Finding NIMo

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

ABSTRACT

The Board of Governors of the Federal Reserve System, via the Comprehensive Capital Analysis and Review (CCAR) program, has opened up a $100+ billion annual market in optimizing the Net Interest Margin at significant banks in expected case scenarios. This note describes a novel Control Theory implementation of the Net Interest Margin Optimization (NIMo) algorithm that is most relevant to banks with 1+ trillion dollar balance sheets and subject to CCAR regulatory reporting. Broadly, the idea is to automate the Bank's discretionary capital allocation plan to optimize the new business revenue (e.g., funding loans with deposits) relative to dynamic operational and market constraints. 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.

Keywords

CCAR, FP Code Optimization, Net Interest Margin Optimization. Monte Carlo Simulation, Balance Sheet, Accrual Portfolio, parallel NLP/LP.

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. We review the background of the capital allocation plan and then examine the elements of the trifecta and how they enable the new computation. Finally we supply the economic context to give some sense why this Net Interest Rate Margin problem is important.

Banks can hold, in aggregate, tens of millions of contractual positions or securities representing: deposits, commercial loans, consumer loans, and other tradable securities in their Accrual Portfolio. Some Banks could hold 100mm active credit card accounts or contracts, for example. These positions can be originated, held, and managed by thousands of regional branches distributed around the country or around the world. Due to current Banking system's computational limitations, Banks do not rapidly simulate the Accrual Portfolio's balance fluctuations, rates of return, and default charges at the position/security/contract level. It can take a couple hours to execute on a large multiprocessor box/grid for a worst-case risk scenario. Newer competitive systems can rapidly simulate the Accrual Portfolio at the security/contract level on a single core in a couple of seconds.

In the Bank's view, each of these Accrual Portfolio positions belong to one of two sets, Assets or Liabilities. The Liabilities, including deposits and Fed Funds, provide cash needed to fund the Assets, including credit cards and mortgages. Assets generally throw off 5-6x more cash than the Liabilities consume. The customer paying 3% on his $1000 credit card line to the Bank throws off more cash than the 0.5% rate the Bank pays on a $1000 deposit account. The Bank Treasury manages the cash flows from Assets to Liabilities, maintains the Balance Sheet of Assets and Liabilities making sure all assets are funded, and implements the Bank's discretionary capital allocation plan with the excess Asset cash flow. The excess Asset cash flow is the cash left over after meeting Liability contractual obligations and regulatory reserve level requirements. If it is known accurately when excess cash is available the Bank is free to use it to purchase new assets according to the discretionary capital allocation plan. If the Bank purchases a new asset before the excess cash is available the Treasury must fund some of the position from the money market. Funding from the money market is slightly more expensive than funding with free cash, and thus drives down the Bank's Net Interest Margin. That price differential will get larger as the Fed starts to raise interest rates. If the Bank waits too long to purchase a new asset, the realized Net Interest Margin will somewhat smaller that it could have been in theory.

Additionally, one could model the default cost or the variations in making new investments. You can see how the NIMo problem decomposes to:

1.) A computationally intensive daily Balance Sheet simulation of millions of securities

2.) A less computationally demanding numerical optimization of what and when new investments are to be made, and

3.) Embedding this optimization computation in a control loop allowing feedback to 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 periodically 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 using NIMo. 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. Not the simplest thing you can do, but not rocket science either. Historically, this has only been sporadically accomplished in Wall Street shops. The Street's serial quantitative libraries, remote vendor servers, interpreted languages, functional languages each provide certain significant benefits. Unfortunately, as of 2015 competitive floating-point execution speeds on x86 commodity microprocessors is not one of those benefits. 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 algorithm designers. The Golden Age of Floating Point computing could continue to increase the FLOPS supply through 2020. Another 8x in per core FP 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 contract-by-contract simulation faster on a small commodity microprocessor core footprint. The Balance Sheet MC simulation that will consume the vast majority of the FP cycles and largely determine the hardware cost of the entire NIMo system. Even though the numerical optimization starts off as cubic expected asymptotic complexity, since the number of new investments is relatively small, the computation is very tractable and can probably run on a desktop computer.

The key assumption to make NIMo profitable is that the Accrual Portfolio security's balance, rate of return, and default models can be improved to pull down the expected case approximation error measured out to the simulation horizon. You want solid estimates on when sufficient cash is available to enter into funded new investments. Sytematically important Bank's are typically quite guarded about disclosing their proprietary models so there is not a large literature to directly guide development. There are, however, very closely related fields that can provide guidance. For example, you can use standard Mortgage Backed Security prepayment model techniques applied to pools of deposits and credit card accounts to obtain accurate balance and rate of return forecasts.

The NIMo profitability breakeven assertion is that, in 2015, there are enough "free" FLOPs to simulate a full multi-trillion dollar balance sheet security-by-security for a stochastic market model (e.g., Libor Market Model - LMM) Monte Carlo derivation of the expected Net Interest Margin or Net Interest Revenue. The effective argument is that you can computationally manage the Bank's accrual portfolio like a very large Front Office trading book. The main technical challenge with conventional x86 cores is to maintain good memory hierarchy performance as the security models become more complex and data driven. In NIMo the tension arises because the quantitative models typically increase the input data size to improve forecasting accuracy. If the mandatory input size forces the cache misses too high then the FP execution pipelines start to idle waiting for data. Generally, the interest rate trading book runs much more efficiently than the mortgage backed trading book on x86 code. The pressure on the memory hierarchy is typically one of the main reasons for the performance difference. For big Accrual Portfolios this the main breakeven analysis point for NIM optimization running on x86 cores.

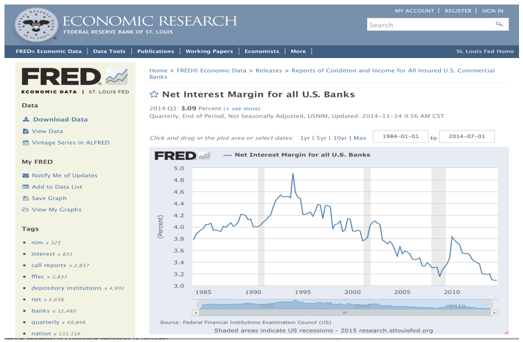

ECONOMIC RESEARCH
FEDERAL RESERVE BANK OF ST. LOUIS
FRED
Net Interest Margin for all U.S. Banks
2014:Q3: 3.09 Percent
Quarterly, End of Period, Not Seasonally Adjusted, USNIM, Updated: 2014-11-14 9:56 AM CST
(Percent)
1985
1990
1995
2000
2005
2010
Source: Federal Financial Institutions Examination Council (US)
Shaded areas indicate US recessions - 2015 research.stlouisfed.org

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

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

NIM is reported in basis points (bps) or in 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?

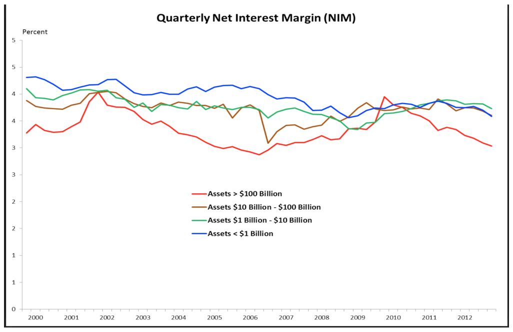


Figure 3: Historical Net Interest Margin

(FDIC, Gruenberg, 2013)

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. Not all of this dispersion is due to the capital allocation plan or its implementation. Occasionally banks incur unexpected costs that can drive down the NIM. Some of that dispersion, however, is due to the Firm's discretionary capital allocation plan and its implementation.

The idea in this note is to provide an alternative to banks currently steering a rocket with a joystick. CCAR will centrally provide the accrual positions and indicatives, "free" FP will provide the expected values in the Balance Sheet simulation, and standard numerical optimization will determine the new investment timing and size. 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. Banks certainly have state-of–the-art capital planning processes in place, but not driven directly from the Accrual Portfolio security level. If it were otherwise CCAR would be completed. We are still gathering hard data but we plausibly expect to automate Bank NIM Growth on the order of 10 bps per annum. That is worth about the same as a billion dollar a year annuity in perpetuity to any of the Banks listed in Figure 2. 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

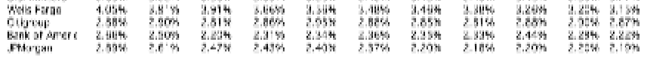
The most relevant literature for Net Interest Optimization concerns the elements of the trifecta: U.S. Fed. CCAR Balance Sheet Simulation, "Free" FP in Finance, and Optimization and Control Theory.

2.1 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? 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. Buehler, et.al. has a good discussion of U.S. Bank Liquidity and funding. Both Buehler, et.al. at 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.

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.



	Q1 2011	Q1 2012	Q2 2012	Q3 2012	Q4 2012	Q1 2013	Q2 2013	Q3 2013	Q4 2013	Q1 2014	Q2 2014
U.S. Bancorp	3.59%	3.60%	3.58%	3.59%	3.55%	3.48%	3.45%	3.43%	3.40%	3.35%	3.27%
Wells Fargo	4.05%	3.81%	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	2.98%	2.50%	2.23%	2.31%	2.34%	2.36%	2.35%	2.33%	2.44%	2.29%	2.22%
JPMorgan	2.59%	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)

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.

2.2 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. Lots of Chebyshev approximations evaluated through Horner's rule optimized for the memory hierarchy.

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 Fused Multiply Add (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 & Patterson, 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.

2.3 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. It seems inevitable that this optimization problem will be forced to NLP eventually.

For the open-loop optimal control problem, GPOPS is a standard commercial code. If you want closed-loop or feedback GPOPS solution you can implement receding horizon control or model predictive control. Closed-loop control is more robust with regard to perturbation and noise.

3. BALANCE SHEET CONTROL THEORY

The main idea in this note is that it is just now feasible with off the shelf microprocessors to use established numerical optimization algorithms off a security–by-security representation of the Firm's Accrual Portfolio to search for and find the expected maximum Net Interest Revenue or the Net Interest Margin over a given simulation horizon. Given that you can do this, the obvious application is to automate the capital allocation plan in a control loop to first optimize the implementation of the current capital allocation plan and second to aid in the amendment of the capital plan consistent with market expectations in the event of a large global market surprise (e.g., Russian interest rate hike, Argentina default, or Swiss Central Bank EUR depeg).

The capital allocation process for the banks holding runoff positions is a plan for buying new investments with cash flows sourced from the Accrual Portfolio. A bank may want to source more funding by obtaining deposits in Texas and use the aggregate funds to issue credit cards in California or mortgages in New York. There are Treasury transactions for short term funding conducted on exchanges with two-way markets but these deposit, cards, and mortgages transactions are not conducted through an exchange – they are new business goals for a retail/consumer bank branch. There is a stochastic element to both the timing and level of these new investments. The Texas branch may get all the deposits but the New York branch may only reach 75% of the target mortgage issuance according to the capital plan. The excess asset or liability shortfalls are covered in the short-term debt market by the Treasury or by Treasury operations. The baseline (start of year) NIM of the runoff portfolio is incrementally adjusted by the new investment cash flows over a year to derive the end of year NIM. The object of the capital plan is to simultaneously: anticipate the variations in the implementation of the capital plan, place high return assets funded by stable low cast liabilities, and anticipate market changes. All banks have a very detailed capital allocation plan that is periodically updated.

Simulation amounts to projecting contract or contract pool balances, rates of return, and default realizations for the 1) positions in the runoff portfolio and 2) the set of prospective new investments. The contract's balance, return rate, and default cost realization over time will be represented by a set of time series functions computed apriori. The composition of the inception face amount and type of contracts in the accrual portfolio follows the statistics available at the Fed, in our simulations. A better filled out set of accrual portfolio statistics can be seen at the Fed web site. The challenge is to determine when, with some precision, the net cash flows first become available to implement the discretionary capital allocation plan so that the implementation of the plan, or even the capital allocation plan itself, can be optimized as a stochastic program [see Shapiro, et. al., Lectures on Stochastic Programming].

A key part driving almost all the Balance Sheet simulation analytics is the balance, return rate, and default cost modeling for the contracts in the Accrual Portfolio. These models are to be run against a stochastic market model, like LMM, to generate expected balances, return rates, and default costs to the simulation horizon. Separately, a stochastic market model, that we select, generates the market data that is fed to the balance model to output the corresponding balance. If the stochastic market model can use a random number generator to uniformly sample the whole space of future market/econometric data then you can use a Monte Carlo algorithm to derive the expected balances, rates, and default costs. Let's look at some example models from the simplest in U.S. Treasuries to the more complex in deposits and mortgages. For a U.S. Treasury bond (UST) the balance model is a constant function returning par or 100 until the bond maturity. The UST rate model, ignoring any up front brokerage fees, a function equal to the semiannually paid coupon times par on scheduled coupon payment dates, otherwise it is zero. The UST default model gives us expected default payments of zero. The error in this set of UST models is small.

For a deposit contract/account the balance, rate and default models are not so straightforward. Three main features of deposit contracts complicate the modeling:

1. many deposits have no contractual maturity date
2. the account balance fluctuates with individual deposits and withdrawals
3. the cost of the account may fluctuate with periodic fee assessments

In the U.S., FDIC Insurance covers some checking, savings, and money market deposit accounts up to a posted maximum. So the default cost is zero for these insured accounts under the specified amounts. In the recent Cyprus and Greek banking crises there appears to be greater uncertainty about the possibility of default on deposit contracts and the default cost may be non-zero for deposit contracts in those institutions. Frequently deposit models pool many deposit contracts and then use single or multivariable regression to fit the aggregate historical balance, return rate, and default figures. The regression fit equations are then used with forecast data to provide forward estimates of the modeled parameters, consistent with the historical data.

If the pooled balance at the conclusion of the last holding period is known, then assuming I have fit a regression based on Fed Funds: Known Deposit Balance * (c1+ c2 * (projected Fed Funds – previous Fed Funds) = new Deposit Balance. I get a recursion defining a time series of projected deposit balances recorded as the balance time series. Return rate and default models can be similarly defined. The error in the regression fit can be large due to dispersion of the actual data to the fit as well as changes to observed behavior, contract definitions, or economic conditions causing a deviation from historically derived expectations. The execution time should be short since you are trading off accuracy for analytic simplicity and fast execution. The performance will not be that different from the UST case although the approximation error will be much higher.

Finally, what are the modeling alternatives if greater accuracy is required than is available through a pooled contract regression fit? These modeling problems arise when the contracts in the accrual portfolio have common features such as: embedded call options, amortization schedules, sink schedules, facility draw down agreements, prepayment options, structured product tranches, default protection, or cash flow waterfalls. Large portions of the Accrual Portfolio hold deposits, commercial loans, credit cards, and mortgages and all these contracts have some of these features and they are difficult to model accurately. Any of these contract features can shape the future balance, return rate, and default costs aggregated into the Accrual Portfolio balance sheet. The timing of when there is expected to be sufficient runoff cash on hand to enter into a new investment can depend on how well these features are modeled. Fortunately, there is a mature literature on mortgage prepayment models used for mortgage backed security trading [see Hayre or Reel]. Almost all these features are encountered in the prepayment literature, with possible exceptions being the more credit oriented features like default protection. The modeling accuracy and the computational performance of the models can vary widely. However, note that sell side mortgage desks have been trading in a competitive OTC market and in volume against these prepayment models for 30 years. The chances are good that there are solid accurate factor structure models for the expected behavior of all of these accrual contracts. The models may be proprietary, require frequent recalibrations, and be somewhat data intensive but it is implausible to assume they do not already exist.

At some level (not necessarily at the individual contract level), every bank has analytic models for these quantities, otherwise they would have no current capacity to forecast the NIR or NIM that their current discretionary capital allocation plan is expected to achieve. In the absence of reasonable models, one would be hard pressed to offer a compelling argument asserting one capital allocation plan is better or worse than another. Moreover, in the absence of good models the bank's capacity to track how well the actual Treasury selected capital allocation plan was being implemented, would be limited. For a large bank with thousands of geographically dispersed centers and branches the lack of solid balance, return rate, and default models would amount to flying blind. Fortunately, with CCAR, the modeling fidelity is rapidly improving to an even more granular scale to provide greater forecasting ability and automated monitoring visibility.

Let's assume we run daily simulation periods. First, simulate the runoff portfolio over the simulation horizon. The simulation is particularly interesting one year out because this is the NIM you report to the shareholder. There is a case to be made to simply optimize the 1Y NIM. The problem is that the credit default charges for new investments may only be realized after Year 1. It could be the case that this year's optimal NIM causes a depressed NIM in subsequent years. Thus it makes sense to extend the simulation horizon to allow the deferred default cost to enter the optimization evaluation. Let's push the simulation horizon out to five years and use some weighted sum of the annual NIM (or NIR) as the objective function.

Simulate the universe of potential new investments over the simulation horizon (previously determined to be 5Y). For simplicity assume we have a total of approximately 10K potential new investments with parameterized trade inception and trade size. Let the LP decide when and how much of each new investment to print. There are some problems with these assumptions. One is the new investment contracts may not be fungible with respect to modeling the trade inception date. Another is the new investment contract characteristics that we are selecting in year 2 of the simulation need to be known at simulation time. For example you have MBS to purchase in year 2 with mortgage pool collateral assembled in simulation year 1. At the simulation inception you have no particular information of what pools might be assembled in the course of the simulation. For now let's argue that these effects will be small. Run all these 10K balance, return, and default models through the MC simulation to get the corresponding time series of expected values. Save these time series for input to the LP.

Solve a multi-period LP optimization selecting the future new investments using the NIR as an inexact proxy for NIM in the objective function. All the Asset and Liability aggregated expected balances, returns, and defaults form a basis for the search. There is a single set of balance return, and default time series for all the runoff Assets and Liabilities. Since there are no sales in the Accrual Portfolio (not completely true take AFS portfolios, for example) the expected values from the runoff portfolio are constant with respect to the LP optimization. This is a big deal because it means that the runtime complexity is not dependent on the size of the runoff portfolio. The runtime simulation cost in determining the expected balances, returns, and defaults is dependent on the 10mm runoff positions the LP is independent. That is what has the potential to make this NIMo computation tractable even at the whole market level (the all USD asset and liabilities level). Informally, you simply have to run balance, return, and default models for 100mm positions through a 10K path Monte Carlo.

The LP determines the identity, timing, and size of new investment with the simulated new investment data we computed previously. The runtime complexity of the LP depends on the 10K positions and the daily simulation periods out five years (we will probably go to a variable grid). Mittleman's parallel benchmarks seems to show I can get commercial LP codes that will run 1mm variables in 1000 seconds on a Linux desktop w multithreading. I am guessing that I need at most 10mm variables to run all USD assets and liabilities through parallel Simplex LP. Seems to be well feasible even if I have to throw massive hardware at it. Plausibly the single bank LP cases are well tractable with off the shelf LP codes.

To summarize, 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 at the Firm level 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 per Firm;

daily simulation periods; 10K paths in the Monte Carlo; we estimate somewhat optimistically ~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.

Figure 5 shows the Balance Sheet control flow for NIMo. The idea is to use NIMo for a couple obvious Control Theory applications in the automation of the Treasury capital plan implementation.

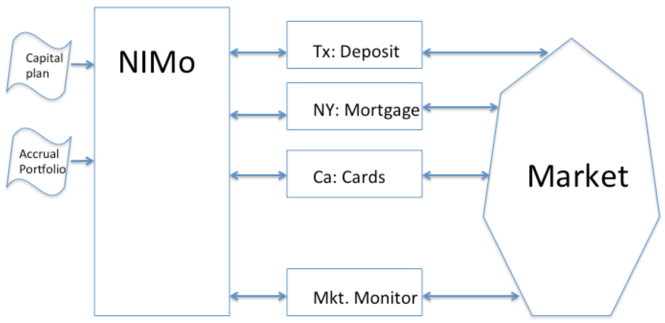


Figure 5: Balance Sheet Control Theory

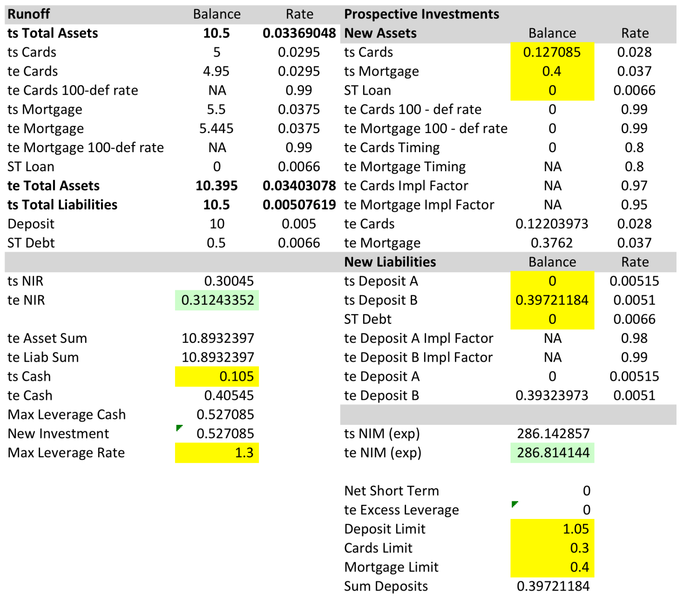
There are a couple obvious applications of a Control Theory framework for Capital Allocation. First, use a given capital allocation plan specification and let NIMo determine the explanatories for the realization of the given capital allocation plan. For example, a bank could begin to accumulate historical 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 (see Figure 6). Figure 7 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. For simplicity we will focus on the one period model.

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.



Runoff	Balance	Rate	**Prospective Investments**		
ts Total Assets	**10.5**	**0.03369048**	**New Assets**	Balance	Rate
ts Cards	5	0.0295	ts Cards	0.127085	0.028
te Cards	4.95	0.0295	ts Mortgage	0.4	0.037
te Cards 100-def rate	NA	0.99	ST Loan	0	0.0066
ts Mortgage	5.5	0.0375	te Cards 100 - def rate	0	0.99
te Mortgage	5.445	0.0375	te Mortgage 100 - def rate	0	0.99
te Mortgage 100-def rate	NA	0.99	te Cards Timing	0	0.8
ST Loan	0	0.0066	te Mortgage Timing	NA	0.8
te Total Assets	**10.395**	**0.03403078**	te Cards Impl Factor	NA	0.97
ts Total Liabilities	**10.5**	**0.00507619**	te Mortgage Impl Factor	NA	0.95
Deposit	10	0.005	te Cards	0.12203973	0.028
ST Debt	0.5	0.0066	te Mortgage	0.3762	0.037
			New Liabilities	Balance	Rate
ts NIR	0.30045		ts Deposit A	0	0.00515
te NIR	0.31243352		ts Deposit B	0.39721184	0.0051
			ST Debt	0	0.0066
te Asset Sum	10.8932397		te Deposit A Impl Factor	NA	0.98
te Liab Sum	10.8932397		te Deposit B Impl Factor	NA	0.99
ts Cash	0.105		te Deposit A	0	0.00515
te Cash	0.40545		te Deposit B	0.39323973	0.0051
Max Leverage Cash	0.527085				
New Investment	0.527085		ts NIM (exp)	286.142857	
Max Leverage Rate	1.3		te NIM (exp)	286.814144	
			Net Short Term	0	
			te Excess Leverage	0	
			Deposit Limit	1.05	
			Cards Limit	0.3	
			Mortgage Limit	0.4	
			Sum Deposits	0.39721184	

Figure 6: LP Solver for toy NIMo model

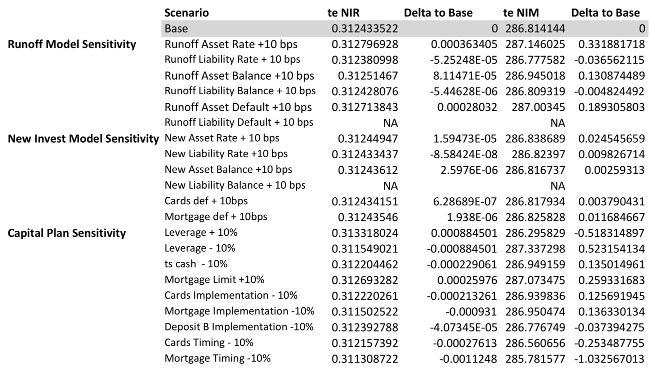
The new investments include Mortgages at 370 bps and Cards at 280 bps as well as short-term debt at 66 bps. The optimization model will let you go long or short any quantity of short-term debt at 66 bps. The assumed 100 bps default charge is realized at the end of the simulation period for the new investments 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 7 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.



	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 7: NIM Optimization sensitivity - one period

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 quite sensitive to perturbations in the Mortgage Limit and the Mortgage Timing.

Perhaps the most notable sensitivity presented in Figure 7 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] Basel Committee, http://www.bis.org Basel Committee on Banking Supervision.

[3] Blaze, https://code.google.com/p/blaze-lib/.

[4] Brown, Ian, Regression Model Development for Credit Card Exposure at Default (EAD) Using SAS/STAT.

[5] Buehler, et.al., McKinsey, Between deluge and drought: The future of US bank liquidity and funding, July 2013.

[6] Canals-Cerda & Kerr, Federal Reserve Bank of Philadelphia, Forecasting Credit Card Portfolio Losses in the Great Recession: A Study in Model Risk, 2014.

[7] Colwell, Hot Chips 25, The Chip Design Game at the End of Moore's Law, https://www.youtube.com/watch?v=JpgV6rCn5-g.

[8] CPLEX, http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/.

[9] Dixon, http://www.cs.usfca.edu/~mfdixon/preprints/xcl_hestoncalibration.pdf.

[10] Giles, https://people.maths.ox.ac.uk/gilesm/.

[11] Goldberg, What Every Computer Scientist Should Know About Floating-Point Arithmetic, http://perso.ens-lyon.fr/jean-michel.muller/goldberg.pdf.

[12] GPOPS, http://www.gpops2.com.

[13] Gurobi, http://www.gurobi.com.

[14] Hager, http://blogs.fau.de/hager/.

[15] Hanweck and Ryu, https://www.fdic.gov/bank/analytical/working/wp2005/WP2005_2.pdf.

[16] Hayre, Lakhbir, SSB Guide to Mortgage-Backed and Asset Backed Securities, 2007.

[17] Hennessey and Patterson, Computer Architecture, 5th Ed., Morgan Kaufmann, 2011.

[18] Higham, Accuracy and stability of Numerical Algorithms, SIAM, 2002.

[19] Hirtle, Kovner, et.al., Federal Reserve Bank of New York, The capital and Loss Assessment Under Stress Scenarios Model, 2014.

[20] Kahan, http://www.cs.berkeley.edu/~wkahan/.

[21] Intel, Code Modernization, https://software.intel.com/en-us/code-modernization-enablement/developer-resources.

[22] Klein & Neira, Nelder Mead Simplex Optimization Routine for Large Scale Problem, 2013. http://www.cs.ucsb.edu/~kyleklein/publications/neldermead.pdf.

[23] Mittlemann, Hans, Benchmarks for Optimization Software, http://plato.asu.edu/bench.html.

[24] Mosek, https://mosek.com.

[25] Muller, et.al. Handbook of Floating –Point Arithmetic, Birkhauser , 2010.

[26] Patterson and Rao, GPOPS-II, 2014, http://dl.acm.org/citation.cfm?id=2558904.

[27] Reel, et. al., Prepayment Modeling – Melding Theory and Data into One, 2013.

[28] Shapiro, http://www2.isye.gatech.edu/people/faculty/Alex_Shapiro/SPbook.pdf, Lectures on Stochastic Programming, 2009.

[29] US Federal Reserve, http://www.federalreserve.gov/newsevents/press/bcreg/bcreg20141023a1.pdf Fed Supervisory Stress Scenario.

[30] US Federal Reserve, http://www.federalreserve.gov/aboutthefed/boardmeetings/gsib-methodology-paper-20150720.pdf.

[31] van Deventer, Kamakura, Transfer Pricing Systems Design, 2002. http://www.kamakuraco.com/Portals/0/doclibrary/WP5.pdf.

[32] 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.