1998 VALUATION ACTUARY
SYMPOSIUM PROCEEDINGS

SESSION 42PD

ECONOMIC SCENARIO GENERATORS (ESG) AND ACTUARIAL PRACTICE

Steven L. Craighead, Moderator

Michael F. Davlin

Vladimir S. Ladyzhets

Mark Tenney

Hongfei Zhang

ECONOMIC SCENARIO GENERATORS (ESG) AND ACTUARIAL PRACTICE

MR. STEVEN L. CRAIGHEAD: Mark Tenney will be speaking first, and he will be speaking at a general level. He has been working with this material off and on for several years. The next speaker will be Vladimir Ladyzhets. He'll be talking about the SS&C model that the company has developed. The topics will move from general to more and more specific. Third, Hongfei Zhang will discuss a low discrepancy model, which is not necessarily an economic issue; however, it does discuss certain problems that one must address with economic scenario generators. Finally, Mike Davlin will speak. He has been developing software for several years in many different capacities. He has been involved since the early 1980s doing software development at all different levels, and he will be talking about software standards for economic scenario generation. This is relevant because software ultimately is where the rubber meets the road.

Economic scenario generation is both an art and a science. Mark Tenney will now discuss the actual complexity of the issues backing economic scenarios.

MR. MARK TENNEY: I'll discuss the state of the art in applying economic scenario generators in the life insurance business in the U.S. I'll relate a little bit to the experience I've had over the last few years, much of it has been through working with Steve, Mike Davlin, Hongfei Zhang, and some of the others in the audience. I wanted to talk about some of the general conceptual issues, some of the problems, and some of the issues that are a little more touchy-feely, but critical to applying these generators. I wanted to talk about the practice of applying economic scenario generators in the U.S. life business. Before getting to that, I'd like to just make a couple of comments. In a recent *Wall Street Journal* issue, there was an article about Long-Term Capital, a company that is partly a group of academics and partly Wall Street traders who are some of the leading practitioners of quantitative trading. They worked on Solomon Brothers and then went off and formed their own firm. That firm, basically, had to be recapitalized by the Wall Street firms to the tune of $3.5 billion. That was done in a meeting of the senior leadership of Wall Street, the head of Goldman Sachs and so forth.

Basically, they had to ante up about $300 million for the larger investment banks and $50 million to $100 million for some of the smaller ones. They were counterparties to Long-Term Capital on derivative deals. Many of them were very complex derivative deals for both U.S. economy-based variables and also overseas.

The question comes up, why did the investment banks bail out this hedge fund? Why did this hedge fund get into these problems? The hedge fund had been making bets on the movements of different derivative prices and interest rates. In particular, it made convergence bets between yields in different economies or different points on the yield curve. They had been using these high-tech models and trying to understand the relative pricing of securities and risk and the risk/return trade off. They made these bets and then they lost an enormous amount of capital. The leaders of Wall Street, basically, were in a situation where they didn't know what the consequences would be of letting Long-Term Capital go bankrupt. It might have been that it could have been liquidated in an orderly manner, but they were afraid of doing that. It points up the complexity of applying these sorts of models to security markets. Long-Term Capital, obviously, made some mistakes, with regard to both its capital level and its risk and as far as the probabilities of things that could happen to it. They thought markets would move in a certain direction and they placed their bets. When markets kept moving in the opposite way, they didn't have the capital to survive.

It's sort of an object lesson to us as we apply economic scenario generators and look at the decisions we're making, where typically, we have mismatches between assets and liabilities. There is a lot of leverage in the life insurance business. Our capital is 5% to 10% of our assets and our liabilities. We are subject to making large leverage bets just like Long-Term Capital did. We are making decisions based on our understanding of the world and what could happen. If that understanding is wrong, then we potentially could be wrong on a highly leveraged basis.

I'll try to refer back to this sort of issue in my comments. We should take as a warning the fact that the leading firm of practitioners of derivative pricing with two Nobel prize winners as part of their board went bankrupt by misapplication of these models. I think it's a sobering thought and simply reminds us that we need to pay a lot of attention to what we're doing when we make these decisions.

In terms of the life insurance business in the U.S., there's a lot of experience with using stochastic asset/liability modeling (ALM) systems and different generators. Many different companies have been using those for a number of years. We have the PTS and TAS systems and other systems that people have used, both home-grown systems and systems from other vendors. People have used stochastic generators, which may be part of the systems or externally supplied. They have generated scenarios of interest rates and other economic variables from the systems and gotten results. The question is, what experience have we developed with this? What's the state of the art? What's our understanding? How good are they? What do we do with them? How do we evaluate them?

Wall Street has taken an approach of not really looking at things the same way the life insurance industry has. The life business tends to look at stochastic ALM systems, which are cash-flow systems, which model on a multi-year basis. Wall Street tends to look at things like value-at-risk or deltas and gammas, which are basically sensitivity to price change and have concentrated on fitting current market prices with complex models by having lots of parameters in those models. That approach reached its day of reckoning with the near bankruptcy of Long-Term Capital. What we have to do, in our business, is understand what are the probabilities of scenarios out of these economic scenario generators, and what are the true risks?

Because we don't have the true model, we can't separate the business problem from the generator. It would be nice to say let's get the correct generator, or let's work a few months calibrating it. Let's understand it and then apply it to all of our different business problems. The problem is generators have flaws and those flaws can interact with the business problem. It's not such a simple thing to apply a generator to a business problem.

An example of a business problem is capital allocation. You might be looking at capital allocation for a line of business based on solvency, according to some measure like keeping your credit rating over the next five years. You might be interested in 95% probability to keep your credit rating above a certain level. You'd look at your capital needs based on that measure. The product that you're dealing with, if it's crediting based on a five-year rate, is sort of your benchmark that your credit is based on. You then have some investment strategy that differs from investing in five-year assets.

You also have a reasonably complex problem with a complex output—the 95th percentile. The question is, what in the generator drives the results of that business problem? Figuring out the properties of the generator, which correspond to that 95th percentile is complicated. Ideally, you'd like to be able to say, it's the average level rates or it's the probability that rates are lowered or it's the probability that the yield curve is inverted. But there are a number of scenarios that can produce a bad outcome and produce outcomes between the 95th percentile and 100th percentile. Trying to understand what aspects of the generator caused you to get those cases between 95% and 100% is difficult. In fact, for a complex business problem, you can never really say, well, it's the lower tail or this or that, because it's a combination of factors.

We can't separate the business problem from the economic scenario generator. As we look at these different business problems, we have to understand the properties of the generator and how those link to not only the business problem, but also the parts of the business problem we're interested in looking at. That could potentially mean understanding the full path of rates and the full set of paths of rates from the start of the horizon to the terminal point that we're looking at. That essentially gives us no meaning or intuition to grasp. To say it depends on the whole path, really doesn't tell us what drove bad results on a particular problem.

To understand that, we have to develop heuristic concepts or concepts that are not exact explanations of what happens but that relate, at least on a crude basis, what aspects of the generator are driving which aspects of the business results. If we're crediting based on a five-year rate, the accumulated account values is, obviously, one thing you would be looking at. This reduces the state of the world from the path to something smaller. That still doesn't solve our problem, but it's the sort of measure that we can start looking at and thinking about that helps us understand the problem.

We have to develop some way of understanding what in the generator is driving what in the business problem. Figuring that out is hard. Some people have been working on that pretty hard. Some of those people are on our panel. I think Steve Craighead, in particular, has probably done more on this than anyone. He has been studying generators at Nationwide and their different implications and looking at a whole host of business problems along their whole business very intensively over the last few years. It has been quite a privilege for me to work with him on this work. He has written an enormous body of papers on the outcome of generators and things like fitting distributions to get the tails of distributions. I'd just like to say that I think some of the things he's done, which I'm going to talk about in the remainder of the talk, are really critical to trying to understand the conceptual length between the generator and the output.

One of those things is what Steve calls "razors" and it's named after the term *Occam's Razor*, in which you reduce a problem to what actually drives the results and that's all you keep. The idea is we can develop these razors based on what properties of generators drive results and we can understand what's going on. Approaches to that have been taken by various people. I'll get to those a little later. They have tried to look at the properties of generators like frequency of yield curve inversions and the histogram of interest rates of three months or ten years and so forth.

If we had the perfect model, we wouldn't have to do this. You wouldn't have to think about what the relationship was of the generator to the business problem quite as deeply. You still have to think about it, but you wouldn't have to worry about what's wrong with the generator, which makes it much harder to think about. When you're running a generator and you get results and then you get a 95th percentile on a certain capital, on a certain expected return, then you have to ask what in the generator drove this result and what could be wrong to produce a different result. If we calibrated the generator to produce interest rates that are relatively low, what if the reality is that interest rates could, in fact, be high? We might have one calibration that we thought is more accurate, but another one that produces higher rates. That other one might correspond more truly to the risk in the tails and be more appropriate for setting capital based on extreme percentiles of the distribution. We might be in a world where we have to think about one calibration for risk in the tails and another calibration for getting expected return on that same capital level.

In order to solve this problem, we could think that it really doesn't matter which generator you have. They're all wrong. We start with that. We know the model is wrong, but the problem is, the worse the model is, the harder it is to think about it, and the more you have to think about what is wrong with the model.

The other thing with incorrect models is that typically, the structure of the model doesn't correspond to reality or any way you think about validity. So when you try to think about what is wrong with the model, you don't have a very good starting point. Some of these Wall Street models like Black, Derman, and Toy were basically just pushing parameters at fitting yield curves and derivative prices, and are the worse sort of case of that because all you have is a bunch of parameters there that you're pretending are constants. In fact, they are stochastic. You re-fit them every day, and then you ignore that fact when you generate scenarios out of the model because you only treat one of the factors as stochastic, the short-term rate. With a model like that, you have to think about how all those weird little parameters shift as the economy could move around? Since you can't figure out how that happens, you can't think with that model, and you can't really do a very good job of doing any sort of sensitivity analysis or understanding what in that model is driving your results.

You really need to start with a model that's about as good as you can get and that has a structural logic to it that allows you to think about limitations of the model and what interacts with the business problem.

In an ideal world, we could take this generator—even a bad generator—and think about the business problem and the properties of the generator and figure out what's driving what. The problem is we'd never have time to do that. Steve Craighead has a lot of time to do that, but even he doesn't have time to do it all the time. So what are we going to do to get around that problem? What people do is, when they get a new generator, they spend a couple months working with it and seeing what the results are. They run it through some business problems. They might model their company and see what the results are, or try to tweak it and get some understanding of it and then move on. After that process, they are basically going to apply it to different business problems and maybe do a little bit of tweaking again. Realistically, there is that one upfront stage, and then it's going to be a long time before people have time to really work with it again. As a practical measure, you just run with the best generator you have. If it's not the best, and you really don't have time to figure out what in the generator could be wrong and what could be driving the results, you go with it anyway.

We need to have the best generator, and it has to have the best calibration we can have. Most of the time, that's really what we're stuck with. We have some understanding of some of the problems with its properties. If it's based on some sort of public research, then we might get the advantage of some research that other people have done at other companies through forums or publications like this. But, we're basically going to go with what we have. When you do that, you really need to have the best available generator in order to do that.

How do we get the best generator? Panels like this help as does some of the work that some of these and other people have done. It's a slow process. It can't be done by one company. It takes an industry and it takes an academic profession. It also takes time, and it takes a certain amount of demand from customers as to the properties of these things and what drives the results.

As I said before, there are these heuristic concepts, which Steve calls "razors." I introduced the term *stylized facts*. Other people have commented on that. Mike Davlin has been involved in this sort of work for a number of years both at Chalke and subsequently. It's a set of properties like frequency of yield curve inversion or how interest rates are more volatile when rates are higher. They help us understand the properties of generators and what relates to the outcome of systems.

I'll just flip over these contributions. What goes into the best generators now? First, they are arbitrage-free. You might not think that's too important, but if a generator is arbitrage-free that means it is logically consistent. If it's not logically consistent, and you only have a certain amount of time to spend thinking about it, and if you have to use it, then you don't want some logical inconsistencies down there that no one can possibly understand the implications of. Therefore, you need something that's arbitrage-free.

The model needs a logical structure that's meaningful with respect to the history of interest rates. You need to be able to look at a graph of interest rates and see that in the 1960s interest rates were low, and in the 1980s, they were high. Then you can see where in the mode those results can be produced. An illogical or inconsistent model is not going to help you think about the problems.

Finally, we need to expand these models to other variables, such as inflation, stock index, and things like that. If the model has a structure that corresponds to the way the economy looks, you can think about it, and it's going to be a good model for developing models of inflation in relation to the stock index and so forth.

In conclusion, we need the best generators available. Ideally, you'll have more than one that are of good quality, because running one generator against another gives you a lot of information, and it helps you think about what is the difference in the generators that drove the results. It helps you to get a good understanding of the published or unpublished research that exists. It's very important that whenever you get a generator, you work with it. If you can have more than one at the same time and run different generators, then you can see that they get different results and you can ask why. That's very powerful. You can tinker around with things and try to make them the same and see how close you can get.

Finally, it's very important for customers to be demanding of vendors. If you're not, you're not going to get good quality, and you're not going to understand how to use the software models.

MR. VLADIMIR S. LADYZHETS: In his presentation, Mr. Tenney gave us a general description of the problems you can face when developing, calibrating, or running an economic scenario generator. I'm going to talk about a particular generator that has been developed at SS&C Technology Company. That might provide you with some specifics for what you just heard from Mr. Tenney. Probably the most interesting thing about the model, which underlies the SS&C Technologies generator, is that it ties together the debt market with the equity market.

I would like to start by briefly mentioning different types of interest rate models that are currently used. The objective of this is just to be sure that terminology is established. I am using the paper "The Four Faces of Interest Rate Model," written by P. Fitton and J. McNatt. In accordance with the paper, we distinguish an arbitrage-free model from an equilibrium model. Here you can see some basic or major differences between arbitrage-free and equilibrium models.

- Arbitrage-Free Model
  - Takes certain market prices as given and adjusts model parameters in order to fit the prices exactly,
  - Requires as least one parameter for every market price used as an input to the model,
  - Does not in reality attempt to emulate the dynamics of the term structure.

- Equilibrium Model
  - Employs a statistical approach, assuming the market prices are observed with some statistical error, so that the term structure must be *estimated*, rather than taken as given,
  - Uses a small set of state variables,
  - Is more tractable and easier to generalize and to make consistent with the observed features of interest rate behavior.

Each type of model can be used with two different parameterizations: risk-neutral parameterization and realistic parameterization. Here you can see some basic or major differences between risk-neutral parameterization and realistic parameterization.

- Risk-Neutral Parameterizations
  - *Premia*, whatever their values, that exist in the marketplace are embedded in the interest rate process itself, so that the expected discounted value of a cash-flow at the risk-adjusted short rate is equal to the discounted value of the cash-flow at the spot rate.
  - An interest-sensitive instrument's price can be estimated by averaging the present value of its cash-flows, discounted at the short-term interest rates along each path of the short rate under which those cash-flows occur.

- Realistic Parameterizations
  - *Premia* are distinguished from the interest rate process.
  - They simulate realistic environment where the more risk-inclined investors are, the higher the term premium will be.

Table 1 shows how different types of models and their parameterizations can be used in actuarial practice. The model I'm presenting is an equilibrium model with realistic parameterization, which is shown in the right bottom quadrant of the table. In accordance with what you see in this quadrant, our model is mostly used for stress testing and reserve/asset adequacy testing.

TABLE 1
Model Classification

Model Classification	Risk-Neutral Parameterization	Realistic Parameterization
Arbitrage-free	Current pricing (Unreliable data)	Unusable
Equilibrium	Current pricing (Unreliable data) Horizon pricing	Stress testing Reserve/asset adequacy testing

The model I am presenting is called the five-factor debt equity model. The model and the stochastic generator implemented in this model were developed in SS&C Technology by Vladimir Fishman, Yuri Galperin and Anatoly Reynberg. At the present time, I'm responsible for the maintenance development of the generator, which includes making some modifications and changes to the model. At the end of this presentation, I will briefly mention what I recently did with the five-factor debt equity model.

The five-factor debt equity model (shown below) ties together the debt market and the equity market. As the model name implies, it deals with five major economic factors: a short-term interest rate, a long-term interest rate, the S&P 500 index, volatility of the S&P 500, and the inflation rate. The major objective for the development of this model was to provide the company with a complete picture of the economic environment. Each of five factors has its own stochastic differential equation.

Five-Factor System of Equations

$$dp = [b_u u - b_r r + d_p + b_p v]\, p\,dt + p\sqrt{v}\, dz_p \qquad (1)$$

$$dv = k_v (d_v - v)\, dt + \sigma_v \sqrt{v}\, dz_v \qquad (2)$$

$$dr = k_r (u - r)\, dt + \sigma_r \sqrt{r}\, dz_r \qquad (3)$$

$$du = k_u (d_u + a_v v - u)\, dt + \sigma_u \sqrt{u}\, dz_u \qquad (4)$$

$$di = k_i [(1 - a_i)\, r + a_i u - d_i - i]\, dt + \sigma_i\, dz_i \qquad (5)$$

(1) p – S&P 500 index
(2) v – volatility of S&P 500
(3) r – short-term interest rate
(4) u – long-term interest rate
(5) i – inflation rate

The first equation is for the S&P 500 index; the second is on the volatility of this index; the third is on the short-term interest rate; the fourth is on the long-term interest rate; and the fifth is on the inflation rate. These equations describe the stochastic processes known as the geometric Brownian motion. These type of equations were introduced to finance by Black, Scholes, and Merton. In physics, these equations have been known and used mostly for the kinetic theory of gases for at least 100 years.

Each equation of the five-factor model is complemented by the initial or starting value. So for each economic factor, we have the equation, which governs its development in the course of time and the initial position from which we start projecting its future values.

The output of the generator is monthly projections for all five factors and, nine yields given by the Treasury yield curve.

Mr. Tenney mentioned "razors" or stylized facts providing a theoretical basis for the development of economic generators. He specifically referred to the stylized facts, which were found by David Becker from Lincoln National Life. Those stylized facts are the foundation for our five-factor model. Among them, the following seven stylized facts were especially important for the design of the model and development of the generator:

- Short-term and long-term interest rates move together most of the time.
- The S&P 500 rate (excluding dividends) has an historical average of 6% above the risk-free short-term interest rate.
- The spread between long and short interest rates has a week but positive correlation with the S&P 500 rate.
- Over short time intervals, there is an inverse relationship in the behaviors of short-term interest rate and the S&P 500.
- The correlation between the S&P 500 rate and interest rates is not significant.
- The short-term interest rates and inflation are strongly positively correlated.
- The long-term and short-term interest rate spread is negatively correlated with inflation.

The SS&C five-factor stochastic model is consistent with the stylized facts listed above as well as with the other empirical dependencies between major economic factors observed by David Becker and other researchers and practitioners. The stochastic generator implementing the five-factor model exhibits the following main features: It efficiently generates realistic scenarios of five major economic factors as realization of the common interrelated stochastic processes. Efficiently means that scenario generation can be done very fast. For example, if I run 1,000 scenarios, each with a ten-year horizon, it takes between three and five minutes on the PC.

The parameters of the model and the initial values of the state variables (state variable is another name for the model factor) can be fitted to a given market environment. For example, if the user believes that the next, say ten years, will be similar to some particular ten years in the past, he or she can find a set of generator parameters, which allows the user to "replicate" the past. This generates a number of stochastic scenarios featuring similar historical properties but starting from the current market conditions.

- The generated scenarios are consistent with the results of statistical analysis of observed data. Based on the previous example, this means that statistics, such as annual volatilities, spreads, and the percentage of inverted Treasury yield curves, calculated for projected values, will be close to those calculated for historical data.
- The bond equivalent yields and spot yields for maturities presented on the Treasury yield curve are calculated at each projection time. I would like to note that the bond equivalent yields and spot yields can be calculated for an arbitrary set of maturities, and the choice of the Treasury yield maturities is due to the available market data and the user's convenience.

What are you supposed to do in order to generate scenarios? Since the market situation is changing, you cannot keep the same set of initial values and parameters in your generator all the time. First, you need to calibrate the generator in accordance with the historical data and/or your analysis of the market tendencies. The calibration of the generator means that you are finding values for all coefficients of the five equations incorporated into the model (there are a total of 17 coefficients) plus correlation coefficients specifying interrelations between stochastic behaviors of the model five factors (there are seven correlation coefficients).

The next step is fitting input or initial values of the state variables into the current market. After that, you generate scenarios. The final step is the valuation of the generated scenarios. Scenario valuation consists of the calculation of various statistics on the generated scenarios, volatilities, spreads, returns, etc., and comparison of these statistics with those calculated on the historical data. The valuation is a crucial step. If it shows that the scenario statistics do not match the historical ones, you need to go to the very beginning and start the process over: calibration, initial data fitting, scenario generation, and scenario valuation.

Obviously, the calibration is the most complex and time-consuming part of the scenario generation. As I mentioned early, to calibrate the five-factor model the numerical values for 24 coefficients have to be found. However, our experience shows that once you get the model calibrated, you can keep the same set of parameters, with very minor modifications, for a period of five years or more.

At the present time, the five-factor generator calibration is based on historical data on the interest rates, the equity index, and the inflation rate of the period 1956 through 1995 with emphasis on the decade of 1986 through 1995. This calibration explicitly or implicitly assumes that what's going to happen in the next ten years will be similar to what happened during the last ten years. In other words, it can be said that the calibration procedure is equivalent to the transformation of the qualitative expert judgement into a quantitative set of parameters.

In order to obtain the current set of model parameters, a combination of different heuristic methods was used. We started with a set of "realistic" parameter values, the best guess, and then performed a number of numerical experiments with various modifications of the initial parameter values until the final set of parameters was established. Charts 1–14 show an almost ideal match between various statistics calculated for the historical data and those calculated for generated scenarios.

We used all available historical data for the period 1956–95 in connection with model validation with the emphasis on the decade of 1986–95. After model parameters were calibrated to the economic condition of the last decade, the fitting procedure was used to find the best initial values for the current situation. Any yield curve of this period can be fitted with high accuracy. In our experiments the discrepancy between initial and fitted curves before application of the yield residuals did not exceed 15 basis points. Chart 1 illustrates this fact displaying the results of the fitting to the yield curve as of January 1, 1986.

CHART 1
Actual vs. Fitted BEY Curves as of 1/1/1986

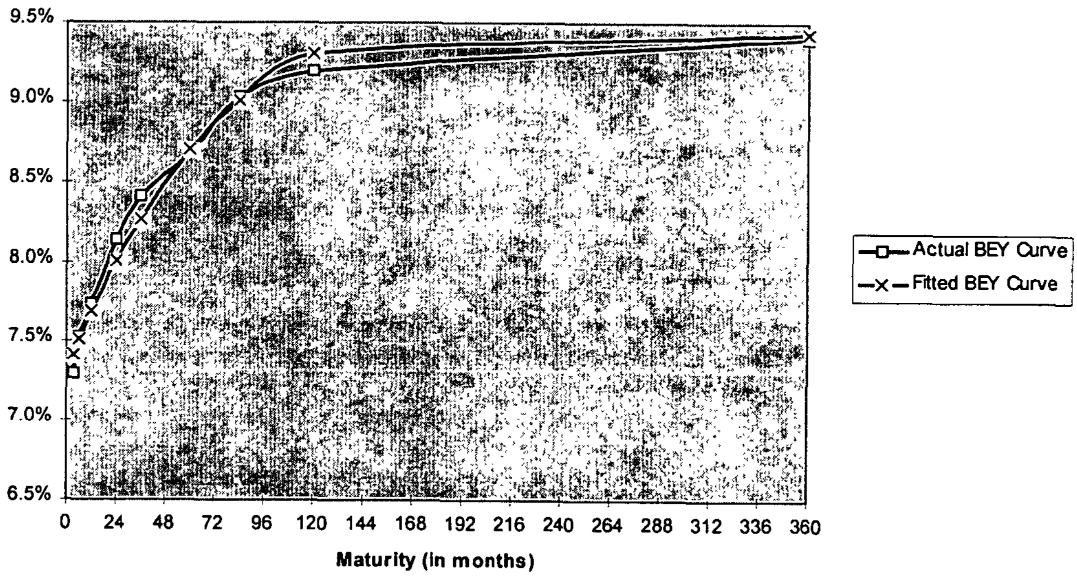


CHART 2
Five-Year Forward Yield Curves (as of 1/1/91)

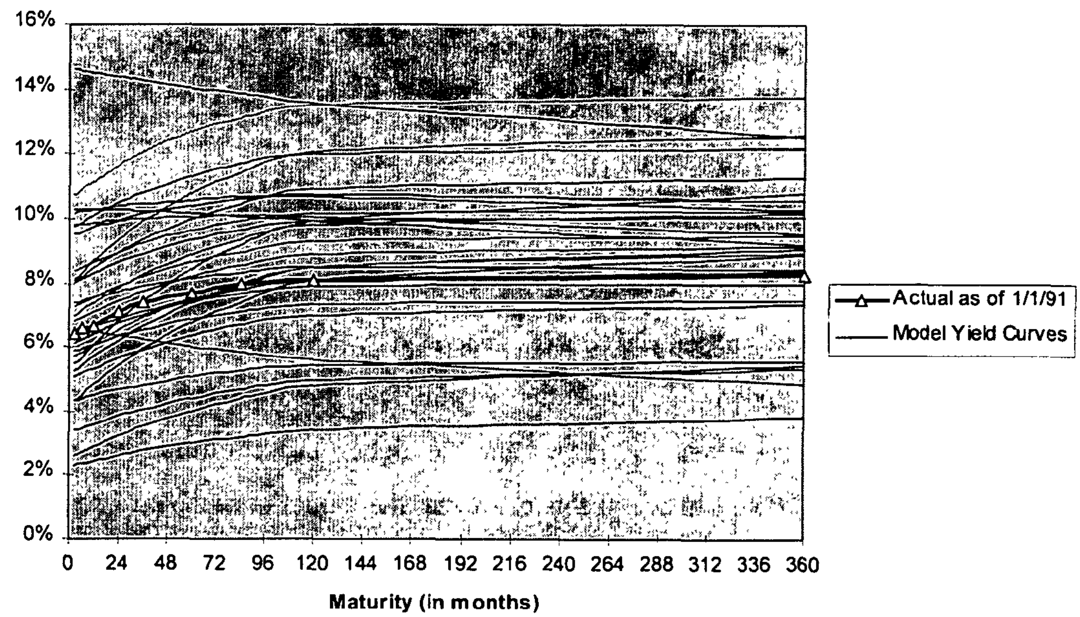


CHART 3
Actual Correlations Between Yields For Different Maturities

	3m	*6m*	*1y*	*2y*	*3y*	*5y*	*7y*	*10y*	*30y*
3m	1 00	1 00	0 99	0.97	0.96	0 93	0 92	0 90	0.87
6m	1.00	1 00	1 00	0.99	0 98	0 95	0 94	0 93	0 90
1y	0 99	1.00	1 00	0.99	0 99	0 97	0 95	0 94	0.91
2y	0 97	0 99	0 99	1 00	1.00	0 99	0.98	0 97	0.95
3y	0.96	0 98	0.99	1 00	1 00	0 99	0 99	0 98	0 96
5y	0 93	0.95	0.97	0 99	0.99	1 00	1.00	1 00	0 98
7y	0 92	0 94	0 95	0.98	0 99	1 00	1 00	1 00	0 99
10y	0 90	0.93	0 94	0 97	0.98	1 00	1.00	1.00	1.00
30y	0 87	0 90	0.91	0.95	0.96	0 98	0 99	1 00	1 00

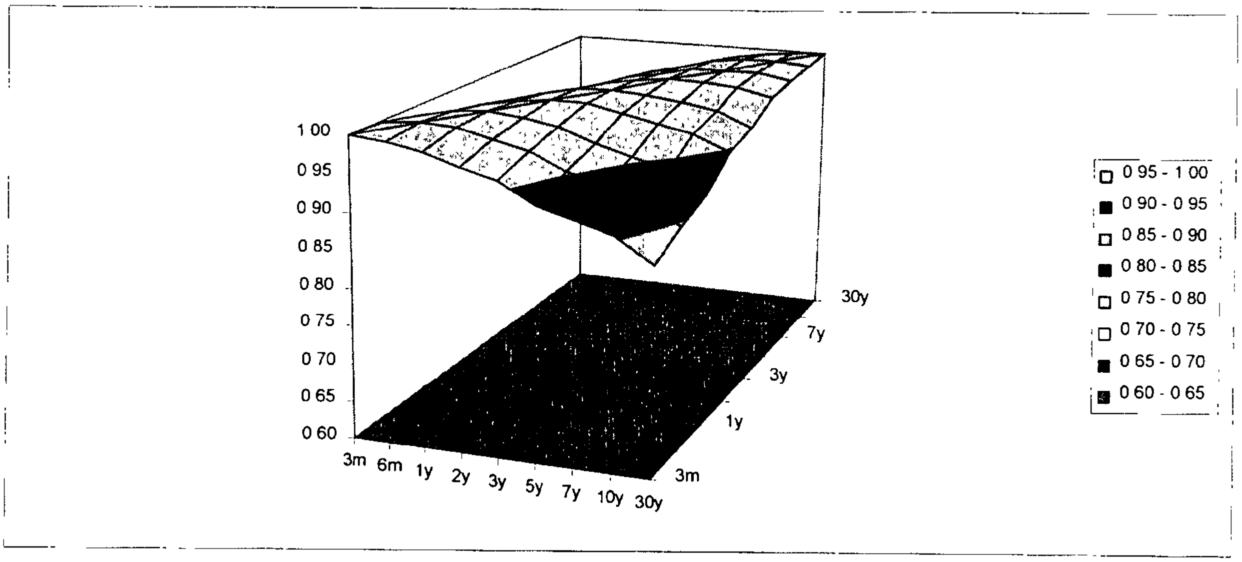


Chart 2 shows randomly chosen yield curves generated by the model. The actual yield curve for the same January 1, 1991 epoch (five-years after the initial point in time) is super-imposed on the chart. In addition, the extreme cases (the lowest and the highest curves) from the entire set of 250 are included in Chart 1. One can see in this chart that the model is capable of reproducing the rich variety of possible yield curve shapes—there are both inverted and normally shaped yield curves.

Charts 3 and 4 show the correlation between yields with different maturities for actual and model data. As it can be seen, the yields are strongly but not perfectly correlated, which is consistent with the stylized fact 1. The values of model correlation coefficients are reasonably close to the actual ones.

Chart 5 demonstrates the closeness of the actual and model-based values of the average spreads between yields with different maturities.

CHART 4
Correlations Between Yield Rates With Different Maturities
Based On Modeling Results

	3m	*6m*	*1y*	*2y*	*3y*	*5y*	*7y*	*10y*	*30y*
3m	1.00	1.00	1.00	0.99	0.99	0.97	0.94	0.91	0.83
6m	1.00	1.00	1.00	1.00	0.99	0.97	0.95	0.92	0.84
1y	1.00	1.00	1.00	1.00	0.99	0.98	0.96	0.94	0.86
2y	0.99	1.00	1.00	1.00	1.00	0.99	0.98	0.95	0.89
3y	0.99	0.99	0.99	1.00	1.00	1.00	0 99	0.97	0.91
5y	0.97	0.97	0.98	0.99	1.00	1.00	1.00	0.99	0.95
7y	0.94	0.95	0.96	0.98	0.99	1.00	1.00	1.00	0.97
10y	0.91	0.92	0.94	0.95	0.97	0.99	1.00	1.00	0.99
30y	0.83	0.84	0.86	0.89	0.91	0.95	0.97	0.99	1.00

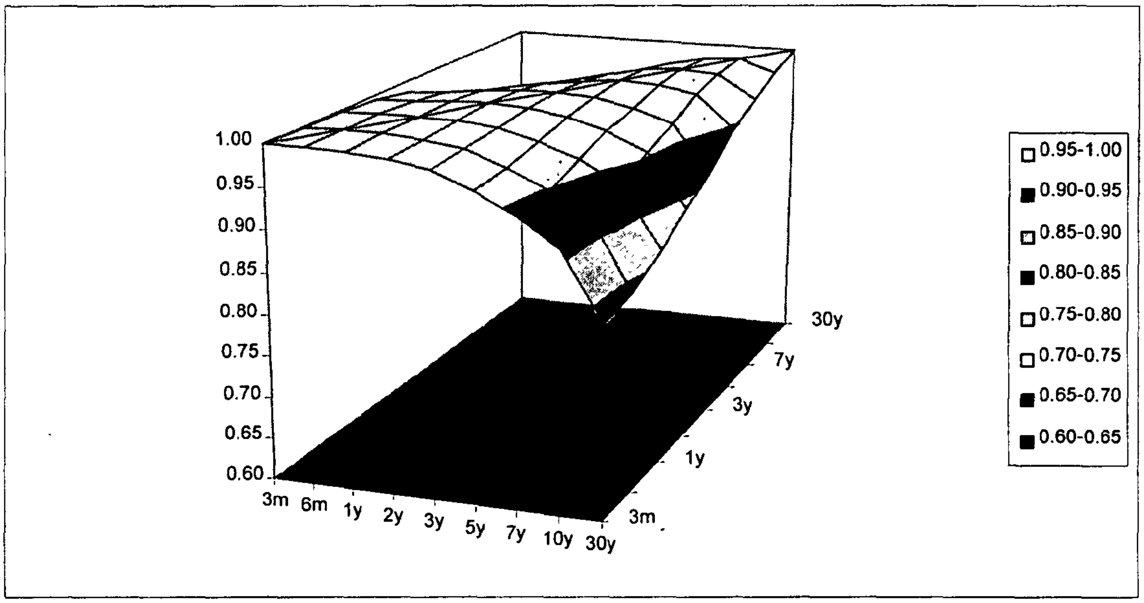


Chart 6 compares the values of the modeled interest rate volatility for different maturities with the actually observed values. Both the pattern and absolute values look satisfactorily consistent.

CHART 5
Actual vs. Modeled Average Spread by Maturity (1986–95)

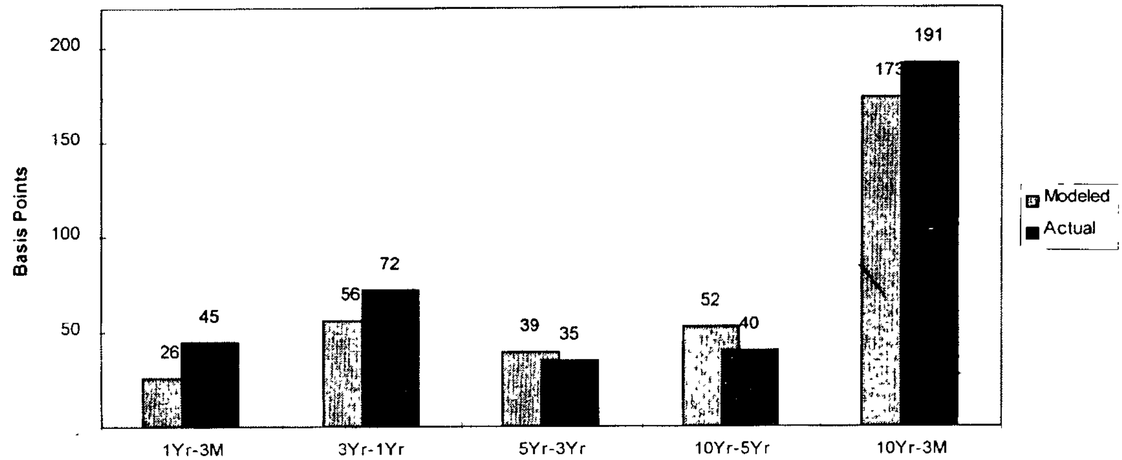


CHART 6
Actual vs. Modeled Average Absolute Monthly Yield Rate Volatility (Annualized, 1986–95)

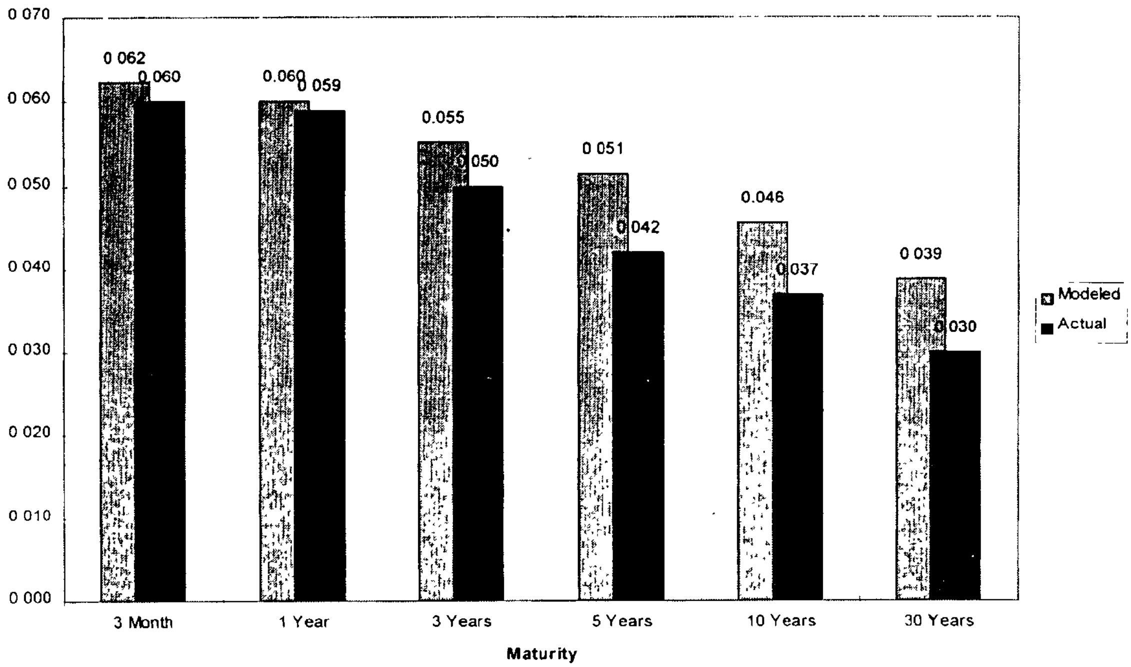
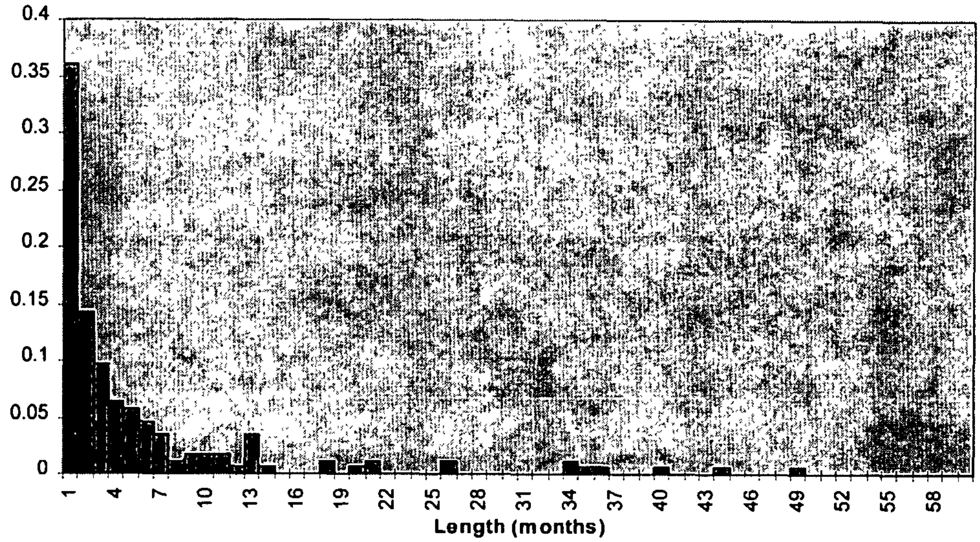


CHART 7
Histogram of the Inversion Period Length
(10 Years Projection, 250 Scenarios)

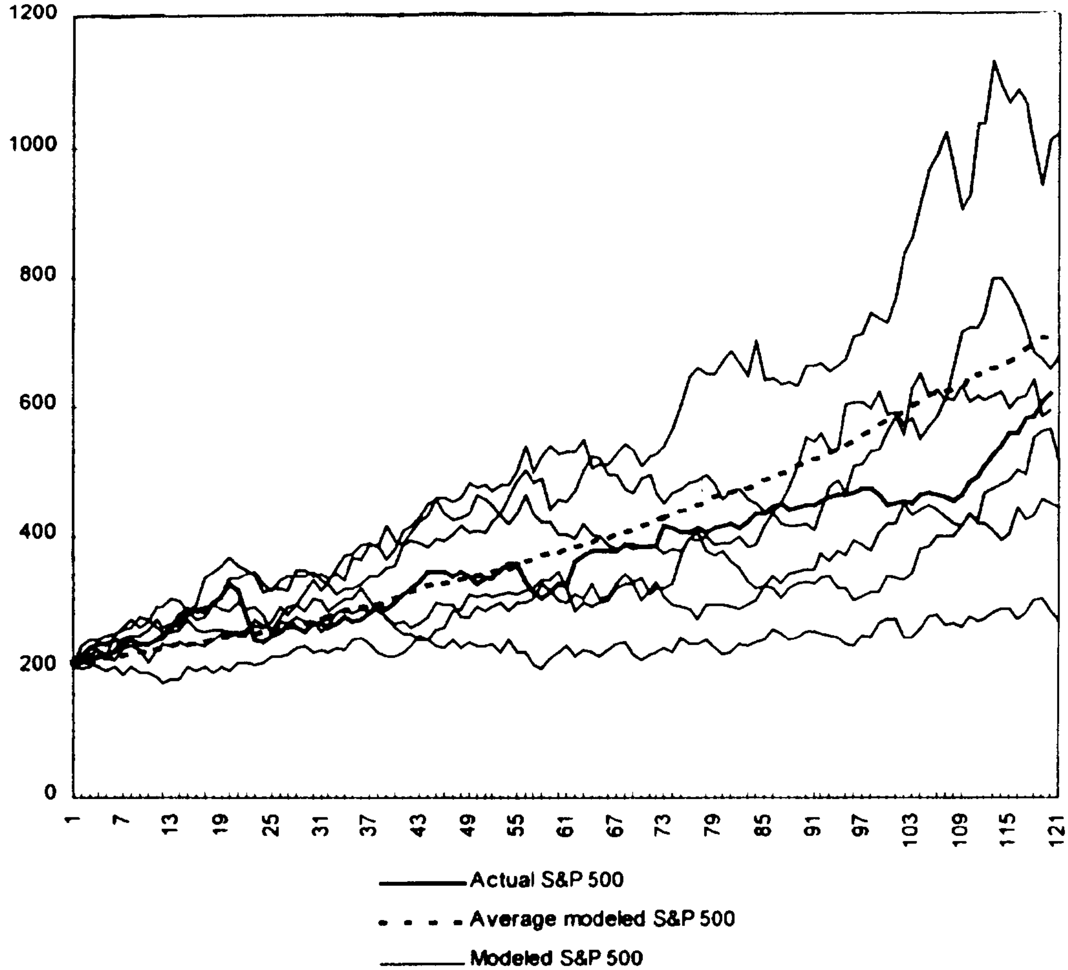


Mean Length of Period:	6.31	Median of Length:	2	% of Inverted Curves Modeled:	7.4%
Mode of Length:	1	STD of Length:	10.5	% of Inverted Curves Actual:	4.2%

The modeled yield curves can be inverted with some small probability. This occurs in the real world, particularly in connection with anticipated or actual policy of the Federal Reserve. The average recovery period is reasonably small, which is also consistent with the long period observations. Chart 7 shows the histogram of the length of the inversion period among the modeled yield as well as some related statistics illustrating this statement.

Some typical scenarios for the S&P 500 index are shown on Chart 8. We randomly picked a few scenarios to illustrate the visual resemblance between the actual and model produced curves.

CHART 8
Actual vs. Modeled S&P 500 (1986–95)



Charts 9 and 10 show the histograms of the S&P 500 return (without dividend yield) both for actual and model data.

Charts 11 and 12 show the typical scenarios and histograms for the model and actual inflation rate data.

CHART 9
Actual S&P 500 Return Histogram (1956–95)

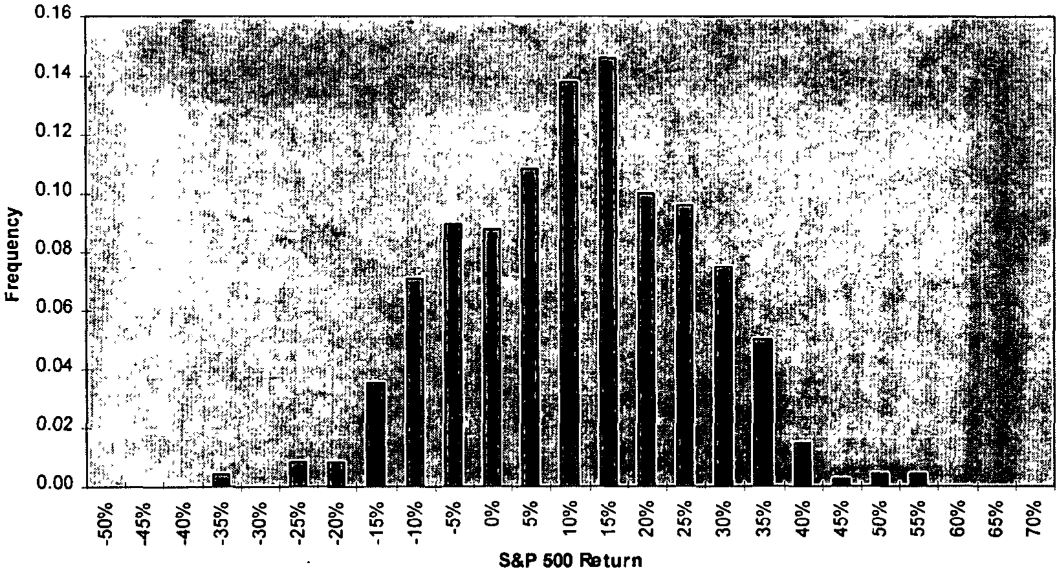


CHART 10
Model S&P 500 Return Histogram

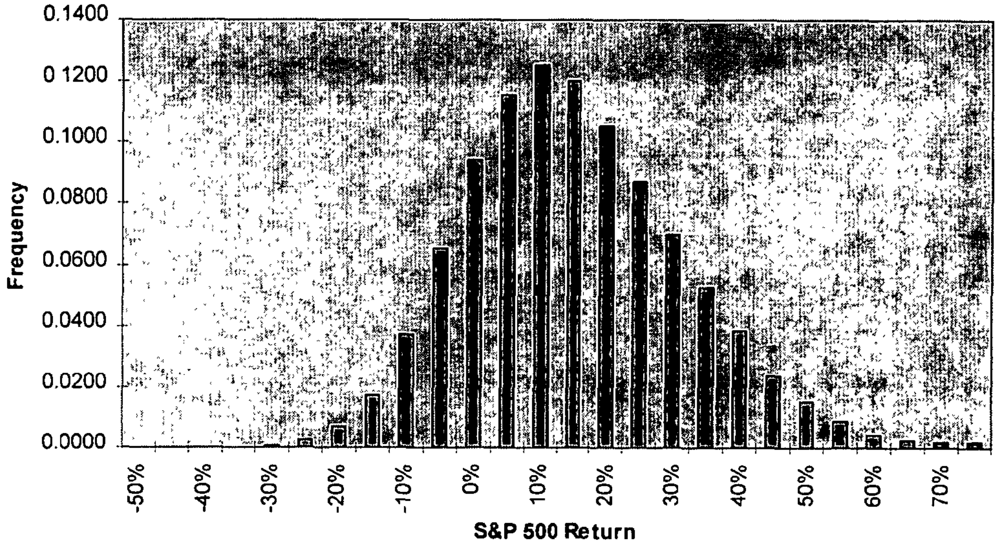


CHART 11
Actual vs. Modeled Inflation Rate (1986–95)

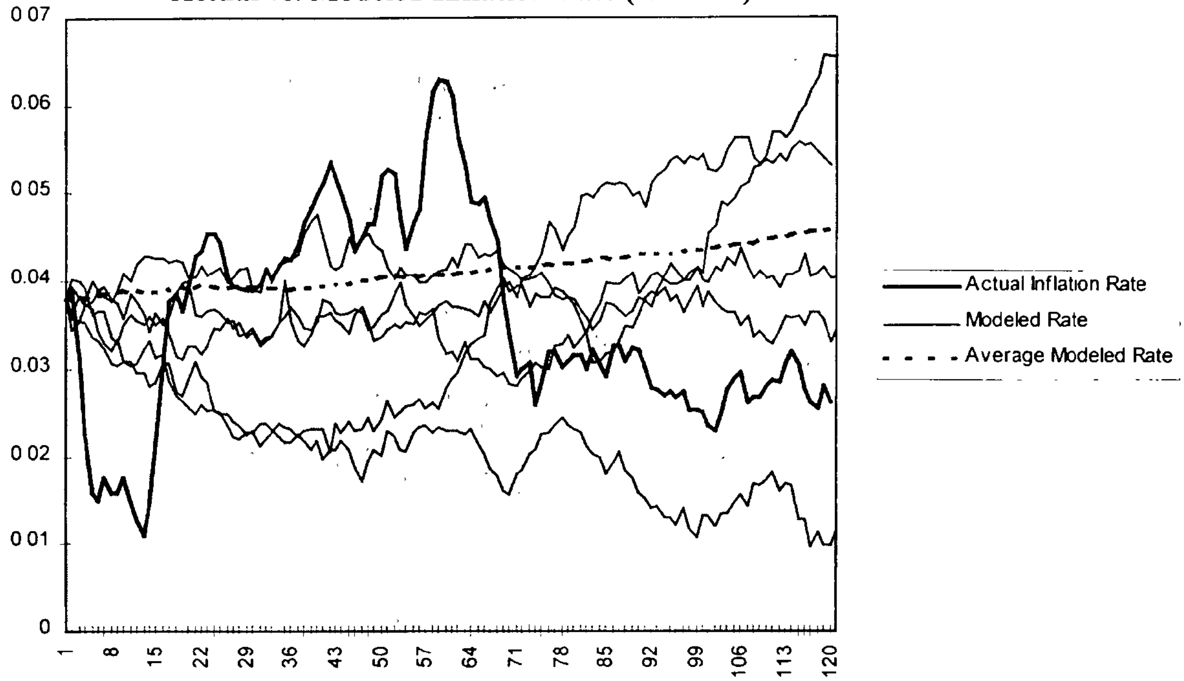


CHART 12
Actual Inflation Rate Histogram (1956–95)

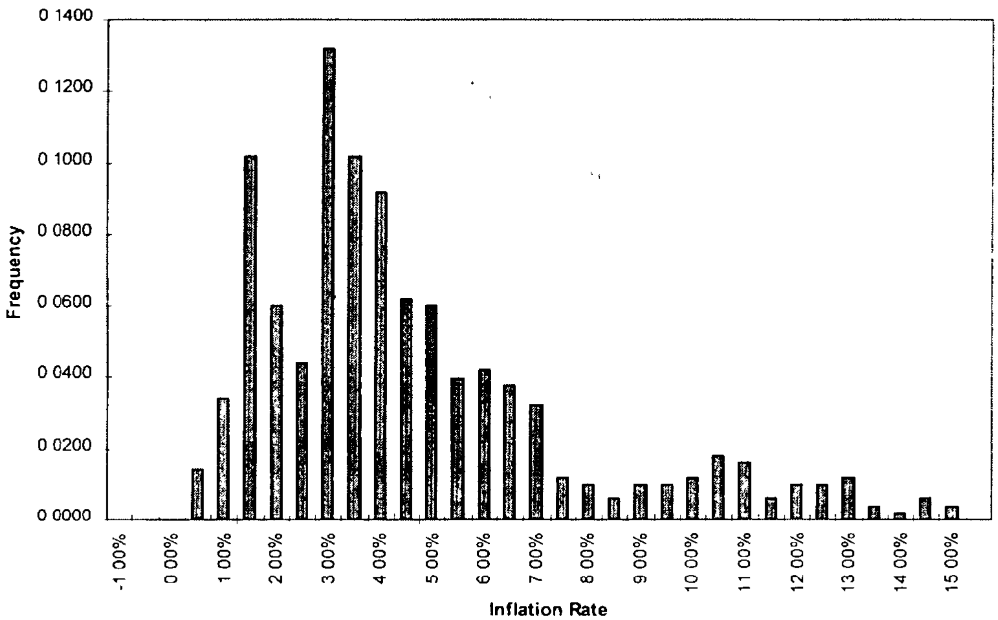


CHART 13
Model Inflation Rate Histogram
(10 Years Monthly Projection)

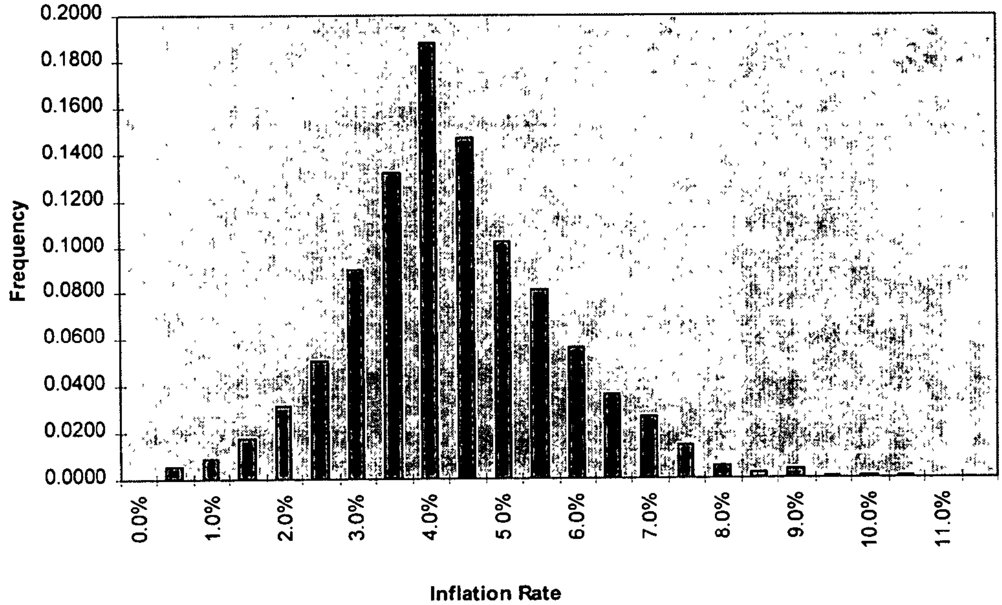


TABLE 2
Correlations to Interest Rate Spreads

	1y-3m	3y-1y	5y-3y	10y-5y	10y-3m
S&P 500 Return, Actual		0.12	0.11	0.15	0.11
S&P 500 Return, Model (Average)	0.13	0.11	0.12	0.13	0.12
Inflation Rate, Actual	-0.16	-0.42	-0.30	-0.27	-0.36
Inflation Rate, Model (Average)	-0.32	-0.34	-0.37	-0.41	-0.37

TABLE 3
Correlations to Yield Rates

	3m	1y	3y	5y	10y
S&P 500 Return, Actual	-0.19	-0.25	-0.28	-0.30	-0.31
S&P 500 Return, Model (Average)	-0.24	-0.25	-0.25	-0.26	-0.26
Inflation Rate, Actual	0.66	0.64	0.62	0.61	0.59
Inflation Rate, Model (Average)	0.63	0.64	0.64	0.63	0.59

TABLE 4
Correlation Between S&P 500 Return and Inflation Rate

Actual (average)	–0.27
Model (average for 250 scenarios, 10 years projection	–0.16

Charts 13 and Tables 2–4 demonstrate the important feature of the model to preserve the actual interrelation between the interest rate, the equity index, and the inflation rate behaviors. According to Becker (1995), the S&P 500 return and spreads are weakly positively correlated (stylized fact 3). The results produced by the model are very close to Becker's figures. The modeled and the actual inflation rates are also correlated with the interest rate spread in a consistent way (stylized fact 7).

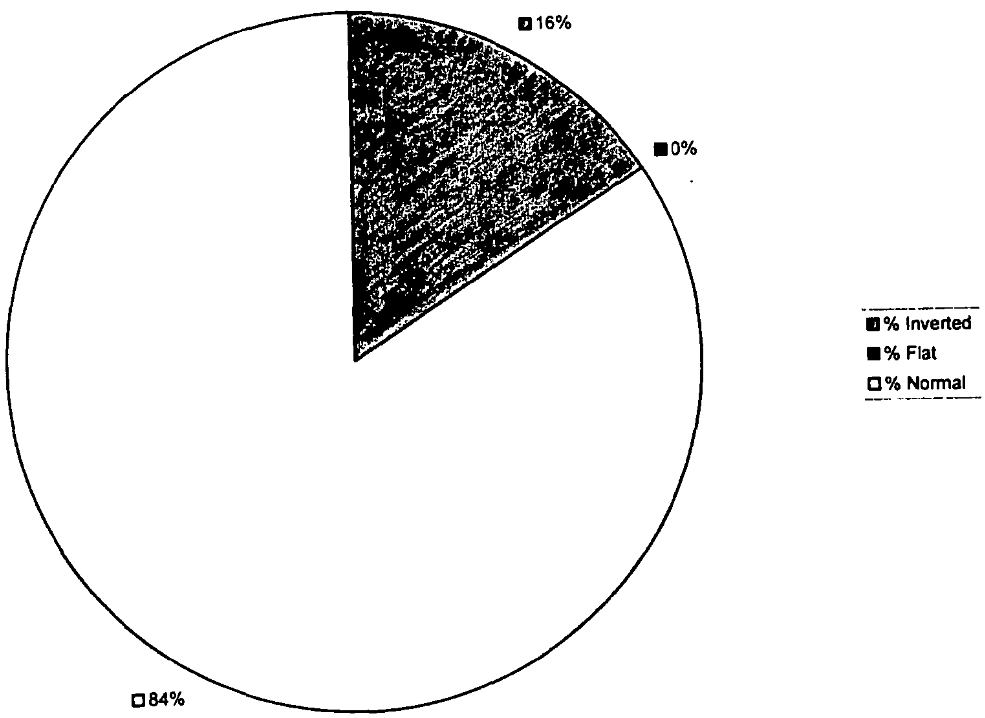
Correlation coefficients between yield rates and the S&P 500 return (stylized fact 5), as well as between yield rates and inflation rate (stylized fact 6) are quite consistent with the actual data. Finally, the correlation between the inflation rate and the S&P 500 return produced by the model is reasonably close to the value calculated based on the actual data.

Finally, Chart 14 shows the percentage of normal/inverted/flat Treasury yield curves. The chart is based on 10 year–3 month criteria.

The next step of the scenario generation is finding initial or starting values for all five model factors, the S&P 500, the volatility of S&P 500, the short-term interest rate, the long-term interest rate, and the inflation rate, corresponding to the current market situation.

There is no problem in finding the initial values for the S&P 500 and the inflation rate. The first is just the current value of the index and the second can be easily calculated from the consumer price index. That is why these two state variables are referred to as "observable" variables. The situation is principally different with the other three variables, the volatility of S&P 500, the short-term interest rate, and the long-term interest rate. They are "unobservable" because there is no source that can provide one with the current market values of these three variables. The initial value for the S&P 500 volatility can be calculated using prices of options on the S&P 500 index or applying the GARCH methodology. The situation with the initial values for short-term and long-term interest rates is more complex, and in our generator, the special fitting procedure was developed for calculating these initial values.

CHART 14
Percentage of Normal/Inverted/Flat Treasury Yield Curves Based on 10 Year–3 Month Criterion



I am going to give a brief description of the fitting procedure, and this is the proper time to explain the exact meanings of the words "short-term" and "long-term interest rates." In accordance with the well-established approach, interest rate models are built around the short-term risk-free interest rate, which is assumed to be a time-dependent random function, $r = r(t)$. More precisely, the short-term interest rate $r = r(t)$, is the rate that applies to an infinitesimally short period of time, $(t\text{-}\Delta t,\ t+\Delta t)$. In the five-factor model, the equation governing the stochastic process $r(t)$ is the commonly used mean-reverse equation of the five-factor system of equations (3). This means that as time evolves, the short-term rate is assumed to converge to its mean reverse. However, what is less common, but is becoming more and more popular, is the mean reverse of the short-term interest rate is not a constant. In its turn, this mean reverse is a stochastic process, $u = u(t)$, which is referred to as the *long-term interest rate*. In the five-factor model, governing the stochastic process $u(t)$ connects the long-term interest rate with the volatility of the S&P 500 [see equation (4)]. Because of this connection, this model is able to deal simultaneously with the debt market and equity market.

In addition to five stochastic differential equations, the model contains a partial differential equation, which is used in order to calculate yields of the Treasury yield curve given the model parameters and values of the five factors. This implies that at time zero, one has two Treasury yield curves, the real market curve and the curve calculated from the initial values of five factors and the model parameters. The objective of the fitting procedure is to find such initial values of short-term and long-term interest rates, which provide the best fit of the calculated initial yield curve into the real market one. Technically speaking, the fitting problem is equivalent to the problem of finding a minimum of a nonlinear functional. The five-factor generator contains a block of numerical routines that solves this problem of implementing the Levenberg-Marquardt algorithm.

The output of the generator fitting process is initial values for short-term and long-term interest rates and a set of time-dependent parameters called adjusting coefficients. The purpose of these coefficients is to eliminate the discrepancy between computed and real market yield curves. The usage of the adjusting coefficient makes the five-factor model a hybrid between the equilibrium and the arbitrage-free model. It means that it allows you to start with a real yield curve when you do your runs.

The five-factor generator uses the classic Monte Carlo method to produce stochastic scenarios. The major numerical procedure is the calculation of the conditional probability density function: $f(p, v, r, u, i, t \mid p_o, v_o, r_o, u_o, i_o, t_o)$, which is equal to the probability that at time t, the S&P 500 index, its volatility, the short-term interest rate, the long-term interest rate, and the inflation rate will take the values $p, v, r, u, i,$ given that at the initial time, t_o, their values are p_o, v_o, r_o, u_o, i_o. The function f is the key for the scenario generation process. As soon as the conditional probability of

f is known, the values of the five factors can be easily calculated. The generator uses the current values of five factors to calculate yields on Treasury securities with different maturities. As I mentioned earlier, the calculation of yields is based on the partial differential equation of the Black-Scholes type.

The generator output file is the sequence of monthly projections of the state stochastic variables along with nine yields of the Treasury yield curve. The user can specify the number of scenarios and the length of the scenario. It takes about 25–30 minutes to generate 1,000 scenarios of a 30-year length.

As I said, the generated scenarios are subject to the valuation procedure. Usually, to make sure that the scenarios are "reasonable" we calculate the following statistics:

- Percentage of normal/inverted Treasury yield curves
- Correlation between yields with different maturities
- Average spread by maturity
- Average annual yield volatility
- Average annual return on S&P 500 (without dividends)
- Average annual volatility of S&P 500

The calculated statistics are compared against statistics obtained form the historical data. Recently, I completed the set of stochastic scenarios based on the Treasury yield curve as of June 30, 1998, and I would like to share with you one interesting observation I made over these scenarios: about 15% of the scenarios show extremely low yield on the Treasury securities with short maturities and they also show deflation.

I'd like to share how the debt-equity generator is used with the other SS&C Technologies' products, such as Finesse and PTS.

- Assets Analytics
  - Basis for creating floating rate models
  - Basis for discounting cash-flows
  - Projecting realistic equity indices
  - Basis for modeling quality spreads
- Liability Analytics
  - Inflationary impact on liabilities
  - Discounting liability cash flows
  - Basis for projecting equity-based policies (equity-indexed annuities and variable annuities)

A few words about the recent modifications that have been made to the model and generator. A three-factor model and implementation of this model generator were developed as a derivation of the five-factor model. This three-factor model is a pure equity model and allows one to calculate monthly projections for the emerging markets, developed markets, and small cap indices. In the nearest future, we are planning to replace or complement the Monte Carlo simulations with a hybrid between Monte Carlo and low discrepancy sequences.

I would like to give you a list of references pertaining to my presentation. I think that the books and papers on this list might provide you with valuable theoretical and practical knowledge on the subject of interest rate stochastic generators.

References

1. Fitton, P. and McNatt, J., "The Four Faces of Interest Rate Model," In F. Fabozzi, ed. *Advances in Fixed Income Valuation: Modeling and Risk Management*, Frank J. Fabozzi Assoc., 1997.

2. Hull, J., *Options, Futures, and Other Derivatives.* 3rd edition. (Prentice Hall, 1997) (See also the lists of references provided in this book).

3. Chen, Lin, *Interest Rate Dynamics, Derivatives Pricing, and Risk Management*, Springer, 1996.

4. Fishman, V., Galperin, Yu, Reynberg, A., "Five-factor Stochastic Debt-equity Model," Submitted for publication.

5. Becker, D. "Stylized Historical Facts Regarding Treasury Interest Rates" *1955 to 1994 Technical Report*, Lincoln National, Fort Wayne, Indiana, 1995.

6. Fishman, V., Fitton, P., and Galperin, Yu., "Hybrid Low Discrepancy Sequences: Effective Path Reduction for Yield Curve Scenario Generation," *Journal of Fixed Income*, (June 1997).

MR. HONGFEI ZHANG: The previous two speakers talked about all these complicated, multi-factor economic scenarios. When it comes down to practical applications and you do your simulation, you realize that it takes a long time to do it. So, what I'll be talking about is a little bit technical. I will talk about one way of reducing the number of scenarios to get the maximum amount of information out of this. In this application, we talk about the cash-flow testing we have done at Nationwide Financial Services. All the computations were done by Steve Craighead.

My objective is to give you simple illustrations of the effective use of low discrepancy sequence (LDS) in cash-flow testing with multi-factor economic models.

You're given a set up for economic scenario generators. What do you do? You do a Monte Carlo simulation using whatever generators you have from Excel or from whatever programs you got from the Internet. Monte Carlo simulation was designed by scientists at Los Alamos, initially, for multi-dimensional integration during the Second World War for testing nuclear bombs. It has consequently been applied to the pricing of financial instruments and the derivatives. You probably have heard a lot about this on Wall Street. Of course, when you generate a random path, you're going to use it. In the insurance setting, you can do asset/liability analysis, cash-flow testing and stress testing and you use different Monte Carlo simulation techniques.

The drawback with the Monte Carlo simulation with the pseudo random number generator is that the computer generated the random number. If you plot those graphs of random numbers, it's not very uniform. It takes a lot of scenarios to cover whole spaces. One of the methods to improve your simulation is to find the analytic formulation, if you have a close formula. You would be lucky to do this. Most of the time, you can't get lucky on this.

You can also do a numerical simulation of partial differential equations. This is very technical. But this is restricted only to a maximum of a three-dimensional case. When you have a five-factor model like SS&C is developing and this method also breaks down, you can also use a combination of variance reduction methods. Antithetic variables, stratified sampling, importance sampling, and control variates. Another method I'm going to talk about is low discrepancy sequences. It's also called quasi-Monte Carlo sequences.

Basically, the LDS is a bunch of different meshed sequences with low discrepancy. Discrepancy is a measure of uniformity of how you cover your space. You know the sequence cover better. Examples of those LDS, which were developed in 1920 and in the 1950s, are Sobol, Halton, Van der Corput or Faure. More recently, there have been a bunch of derivatives of those sequences called a generalized Faure, generalized Sobol and generalized Niederreiter sequences. Those sequences are generated through some very complicated mathematical theories using fields and prime numbers.

I will give you one simple example of LDS. Say you want to generate some sequences to cover the interval from zero to one. How did you do it more efficiently? Excel can do it for you. Another way to generate is to take a mid-point rule:

$$-1/2,\ \{1/4, 3/4\},\ [1/8, 3/8, 5/8, 7/8\},\ \ldots,$$
$$\{2i-1, /2N\},\ \text{where } i = 1, \ldots, N.$$

Notice how I group those things together. When people talk about the application of LDS, many people just think we use random numbers or when they want to do a simulation, they use ten points. Actually, this is a flaw in using LDS because if you want to achieve low discrepancy, you have to use sequences that are called nets. I grouped those together. In the sequence above, you might think you can take half of one-fourth. What about the other one? You probably should take three-fourths. The same applies to the other group.

The use of nets in LDS is really crucial. I want to emphasize that it must be used in bundles of nets, otherwise, you won't achieve the effect you want. Even the bundles of nets are sometimes not enough. An example of a net in generalized Faure sequences in d-dimension is generated by the least prime number larger than or equal to the dimensions. If you want to generate a four-dimensional sequence, you can take up the prime number five, or in 82 dimensions, you pick up the prime number 83. We'll emphasize 82 because that's the number we have used on experiments.

The next size in 82 dimensions is simply the multiple of 83. So you have 83, two times 83, and three times 83, and so on. You might say, "I want to do 100 simulations", but you probably will get very screwy results.

Another technique, which is one we are currently testing right now at Nationwide looks very promising. It is the Brownian Bridge method. What is a Brownian Bridge? We know that all the stochastic differential equations have some sort of Brownian motion. You write on stochastic differential equations and you just generate the path this way:

$$Xn+1=Xn+\mu*\Delta t+\sigma*sqrt(\Delta t)*z, n=0,1,2,...8.$$

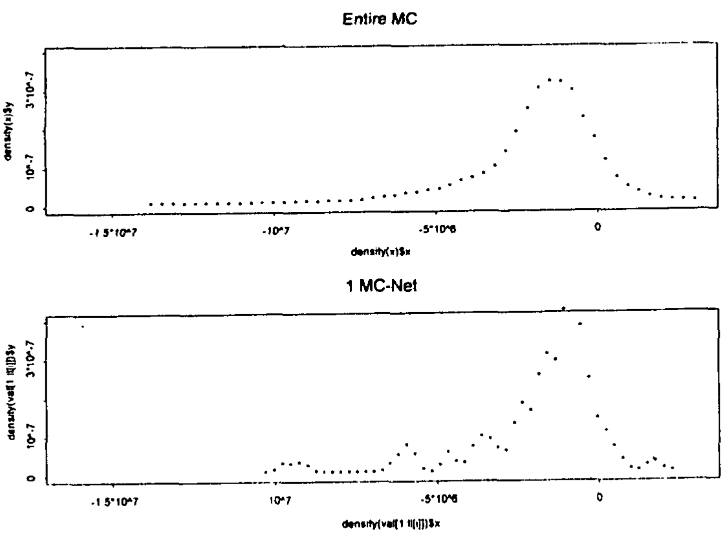
Brownian Bridge is a way to generate a path, but in a different order. When you generate a path with the Brownian Bridge, you sort of scramble the order of generation, but the total variance stays the same. Effectively you are generating the same path from a normally distributed variable, but you re-order them in the way you want. Let's say you have this one- factor model, the standard, stochastic differential equation. You have probably seen this one. So how do you do this? We just take the first point and simulate one again from a random number. You go to the second point, third point, and so forth. They generate a path. You plug in whatever software you have, such as PTS. Then you do your simulation and see what the result comes out to be.

The order is different in Brownian Bridge. The first point is normally given. You're given the initial yield. Then, I would generate the last point on the path but keep the variance checked. Then I would generate once in the middle. Then I would go half-and-half and half-and-half again. You don't have to generate in this order. You can generate in any order you want. The advantage of doing this is, when you have your block of business, you place some importance on your block of business. The seven years is really important in this case. My duration tells me that the first seven years is probably very important, so I want to emphasize the first seventh year. You can do it in this order.

As an example, I will show you some results of cash-flow testing on bank-owned life insurance (BOLI). This line of business is very interest-rate sensitive, so we use Tenney's Double Mean Reverting Process™ (DMRP) interest rate model. We tested this business on various techniques used in LDS, of course. We also tested 10,000 Monte Carlo simulations, and various Brownian Bridge simulations. We tested over 20 years with quarterly time intervals for the first seven years, so this gives you 28 dimensions. Then, we tested annually for the next 13 years, which gives you 41. Now we have a two-factor model. So the total dimension of the problem is 82. I'm going to generate this using the net size of 83.

The top graph of Chart 15 shows the cash distribution. Its present value is discounted at 11%. The option-adjusted distributive earnings are at 11%. The top graph was generated using 10,000 Monte Carlo simulations. We have the luxury of doing this. You can assume this is a real distribution for your business. You must think 10,000 is enough. The bottom graph is the standard Monte Carlo simulation, using only 83 scenarios. You might think it's a little bit off. The tail is not really reflected here.

CHART 15
Entire MC/1MC–Net



Let's double the size of the Monte Carlo simulation in Chart 16. You can use it any way you want to. This is two times the net size, which is two times 83, or about 160 scenarios. You can see that it's getting a little bit better. In Chart 17 I'm going to go to three times that one, and in Chart 18 I'm going to 24 times the Monte Carlo net. Even after 2,000 scenarios, you're still up a little bit from the 10,000. You might think this will be the best estimate you have.

CHART 16
Entire MC/2MC–Net

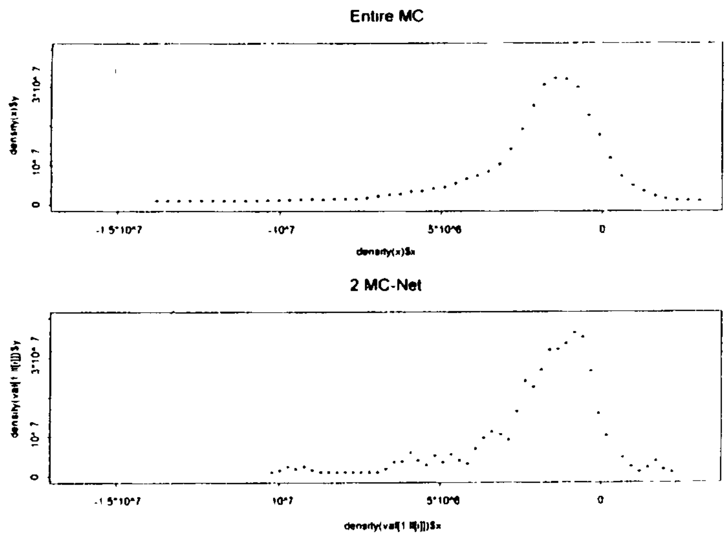


CHART 17
Entire MC/3MC–Net

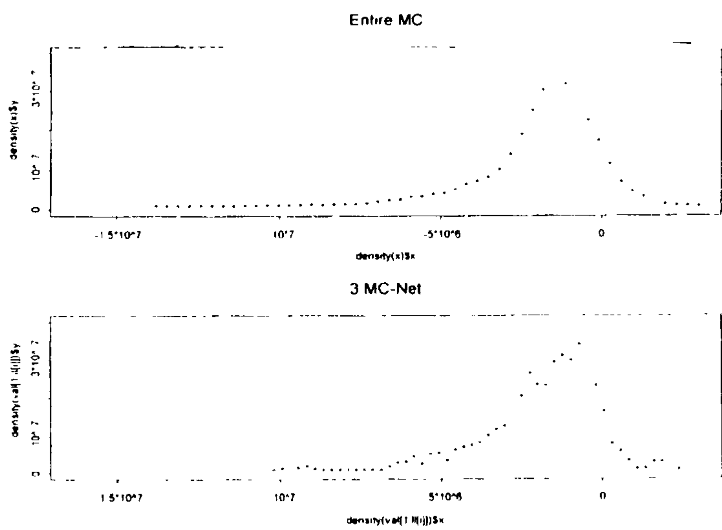
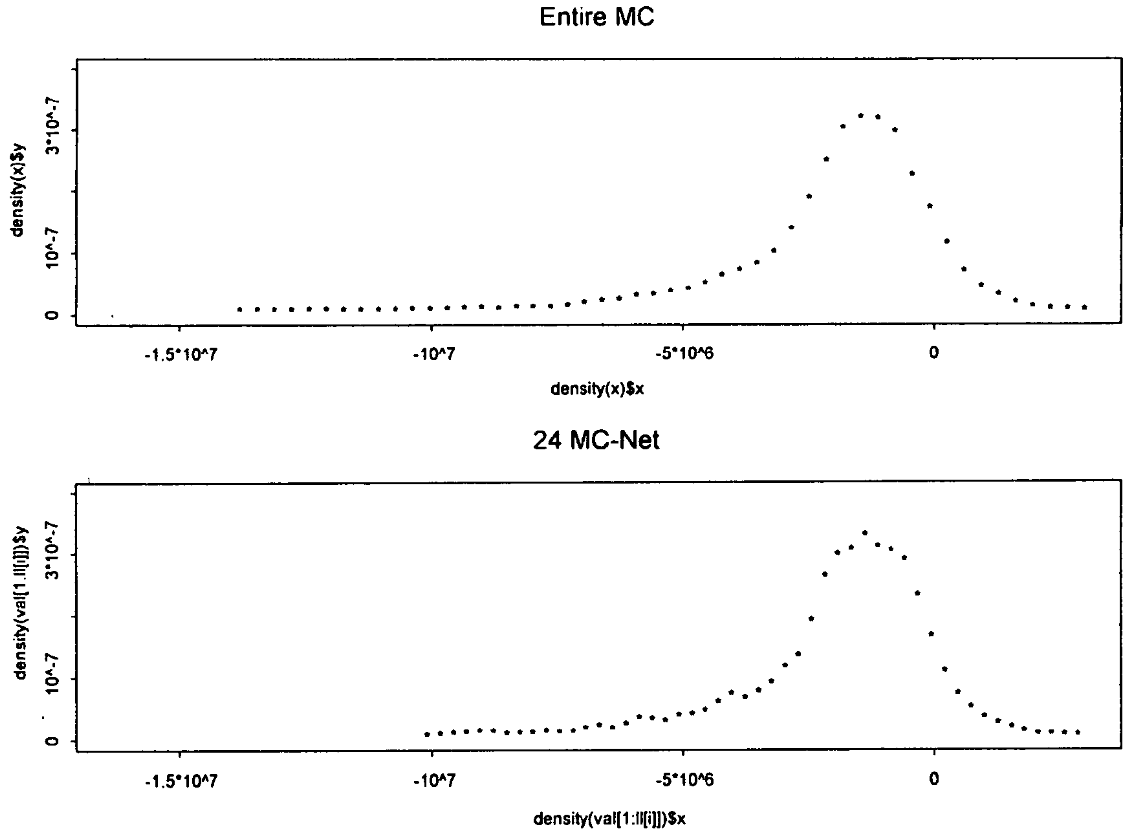


CHART 18
Entire MC/24MC-Net



Let's look at LDS. Chart 19 shows the distribution if you use only 83 LDS scenarios. Chart 20, at two times the LDS, looks a little better. That tail picks up a little bit. At three times LDS (Chart 21), it doesn't change that much. At 24 times LDS or 2,000 scenarios (Chart 22), you see it's okay, but you're probably not going to be happy with that. The tail will be a little bit longer.

CHART 19
Entire MC/1LDS–2

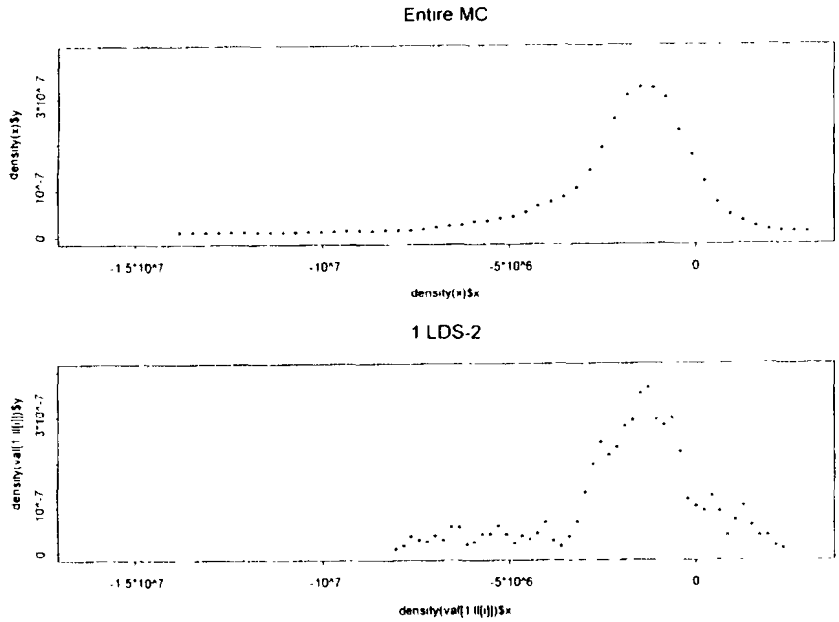


CHART 20
Entire MC/2LDS–2

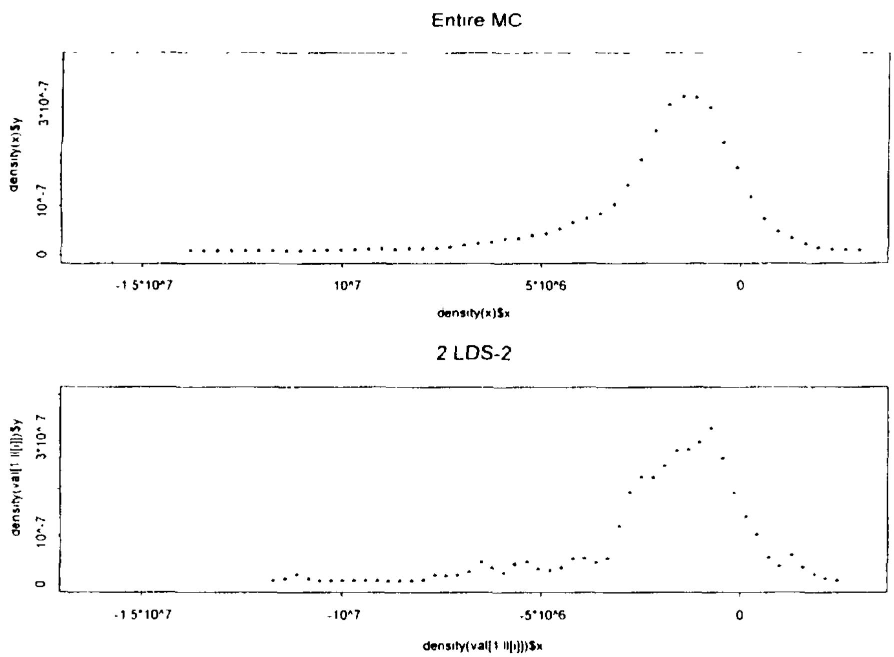


CHART 21
Entire MC/3LDS–2

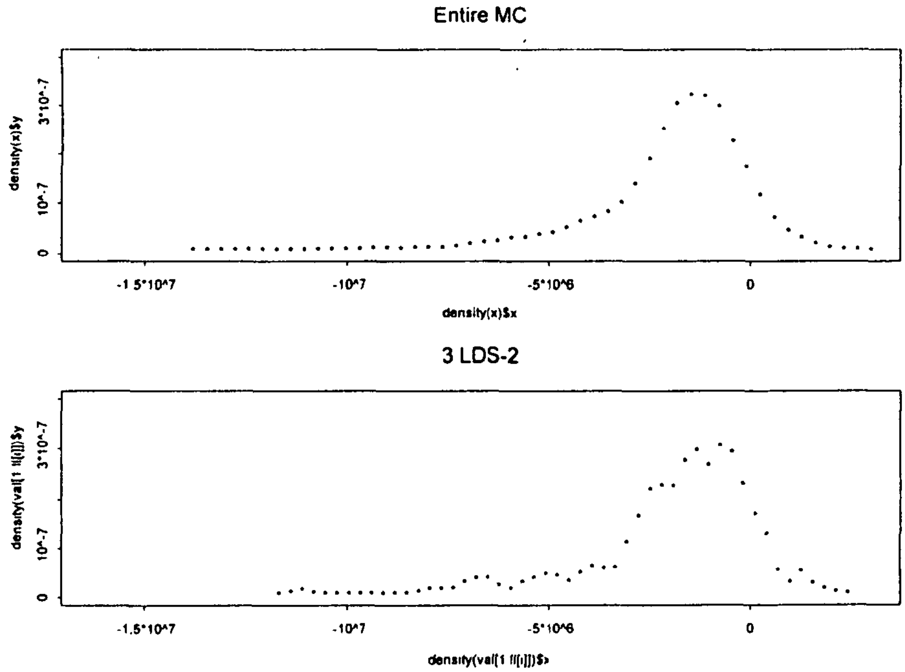
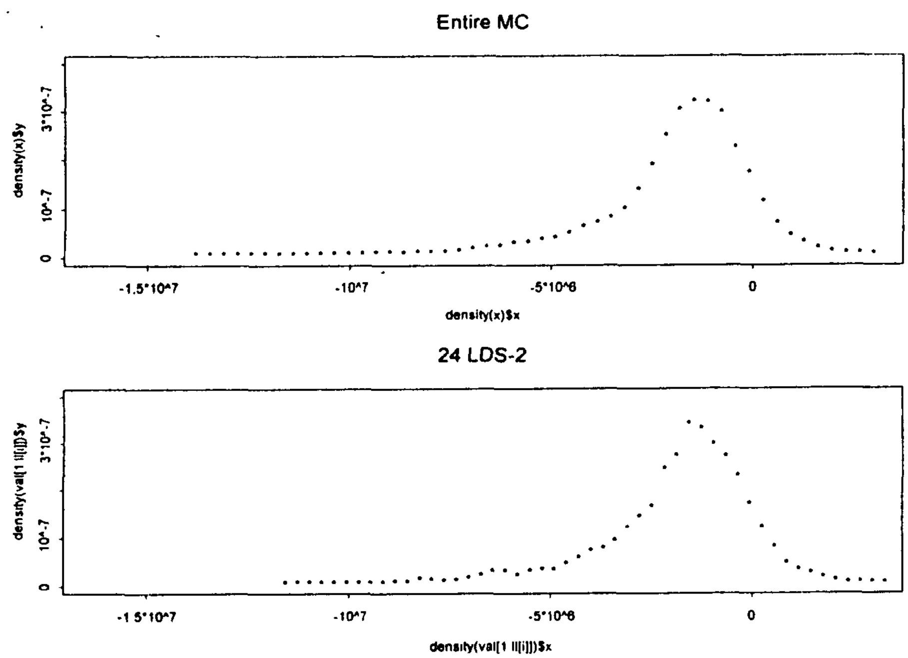


CHART 22
Entire MC/24LDS–2



Now we test Brownian Bridge using the LDS. Chart 23 is just generated randomly using Brownian Bridge, without regard to what business we are talking about. In the distribution, you get this little hump for only 83 scenarios. And Chart 24 is two times 83; Chart 25 is three times 83, and you pick up more tails than you wanted. At 24 times 83, or about 2,000 scenarios, Chart 26 shows the distribution you would get.

CHART 23
Entire MC/1BB–3

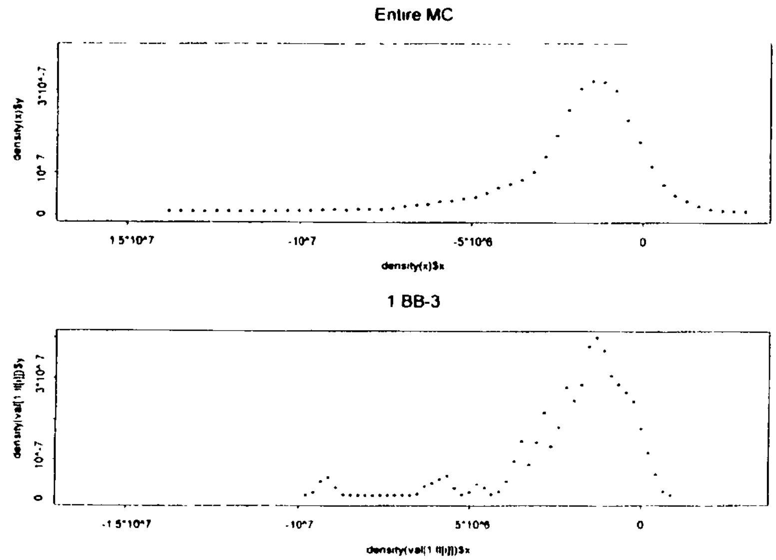


CHART 24
Entire MC/2BB–3

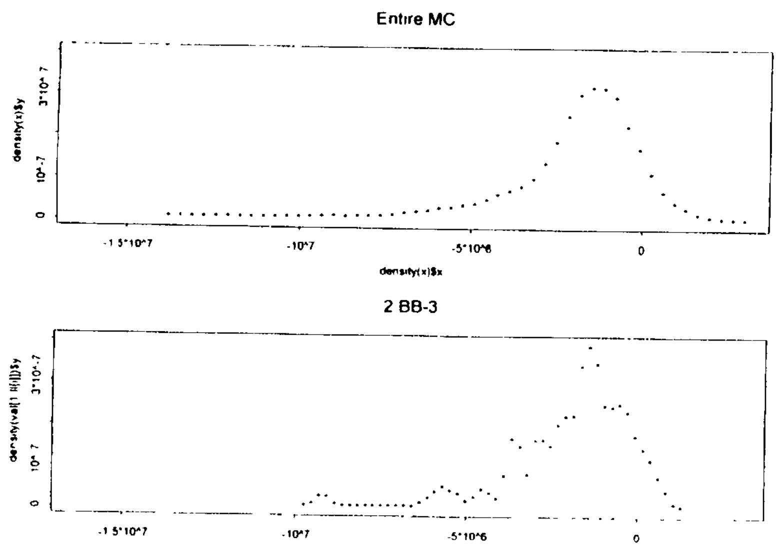


CHART 25
Entire MC/3BB–3

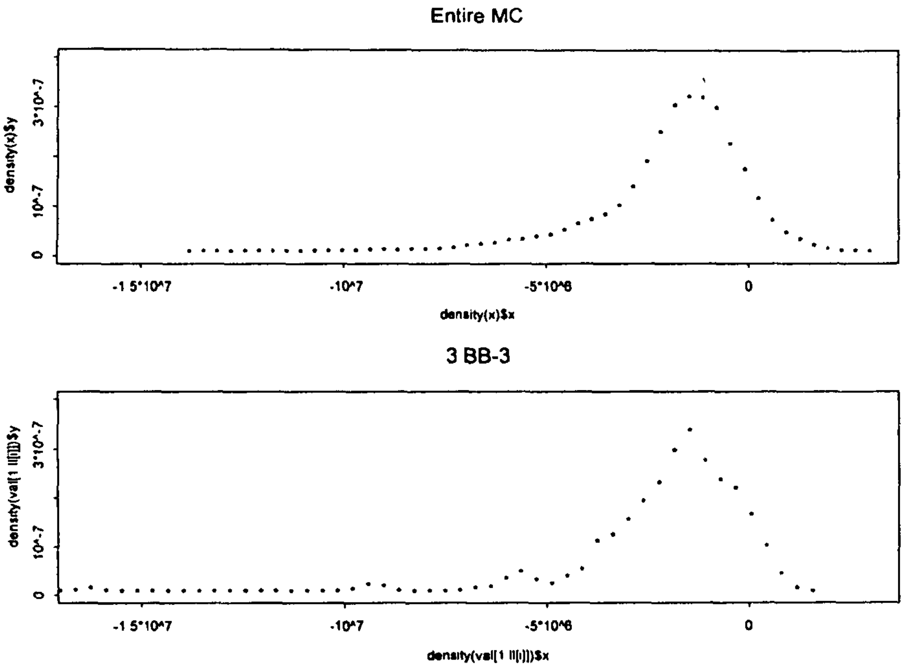
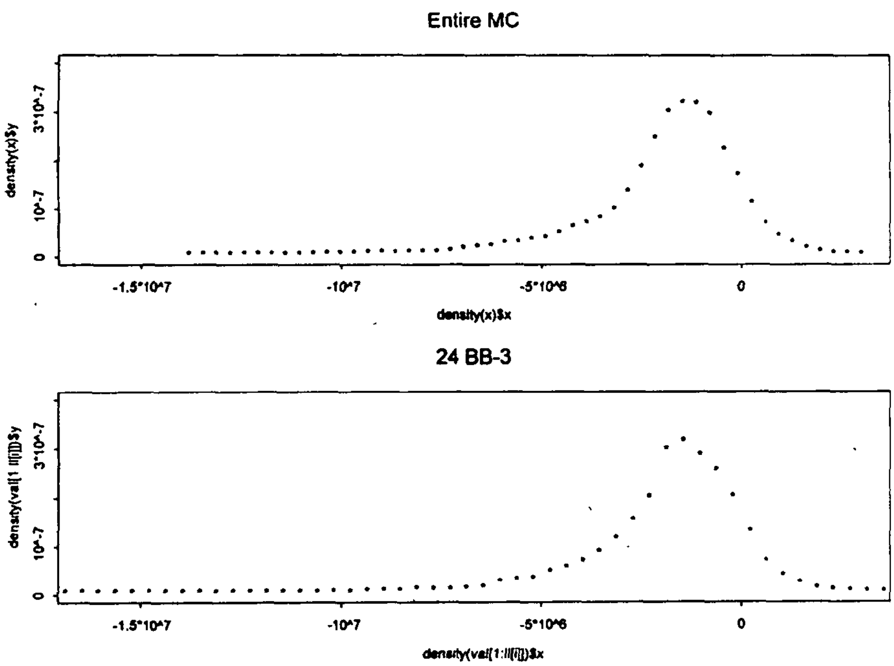


CHART 26
Entire MC/24BB–3



That's a different way of generating Brownian Bridge. The next series of charts we jokingly termed Brownian Bridge 4.5 (Charts 27-30). We want to do a quarterly simulation for the first seven years and an annual one for the next 13 years. I generated Brownian Bridge for the first seven years first, and then I put them together for the next 13 years. You think this would be the way it's done because you place emphasis on the first seven years, so this is what it is. You see 83 scenarios in Chart 27, two times 83 in Chart 28, three times 83 in Chart 29 and is 24 times 83 in Chart 30. It looks pretty good.

CHART 27
Entire MC/1Alt BB–4.5

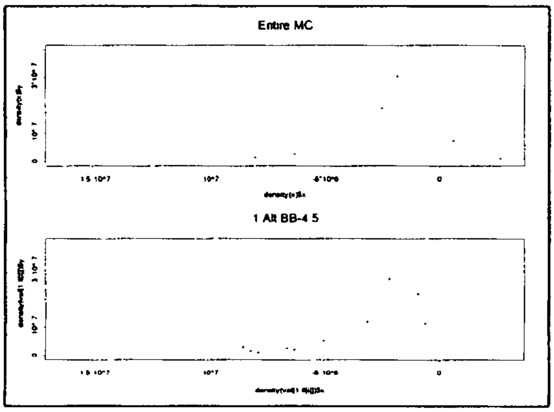


CHART 28
Entire MC/2 Alt BB–4.5

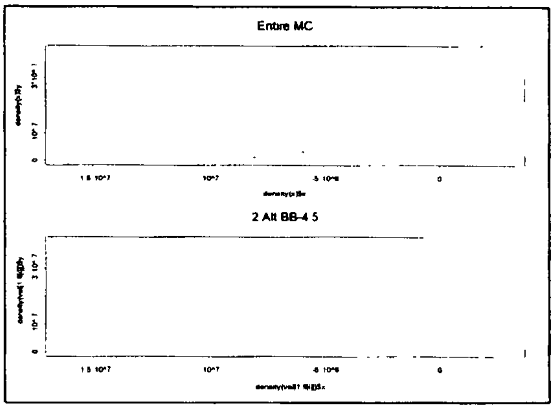


CHART 29
Entire MC/3 Alt BB–4.5

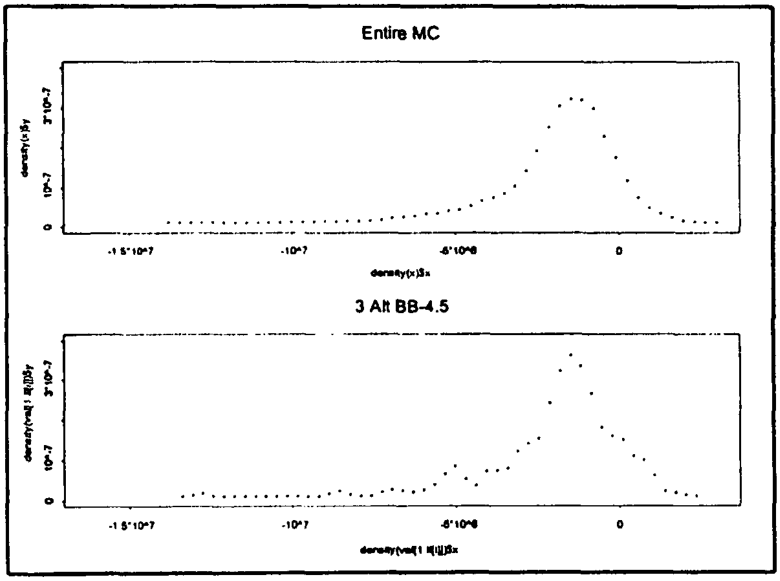
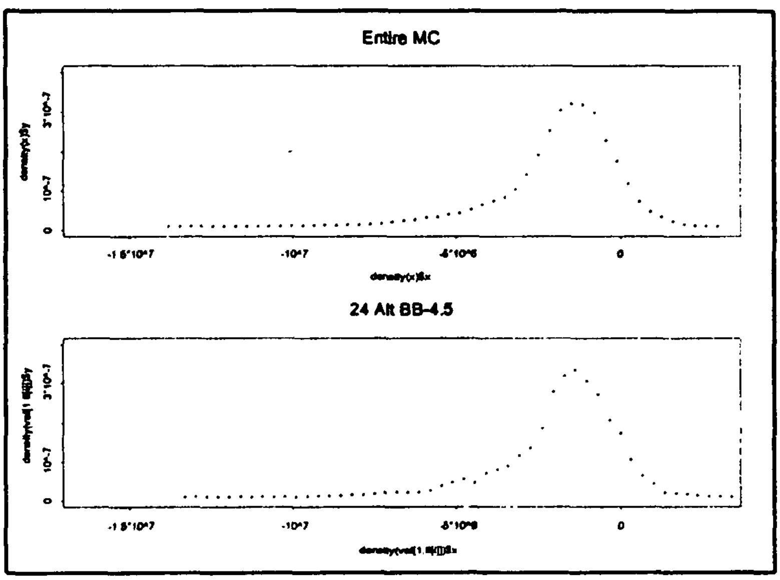


CHART 30
Entire MC/24 Alt BB–4.5



Our future research would apply this Brownian Bridge method to more factor models like five. There might be some other ways of using LDS. There is some research down at the current stage, but it's still very premature to talk about it right now.

MR. MICHAEL F. DAVLIN: As Steve mentioned, I've done a bit of work in this area over the last few years. Actually, the first Monte Carlo simulation that I ever did with an interest rate model was done over 20 years ago when I worked for a large insurance company on the West Coast that sold a lot of term insurance. At that time, we were filing a whole life product in a variety of states. One state wanted us to offer policy loans at an interest rate that was 1% lower than we were offering in other states. That didn't strike me as fair so I did the natural thing, which was to crank up my APL (A Programming Language) interpreter. I created a little Monte Carlo model where I assumed the changes in interest rates had a triangular distribution because I knew how to easily invert that distribution when generating random interest rate paths. I ran a lot of scenarios, calculated what the premiums should be under each scenario with our profit objectives, generated a lot of fancy graphs, and then ran to management to show them what the sizeable premium difference should be. I was surprised to find out that the correct premium difference was, in fact, zero. The reason is because those are good loans and the policyholders are borrowing their own money. So what's the problem?

Since then, models and management's attitudes have changed quite a bit, but that experience definitely set my direction on a different course. I decided I was probably better at implementing tools and letting other people deal with explaining the results to management. So I have been specializing in software. In the early 1990s, my company, subcontracting to Chalke, Inc., developed the first implementation of Mark's model, the DMRP™. We also integrated the INTEX collateralized mortgage obligations (CMO) simulation model into the Profit Test System (PTS). Many of you have probably used software that I've worked on. Additionally, if you've ever gone to a Brunswick bowling alley, rolled three strikes in a row, and had some turkeys slap hands and gobble, that might have been something I worked on, too. I'm so proud.

As Mark, mentioned, I think you really have to be demanding of your software developers these days. I think actuaries have been too easy on software developers, and you are paying the price for it. I'm going to talk about some of the software aspects of economics in our generation. Some of this has already been covered.

There are really three perspectives to economic scenario generation: an economic perspective, a mathematical perspective, and a software perspective. Mark has covered some of the economic desiderata in economic scenario generators in several SOA presentations. These desiderata include the ability to model sovereign debt, or the Treasury yield curve in the United States, or British yield curves or several other countries' yield curves. You may find that you need different interest rate models for different countries. If you are in multiple currencies, you're going to want to have currency exchange models. You might want to have different models for corporate debt. Within the corporate debt category, you might want to have a different model by sector or grade, rather than having to use simple add-on spreads. It depends on how elaborate you want to get.

As Vladimir mentioned, modeling indices are increasingly important for equity-linked products. Debt indices are important as well. This is something I felt we always kind of overlooked when we implemented INTEX in PTS. Many CMO's have cash-flows that are driven by debt indices such as the London Interbank Offered Rate (LIBOR), or the 11th District cost of funds. We never put stochastic models in to model those properly.

Another category of the economic desiderata is the "who knows?" category, and it is probably the most important one of all. Who knew five years ago that we were going to need all of this? Who knows what we're going to want in five or ten years? One of the arguments I want to make is that any economic scenario generator that you build or purchase, ideally, should be extensible and compatible with what you already have. Mark has already mentioned that you want to be able to deal with both realistic and risk-neutral probability measures on all the different components of the model. You're going to want to have some sort of fitting and calibration routines for both parameters and state variables in your models.

One of the mathematical features you might like to have for performance reasons is the ability to exploit analytic solutions whenever those are available. Apparently, Vladimir is using those pretty extensively if he is able to process scenarios as quickly as he said he could. That's really good performance. However, analytic solutions are not always available, so you're going to want to use advanced Monte Carlo techniques in other areas like some of the features that Hongfei mentioned earlier: LDS, antithetic paths, Brownian Bridges, and control variates.

That brings me to some of my pet peeves as a user of software. I don't know if any of these will resonate with you. I don't like to be locked into one vendor. I'd like to be able to mix and match different vendors' products if I possibly can. As an example, I want to purchase an interest rate process from one vendor and a stock model from another. Ideally, I'd be able to take those, compose a larger model that combines the two, correlate the deviates in each of the two submodels and come up with an aggregated interest rate/stock model that actually works. Is that a pipe dream? I don't think so, and I'll get to that later.

I'd like to be able to extend and customize as many aspects of an economic scenario generator system as I can. Sometimes you, as a user, have better ideas than some or all of your vendors. I'm sure you've had that experience. You have probably had vendors who sometimes have priorities other than the ones you have in mind. Maybe they're not in a position to implement the features that you'd like to see implemented in the time frame in which you'd like to see them implemented. I've been on both sides of that fence.

My greatest pet peeve is don't force me to deal with a collection of disparate files. I like my output and my input organized for me. This is a particularly difficult nut to crack if you want to be able to mix and match from different vendors. There's some technology that's available now to enable vendors to work in a cooperative way so you can still have organized input and output.

If I do generate some scenarios out of, for example, Vladimir's system, it would be great if I could get into something like Excel and pull out those scenarios or some portion of them. I just want to deal with the stock index component of it, and do some little one-off analysis or experimentation that would be of great value to me. I don't want to have to try to mimic or reimplement the features that I'm using in my main model just because I'm starting to do some experimentation when I'm developing, perhaps, a new product.

Let me point out a few other software features that are desirable. As I have already mentioned, there is the ability to use the output—or even the internal code—of one system in another system. When I do use the internal code of another system, I want to be able to do that in a very language-independent way. Is there anybody here that works for a company where everyone uses the same computer programming language? I don't see a single hand raised. I'm going to go out on a limb. I predict that there will be peace in the Middle East, Northern and Southern Ireland will be joined together, and the Lion will lie down with the Lamb before the wars over computer languages finally cease. These battles are just too much fun to be stopped. They are just as vicious as conventional wars, but nobody gets physically hurt. There are many good languages out there, and they all have their place in the big scheme of things. The good news is that it is now possible to write software components in such a way that you can use them from any language.

Another feature that's available today that you'd like to have is distributed processing; especially if you're in a model without analytic solutions where you have to use Monte Carlo techniques. Even if you go to low discrepancy techniques, you'd like to be able to distribute the calculations across a network. If you have a multi-processor machine, distribute the calculations across multiple threads. You don't want to be waiting to project what the future's going to be and then find out because the future is realized before your results are done.

I mentioned easy extensibility. By that I mean, I'd even like to be able to take, say, an interest rate model and be able to wrap some code around it, drive it, and maybe change its behavior a little bit. Perhaps I'd like to change its drift term. Maybe I want to put in a time-dependent volatility term. That type of a design is possible. What I would like to have is a set of plug replaceable components. Suppose I intend to use a Cox, Ingersoll, and Ross interest rate process in my model. I have two different vendors, and, if I don't like the way one of their implementations runs, (maybe it's too slow or maybe it's not accurate) I'd like to be able to substitute the second for the first and still have my system work.

Another feature you might like to have is support for component versioning. I don't know if you have been bitten by this situation. Have you gotten a new version of software from a company and found some things don't work anymore? There has been a lot of work in this area that I'm going to talk about. There now exists a technology that does a better job of supporting versioning of different components.

Model input and output is a topic I alluded to a few moments ago. If you're going to have a system —especially a system that comes from multiple vendors—and you want to control where things are stored, then your components have to be able to be instructed as to when to store something to memory or disk and when to get something out of storage. Components have to be able to save their internal data and then reinitialize themselves so that you can ask them for their input or output data. The client with the software, the one that's driving this process, really doesn't need to know how a component stores its data. All you need to be able to do is ask it, in a standardized procedure, to store it in a particular area and get it back. These are often referred to as persistable components. That feature is also possible.

The other thing I like to look for in software is whether it is easily programmable from everyday tools. I tend to work in the Microsoft Office suite of tools. That means I would like to have software components that I can drive from a Visual Basic (VB) or Visual Basic for Applications (VBA) from within the Office suite, such as with tools like Excel or even Word. If I'm working on a Word document, I might like to be able to point to some components and do some quick analysis right from within Word, and have its results put in a table in my document.

There have been several solutions proposed for this problem. The first one was object-oriented programming (OOP). There is a computer scientist, Dijkstra, who wrote a famous paper called "There Are No Silver Bullets." That's true. There's no silver bullet that's going to cure all of our software ills, but some bullets are much better than other bullets. OOP is an appealing bullet that has missed the mark already. If any of you are just starting to get into objective-oriented programming, I would advise that it has its place, but you're going to ultimately be disappointed. I've gone to conferences on object-oriented programming, and the panelists whisper to each other, "Have you seen any code re-use, yet." And the response goes, "No. Have you?" "No." Then they go on and talk about the great benefits of code re-use from object-oriented programming.

All you really get, in some circumstances, is some source code reuse. It can be extendable. You can inherit some functionality from some parent or base classes. There is still the problem of "fragile base classes," where a new derived class requires changes to a common base class so that all of the executables containing the base class must be rebuilt and re-tested. Versioning of software is a real problem in object-oriented programming. I'm not going to spend a lot of time dwelling on this, but the bottom line is you may get a new dynamic link library (DLL) from a vendor, and your code that worked perfectly for a year might not work anymore. It's often hard for a client program to determine which version of a DLL is present on a system. This is such a problem that—especially with Microsoft Windows—many times you find, over the course of a year or two, there's only one solution to an unstable system, and that's to reformat its disk and start all over.

Object-oriented programming is a paradigm that's really useful only from within a single language. Even then, it often is only practical if you stick to development tools from one vendor. It's often a big surprise for people when they get into OOP to find out that they can create a C++ object library full of beautiful object-oriented classes. They give a compiled library to their friend who uses a different C++ compiler and he can't use it. There's a reason for that, and it stems from how the names of classes and variables are put into object files. The problem is called name mangling, and there is now a standard that defines how it should be done. The C++ language is standardized at a source code level, but not at the binary code level for any given platform. The same is true of other languages, with the possible exception of Java.

The bottom line is that a viable third party marketplace in object-oriented libraries never materialized as all the pundits hoped and truly expected. This was a big disappointment for many people, including me. In addition, a standardized approach to enable end-user programmability never developed. If somebody develops some OOP software components, how can you use them in the software you write? There's no standard for OOP components.

Microsoft came out with Visual Basic 3.0 (VB3) back in the early 1990s. They implemented a feature for their own internal use called Visual Basic Extension (VBE) controls. As usual, they really didn't document it much. Some hard core developers kind of rooted it out and determined how to duplicate it. Much to Microsoft's shock, there sprang to life a cottage industry that exploded with reusable components. VBX controls weren't a perfect solution, but everybody scrambled to develop and use them. They truly saved a lot of people a lot of time developing software. Even small niche languages like APL implemented support for VBX controls.

The VBX approach was a 16-bit solution, and it didn't work on a 32-bit Windows platforms. So Microsoft, when they were moving over to Windows 95 and Windows New Technology (WinNT), came up with an enhancement to the VBX "standard." They called their new 32-bit components Object Linking and Embedding (OLE) custom controls, or OCX controls. These were a big improvement over VBX controls, and they were better documented. The OCX specification was based upon an even more fundamental software specification called Component Object Model (COM). The important thing about COM is that, unlike object-oriented programming, COM is a standard specification for reusing binary code. It has nothing to do with source code. It's a binary reuse standard.

Meanwhile, Microsoft accidentally stumbled on the World Wide Web in 1996, although Gates claims to have laid down their strategy in mid-1994. The Web seems to have kind of snuck up on Microsoft, so its immediate reaction was to launch a marketing campaign. Microsoft took all of its COM technology and renamed it ActiveX which confused everybody. Regardless of the ActiveX labels, everything is still based upon COM. COM within Microsoft is almost a religion at this point. Everything it is developing, specifically all its new technology, is based on COM and reusable components under the surface.

COM defines a standard method for one piece of software to reuse functions and data in another piece of software. It's a binary standard. This makes it language independent. It has support for versioning, which is one of the desired features I mentioned. It supports end-user programmability. You can distribute these COM objects over a network, so you can get your distributed processing working. It also defines some standards, something called persistence interfaces, so that you can tell what component persisted your data or persists your output, or, after you've instantiated a component, you can essentially say, "Here's where you can go get your data; here's where you can place your output." You don't need to understand how data are stored by your components in order to get at it. You simply need standardized persistence interfaces, and COM provides them.

The following are development tools that support COM: assembly language, C and C++ compilers, Delphi, Microsoft Java, Microsoft Visual Basic, Fujitsu COBOL. We've worked in all of these, except for Fujitsu COBOL. I put that on the list just to show you that even the ancient languages are adopting COM. There actually could be some use for that if you have some legacy COBOL code. But experimentation is left as an exercise to the listener; I'm not going to touch that one. I have a list of end-user tools that support COM programming. A variety of C++ compilers, the upcoming version of APL 2000, and a lot of web browsers support the use of COM objects.

The main tools I focus on in my company are Microsoft Visual Java and Visual Basic, as well as Visual Base for applications. It doesn't seem like it by looking at them, but these two development products are built on COM from top to bottom. There is going to be a new version of COM coming out soon. The release of NT 5, COM+ will make it easier to develop and use these binary objects. This may not be of much interest to you as an end-user. The benefit of COM+ to you is that it's going to lower the bar in the cost for vendors to adopt what I'm going to advocate here. It will also make it easier for you to install, find, and use components on your machine and network.

I believe that it's time for us to look abstractly at the problem of actuarial modeling. For example, we should define an abstract solution to the problem of economic scenario generators. That specification should be freely published to allow companies to competitively provide implementations programmed to that standard functionality. This is being done in other areas. About three years ago, we started something called OLife™ (OLE for Life insurance). This was back when they referred to COM technology as OLE. It has since been taken over by a nonprofit industry group called ACORD. Right now, they have defined a fairly complete object hierarchy for software that deals with objects related to agency management, such as illustration systems and client contact systems. Some of this may be of interest to us, as actuaries, in the future.

Playing off of Microsoft's ActiveX marketing thrust, my company decided to take a stab at coming up with what we call the ActuaryX™ set of interfaces. ActuaryX™ is a COM specification for a set of programmable objects that actuaries might find useful. We're going to make the specification freely available as we develop it. Our implementations won't be free, of course, but the specification, which gives anyone a leg up on building compatible components, will be free.

The goal here is to allow users like you to cherry pick components from the best of the breed. A ScenarioFactory enables you to have a collection of EconomicModel components that you can use together to construct a set of random scenario objects. The ScenarioFactory has components that enable you to iterate through a scenarios collection that, in turn, might contain a scenarios collection in each of its elements for use in option pricing. Each scenario has a realization; at each node, there is a collection of references to output objects that are stored wherever you decided to have them placed by the different components that have generated the output.

One of the interesting things we discovered is that it is useful to develop purely algorithmic components, such as calibration routines and Monte Carlo yield curve calculators. This is completely contrary to traditional OOP theology, where algorithms are inextricably wedded to data as objects. For example, if you define your interest rate process and random number components correctly, you can create a stand-alone algorithm for determining the yield curve at any state of your interest rate process.

The bottom line to my presentation is this: I have worked with the problem of economic scenario generation long enough that I believe this problem is amenable to an abstract specification. Microsoft's component object model is a sufficient technology to implement it. If I can convince enough people to adopt this approach to developing actuarial software, then I think you are going to benefit by having a set of tools that you can use that are much more flexible and ultimately of much higher quality and reliability.

FROM THE FLOOR: I have a question for Dr. Zhang concerning the Brownian Bridge. I've read some papers by Professor Mark Yor. He mentioned the Brownian Bridge as a process that has known beginning and ending points. Basically, a Brownian motion just starts at Point A and ends at a specified Point B at a certain specified time. I saw some papers by Ball and Taurus in which they used that idea to model the motion of a default-free, zero-coupon bond because it would end at par. I didn't see the connection between the Brownian Bridge you were speaking of and the Brownian Bridge that I saw in the literature.

MR. ZHANG: Brownian Bridge is just a name. Suppose you go from the first point to the last point, so it's like playing bridge. You fill something in the middle. But that's not the only way to generate them, and there is generalization about this mathematically. You can prove this. This is very rigorous by Professor Caflisch from UCLA. What we have done is generalize this using a mathematical means. We still call it Brownian Bridge, but it's not really Brownian Bridge. You don't see this from beginning to end. But what we did is was from the beginning to any point I want it and we can generate in any order we want it. So I hope that this will answer your question.

MR. DAVLIN: I think there's a distinction between the two examples you gave. I think Ball and Taurus were doing a Brownian Bridge on the value of the bond, knowing that it matured at par. Whereas the Brownian Bridges that Hongfei has been talking about are the Brownian Bridges between positions and state space or your state variables of your interest rate process.

MR. ZHANG: Right. But they are only interested in the terminal price. So we are also interested in the process in the middle. I have also done this for pricing. That's the first time I've used it in cash-flow testing.

MR. DAVLIN: I have one question. It's for Vladimir and Hongfei, and it pertains to Brownian Bridges. The first article I ever read about this construct was written by Russell Caflisch at UCLA. Caflisch used a Brownian Bridge to generate random paths between one known state and a second known state. Vladimir, in your model, don't you actually develop the transition probabilities forward and backwards and construct a Brownian Bridge based on that distribution? That is a little different than doing a Brownian Bridge on the deviates and then just using those in your process to go from one point to the next

MR. LADYZHETS: As I mentioned at the end of my presentation, we are planning to implement a hybrid between the low discrepancy sequences and Monte Carlo simulations. The idea is to start with low discrepancy sequences to mark the routes for the future projected values of the state variables. After that the forward and backward transition probabilities are used to fill up the gaps. This is very similar to the approach suggested by Caflisch. In our opinion, such an approach will allow us to reduce the number of generated scenarios without inducing significant bias.

MR. DAVLIN: This one is for Vladimir again. If I understood your model correctly, I believe you have a Double Mean Reverting Process for interest rates, which looks very much like a double mean reverting version of the single factor Cox, Ingersoll, and Ross process. You adjust the long-term target mean of the instantaneous rate, so that it varies randomly by the volatility of the stock market; is that right?

MR. LADYZHETS: Yes.

MR. DAVLIN: If you take that term out, and then put it back in, what does that do to the shape of the yield curves that you're able to realize in that model? How does that term change the shape of the realizable yield curves?

MR. LADYZHETS: Let me explain briefly how a Treasury yield curve is calculated in our model. The procedure is based on a partial differential equation of the Black-Scholes type. In the original Black and Scholes equation, you have only price underlying security in addition to time. In the equation used in the five-factor model, you have three variables in addition to time. They are the short-term interest rate, the long-term interest rate and the volatility for S&P 500. The equation provides you with a price of unit cash-flow, which is the amount you need to pay today in order to get one dollar in a month. As soon as you have the price of unit cash-flow, you can calculate all the positions on the Treasury yield curve. Since the long-term interest rate and S&P 500 volatility contribute to the values of yields, the generator demonstrates such a variety of yield curve shapes that cannot be achieved by using any one-factor generator. I would like to note that the way we calculate Treasury yield curves is similar to one described in the monograph "Interest Rate Dynamics, Derivatives Pricing, and Risk Management" (1996) written Dr. Lin Chen.

MR. DAVLIN: In Mark's model, where you have a single mean reverting equation, you can either generate yield curves that are upward sloping, kind of on an exponential form, or downward sloping, depending on the initial state variables. When you add Mark's second equation, it allows you to add a hump creating a term to that so that you can get yield curves that are humped or inverted. When you add an additional term like the stock market volatility, does that just raise or lower the humps due to the second equation a little bit? Do you actually get some kinks or other interesting features in the curve?

MR. LADYZHETS: Since the Treasury yield curves projected by the five-factor generator depend on the volatility of S&P 500, an increase or decrease in the stock market volatility contributes to the vertical shifts and slope changes of the yield curves. It also brings some humps that, I believe, we would not observe, if there was no connection between the yields and stock market volatility. In our opinion, the debt market "feels" the equity market through its volatility and this is explicitly incorporated into the five-factor model.

MR. DAVLIN: So if the stock market became more volatile, then I'm more likely to have a humped curve than I would otherwise.

MR. LADYZHETS: Yes.

MR. DAVLIN: That's very interesting.

MR. LADYZHETS: The first statistic I check doing the scenarios valuation is the percentage of inverted yield curves. The inversion of yield curve is defined by the difference between ten-year yield and three-month yield. It is inverted if this difference is negative. Scenarios projected by the five-factor generator exhibit between 15% and 18% of the inverted yield curves.