 10 bps	0.31244947	1.59473E-05	286.838689	0.024545659
	New Liability Rate +10 bps	0.312433437	-8.58424E-08	286.82397	0.009826714
	New Asset Balance +10 bps	0.31243612	2.5976E-06	286.816737	0.00259313
	New Liability Balance + 10 bps	NA		NA	
	Cards def + 10bps	0.312434151	6.28689E-07	286.817934	0.003790431
	Mortgage def + 10bps	0.31243546	1.938E-06	286.825828	0.011684667
Capital Plan Sensitivity	Leverage + 10%	0.313318024	0.000884501	286.295829	-0.518314897
	Leverage - 10%	0.311549021	-0.000884501	287.337298	0.523154134
	ts cash - 10%	0.312204462	-0.000229061	286.949159	0.135014961
	Mortgage Limit +10%	0.312693282	0.00025976	287.073475	0.259331683
	Cards Implementation - 10%	0.312220261	-0.000213261	286.939836	0.125691945
	Mortgage Implementation -10%	0.311502522	-0.000931	286.950474	0.136330134
	Deposit B Implementation -10%	0.312392788	-4.07345E-05	286.776749	-0.037394275
	Cards Timing - 10%	0.312157392	-0.00027613	286.560656	-0.253487755
	Mortgage Timing -10%	0.311308722	-0.0011248	285.781577	-1.032567013

Figure 5: NIM Optimization sensitivity from a one period sample model

Perhaps the most notable sensitivity presented in Figure 5 shows if the timing of the peak mortgage new investment varies by 10% the NIM goes down by a basis point. (see the lower right corner of the row Mortgage Timing – 10%). For some banks that's a $200mm loss of revenue for a 45-day delay closing mortgage loans.

5. Summary

Net Interest Margin Optimization (NIMo) is a new computational algorithm for Finance, specifically for automating the Treasury Capital Allocation process across retail and commercial banking. The implementation of NIMo for a multi-trillion dollar balance sheet requires a massive in-memory platform for large scale Monte Carlo simulation feeding a significant parallel NLP/LP optimization problem requiring upwards of half a million variables.

6. References

1. Albanese, High Performance Pricing With Fourth Level BLAS Extensions, 2010. http://www.albanese.co.uk.
2. Colwell, Hot Chips 25, The Chip Design Game at the End of Moore's Law, https://www.youtube.com/watch?v=JpgV6rCn5-g.
3. Dixon, http://www.cs.usfca.edu/~mfdixon/preprints/xcl_hestoncalibration.pdf.
4. Giles, https://people.maths.ox.ac.uk/gilesm/.
5. Hanweck and Ryu, https://www.fdic.gov/bank/analytical/working/wp2005/WP2005_2.pdf.
6. Hayre, Lakhbir, SSB Guide to Mortgage-Backed and Asset Backed Securities, 2007.
7. Intel, Code Modernization, https://software.intel.com/en-us/code-modernization-enablement/developer-resources.
8. Klein & Neira, Nelder Mead Simplex Optimization Routine for Large Scale Problem, 2013.
http://www.cs.ucsb.edu/~kyleklein/publications/neldermead.pdf.
9. McKinsey - Buehler, et.al., Between deluge and drought: The future of US bank liquidity and funding, July 2013.
10. Mittlemann, Hans, Benchmarks for Optimization Software, http://plato.asu.edu/bench.html.
11. Reel, et. al., Prepayment Modeling – Melding Theory and Data into One, 2013.
12. US Federal Reserve, http://www.federalreserve.gov/newsevents/press/bcreg/bcreg20141023a1.pdf Fed Supervisory Stress Scenario.
13. van Deventer, Kamakura, Transfer Pricing Systems Design, 2002.
http://www.kamakuraco.com/Portals/0/doclibrary/WP5.pdf.
14. Wittman, et.al., Short Note on Costs of Floating Point Operations on current x86-64 Architectures: Denormals, Overflows, Underflows, and Division by Zero, http://arxiv.org/pdf/1506.03997.pdf.