INFORMATION TO USERS

The most advanced technology has been used to photograph and reproduce this manuscript from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

U·M·I

University Microfilms International
A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
313/761-4700 800/521-0600

Order Number 9108136

Market volatility and trading strategies

Krull, Steven Brian, Ph.D.

City University of New York, 1990

U·M·I

300 N. Zeeb Rd.
Ann Arbor, MI 48106

A

MARKET VOLATILITY AND TRADING STRATEGIES

by

STEVEN KRULL

A dissertation submitted to the Graduate Faculty in Business in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York.

1990

This manuscript has been read and accepted for the Graduate Faculty in Business in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

7/31/90
Date

Harry Markowitz
Chair of Examining Committee

8-1-90
Date

Ronald Vedenbaugh
Executive Officer

Ronald Anderson

Giora Harpaz

Ashok Vora

Supervisory Committee

The City University of New York

Abstract

MARKET VOLATILITY AND TRADING STRATEGIES

by

STEVEN KRULL

Advisor: Professor Harry Markowitz

Two models of securities markets are simulated. First is a model of the stock market with investors using rebalancing and option replicating portfolio insurance strategies. Second, the above model is extended to include a futures market and index arbitrageurs. Simulation results show that more portfolio insurers leads to greater market volatility, index arbitrage does not affect market volatility, portfolio insurance strategies do not appear to be mean-variance efficient, and high daily volatility does not appear to be related to high quarterly volatility. Implications are that rational investors would not use portfolio insurance so there is no need to restrict program trading, and high daily volatility is not necessarily bad since quarterly volatility is largely unaffected.

PREFACE

I would especially like to thank Harry Markowitz for his enthusiasm, intellectual guidance and generous assistance. Also, I would like to thank the other members of my committee, Ron Anderson, Giora Harpaz and Ashok Vora, as well as the participants at the Baruch Finance Seminar Series. Of course, any remaining errors are my own.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

BACKGROUND AND LITERATURE REVIEW

I. INTRODUCTION

I.A. RECENT HISTORY

Various researchers have been examining the events and market structure around October 19, 1987 in an attempt to understand the possible causes of 'the crash.'[1] The crash is but a small part of the big picture however. Market volatility is important for several reasons. It affects decisions regarding the necessity to hedge. If volatility is not constant, options pricing models and empirical research are also greatly complicated. Thus an understanding of the causes and effects of market volatility are necessary.

This paper will attempt to extend the market structure model used in the recent work by Kim and Markowitz [1988] (hereafter KM) as a reference point. The KM paper consists of a discrete event simulation using SIMSCRIPT II.5[2] of a simple market with two types of investors and two securities. The investors are portfolio rebalancers and portfolio insurers using constant proportion portfolio

[1] See for example Anderson and Tutuncu [1988], Edwards [1988a,b], Grossman [1988a,b], Grossman and Miller [1988], and West [1988]. This list is by no means meant to be exhaustive.

[2] SIMSCRIPT II.5 is the trademark of CACI, International. It is described in Kiviat, Villanueva and Markowitz [1983]. A listing of the KM model is contained in Markowitz [1988].

insurance (CPPI for short).[3] The securities are cash and stock (or market portfolio). By varying the proportion of the two types of investors, KM show that under certain conditions it is possible for market volatility to become quite large. This result obtains with a large proportion of portfolio insurers. In the presence of margin, the volatility becomes explosive.

I.B. GOALS

The goals of this paper are several. The KM framework will be expanded by including more than one type of stock, and by the addition of index futures and options. The use of derivative securities is justified by their lower transaction costs. Edwards [1988b] estimates that transaction costs of trading on the NYSE are approximately seven to ten times larger than trading on the CME with S&P 500 futures. Also considered will be the presence of 'information traders' or 'noise traders.' These traders receive private signals regarding the desirability of the stocks (although the signals are not presumed to be valid). They invest more in stocks with positive signals and less in stocks with negative/less positive signals. This model is simulated and the effect of the presence of futures is analyzed.

The portfolio insurers are also modeled using the

[3] See KM for a complete description of the two.

option replication strategy of Rubinstein and Leland [1981] and Rubinstein [1985] in addition to the CPPI method utilized by KM. This is done since, as is noted by Perold [1986] and Bookstaber and Langsam [1988], CPPI has no analytical foundation for its use and parameter choices are arbitrary.

In addition to using the pure simulation model, historical market data is also used as input to the model. In other words, price here is exogenously determined. The actions of the various investors is again analyzed.[4] Results of the application of trading strategies on market data is compared with the pure model simulation by comparing return and risk of the investors.

In addition, the alternative strategies are compared for effectiveness based on realized risks and returns (for example, Sharpe's ratio, Treynor's ratio and Jensen's alpha are used). This may expose certain strategies as flawed (at least in the context of the model). This of course suggests that all investors may not be totally rational which is consistent with West [1988] and others' suggestions that "fads and fashions" may be important. We will not attempt to pass judgement on this issue here since others including Kleidon [1988] disagree. Another explanation is offered by

[4] This is difficult since the model allows prices to be affected by investors while the empirical tests will assume that investors accept prices as given. However, if the investors in the model are representative, then their actions based on real data may parallel the actual investors, resulting in the subsequent price actions.

Black [1986]. Investors may believe that they are rational and are acting on valuable information, but the presence of *noise* makes it impossible to distinguish between valuable and useless information, and thus between "noise" and "information" traders.

The organization of the paper is as follows: Section II consists of a review of the volatility literature with a discussion of models related and unrelated to this paper. Then there is a brief review of the major characteristics of the KM model in section III. Section IV describes the methodology including descriptions of securities and investors to be modeled. Finally, section V consists of the summary and comments about future research directions.

II. REVIEW OF THE LITERATURE

There is substantial evidence that there is excess volatility in the stock market. Summers [1986] and Poterba and Summers [1987] determine that market fluctuations are much larger than rational valuation methods would suggest. They also comment that the fundamental causes of these fluctuations remain undetermined. Grossman and Shiller [1981] and Shiller [1981] find that stock price volatility is well in excess of what could be explained by dividend information or real interest rates.

Various theories of market volatility have been developed. West [1988] states that these theories fall into three categories: Rational bubbles, small sample bias and

"fads and fashions." Small sample bias refers to statistical estimation problems which need not be related to actual volatility. For a discussion and literature review of this topic see West [1988]. The other two categories will be discussed below.

II.A. MODELS SIMILAR TO KM

Closely related to this paper is that of Anderson and Tutuncu [1988]. They obtain closed-form solutions to a set of differential equations describing the market's operating mechanism. Their market consists of "investors," "program traders" and "portfolio insurers," and two securities, a stock index and stock index futures. The stock index value is determined exogenously, is constant, and the implied risk free rate is zero (although this is not explicitly mentioned).

Investors basically buy and hold, program traders arbitrage differences in the futures and spot prices where larger price differences result in linearly larger demands, and portfolio insurers attempt to set a limit on losses by selling futures in a falling market and buying in a rising market (with the amounts assumed to follow the Rubinstein-Leland [1981] option replication methodology). This model shows that in the absence or with small percentages of portfolio insurers, the market will converge to a stable equilibrium. When the percentage of portfolio insurers becomes large in the presence of program traders, the market

no longer converges to equilibrium although it does not become explosive. However, this is unlike KM where margin is considered and unbounded prices may result.

II.B. OTHER MODELS OF MARKET VOLATILITY

Edwards [1988b] reviews some of the theories advanced to explain recent increases in market volatility. The theories include increases in speculative activity caused by the presence of professionally managed speculative trading programs. Creation of futures markets, which some argue has destabilized cash markets, and "fashions and fads" which indicate capricious actions on the part of investors. The tests of this paper, KM, and Anderson and Tutuncu [1988] are mainly concerned with portfolio insurance which falls in the first category above. Another possibility is the "rational bubble" theory. Other theories relate margin requirements to volatility while some believe volatility depends greatly on market microstructure.

II.B.1. MARGIN REQUIREMENTS

Hardevoulis [1988b] finds that there is a negative relationship between cash market volatility and initial margin requirements set by the Federal Reserve Board. However, he points out that these results should be interpreted carefully since the number of observations is severely limited.

II.B.2. RATIONAL BUBBLES

The topic of "rational bubbles" has recently attracted much research. West [1988] summarizes many of the studies and finds that "rational bubbles" can not explain the excess levels of volatility in the market. Subsequent to this however, Hardouvelis [1988a] finds that using more powerful tests allows one to conclude that excess market volatility in Japan, the United States and Great Britain may be caused by "rational bubbles." It remains to be seen whether this will be supported by future studies.

One may of course wonder how a bubble can be rational. This is explained by a process where in a rational expectations equilibrium, the expected return is the risk free rate. Thus, a market which grows at a rate exceeding the risk free rate if the bubble continues and falls to a low level if the bubble bursts may be consistent. If a bubble exists, it is necessary for it to grow at an ever increasing rate since the potential decline keeps increasing. Thus, bubble tests look for increasing volatility over a time period of mostly positive returns. See West [1988] and Hardevoulis [1988a] for a description of "rational bubbles" and references.

II.B.3. FADS AND FASHIONS

"Fads and Fashions" theories suggest that there are some investors who do not always act rationally. West [1988] calls these investors "noise traders." These traders

are naive and trade without valuable information. Their existence implies more volatility and thus other traders, particularly market makers, will require higher returns to participate in the market. This means that there will be fewer market makers to provide liquidity or the supply and demand of immediacy. Grossman and Miller [1988] model the entry and exit of market makers. With fewer market makers, the market will have greater volatility.

It could be argued that various traders are naive or "noise traders." These include any traders following mechanical rules used by technical traders such as support and resistance levels, trend followers or cycle followers. Most studies done have shown these methods to be useless. In futures markets, public commodities trading partnerships have been found by Elton, Gruber and Rentzler [1987, 1989] to serve no apparent need of investors. Despite this, their size keeps growing suggesting that irrational investors do exist. Most studies of mutual funds investing in stocks show that load funds don't outperform no-load funds *before* sales expenses are included. This lends additional support to the existence of irrational investors. Perhaps there is some as yet unexplained utility to investors to perform poorly.

Finally, it may be argued that strategies such as portfolio insurance are naive, although no doubt Leland, O'Brien and Rubinstein might object. Leland [1980] argues that two types of investors should use portfolio insurance:

"safety first" and above average (positive alpha) investment managers. Replication through dynamic trading is necessary (Rubinstein [1985]) since the necessary options may not be available. There are also position limits which may affect one's ability to insure with options. Even transaction costs can be adjusted for (Leland [1985]).

But as Grossman [1988a,b] clearly articulates, the substitution of trading strategies (dynamic trading) for securities is fraught with danger. Where options allow one to lock in a volatility, a dynamic trading strategy does not. As Anderson and Tutuncu [1988] point out, mechanically following the dynamic trading strategy of portfolio insurance might succeed in locking in the market bottom when selling out. This may in fact explain why there has been a significant reduction of investment managers using portfolio insurance since the October '87 crash. Also, Benninga and Blume [1985] argue that under reasonable assumptions, portfolio insurance is precluded except when there are incomplete markets and no risk free asset.

It is these features of portfolio insurance that lead it to be considered as a cause of the crash. However, as Edwards [1988b] points out, portfolio insurance was accompanied by large mutual fund sales caused by huge fund-holder redemptions. This implies that there was either some major economic change or extreme irrationality. With the benefit of 20-20 hindsight, little has happened to the economy, leaving irrationality as the apparent major cause

of the crash. Perhaps the fund redemptions were caused by investors pursuing portfolio insurance on their own. The preceding discussion justifies the modeling of portfolio insurance as factor in market volatility.

III. A BRIEF SUMMARY OF THE KIM-MARKOWITZ MODEL

The KM model consists of two investor types: rebalancers and portfolio insurers, two securities: stock and cash, margin in some cases, and a market trading mechanism. The securities include non-interest paying cash and one non-dividend paying stock. Investors review their portfolios randomly with a mean of five days and exponential distribution.

The rebalancers are assumed to attempt to maintain a portfolio of half cash and half stock. This is accomplished by selling stock when stock comprises over 55% of their portfolio or by buying stock when it comprises under 46% at a review time. The size of the order placed is the amount needed to attain the 50-50 level. Portfolio insurers use Constant Proportion Portfolio Insurance with 65 day plans with different insurers having different plan expiration dates. A Floor is chosen at the beginning of a plan. The Floor is the minimum value the insurers wish to maintain. Essentially, as stock prices rise, more stock can be purchased while maintaining the floor, and as prices fall, stock must be sold to protect the floor. For details of KM's parameters, see KM [1988, p. 5-6]. For details about

CPPI, see Perold and Sharpe [1988] and Black and Jones [1987].

In addition to the above, all investors begin with a $100,000 portfolio and receive cash randomly with exponentially distributed mean times of ten days and amounts uniformly distributed between -8,000 and +9,000. This supplies a slight upward bias to prices by including more inflows than outflows on the average.

Finally, trading procedures must be explained. When an investor receives a signal to buy or sell, an estimate of the current market price is obtained. This estimate is the average of the (highest) bid and (lowest) asked price if both exist. If only bids exist, the price is 1.01 times the highest bid; if only offers exist, the price is 0.99 times the lowest offer. If neither bid nor offer exist, the price is the last sale price (set to 100 at the start of the simulation). Orders unfilled after 1 day are reconsidered. The buy(sell) quantity is recalculated using a new bid(offer) which is 1.01(0.99) times the last bid(offer). Orders not executed are placed in the limit order book until filled or cancelled.

In their simulations, KM begin the trading with half the rebalancers above equilibrium and half below in order to kick off trading. This procedure is not in fact necessary for a reasonable amount of trading to occur. It has been found that the KM model may be started in equilibrium, and the randomness of cash flows and portfolio reviews is

sufficient to cause trading activity. For a detailed example of why and how orders are placed and how the resulting trades are executed, see the appendix.

IV. METHODOLOGY

Initially, the KM model is employed without modification. First, some of their simulations are duplicated with one change as discussed above: all rebalancers begin with the same portfolio. Thus trading begins in equilibrium as opposed to out of equilibrium in KM. Another modification will be to track portfolio values, returns and risks of the trading strategies to observe strategy effectiveness.

IV.A. SIMULATIONS

The 'market portfolio' will be split into two securities and a third trading strategy will be added. The market rebalancers and portfolio insurers will continue to buy and sell the market portfolio i.e. they will hold the two securities in the same proportion as in the market portfolio. The third type of traders will receive signals about the two stocks. The arrival of signals will be distributed exponentially with mean time of 10 days, and the size of the signals will be drawn from a uniform distribution with a range of 0 to +0.5. When receiving a signal, the 'information investors' will invest a fraction of their wealth (which may be altered by the data input)

dependent on the signal size in that stock. Although not modelled here, the availability of margin would increase the amounts invested proportionately. Any new signal 'cancels' the old signal and sets the new level of investment.

The existence of two or more securities makes things interesting. For different reasons, two types of investors may both be bidding for the same security at the same time, driving up the security's price. Thus it may be advantageous for investors who wish to hold only the market portfolio to trade index futures or options. For this reason, index futures are now added to the model. To maintain the futures at a proper price relative to the underlying securities, a fourth investor type is added: the arbitrageur (called arbitrager in the programming).

Initially, the index futures contract added is perpetual, i.e. they have no expiration date. Thus, under the assumption of zero dividends and zero interest rate, the futures price should be equal to the spot price at all times. This contract is in fact similar to one recently proposed by the NYSE. Since daily settlement is required in futures markets, it is be required here causing daily cash inflows and outflows.

Margin is not be explicitly considered, but the 'floor' held by insurers and the cash held by rebalancers would normally be sufficient. The portfolio insurers and rebalancers will now only trade the futures contract. This can be justified by the lower transactions costs of the

futures markets. The arbitrageurs will compute the difference between the market index and the futures price and take opposite stock/futures positions when a minimum differential is observed. Initially this differential will be set to zero. Arbitrageurs will have a linear demand function as in Anderson and Tutuncu [1988] so that as the gap increases, their demand increases.

IV.B. REAL MARKET DATA AS THE INPUT

The above exercise basically assumes that there is no external value to the market. Investors simply buy and sell in order to follow their trading strategy or to adjust their portfolio due to cash inflows or outflows. If we assume that real market prices include some information about value, then there should be some knowledge gained by incorporating this into the model. Merrick [1988] does something similar to this. He compares portfolio insurance with "cost of carry" futures prices and actual futures prices and finds significant differences. The "cost of carry" futures prices are theoretically determined by the spot index value, interest rates and dividends.

The basic premise in this model is that historical prices were determined by a combination of traders following their chosen strategies. Thus by applying presumed actual strategies to historical prices, we may determine how traders were acting in reality. This will be accomplished by the use of the daily S&P 500 index derived from the CRSP

tapes, in the trading strategies of the model's investors. Ex-post (use today's price to determine trades and trade at today's price) tests are used.

IV.C. OPTION REPLICATION FOR PORTFOLIO INSURANCE

Rather than holding stock and creating a put through dynamic trading, portfolio insurers in this model will hold cash and replicate a call. The amount of futures (stock) to be purchased will be the hedge ratio, N(D1), determined from the Black-Scholes [1973] call option pricing model times the portfolio value. Thus at high market levels N(D1), will approach one (fully invested) and at low levels it will approach zero (all cash). The strike price will be initially set equal to the portfolio value at the beginning of each insurance plan (although this amount could be altered). Since quarters are assumed to have 65 days, a year will be defined as 260 days. Volatility may be derived from "real world" experience or from actual model history. D1 is defined as:

$$\{\ln(P/E)+(r-d+0.5*s^2)*t\}/(s*t^{0.5}).$$

Where P is the index price, E is the exercise price, r is the risk free rate, d is the continuously compounded dividend rate of dividends to be paid prior to maturity, s is the standard deviation (volatility), and t is the time to maturity of the plan. The initial hedge at the start of an insurance plan under the assumption of a zero risk free rate and no dividends will start out with $D1=0.5*s*t^{0.5}$. Of

course, this will change over the life of the plan. $N(\cdot)$ is the cumulative normal distribution function.

V. COMMENTS AND SUMMARY

One problem which is not addressed is caused by changing strategies. October 19 exposed some trading strategies to be apparently less than perfect. In the real world one would expect such investors to revise their trading strategies (for example, portfolio insurance was reduced substantially subsequent to October 19). We assume that strategies are held constant.

Unfortunately, the model can not be tested using actual proportions of investor types in the real world since this information is not readily available. Although the dollar value managed by portfolio insurers has been estimated on occasion, it has not been done regularly. Thus, various proportions of investor types are tested leaving actual proportions for future research if and when they can be determined.

CHAPTER 2

THE KIM-MARKOWITZ MODEL WITH OPTION REPLICATING PORTFOLIO INSURERS

I. INTRODUCTION

I.A. BACKGROUND

This section reviews some of the results of Kim and Markowitz (1989, hereafter KM) with the substitution of Option Replicating Portfolio Insurers (ORPIs) for the Constant Proportion Portfolio Insurers (CPPIs). This is done since ORPIs are probably more common in the marketplace. ORPIs follow the strategy of Leland and Rubenstein (1981) which is to attempt to replicate a call option (which is equivalent to holding a portfolio of stocks and a put). Essentially, this strategy creates a floor similar to that of the CPPI, with the potential for unlimited gains. Both strategies reduce their market exposure as prices fall and increase their market exposure as prices rise. Thus, both strategies claim to be able to limit losses while not limiting gains.

Unfortunately, as Grossman (1988a,b) points out, replicating an option is not equivalent to actually buying an option. If enough investors attempt to replicate the option, when the time comes to sell, the price series becomes discontinuous, thus violating the underlying assumptions of Leland and Rubinstein (1981). The result may be paying more for the replicated option than the actual

option as the insurance turns out to be more expensive than anticipated. This fact can be illustrated by using the discrete time simulation of this paper and KM. After a brief comparison of the market effects of the CPPIs and the ORPIs, the performance of the two will be compared to the Rebalancers.

I.B. METHODOLOGY

For consistency, the ORPIs are created so that they are very similar to the CPPIs. They attempt to insure their portfolio for one quarter (65 trading days). They begin with their portfolio half in stocks and half in cash. They receive cash flows that are received at an exponentially distributed rate with a mean of ten days and a uniformly distributed amount of -8000 to +9000. They also review their portfolio on the average every five days, again exponentially distributed. Since they are replicating an option, an annual standard deviation of 20% is assumed which is reasonably close to the real world. This could be internalized to depend on the historical price series, but ex-ante and ex-post volatility need not be the same (resulting possibly in even more error than a naive long run market average volatility).

Additionally, if an order is placed but not executed, the investor revises the order in one day. Although there are no transaction costs in this model, it is assumed that an order is placed only if the portfolio deviates from the

target investment ratio by more than one percent (varying this appears to have little overall effect on the individual investors or the market). Trades by an ORPI are also temporarily suspended in the event of large market moves (five percent or more). This is done since in early simulations, ORPIs lost unreasonably large amounts of money without this cap, and this strategy is claimed to have been followed on October 19, 1987 by some portfolio insurers.

II. MARKET COMPARISON OF CPPIs VERSUS ORPIs

II.A. VOLATILITIES

Comparing the market volatilities in Table 1 with those presented in KM (Table 2, p.46)[5], small numbers of insurers cause the market to be more volatile if they are ORPIs, while with large numbers of insurers, the ORPI case is less volatile. In both cases however, there is a monotonic increase in volatility as more portfolio insurers are added to the market.

Trading volume in Table 2 is compared to KM (Table 5, p.47). As with the CPPIs, as more ORPIs are added to the market, trading volume increases. However, in both cases volume decreases over time. This is the result of the

[5] Note that KM start their simulation in a state of imbalance resulting in initially high volatility and trading volume. This simulation starts off in approximate equilibrium. Thus it is necessary to compare the latter stages of the simulations for uniformity. KM was simulated with an initial state of equilibrium and the results in the second halves of the simulations were quantitatively similar.

average portfolio insurer losing market share to the average rebalancer over time. Thus since the insurers are the more active traders and their relative influence falls over time, trading volume decreases over time. This also explains the decrease in volatility over time; since the insurers appear to be the cause of the market volatility, their loss of market share results in a less volatile market. This result is discussed further in Chapter 4.

Below are comparisons of betas, alphas, Sharpe measures, Treynor measures and standard deviations for individual portfolios and investor types. All computations consist of 100 quarterly returns observations. Returns are computed as:

$$R_{i,t}=(V_{i,t}-V_{i,t-1}-D_{i,t})/(V_{i,t-1}+D_{i,t}/2) \qquad (1)$$

where $R_{i,t}$ is the return in period t on investor i, $V_{i,t}$ is the portfolio value of investor i at time t, $D_{i,t}$ is the net deposit for investor i in time period t. Thus an investor's return is equal to the non-deposit increase in portfolio value divided by the initial portfolio value plus one half the net deposit (which assumes that the deposits occur on the average in the middle of the period).

II.B. BETAS

Tables 3A and 3B compare the betas of ORPIs and CPPIs to one, respectively, as a measure of systematic risk. Also

the Rebalancers are examined in both cases. For small numbers of ORPIs (two and three with the one ORPI beta of 1.22), their betas are significantly greater than one. However, as their numbers increase, their average beta is statistically insignificantly different from one. Most of the simulations have average Rebalancer betas of less than one, although they are nearly equal to one.

The simulations with CPPIs, perhaps surprisingly, show that their betas are significantly less than one for most cases with less than 25 CPPIs, but are greater (but insignificant) from one for large numbers of CPPIs. Rebalancers in these simulations often have betas significantly greater than one except for the cases of 2/3 and 5/6 CPPIs where rebalancer betas are significantly less than one. Overall, it is not clear that there is a relationship between betas and relative proportions of insurers and rebalancers.

II.C. ALPHAS

Rather than compare raw investor returns, it is preferable to compare risk adjusted returns. This will be done using alphas, Sharpe measures and Treynor measures. Alphas (sometimes called Jensen's Alpha) are the intercepts of regressions of market returns on portfolios or individual stocks returns. If alpha is statistically different from zero, it implies that a portfolio has outperformed(underperformed) the market if it is

greater(less) than zero. Tables 4A and 4B examine this for ORPIs and CPPIs respectively.

With the exception of a market consisting almost completely of ORPIs, ORPIs always significantly underperform the market. Although this could be the result of the simulation methodology, these investors were modelled using the standard set-up with some fine tuning to prevent them from doing even worse. Not only were the average alphas significantly less than zero, but the majority of the individual ORPIs had negative alphas for all cases except for 145 out of 150 investors. The converse in these simulations was also true; the Rebalancers' average alpha was generally significantly greater than zero with the majority of the Rebalancers having positive alphas for all cases of ORPIs equal to or exceeding five investors out of 150 and significantly positive alphas for 25 or more ORPIs. Thus it would appear that the ORPI strategy was dominated by the Rebalancer strategy.

CPPIs did approximately as poorly as the ORPIs with large numbers of individual investors with significantly negative alphas for cases of 25 or more CPPIs. The average alphas are generally not different from zero, although virtually always negative, apparently as the result of the high volatility of their strategy. As with the ORPIs, Rebalancers have significantly positive average alphas for all cases of 25 or more CPPIs with the virtually all individual alphas positive in these simulations. Thus it

appears that the Rebalancer strategy also dominates the CPPI strategy.

II.D. SHARPE MEASURES

Since we are dealing with 'diversified' portfolios, a comparison can be made using the return above the risk-free rate divided by the standard deviation, the Sharpe measure. Since there is no interest or dividends in this model, the risk-free rate is zero. Tables 5A and 5B examine the Sharpe measures.

In both ORPI and CPPI simulations, as the number of insurers increases beyond five or ten, the market's Sharpe measure declines monotonically. This suggests an overall utility loss in a mean-variance world. However, in both cases, as the number of insurers is increased, the average quarterly market return increases (see the first line of tables 6A and 6B which are just the arithmetic average quarterly market returns). This is only a phantom gain, since over time one must use a geometric average. The market values at the end of the simulations are typically very close.

In the ORPI simulations, ORPIs average significantly below the market Sharpe measure as well as the individual ORPIs having Sharpe measures predominantly below the market. On the other hand, the Rebalancers for cases with five or more ORPIs on the average significantly outperform the

market with a majority of the individuals outperforming the market in all cases.

The CPPIs appear to do as bad if not worse than the ORPIs, again with most cases having significantly worse than market Sharpe measures with nearly all individuals underperforming the market. Except for cases with ten or less CPPIs, the Rebalancers outperform the market on the average and for most individual investors. So for both CPPIs and ORPIs, the insurers appear to be dominated by the Rebalancers using the Sharpe measure.

II.E. TREYNOR MEASURES

Since the standard CAPM suggests that beta is the relevant risk measure, the Treynor measure is also used. The Treynor measure is equal to the return minus the risk free rate, all divided by the portfolio beta. For the market portfolio with a risk-free rate of zero as in these simulations, this is equivalent to the market rate of return. The results here, shown in Tables 6A and 6B, only confirm those of the Sharpe measure and alpha above.

As mentioned above, the market Treynor measure generally increases as the number of insurers increases, but this is the result of increasing market volatility and not higher geometric returns over time. Thus the market does not necessarily benefit from increasing numbers of insurers.

ORPIs again average significantly worse than the market with most of the individuals underperforming except when

there are 145 ORPIs. In nearly all cases the average Rebalancer outperforms the market with a majority of individuals outperforming in every simulation.

CPPIs also generally do worse than the market on the average, and do worse for most individuals. The average Rebalancer outperforms significantly for cases with 25 or more CPPIs and the majority of individual Rebalancers outperforms for most cases. Again, the rebalancers appear to dominate the insurers of both types.

II.F. STANDARD DEVIATIONS

Although no specific statistical tests were run comparing standard deviations, Tables 7A and 7B show how simulations and investors compared. As mentioned above for the case of daily standard deviations, quarterly standard deviations increase almost monotonically with increasing numbers of insurers of either type. Also, for ORPI simulations, average Rebalancer standard deviations nearly always were less than the ORPIs'.

For the CPPI simulations, only cases of 50 to 125 CPPIs resulted in lower average Rebalancer standard deviations, with most of the others cases appearing about the same.

However, a very interesting result is obtained by comparing the last quarter daily standard deviation of returns in Table 1 with the quarterly standard deviations of Table 7A. Typically the quarterly standard deviation is about one and one half times the daily standard deviation

(even lower if we compare the average quarter's daily standard deviation with the quarterly standard deviation). Statistical theory suggests that the 65 day quarterly standard deviation should be approximately the square root of 65 (about 8) times the daily standard deviation. This is clearly not the case here. The low ratio here suggests that most of the daily volatility is transitory. In other words, on a day to day basis volatility may be high, but over longer periods it is not much higher.

In fact, since the market values were nearly the same at the end of all the simulations, the 100 period volatilities were nearly the same. Thus, whatever short term volatility portfolio insurer strategies appear to add to the market, insurers don't appear to have a lasting effect (except for the case of CPPIs using margin perhaps). Thus the standard practitioner argument, that the stock market volatility (probability of low returns) is low for long periods of time, appears to be warranted.

II.G. OTHER STATISTICAL TESTS

Two other tests are run to compare betas, alphas, Sharpe measures and Treynor measures of the insurers and the rebalancers. First means difference tests are run for both types of insurers versus the rebalancers. Then Mann-Whitney non-parametric rank tests are done comparing ORPIs with Rebalancers.

Tables 8A and 8B show the results of the means difference tests. As earlier results show, there is no conclusive differences in betas. For alphas, Sharpe measures and Treynor measures however, Rebalancers outperform both CPPIs and ORPIs for most of the simulations and generally statistically significantly. This supports the above results of rebalancers dominating insurers.

Table 9 reports the Mann-Whitney test results. These tests are run to see whether the normality assumptions seems reasonable. The results of these tests mirror the above results, perhaps even more strongly favoring the rebalancers over the insurers.

III. SUMMARY AND CONCLUSIONS

From the above results, it appears that from an individual investor viewpoint, rebalancer strategies appear to dominate insurer strategies whether CPPI or ORPI. Rebalancers tend to get higher returns with lower risks on the average.

Additionally, although more short-term volatility may result from the presence of more portfolio insurers, it is not clear that insurers cause any lasting effects. In fact, their presence clearly appears to benefit the rebalancers. Thus in an efficient market, one would expect portfolio insurance to lose favor as its performance became known. Perhaps this explains the large drop in its use after October 19, 1987.

CHAPTER 3

SIMULATION USING REAL MARKET DATA

I. INTRODUCTION

I.A. MOTIVATION

The previous chapter has shown that in general, investors using a rebalancing strategy outperform investors using portfolio insurance strategies when trading head to head. However, this result may be considered by some to be suspect since the trading strategies were not applied to real world data. That is the topic of this chapter.

Although the results in the previous chapter appear to be robust for a range of investor parameters, since many investors 'test' potential trading strategies on market data, it would be interesting to confirm the results in the previous chapter using historical data. It should be noted that testing any strategy on historical data does not mean that it will work in the future. However, since the strategies employed in this paper's model were not optimized using historical data, any performance results would appear to be valid.

I.B. METHODOLOGY

The daily S&P 500 index returns for the period of January 1962 to December 1988 were obtained from the Center for Research in Securities Prices (CRSP) database. These returns were then converted in to a price series with the

first price standardized at 100. Simulations were run with each trading strategy taking the trade prices as given. It was assumed that all trading decisions and subsequent trades were made at the same price. Of course, implicit in these simulations is the fact that individual investors have no effect on trading prices.

II. RESULTS

Results of the simulations are contained in Table 10. Although not as powerful as in the previous chapter, the same general conclusions can be drawn. As before, rebalancing investors have generally positive alphas while portfolio insurers have negative alphas (with only the CPPIs being statistically significant).

Surprisingly, all betas are very close to one although CPPIs are significantly less and ORPIs are marginally significantly greater than one.

All three types of investors have Sharpe measures significantly worse than the market with Rebalancers having the highest ranking. This result suggests that a buy and hold strategy may have been superior to all three strategies tested here.

Treynor measures yield yet another ranking with the ORPI investors apparently doing best (even better than the market). The Rebalancers and CPPIs both underperformed the market based on this measure, but only the CPPIs' difference was significant.

Finally, an examination of the standard deviations indicates that the Rebalancers had slightly more volatility than the market, CPPIs had significantly less volatility than the market, and ORPIs had significantly more volatility than the market.

Figure 1 shows the S&P 500 index over time and the trading volumes of each of the three investor types. Despite general upward trend in the index, the trend following strategy of portfolio insurers does not appear to give them any edge over the rebalancers.

An alternative way of comparing the three trading strategies is shown in Figures 2 and 3. Figure 2 plots the end of quarter portfolio values for 1962-1988 for the average of the 150 investors of each type of trading strategy. As can be seen, the three strategies track very closely with the insurers performing slightly better over the entire period. Figure 3 plots the quarterly returns which also track closely.

Table 11 offers another angle on the quarterly returns and standard deviations of the three trading strategies. Over the entire period, the CPPIs have the lowest average return and standard deviation, the Rebalancers fall in the middle, and the ORPIs have the highest average return and standard deviation. Thus higher risk receives higher return. This appears to contradict other results suggesting that the rebalancing strategy is the best.

However, when the sample period is split into two parts (1962-1979 and 1980-1988), different results are obtained.[6] In the first period, Rebalancers have the highest return and standard deviation, followed by the CPPIs and the ORPIs. In the 1980s however, the Rebalancers have the middle return and the lowest standard deviation, the CPPIs have the lowest return and middle standard deviation, and the ORPIs have the highest return and by far, the highest standard deviation. This weakly suggests that the use of portfolio insurance strategies may in fact have affected market prices to the traders' own detriment.

III. CONCLUSIONS

This chapter has demonstrated that the results in the previous chapter do not appear to be anomalous. The rebalancing strategy appears to have an edge over portfolio insurance strategies. However, it does not appear that any strategy clearly outperforms a buy and hold strategy. This is especially interesting in light of the fact that there are no transaction costs in this model. Such costs would only make the buy and hold strategy appear even better by comparison.

[6] This split is chosen since 1980 approximately marks the date when portfolio insurance began to enter the academic literature.

CHAPTER 4

FUTURES MARKETS AND INDEX ARBITRAGE: EFFECTS ON MARKET VOLATILITY

I. INTRODUCTION

I.A. INCLUSION OF FUTURES AND INDEX ARBITRAGE

The purpose of this chapter is to extend the results of the earlier chapters to include the effects of futures markets and index arbitrage on the volatility of the stock market. As discussed in Kim and Markowitz (1989), adding a futures market to the current model would be redundant. Thus, it is necessary to make some modifications.

I.B. MODIFICATIONS

At this point, the market is modified to contain two types of stock and an index futures market. Both stocks have the same number of shares and initially have the same value. The index futures contract is based on the market portfolio i.e. the value weighted index of the two stocks. For purposes of the simulation, the futures contract can be thought of as settling each day (much like the 'market baskets' created by some of the stock exchanges). Thus, the futures price should always be equal to the market index to avoid arbitrage opportunities. Another simplification is that margin is nominally at 100 percent. Since the model pays no interest on cash, this effectively does not influence the results.

Since rebalancers and portfolio insurers would hold all available stocks in proportion to their weights in the market portfolio, it is necessary to introduce another investor, the 'speculator.' Speculators hold the two stocks in proportion to signals they receive regarding the stocks values. Each speculator will begin with the same proportions of their portfolio in stock and cash as the rebalancers and insurers. If a speculator receives a very positive signal, he may attempt to hold half his portfolio in that stock. A very negative signal will cause the speculator to attempt to sell all of that stock. Potentially a speculator can be 100 percent invested if favorable signals are received for both stocks or 100 percent in cash at the other extreme. These speculators are similar to Black's (1988, 1990) unintentional noise traders.

Although there are no transaction costs in this model, it is assumed that rebalancers and portfolio insurers will adjust their respective market exposures only in the futures market. The futures market still has the advantage of necessitating one order instead of two, simplifying the trading process.

To ensure that the futures market and stock market prices are linked, the 'arbitrager' is added to the model. Like the other investors, the arbitrager begins with the same exposure to the market. This exposure will be split between stock and futures so that the arbitrager begins with positive endowments of each one. If the futures price

exceeds the index value by more than one percent, the arbitrager will place orders to buy stock and sell futures. The converse holds if the index value exceeds the futures price by one percent. The arbitrager's portfolio will thus vary according to the relative prices of the futures market and the stock market.

I.C. IMPORTANT CAVEATS

In order to facilitate the model's construction, short sales are not allowed. Thus, the speculators and the arbitragers can not sell stock (or futures) if they don't already own some. Because of this, the model does not purport to answer any short sales restrictions questions, although this is certainly a topic for further research.

Perhaps more important, it must be remembered that this model only allows limit orders. This could result in an arbitrager not having a 'risk-free' position due to the execution of only one or two of the three legs of any attempted arbitrage transaction in this model. In the real world, one would expect greater liquidity and thus almost certain order execution with minimal price slippage. The major result of this limitation is that one would not necessarily expect arbitragers to outperform other investors in this model despite their theoretically superior strategy. Fortunately, arbitrager performance is not a major objective of this research.

I.D. MODEL SPECIFICATIONS AND METHODOLOGY

Speculators and arbitragers begin with the same wealth as the other investors and receive cash inflows and outflows in the same manner. Speculators and arbitragers also review their portfolios in the same manner as the rebalancers and insurers.

When speculators do a portfolio review, they draw a signal for each stock from a uniform distribution with a minimum of 0.0 and a maximum of 0.5. The number drawn determines the fraction of total wealth that the speculator wishes to hold for a particular stock. Although this may result in unrealistic reversals in holdings, the purpose of the speculators in this model is to perturb the system. In particular, their 'job' is to keep the two stocks from moving in tandem. Based on these specifications, it is expected that speculators would perform poorly since they would on the average buy high and sell low when dealing with the arbitragers.

Arbitragers compare the futures price to the market index when doing a portfolio review and then place the appropriate order if prices deviate as described earlier. The appropriate order will always consist of two buys (stock) and a sell (futures) or two sells (stock) and a buy (futures).

Since arbitragers are users of 'program trading,' they are often implicated as a cause of increased volatility. By

varying the number of arbitragers while holding the numbers of other investors fixed, this implication can be tested.

II. RESULTS

II.A. MARKET VOLATILITY

II.A.1. EFFECTS ON VOLATILITY BY ARBITRAGERS

Tables 12 to 21 summarize the market volatility and trading volume for a fixed proportion of rebalancers, ORPIs and speculators in each table with varying numbers of arbitragers. All reported values refer to the futures market where the rebalancers and insurers trade. Various numbers of speculators were tried, and ten seemed to be sufficient to provide sufficient trading without enormously slowing down the simulations.

Table 12, which has no insurers, suggests that increasing numbers of arbitragers result in increasing market volatility. However, Table 13 and Table 14 with five and ten ORPIs respectively, do not show any particular pattern to market volatility relative to arbitrager numbers.

As the number of ORPIs is increased in Table 15 through Table 18, it appears that market volatility is generally *decreased* by increasing numbers of arbitragers. Thus, it seems that in a world with substantial amounts of portfolio insurers, index arbitragers may in fact be a stabilizing influence.

In Table 19 and Table 20, although large numbers of arbitragers (10, 25 or 50) appear to be stabilizing, smaller

numbers are sometimes stabilizing and sometimes destabilizing. Finally, Table 21 (all ORPIs) is an example of extremely unstable markets where the arbitragers appear to be largely irrelevant. These last three tables are not particularly conclusive since large numbers of portfolio insurers lead to extreme price movements. Results in simulations using more speculators are qualitatively similar.

In addition to the above tables, Figures 4 through 13 (analogous to Tables 12 through 21) plot the quarterly returns (instead of daily returns) of the market for three levels of arbitragers. Like Table 12, Figure 4 suggests that in a market with no portfolio insurers, more arbitagers add volatility. Figures 5 through 12 show no apparent affect of arbitragers on volatility. Finally, Figure 13, a no insurer simulation, suggests strongly that arbitragers can be highly stabilizing when trend following strategies (insurers for instance) dominate the market.

II.A.2. MARKET VOLATILITY SUMMARY

The above simulations demonstrate that the common complaint that market volatility is caused by program trading is only partly correct. Program trading by portfolio insurers is clearly a volatility increasing strategy as was shown in Kim and Markowitz (1989) for constant proportion portfolio insurance and early in this paper for option replicating portfolio insurance. The

results of this section only confirm this. If one compares market volatility with a fixed number of index arbitragers and increasing numbers of portfolio insurers (across tables), there is a nearly monotonic increase in market volatility.

However, program trading by index arbitragers does not appear to be a source of market volatility, and in fact may be a market stabilizer. Thus, it is important that market regulators be careful not to throw the baby out with the bath water by eliminating program trading and/or more drastically, stock index futures contracts.

II.B. TRADING STRATEGY PROFITABILITY

II.B.1. REBALANCERS VERSUS INSURERS

Figures 14, 15 and 16 compare the profitability of rebalancers and ORPIs for two levels of arbitragers. Essentially, the results in all three figures are identical. First, there is an enormous wealth transfer to the rebalancers from the ORPIs. Second, the more ORPIs, the faster the wealth transfer takes place. The second effect is most likely caused by more ORPIs causing greater mispricings and thus greater rebalancer gains sooner.

The third effect is that more arbitragers slightly slower wealth transfer but ultimately better average rebalancer performance. This may be due to speculator influence. With no arbitragers, the speculators have no effect on the rebalancers and ORPIs. With arbitragers

present, the speculators may dampen the mispricings caused by ORPIs since the two trading strategies are uncorrelated. Although this causes a slower wealth transfer, the speculators ultimately lose some money which is captured by the rebalancers.

II.B.2. ARBITRAGERS AND SPECULATORS

Figure 17 reports the performance of the average arbitrager and speculator for three levels of rebalancers and ORPIs. Typically, the arbitragers do quite well and the speculators do poorly. In fact, the arbitragers do nearly as well as the rebalancers. The speculators, on the other hand, end up with somewhat more wealth than they began with. This is due to the positive net deposit inflow which offsets poor trading performance.

II.C. DAILY VERSUS QUARTERLY VOLATILITY

Figures 18, 19 and 20 compare daily volatilities with adjusted quarterly volatilities (quarterly volatility divided by the square root of 65-the number of days in a quarter) for various levels of arbitragers. In general, there does not appear to be any relationship between the two types of volatility. Figure 19 has lower daily volatility for larger numbers of arbitragers with no apparent pattern in the quarterly numbers. The other two figures show no patterns at all.

The only observation that can be made is that quarterly volatilities are much lower than one would expect from looking at the daily volatilities. This suggests that daily fluctuations tend to be random and are averaged out over time. This was noted earlier in the paper.

III. CONCLUSIONS

All the results in this research suggest roughly the same things:

1: Portfolio insurers increase market volatility while rebalancers and arbitragers reduce or have no effect on market volatility.

2: Portfolio insurers appear to be using a mean variance inefficient trading strategy while rebalancers and arbitragers appear to have very efficient strategies.

3: Short term market volatility is largely transitory and smart investors will either ignore it or buy(sell) when the volatility has pushed the market down(up).

4: Since program trading is used by both portfolio insurers and index arbitragers, it is not clear that program trading is harmful. Related to this is that futures markets are not inherently harmful due to either of the above.

The above conclusions strongly suggest that requlators should be very careful before putting restrictions on the market. Restricting program trades would appear to eliminate both 'bad' and 'good' traders. Similarly, if

daily volatility is transitory, there appears to be no justification for trading limits or stoppages.

IV. FUTURE RESEARCH

This paper has not considered the use of margin nor the short sale uptick rule. These two items may be significant factors in market volatility. Also, a market where market and stop orders are placed in addition to limit orders was not considered. Such a market may behave somewhat differently. Thirdly, specialists and/or competing market makers were not modelled. Finally, more research is needed to determine how investors trade so that better models of the market can be made.

TABLE 1
Period Standard Deviation
Simulation Runs with Various Numbers
of ORPI Investors of 150 Total

ORPI Investors adjust their portfolio if they deviate by more than 1% and less than 10% of target stock to total value.

	Number of ORPI Investors							
Quarter	0	1	3	5	10	20	50	100
1	0.0227	0.0213	0.0283	0.0258	0.0275	0.0280	0.0363	0.0468
2	0.0258	0.0217	0.0249	0.0245	0.0282	0.0326	0.0341	0.0397
3	0.0219	0.0260	0.0297	0.0248	0.0231	0.0308	0.0344	0.0409
4	0.0271	0.0205	0.0260	0.0215	0.0257	0.0281	0.0344	0.0401
5	0.0229	0.0240	0.0257	0.0203	0.0216	0.0261	0.0422	0.0464
10	0.0142	0.0189	0.0182	0.0191	0.0237	0.0220	0.0290	0.0461
20	0.0190	0.0150	0.0167	0.0185	0.0142	0.0237	0.0248	0.0302
30	0.0102	0.0100	0.0146	0.0162	0.0250	0.0213	0.0278	0.0447
40	0.0077	0.0077	0.0134	0.0129	0.0159	0.0258	0.0313	0.0459
50	0.0050	0.0103	0.0061	0.0117	0.0193	0.0232	0.0247	0.0440
60	0.0071	0.0064	0.0097	0.0125	0.0168	0.0161	0.0313	0.0395
70	0.0048	0.0101	0.0072	0.0087	0.0119	0.0135	0.0299	0.0361
80	0.0037	0.0036	0.0065	0.0086	0.0066	0.0268	0.0259	0.0405
90	0.0033	0.0012	0.0100	0.0099	0.0107	0.0164	0.0317	0.0416
100	0.0025	0.0025	0.0048	0.0046	0.0125	0.0175	0.0225	0.0450
Overall	0.0113	0.0113	0.0128	0.0145	0.0168	0.0222	0.0310	0.0437

TABLE 2
Period Trading Volume
Simulation Runs with Various Numbers
of ORPI Investors of 150 Total

ORPI Investors adjust their portfolio if they deviate by more than 1% and less than 10% of target stock to total value.

	Number of ORPI Investors							
Quarter	0	1	3	5	10	20	50	100
1	4627	4745	4760	4969	6066	7438	8967	9993
2	4202	4998	5084	4955	5432	6134	7882	10236
3	3919	3989	4377	4843	4962	5854	8846	10828
4	4276	3904	3927	4504	5892	5544	8998	6015
5	3888	4234	3423	3798	4164	5919	8638	7362
10	2097	2193	1996	3476	3429	3381	5756	8157
20	1636	1408	1598	1659	2188	3411	3146	5829
30	772	680	1182	1255	1950	3297	4260	6576
40	639	503	1017	863	1338	3230	4375	6457
50	249	305	390	709	1220	2445	4180	7798
60	302	330	595	836	1028	1942	5127	7664
70	143	289	384	380	1156	1742	4357	8284
80	187	198	248	470	816	1965	2979	7918
90	218	24	366	453	610	1504	4331	6479
100	148	181	270	174	533	1698	2681	7460
Overall	86771	85492	97688	120130	158691	261334	457448	785083

TABLE 3A
Portfolio Betas
Simulation Runs with Various Numbers
of ORPI Investors of 150 Total

Number of ORPI Investors	1	2	3	5	10	25	50	75	100	125	145
ORPIs											
Average Beta	1.22	1.10	1.10	1.06	1.00	-11[a]	0.99	1.02	1.00	0.99	1.02
S.E.	N.A.	0.02	0.02	0.04	0.05	11[a]	0.03	0.03	0.03	0.03	0.02
t-Val[b]	N.A.	6.06	5.23[d]	1.50	0.00	-1.0	-0.3	0.78	0.07	-0.3	1.09
G.T.M.	1	2	3	4	5	15	24	37	50	63	78
%	100	100	100	80.0	50.0	60.0	48.0	49.3	50.0	50.4	53.8
Rebalancers											
Average Beta	1.00	1.00	1.00	1.00	1.00	0.99	1.00	0.97	0.99	1.00	0.74
S.E.	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
t-Val[b]	-1.0	-1.8[c]	0.75	-2.8[d]	-0.6	-21.[e]	-3.5[e]	-30.[e]	-12.[e]	-1.6	-61[e]
G.T.M.	72	61	66	61	67	2	33	0	3	10	0
%	48.3	41.2	44.9	42.1	47.9	1.6	33.0	0	6.0	40.0	0

[a] There was one investor among the 25 with an extremely negative beta due to large net cash outflows and thus small portfolio value. Since betas are equally weighted, this one outlier significantly affected the results.

[b] t-Value testing whether average investor type beta was different from one.

[c] Significant at the 10% level (two-tailed).

[d] Significant at the 5% level (two-tailed).

[e] Significant at the 1% level (two-tailed).

TABLE 3B
Portfolio Betas
Simulation Runs with Various Numbers
of CPPI Investors of 150 Total

Number of CPPI Investors											
	1	2	3	5	10	25	50	75	100	125	145
CPPIs											
Average Beta	0.98	0.97	1.00	0.97	0.98	0.96	1.67	2.02	2.94	1.05	0.97
S.E.	N.A.	0.00	0.01	0.01	0.00	0.01	0.87	0.69	1.27	0.51	0.26
t-Val[b]	N.A.	-49.[d]	-0.4	-4.2[d]	-3.6[e]	-6.4[e]	0.77	1.49	1.53	0.09	-0.1
G.T.M.	0	0	1	1	1	2	13	13	26	30	19
%	0	0	33.3	20	10	8	26	17.3	26	24	13.1
Rebalancers											
Average Beta	1.00	1.00	0.99	1.00	1.00	1.01	1.06	1.02	0.89	0.81	2.17
S.E.	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.01	0.02	0.33
t-Val[b]	0.20	1.77[c]	-1.0	1.60	2.09[d]	5.29[e]	17.3[e]	8.82[e]	-15.[e]	-10.[e]	3.58[d]
G.T.M.	79	77	70	81	86	87	94	65	0	0	5
%	53.0	52.0	47.6	55.9	61.4	69.6	94	86.7	0	0	100

[b] t-Value testing whether average investor type beta was different from one.

[c] Significant at the 10% level (two-tailed).

[d] Significant at the 5% level (two-tailed).

[e] Significant at the 1% level (two-tailed).

TABLE 4A
Portfolio Alphas (in %)
Simulation Runs with Various Numbers
of ORPI Investors of 150 Total

	Number of ORPI Investors										
	1	2	3	5	10	25	50	75	100	125	145
ORPIs											
Average Alpha	-0.4	-0.3	-0.3	-0.2	-0.2	-9.7	-0.2	-0.2	-0.2	-0.1	0.23
S.E.	N.A.	0.02	0.01	0.03	0.03	9.31	0.02	0.02	0.02	0.02	0.13
t-Val[a]	N.A.	-12.[c]	-28.[e]	-7.4[e]	-6.0[e]	-1.0	-11.[e]	-12.[e]	-9.7[e]	-6.0[e]	1.72[c]
G.T.M.	0	0	0	0	0	1	3	6	14	29	73
%	0	0	0	0	0	4.0	6.0	8.0	14.0	23.2	50.3
Rebalancers											
Average Alpha	0.00	0.00	-0.0	0.01	0.01	0.05	0.10	0.21	0.41	0.81	0.69
S.E.	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.01	0.01	0.01	0.03
t-Val[a]	1.18	3.32[e]	-0.8	8.15[e]	13.3[e]	24.4[e]	29.9[e]	37.3[e]	39.1[e]	62.3[e]	22.0[e]
G.T.M.	53	66	73	88	106	125	100	75	50	25	5
%	35.3	44.6	49.7	60.7	75.7	100	100	100	100	100	100

[a] t-Value testing whether average investor type alpha is different from zero.

[c] Significant at the 10% level (two-tailed).

[d] Significant at the 5% level (two-tailed).

[e] Significant at the 1% level (two-tailed).

TABLE 4B
Portfolio Alphas (in %)
Simulation Runs with Various Numbers
of CPPI Investors of 150 Total

	Number of CPPI Investors										
	1	2	3	5	10	25	50	75	100	125	145
CPPIs											
Average Alpha	-0.0	-0.0	-0.0	-0.0	-0.0	-0.1	-6.6	-2.8	-0.9	-4.2	8.31
S.E.	N.A.	0.00	0.01	0.01	0.00	0.01	4.90	2.11	4.11	4.66	5.80
t-Val[a]	N.A.	N.A.	-4.4[d]	-0.9	-4.0[e]	-17.[e]	-1.4	-1.3	-0.2	-0.9	1.43
G.T.M.	0	0	0	1	0	0	1	6	17	22	54
%	0	0	0	20	0	0	2	8	17	17.6	37.2
Rebalancers											
Average Alpha	-0.0	-0.0	-0.0	-0.0	0.00	0.03	0.44	1.00	1.80	3.38	2.33
S.E.	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.02	0.04	0.21	0.01
t-Val[a]	-0.6	-1.2	-1.1	-0.8	1.14	14.1[e]	65.3[e]	50.5[e]	42.6[e]	15.9[e]	31.1[e]
G.T.M.	45	40	51	51	46	104	100	75	50	25	5
%	30.2	27.0	34.7	35.2	32.9	83.2	100	100	100	100	100

[a] t-Value testing whether average investor type alpha is different from zero.

[c] Significant at the 10% level (two-tailed).

[d] Significant at the 5% level (two-tailed).

[e] Significant at the 1% level (two-tailed).

TABLE 5A
Portfolio Sharpe Measures(in %)
Simulation Runs with Various Numbers
of ORPI Investors of 150 Total

	Number of ORPI Investors										
	1	2	3	5	10	25	50	75	100	125	145
Market Sharpe	59.0	57.4	42.9	53.3	40.2	30.1	25.3	20.8	19.7	16.4	8.81
	ORPIs										
Average Sharpe	41.0	40.1	32.0	39.2	30.5	21.5	18.6	15.4	14.8	12.2	8.33
S.E.	N.A.	0.93	0.12	1.14	0.99	1.36	0.38	0.29	0.31	0.35	0.26
t-Val[a]	N.A.	-19.[d]	-88.[e]	-12.[e]	-9.8[e]	-6.3[e]	-18.[e]	-19.[e]	-16.[e]	-12.[e]	-1.8[c]
G.T.M.	0	0	0	0	0	0	0	0	1	16	66
%	0	0	0	0	0	0	0	0	1	12.8	45.5
	Rebalancers										
Average Sharpe	59.0	57.5	42.7	53.6	40.6	31.4	27.5	24.8	26.1	25.9	12.4
S.E.	0.05	0.07	0.31	0.05	0.04	0.05	0.07	0.11	0.16	0.16	0.17
t-Val[a]	-0.1	1.34	-0.6	6.69[e]	12.4[e]	24.6[e]	29.6[e]	37.0[e]	40.1[e]	60.2[e]	21.2[e]
G.T.M.	78	87	100	110	118	125	100	75	50	25	5
%	52.3	58.8	68.0	75.9	84.3	100	100	100	100	100	100

[a] t-Value tests whether investor type average Sharpe measure is greater or less than the market's Sharpe measure.

[c] Significant at the 10% level (two-tailed).

[d] Significant at the 5% level (two-tailed).

[e] Significant at the 1% level (two-tailed).

TABLE 5B
Portfolio Sharpe Measures(in %)
Simulation Runs with Various Numbers
of CPPI Investors of 150 Total

	Number of CPPI Investors										
	1	2	3	5	10	25	50	75	100	125	145
Market Sharpe	73.2	68.3	60.3	67.6	68.9	51.1	22.9	19.2	14.5	12.5	9.83
CPPIs											
Average Sharpe	70.5	66.6	57.5	66.6	65.6	41.8	-21.	-29.	-24.	-17.	-5.8
S.E.	N.A.	0.31	0.54	0.52	0.70	0.45	1.13	2.09	2.12	1.77	1.45
t-Val[a]	N.A.	-7.7[c]	-5.1[d]	-1.8	-4.8[e]	-21.[e]	-39.[e]	-23.[e]	-18.[e]	-17.[e]	-11.[e]
G.T.M.	0	0	0	1	0	0	0	0	0	4	27
%	0	0	0	20	0	0	0	0	0	3.2	18.6
Rebalancers											
Average Sharpe	73.1	68.0	59.6	67.4	68.9	52.2	30.4	30.2	27.2	19.6	11.0
S.E.	0.07	0.10	0.58	0.09	0.05	0.09	0.13	0.21	0.31	0.79	0.17
t-Val[a]	-2.4[d]	-2.5[d]	-1.2	-2.1[d]	-1.0	11.9[e]	59.8[e]	51.9[e]	40.1[e]	8.98[e]	7.20[e]
G.T.M.	64	63	64	67	68	105	100	75	50	25	5
%	43.0	42.6	43.5	46.2	48.6	84	100	100	100	100	100

[a] t-Value tests whether investor type average Sharpe measure is greater or less than the market's Sharpe measure.

[c] Significant at the 10% level (two-tailed).

[d] Significant at the 5% level (two-tailed).

[e] Significant at the 1% level (two-tailed).

TABLE 6A
Portfolio Treynor Measures(in %)
Simulation Runs with Various Numbers
of ORPI Investors of 150 Total

	Number of ORPI Investors										
	1	2	3	5	10	25	50	75	100	125	145
Market Treynor	1.05	1.02	1.07	1.07	1.04	1.08	1.12	1.13	1.29	1.34	1.86
ORPIs											
Average Treynor	0.75	0.75	0.83	0.84	0.85	0.92	0.92	0.93	1.09	1.13	2.01
S.E.	N.A.	0.03	0.01	0.03	0.03	0.05	0.02	0.02	0.02	0.03	0.12
t-Val[a]	N.A.	-10.[c]	-42.[e]	-7.5[e]	-5.9[e]	-3.6[e]	-11.[e]	-12.[e]	-9.6[e]	-6.7[e]	1.24
G.T.M.	0	0	0	0	0	2	3	7	14	33	84
%	0	0	0	0	0	8	6	9.33	14	26.4	57.9
Rebalancers											
Average Treynor	1.05	1.03	1.06	1.08	1.06	1.13	1.21	1.35	1.71	2.15	2.79
S.E.	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.01	0.01	0.01	0.04
t-Val[a]	1.44	3.12[e]	-0.6	8.62[e]	13.5[e]	25.0[e]	29.7[e]	36.9[e]	39.8[e]	60.7[e]	23.4[e]
G.T.M.	82	99	109	118	120	125	100	75	50	25	5
%	55.0	66.9	74.1	81.4	85.7	100	100	100	100	100	100

[a] t-Value tests whether investor type average Treynor measure is greater or less than the market's Treynor measure.

[c] Significant at the 10% level (two-tailed).

[d] Significant at the 5% level (two-tailed).

[e] Significant at the 1% level (two-tailed).

TABLE 6B
Portfolio Treynor Measures(in %)
Simulation Runs with Various Numbers
of CPPI Investors of 150 Total

	Number of CPPI Investors										
	1	2	3	5	10	25	50	75	100	125	145
Market Treynor	1.03	1.02	1.04	1.03	1.03	1.04	1.18	1.07	1.04	1.13	9.21
CPPIs											
Average Treynor	1.03	1.00	1.00	1.02	0.98	0.88	-2.9	0.63	-24.	-3.1	-131
S.E.	N.A.	0.00	0.01	0.01	0.01	0.01	1.56	4.10	14.3	3.97	217.
t-Val[a]	N.A.	-7.3[d]	-4.9[d]	-1.2	-4.2[e]	-18.[e]	-2.6[d]	-0.1	-1.8[c]	-1.1	-0.6
G.T.M.	0	0	0	1	0	0	1	9	17	41	84
%	0	0	0	20	0	0	2	12	17	32.8	57.9
Rebalancers											
Average Treynor	1.03	1.02	1.04	1.03	1.03	1.06	1.60	2.06	3.08	5.50	10.4
S.E.	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.02	0.07	0.36	0.15
t-Val[a]	-0.7	-1.3	0.74	-0.7	1.19	13.8[e]	62.8[e]	47.7[e]	30.9[e]	12.2[e]	7.69[e]
G.T.M.	74	79	70	75	81	111	100	75	50	25	5
%	49.7	53.4	47.6	51.7	57.9	88.8	100	100	100	100	100

[a] t-Value tests whether investor type average Treynor measure is greater or less than the market's Treynor measure.

[c] Significant at the 10% level (two-tailed).

[d] Significant at the 5% level (two-tailed).

[e] Significant at the 1% level (two-tailed).

TABLE 7A
Portfolio Standard Deviations(in %)
Simulation Runs with Various Numbers
of ORPI Investors of 150 Total

	Number of ORPI Investors										
	1	2	3	5	10	25	50	75	100	125	145
Market											
S.D.	1.77	1.78	2.49	2.01	2.60	3.59	4.40	5.44	6.51	8.20	21.1
	ORPIs										
Average											
S.D.	2.24	2.07	2.86	2.27	2.79	4.16	4.81	6.13	7.27	9.15	24.7
G.T.M.	1	2	3	5	8	18	34	55	65	78	84
%	100	100	100	100	80	72	68	73.3	65	62.4	57.9
	Rebalancers										
Average											
S.D.	1.78	1.78	2.90	2.01	2.60	3.54	4.40	5.30	6.46	8.26	16.7
G.T.M.	81	78	75	75	76	3	42	0	10	19	0
%	54.4	52.7	51.0	51.7	54.3	2.4	42	0	20	76	0

TABLE 7B
Portfolio Standard Deviations(in %)
Simulation Runs with Various Numbers
of CPPI Investors of 150 Total

Number of CPPI Investors											
	1	2	3	5	10	25	50	75	100	125	145
Market S.D.	1.40	1.49	1.72	1.52	1.49	2.03	5.14	5.59	7.15	9.06	93.7
CPPIs											
Average S.D.	1.39	1.46	1.72	1.49	1.48	2.01	11.6	36.9	67.0	92.9	103.
G.T.M.	0	0	1	1	3	8	28	74	99	111	29
%	0	0	33.3	20	30	32	56	98.7	99	88.8	20
Rebalancers											
Average S.D.	1.40	1.50	1.75	1.53	1.49	2.05	5.56	6.99	10.2	23.7	205.
G.T.M.	83	86	81	87	93	100	100	75	50	25	5
%	55.7	58.1	55.1	60	66.4	80	100	100	100	100	100

TABLE 8A
Means Difference Tests of
Rebalancer minus ORPI Portfolios
Simulation Runs with Various Numbers
of ORPI Investors of 150 Total
(Alpha, Sharpe and Treynor in %)

	Number of ORPI Investors										
	1	2	3	5	10	25	50	75	100	125	145
Beta											
Diff.	-0.2	-0.1	-0.1	-0.1	-0.0	2.29	0.01	-0.1	-0.0	0.01	-0.3
t-Val[a]	-18.[e]	-16.[e]	-2.0[d]	-8.2[e]	-0.0	11.5[e]	0.36	-1.6	-0.3	0.09	-2.5[d]
Alpha											
Diff.	0.36	0.30	0.24	0.26	0.20	9.77	0.29	0.41	0.58	0.95	0.46
t-Val[a]	32.4[e]	30.2[e]	1.68[c]	34.1[e]	22.8[e]	2.38[d]	21.9[e]	23.0[e]	22.9[e]	17.7[e]	0.64
Sharpe											
Diff.	18.0	17.4	10.7	14.5	10.1	9.89	8.91	9.36	11.2	13.8	4.09
t-Val[a]	29.2[e]	29.9[e]	5.00[e]	46.2[e]	34.9[e]	16.2[e]	31.0[e]	30.1[e]	24.9[e]	17.5[e]	2.90[e]
Treynor											
Diff.	0.29	0.27	0.23	0.24	0.20	0.21	0.30	0.42	0.62	1.02	0.79
t-Val[a]	*[e]	28.7[e]	3.05[e]	34.0[e]	22.6[e]	10.2[e]	21.5[e]	22.8[e]	20.4[e]	14.6[e]	1.23

* Extremely high.

[a] t-Value tests whether average Rebalancer is higher than the average ORPI. Negative values indicate that average Rebalancer is lower than the average ORPI. It is assumed that the variances are unknown but equal.

[c] Significant at the 10% level (two-tailed).

[d] Significant at the 5% level (two-tailed).

[e] Significant at the 1% level (two-tailed).

TABLE 8B
Means Difference Tests of
Rebalancer minus CPPI Portfolios
Simulation Runs with Various Numbers
of CPPI Investors of 150 Total
(Alpha, Sharpe and Treynor in %)

	Number of CPPI Investors										
	1	2	3	5	10	25	50	75	100	125	145
Beta											
Diff.	0.02	0.03	-0.0	0.03	0.02	0.05	-0.6	-1.0	-2.1	-0.2	1.20
t-Val[a]	N.A.	4.27[e]	-0.1	4.78[e]	5.67[e]	11.4[e]	-1.0	-1.5	-1.1	-0.2	0.87
Alpha											
Diff.	0.03	0.02	0.04	0.01	0.04	0.18	7.08	3.85	2.70	7.53	-6.0
t-Val[a]	2.46[d]	N.A.	1.79[c]	1.00	9.92[e]	30.5[e]	2.04[d]	1.82[c]	0.46	0.72	-0.2
Sharpe											
Diff.	2.56	1.44	2.06	0.77	3.30	10.4	51.8	59.7	51.5	37.1	16.8
t-Val[a]	3.04[e]	1.61	0.51	1.63	12.1[e]	36.0[e]	63.2[e]	28.4[e]	17.1[e]	9.33[e]	2.15[d]
Treynor											
Diff.	0.03	0.02	0.05	0.01	0.04	0.18	4.53	1.43	27.0	8.55	141.
t-Val[a]	2.43[d]	1.60	1.74[c]	1.23	10.7[e]	31.5[e]	4.10[e]	0.35	1.34	0.96	0.12

[a] t-Value tests whether average Rebalancer is higher than the average CPPI. Negative values indicate that average Rebalancer is lower than the average CPPI. It is assumed that the variances are unknown but equal.

[c] Significant at the 10% level (two-tailed).

[d] Significant at the 5% level (two-tailed).

[e] Significant at the 1% level (two-tailed).

TABLE 9
Non-Parametric Means Difference Tests
(Mann-Whitney Test)

	Number of ORPI Investors									
	2	3	5	10	25	50	75	100	125	145
Mann-Whitney values[d]										
Beta	296[c]	441[c]	579[b]	700	1179[a]	2599	2508	2613	1586	650[c]
Alpha	0[c]	0[c]	0[c]	15[c]	116[c]	18[c]	6[c]	71[c]	19[c]	91[c]
Sharpe	0[c]	0[c]	0[c]	0[c]	91[c]	98[c]	0[c]	0[c]	0[c]	27[c]
Treynor	0[c]	0[c]	0[c]	16[c]	224[c]	45[c]	14[c]	12[c]	0[c]	48[c]
Std. Dev.	296[c]	441[c]	725[c]	1116[c]	2375[c]	3399[c]	4263[c]	3311[c]	1933[a]	623[c]

[a] Significantly different at the 10% level (two-tailed).

[b] Significantly different at the 5% level (two-tailed).

[c] Significantly different at the 1% level (two-tailed).

[d] Low Mann-Whitney values indicate that the average Rebalancer exceeds the average ORPI as with Alphas, Sharpe measures and Treynor measures. The reverse is true for Beta and Standard Deviation.

Table 10
Portfolio Statistics for Rebalancers, CPPIs and ORPIs
(150 of each investor type per simulation)

Rebalancers					
	Alpha	Beta	Sharpe	Treynor	Std. Dev.
Average.....	0.000128	0.991933	0.233453	0.008579	0.044532
Standard Error	0.000188	0.008227	0.001363	0.000794	0.004577
t-Value.....	0.682396	-0.98048	-3.10066	-1.19094	0.974353
G.T.M.	54	77	29	24	47
%	36	51.33333	19.33333	16	31.33333
CPPIs					
	Alpha	Beta	Sharpe	Treynor	Std. Dev.
Average.....	-9.9E-05	0.992077	0.233134	0.009332	0.039712
Standard Error	2.69E-05	0.000822	0.000675	2.69E-05	3.29E-05
t-Value.....	-3.68718	-9.63966	-6.73727	-7.15961	-10.9728
G.T.M.	63	32	42	40	32
%	42	21.33333	28	26.66667	21.33333
ORPIs					
	Alpha	Beta	Sharpe	Treynor	Std. Dev.
Average.....	-3.1E-05	1.046315	0.209471	0.00959	0.046687
Standard Error	0.000147	0.02531	0.002998	0.000157	0.000879
t-Value.....	-0.21287	1.829894	-9.40779	0.418299	7.521876
G.T.M.	77	86	27	77	111
%	51.33333	57.33333	18	51.33333	74

NOTES
G.T.M. means greater than the market.
t-Values are as follows:
For alpha: difference from zero
For beta: difference from one
For sharpe, treynor and std. dev.: difference from market

TABLE 11
Portfolio Performance of All Three Trading Strategies in Simulations Using S&P 500 Data

(Numbers below are the means and standard deviations of returns for the average investor of each trading strategy)

	Rebalancers	CPPIs	ORPIs
1962-1988			
Average	0.0209	0.0205	0.0210
Std.Dev.	0.0421	0.0408	0.0430
1962-1979			
Average	0.0205	0.0201	0.0196
Std.Dev.	0.0418	0.0394	0.0354
1980-1988			
Average	0.0216	0.0212	0.0234
Std.Dev.	0.0424	0.0432	0.0542

TABLE 12
Simulation Runs with Various Numbers
of Arbitrager Investors

Panel A: Period Standard Deviation
150 Rebalancers, 0 ORPIs and 10 Speculator Investors

	Number of Arbitager Investors						
Quarter	0	1	2	5	10	25	50
1	0.0219	0.0226	0.0207	0.0295	0.0237	0.0275	0.0226
2	0.0260	0.0172	0.0248	0.0234	0.0211	0.0239	0.0242
3	0.0258	0.0184	0.0205	0.0227	0.0258	0.0210	0.0242
4	0.0228	0.0218	0.0245	0.0219	0.0151	0.0265	0.0201
5	0.0217	0.0239	0.0249	0.0182	0.0209	0.0202	0.0180
10	0.0157	0.0263	0.0176	0.0207	0.0197	0.0143	0.0184
20	0.0119	0.0154	0.0181	0.0133	0.0162	0.0169	0.0181
30	0.0112	0.0141	0.0110	0.0127	0.0093	0.0115	0.0189
40	0.0081	0.0067	0.0150	0.0094	0.0063	0.0235	0.0177
50	0.0074	0.0068	0.0051	0.0084	0.0043	0.0115	0.0206
60	0.0065	0.0074	0.0029	0.0070	0.0168	0.0025	0.0165
70	0.0048	0.0021	0.0079	0.0051	0.0046	0.0045	0.0154
80	0.0045	0.0022	0.0021	0.0030	0.0105	0.0063	0.0131
90	0.0022	0.0030	0.0024	0.0043	0.0090	0.0042	0.0123
100	0.0037	0.0018	0.0022	0.0039	0.0039	0.0041	0.0115
Overall	0.0107	0.0113	0.0109	0.0113	0.0120	0.0132	0.0170

Panel B: Period Volume
150 Rebalancers, 0 ORPIs and 10 Speculator Investors

	Number of Arbitrager Investors						
Quarter	0	1	2	5	10	25	50
1	4663	4442	4154	4231	4109	5430	5814
2	4167	4802	5355	4962	4137	6051	6826
3	3988	3686	4260	4370	3723	5079	6025
4	3771	4366	4121	4261	3209	3665	6235
5	3538	3813	3741	3729	3449	3923	5113
10	2307	2297	2127	2416	3096	2782	4310
20	1117	1307	1064	1190	1439	2073	2790
30	519	1223	703	683	1369	608	2582
40	568	375	798	606	359	2110	1271
50	385	270	315	581	287	779	1311
60	305	434	233	231	1093	150	666
70	257	101	214	207	217	203	1106
80	138	97	153	201	417	337	597
90	99	172	110	116	387	82	507
100	71	83	96	55	67	196	405
Overall	80343	86337	84261	88133	98392	107976	177518

TABLE 13
Simulation Runs with Various Numbers
of Arbitrager Investors

Panel A: Period Standard Deviation
145 Rebalancers, 5 ORPIs and 10 Speculator Investors

	Number of Arbitager Investors						
Quarter	0	1	2	5	10	25	50
1	0.0364	0.0396	0.0352	0.0387	0.0428	0.0476	0.0358
2	0.0318	0.0424	0.0387	0.0338	0.0375	0.0379	0.0376
3	0.0300	0.0332	0.0320	0.0416	0.0357	0.0398	0.0436
4	0.0449	0.0426	0.0358	0.0463	0.0410	0.0387	0.0441
5	0.0418	0.0338	0.0339	0.0371	0.0393	0.0376	0.0379
10	0.0290	0.0283	0.0299	0.0360	0.0360	0.0304	0.0292
20	0.0368	0.0368	0.0171	0.0269	0.0282	0.0298	0.0244
30	0.0245	0.0233	0.0194	0.0175	0.0314	0.0262	0.0331
40	0.0220	0.0148	0.0291	0.0102	0.0261	0.0252	0.0287
50	0.0214	0.0149	0.0149	0.0125	0.0250	0.0353	0.0209
60	0.0352	0.0145	0.0075	0.0148	0.0276	0.0244	0.0233
70	0.0208	0.0069	0.0134	0.0069	0.0158	0.0227	0.0338
80	0.0326	0.0129	0.0086	0.0084	0.0449	0.0218	0.0362
90	0.0206	0.0071	0.0194	0.0067	0.0197	0.0179	0.0223
100	0.0287	0.0153	0.0084	0.0108	0.0310	0.0273	0.0199
Overall	0.0283	0.0231	0.0223	0.0190	0.0269	0.0290	0.0275

Panel B: Period Volume
145 Rebalancers, 5 ORPIs and 10 Speculator Investors

	Number of Arbitrager Investors						
Quarter	0	1	2	5	10	25	50
1	12403	12429	12731	14042	13433	15249	12557
2	11843	11690	12574	10767	11368	12474	11655
3	9729	9030	10167	14964	9585	8882	14688
4	11689	11257	10146	9535	11448	7842	10170
5	8834	6525	10019	9231	9529	9053	9339
10	4150	4077	5140	6584	8378	4209	7960
20	2530	4425	2076	2101	4400	1761	5637
30	1707	1498	1741	1097	3091	2599	3992
40	1145	655	1357	569	1141	3104	2731
50	741	506	537	666	1067	1328	1352
60	1479	540	198	559	729	819	1082
70	700	236	422	229	293	1043	1333
80	645	297	209	203	811	394	535
90	462	144	326	188	348	392	259
100	324	228	184	234	404	277	666
Overall	189026	179132	194495	174302	229563	215013	291490

TABLE 14
Simulation Runs with Various Numbers
of Arbitrager Investors

Panel A: Period Standard Deviation
140 Rebalancers, 10 ORPIs and 10 Speculator Investors

	Number of Arbitager Investors						
Quarter	0	1	2	5	10	25	50
1	0.0551	0.0512	0.0427	0.0562	0.0492	0.0490	0.0473
2	0.0642	0.0453	0.0655	0.0513	0.0423	0.0428	0.0415
3	0.0744	0.0532	0.0477	0.0544	0.0565	0.0602	0.0366
4	0.0431	0.0723	0.0407	0.0402	0.0561	0.0454	0.0422
5	0.0521	0.0527	0.0398	0.0460	0.0447	0.0441	0.0414
10	0.0366	0.0410	0.0463	0.0508	0.0402	0.0443	0.0323
20	0.0273	0.0327	0.0256	0.0378	0.0313	0.0405	0.0315
30	0.0250	0.0332	0.0244	0.0339	0.0262	0.0331	0.0322
40	0.0189	0.0243	0.0213	0.0264	0.0209	0.0517	0.0308
50	0.0115	0.0232	0.0166	0.0271	0.0267	0.0226	0.0265
60	0.0132	0.0290	0.0155	0.0286	0.0075	0.0211	0.0177
70	0.0087	0.0308	0.0102	0.0205	0.0087	0.0142	0.0202
80	0.0076	0.0116	0.0121	0.0422	0.0091	0.0245	0.0207
90	0.0235	0.0131	0.0160	0.0241	0.0045	0.0165	0.0053
100	0.0141	0.0275	0.0234	0.0126	0.0033	0.0231	0.0111
Overall	0.0251	0.0310	0.0266	0.0345	0.0244	0.0288	0.0277

Panel B: Period Volume
140 Rebalancers, 10 ORPIs and 10 Speculator Investors

	Number of Arbitrager Investors						
Quarter	0	1	2	5	10	25	50
1	28207	27594	24787	25968	26690	30569	25939
2	20753	18727	29497	18248	21666	18877	22270
3	15740	18818	14145	19445	18651	22438	20061
4	11472	17305	11626	13208	17391	16155	15624
5	12036	12572	10327	14502	12726	12513	15205
10	4090	6993	7089	6263	8119	9983	8991
20	1819	2547	2338	3365	3352	6879	4726
30	1326	2700	1345	1686	1806	5088	3330
40	524	1286	809	1346	890	3159	3050
50	406	688	686	834	831	1459	1719
60	331	993	450	646	238	874	360
70	238	570	475	385	260	308	1098
80	248	283	404	516	131	473	194
90	494	187	399	525	120	195	78
100	226	728	480	124	73	328	317
Overall	200985	264020	237440	272700	253851	353545	342202

TABLE 15
Simulation Runs with Various Numbers
of Arbitrager Investors

Panel A: Period Standard Deviation
125 Rebalancers, 25 ORPIs and 10 Speculator Investors

	Number of Arbitager Investors						
Quarter	0	1	2	5	10	25	50
1	0.0894	0.0887	0.0752	0.0819	0.0672	0.0654	0.0663
2	0.0722	0.0660	0.0671	0.0594	0.0642	0.0712	0.0797
3	0.0933	0.0768	0.0688	0.0759	0.0693	0.0641	0.0712
4	0.0651	0.0713	0.0700	0.0628	0.0710	0.0572	0.0530
5	0.0769	0.0783	0.0683	0.0598	0.0540	0.0510	0.0423
10	0.0683	0.0557	0.0467	0.0581	0.0469	0.0450	0.0480
20	0.0377	0.0367	0.0234	0.0483	0.0449	0.0378	0.0350
30	0.0322	0.0331	0.0327	0.0260	0.0339	0.0372	0.0255
40	0.0279	0.0434	0.0181	0.0151	0.0384	0.0388	0.0246
50	0.0295	0.0401	0.0161	0.0227	0.0232	0.0374	0.0333
60	0.0469	0.0183	0.0203	0.0369	0.0222	0.0202	0.0193
70	0.0159	0.0148	0.0100	0.0133	0.0092	0.0041	0.0297
80	0.0062	0.0146	0.0063	0.0193	0.0130	0.0039	0.0333
90	0.0025	0.0239	0.0190	0.0197	0.0082	0.0022	0.0181
100	0.0076	0.0124	0.0161	0.0216	0.0058	0.0025	0.0266
Overall	0.0344	0.0366	0.0297	0.0332	0.0321	0.0320	0.0325

Panel B: Period Volume
125 Rebalancers, 25 ORPIs and 10 Speculator Investors

	Number of Arbitrager Investors						
Quarter	0	1	2	5	10	25	50
1	64300	61118	53325	58384	60493	47959	48251
2	38347	41715	42095	42893	44210	51824	45323
3	42517	35008	43282	37596	43323	41130	37946
4	25354	26009	27271	28722	26784	35974	25012
5	23036	27324	27941	20843	21196	28930	21155
10	10533	7504	5742	14834	8536	15162	16846
20	2950	3108	1358	4632	2921	6874	6473
30	1182	1956	1656	2281	3900	4926	2951
40	1049	1457	653	485	2284	1483	2046
50	999	1218	565	680	1003	1115	2363
60	1470	453	493	1062	391	404	593
70	380	220	392	264	458	239	803
80	204	391	92	537	518	96	549
90	103	331	285	395	165	135	227
100	225	163	105	851	200	179	691
Overall	366665	376712	345345	419357	429907	517183	500501

TABLE 16
Simulation Runs with Various Numbers
of Arbitrager Investors

Panel A: Period Standard Deviation
100 Rebalancers, 50 ORPIs and 10 Speculator Investors

	Number of Arbitager Investors						
Quarter	0	1	2	5	10	25	50
1	0.1050	0.1188	0.0812	0.1111	0.0943	0.0901	0.0984
2	0.0825	0.0931	0.0766	0.0772	0.0818	0.0663	0.0504
3	0.0886	0.0913	0.0917	0.0802	0.0699	0.0753	0.0814
4	0.1035	0.1277	0.0964	0.0948	0.0778	0.0647	0.0574
5	0.0851	0.1205	0.1042	0.0789	0.0712	0.0625	0.0599
10	0.0789	0.0733	0.0626	0.0664	0.0578	0.0566	0.0458
20	0.0356	0.0346	0.0270	0.0475	0.0394	0.0460	0.0472
30	0.0294	0.0243	0.0274	0.0339	0.0341	0.0328	0.0249
40	0.0216	0.0208	0.0069	0.0326	0.0412	0.0271	0.0330
50	0.0085	0.0109	0.0079	0.0187	0.0427	0.0349	0.0381
60	0.0094	0.0076	0.0124	0.0298	0.0179	0.0119	0.0208
70	0.0022	0.0057	0.0042	0.0225	0.0285	0.0207	0.0367
80	0.0039	0.0072	0.0067	0.0174	0.0128	0.0199	0.0115
90	0	0.0064	0.0012	0.0130	0.0110	0.0157	0.0064
100	0.0118	0.0075	0.0039	0.0141	0.0021	0.0022	0.0043
Overall	0.0367	0.0399	0.0356	0.0416	0.0362	0.0354	0.0349

Panel B: Period Volume
100 Rebalancers, 50 ORPIs and 10 Speculator Investors

	Number of Arbitrager Investors						
Quarter	0	1	2	5	10	25	50
1	76961	81715	66832	74654	74182	70412	77606
2	54140	62078	57629	57871	61259	56968	49759
3	55956	51059	60202	52316	52716	62715	63016
4	47107	54958	53271	55720	53763	52799	48241
5	44928	48149	52792	40414	50420	52084	43759
10	16144	14037	10583	15994	23749	27156	22042
20	1986	2637	1690	3311	6155	11941	10265
30	887	1006	1073	1619	2031	3804	3988
40	728	591	207	1383	1467	1749	2387
50	282	340	406	387	1112	1455	1686
60	188	259	109	660	315	244	866
70	94	166	215	505	732	270	583
80	66	111	289	500	82	228	191
90	0	150	63	287	116	246	152
100	197	216	96	292	144	73	186
Overall	505691	517661	584190	577570	676363	733595	741399

TABLE 17
Simulation Runs with Various Numbers
of Arbitrager Investors

Panel A: Period Standard Deviation
75 Rebalancers, 75 ORPIs and 10 Speculator Investors

	Number of Arbitager Investors						
Quarter	0	1	2	5	10	25	50
1	0.3932	0.1434	0.1545	0.1325	0.1214	0.1220	0.1316
2	0.1192	0.0908	0.1168	0.0975	0.0807	0.0732	0.0810
3	0.1099	0.1311	0.1223	0.1159	0.0817	0.0836	0.0691
4	0.0917	0.1109	0.1031	0.0753	0.1268	0.0663	0.0820
5	0.1014	0.1011	0.0962	0.0844	0.0929	0.0696	0.0806
10	0.0889	0.0673	0.0723	0.0703	0.0461	0.0716	0.0485
20	0.0356	0.0633	0.0295	0.0658	0.0557	0.0409	0.0383
30	0.0155	0.0437	0.0374	0.0457	0.0316	0.0511	0.0346
40	0.0126	0.0446	0.0285	0.0561	0.0577	0.0304	0.0355
50	0.0198	0.0178	0.0308	0.0201	0.0237	0.0491	0.0311
60	0.0094	0.0272	0.0188	0.0057	0.0152	0.0063	0.0266
70	0.0045	0.0381	0.0258	0.0017	0.0128	0.0012	0.0018
80	0.0054	0.0362	0.0212	0.0012	0.0282	0.0041	0.0058
90	0.0012	0.0284	0.0078	0.0021	0.0180	0.0012	0.0017
100	0	0.0017	0.0186	0.0027	0.0393	0.0040	0.0012
Overall	0.0546	0.0482	0.0422	0.0430	0.0428	0.0386	0.0342

Panel B: Period Volume
75 Rebalancers, 75 ORPIs and 10 Speculator Investors

	Number of Arbitrager Investors						
Quarter	0	1	2	5	10	25	50
1	93152	97604	108626	97360	88159	93320	90547
2	78684	65854	73900	74084	72530	66746	76064
3	65143	72721	68752	70472	64074	64766	70439
4	53782	62412	56288	57383	68348	50216	76896
5	53571	52322	43575	56773	53714	50289	61178
10	14010	13647	11643	27585	22560	30384	23687
20	1546	4168	2139	10535	8790	11891	9196
30	369	840	1149	2648	3252	4983	4225
40	387	1678	830	1564	1795	2756	2163
50	461	461	1078	380	337	2016	918
60	275	667	983	202	853	509	410
70	200	860	535	32	314	8	50
80	207	614	341	42	380	99	301
90	75	297	257	107	293	78	63
100	0	55	261	95	661	47	46
Overall	564290	611824	561184	803597	778228	878810	776095

TABLE 18
Simulation Runs with Various Numbers
of Arbitrager Investors

Panel A: Period Standard Deviation
50 Rebalancers, 100 ORPIs and 10 Speculator Investors

	Number of Arbitager Investors						
Quarter	0	1	2	5	10	25	50
1	0.2131	0.1170	0.1977	0.2028	0.1852	0.1952	0.1731
2	0.1280	0.1267	0.1082	0.1036	0.1273	0.0880	0.0833
3	0.1338	0.1219	0.1125	0.1505	0.0739	0.0774	0.0767
4	0.1227	0.1323	0.1171	0.1117	0.0815	0.0795	0.0795
5	0.1492	0.1418	0.1033	0.1126	0.0825	0.0856	0.0876
10	0.0857	0.0937	0.0783	0.0674	0.0651	0.0715	0.0649
20	0.0411	0.0639	0.0596	0.0632	0.0400	0.0441	0.0365
30	0.0313	0.0018	0.0342	0.0503	0.0224	0.0329	0.0325
40	0.0438	0	0.0293	0.0211	0.0497	0.0251	0.0246
50	0.0232	0	0.0265	0.0122	0.0316	0.0235	0.0223
60	0.0252	0.0080	0	0.0267	0.0093	0.0441	0.0138
70	0.0098	0.0030	0	0.0226	0.0054	0.0305	0.0155
80	0.0198	0.0054	0.0012	0.0288	0.0039	0.0252	0.0018
90	0.0183	0	0	0.0069	0.0064	0.0401	0
100	0.0256	0.0043	0	0.0203	0	0.0205	0
Overall	0.0513	0.0464	0.0506	0.0507	0.0430	0.0462	0.0412

Panel B: Period Volume
50 Rebalancers, 100 ORPIs and 10 Speculator Investors

	Number of Arbitrager Investors						
Quarter	0	1	2	5	10	25	50
1	107943	100414	99888	102290	102882	100956	103232
2	82968	83230	86187	82236	85748	85658	83097
3	83549	77339	73679	85198	73935	74967	81135
4	65408	69831	62965	62962	61905	86745	68933
5	62903	67503	61298	62282	64249	63125	70069
10	16085	16372	20918	17437	28102	40882	32145
20	779	2129	3157	7843	10847	7646	7909
30	724	54	1066	997	2751	3573	3838
40	1487	0	973	1187	1763	1470	2043
50	195	133	338	1979	924	836	979
60	368	324	0	837	413	932	439
70	163	174	0	586	139	448	607
80	302	102	37	282	233	441	56
90	427	0	0	160	165	728	0
100	163	307	0	171	0	444	0
Overall	620976	623875	655322	772198	904466	957762	941952

TABLE 19
Simulation Runs with Various Numbers
of Arbitrager Investors

Panel A: Period Standard Deviation
25 Rebalancers, 125 ORPIs and 10 Speculator Investors

	Number of Arbitager Investors						
Quarter	0	1	2	5	10	25	50
1	0.2567	0.4598	0.2331	0.7956	0.2317	0.2440	0.3300
2	0.1396	0.1139	0.1888	0.1150	0.1153	0.1091	0.0943
3	0.1308	0.1397	0.1267	0.1260	0.0877	0.0839	0.0840
4	0.1460	0.1812	0.1129	0.0997	0.1067	0.0849	0.0935
5	0.1680	0.1383	0.1121	0.1161	0.1070	0.1047	0.0897
10	0.0878	0.0865	0.0874	0.0821	0.0929	0.0805	0.0606
20	0.0601	0.0701	0.0449	0.0366	0.0475	0.0329	0.0436
30	0.0451	0.0130	0.0198	0.0337	0.0446	0.0306	0.0376
40	0	0.0235	0	0.0253	0.0388	0.0512	0.0347
50	0.0042	0.0321	0.0012	0.0340	0.0012	0.0492	0.0265
60	0.0012	0.0295	0.0037	0.0171	0.0051	0.0267	0.0230
70	0	0.0193	0.0037	0	0.0116	0.0046	0.0230
80	0.0012	0.0017	0	0.0090	0.0110	0	0
90	0	0.0363	0	0	0.0012	0	0.0012
100	0	0.0467	0.0012	0	0	0.0025	0
Overall	0.0546	0.0704	0.0490	0.0896	0.0520	0.0469	0.0492

Panel B: Period Volume
25 Rebalancers, 125 ORPIs and 10 Speculator Investors

	Number of Arbitrager Investors						
Quarter	0	1	2	5	10	25	50
1	114886	113997	112009	115193	117395	120888	119441
2	86800	86616	94279	78428	96077	91257	94363
3	76702	66413	77488	71962	83297	72863	91056
4	73341	79870	73076	65068	72300	70821	74094
5	74521	60855	62218	57272	70988	76696	78340
10	10978	13354	16177	17976	26523	25794	28032
20	3080	1785	723	4203	6201	7029	6641
30	103	400	0	1381	3926	2950	4522
40	0	665	0	663	691	2463	5393
50	98	575	62	408	80	854	2431
60	37	882	77	139	545	504	643
70	0	285	77	0	696	196	85
80	48	248	0	245	837	0	0
90	0	419	0	0	318	0	94
100	0	588	101	0	0	69	0
Overall	610767	642163	650983	678394	896817	853503	916976

TABLE 20
Simulation Runs with Various Numbers
of Arbitrager Investors

Panel A: Period Standard Deviation
5 Rebalancers, 145 ORPIs and 10 Speculator Investors

	Number of Arbitager Investors						
Quarter	0	1	2	5	10	25	50
1	0.4695	0.4412	0.5451	3.1109	0.3565	0.6322	0.3590
2	0.2425	0.1935	0.2016	0.1323	0.1114	0.1340	0.1380
3	0.1421	0.1800	0.0995	0.1913	0.1334	0.1123	0.0962
4	0.1605	0.1320	0.0685	0.1626	0.1096	0.1102	0.0953
5	0.1408	0.1693	0.1168	0.1023	0.2618	0.1454	0.1240
10	0.1129	0.0859	0.0867	0.0559	0.1107	0.0716	0.0669
20	0.0293	0.0499	0.0258	0.0801	0.0971	0.0478	0.0355
30	0.0321	0.0018	0.0109	0.0533	0	0.0214	0.0308
40	0.0317	0	0.0021	0.0368	0.0041	0.0304	0.0163
50	0.0432	0	0	0	0	0.0247	0.0498
60	0	0	0	0	0	0.0231	0.0672
70	0	0	0	0	0	0.0221	0.0859
80	0	0	0	0	0.0018	0.0242	0
90	0	0	0	0	0.0041	0.0201	0.0012
100	0	0.0017	0	0	0.0058	0.0215	0.0039
Overall	0.0718	0.0685	0.0714	0.3150	0.0658	0.0552	0.0570

Panel B: Period Volume
5 Rebalancers, 145 ORPIs and 10 Speculator Investors

	Number of Arbitrager Investors						
Quarter	0	1	2	5	10	25	50
1	88626	99831	83790	83534	101941	122721	118847
2	71406	82419	63131	73622	94479	97998	98077
3	41239	66294	49930	45730	58090	88135	68912
4	70351	51311	49029	61140	69213	68988	72199
5	31528	46150	47318	42000	63809	74622	71926
10	3555	3868	13881	8946	10438	28093	25393
20	608	341	6689	1554	985	9196	6917
30	426	15	1186	1760	0	5102	3604
40	457	0	36	327	607	5143	2015
50	305	0	0	0	0	3362	475
60	0	0	0	0	0	3724	1857
70	0	0	0	0	0	2502	1101
80	0	0	0	0	370	2740	0
90	0	0	0	0	196	2257	26
100	0	130	0	0	594	2040	273
Overall	374269	460452	481726	469735	548841	1072470	878197

TABLE 21
Simulation Runs with Various Numbers
of Arbitrager Investors

Panel A: Period Standard Deviation
0 Rebalancers, 150 ORPIs and 10 Speculator Investors

	Number of Arbitager Investors						
Quarter	0	1	2	5	10	25	50
1	0.4405	0.5521	1.1188	1.9025	0.4796	0.4192	0.3882
2	0.5213	0.4562	2.0496	0.3309	0.1336	0.1832	0.1635
3	0.3340	0.2787	25.3807	0.2224	5.5060	0.1026	0.1484
4	1537.4515	0.0907	0.1710	0.1548	0.2486	0.1025	0.1344
5	0.2600	0.0847	413.275	0.2682	0.4682	0.1091	0.1283
10	0.1168	0.1297	0.0108	0.0472	0.1564	4.7694	2.8571
20	0.1050	0.0544	0	0.0711	1.2721	1.1684	0.0539
30	0	0	0	0.1227	1.4332	0	0.0244
40	0.1143	0	0	0	0	0	0.0219
50	0	0	0	0.0143	0	0	0
60	0	0	0	0.1186	0	0	0
70	0	0	0	0.2751	0	0	0
80	0	0	0	0.0958	0	0	0
90	0	0	0	0.0565	0	0	0
100	0	0	0	0	0	0	0
Overall	157.25	0.6451	41.8285	2.6911	11.3143	1.9247	0.3004

Panel B: Period Volume
0 Rebalancers, 150 ORPIs and 10 Speculator Investors

	Number of Arbitrager Investors						
Quarter	0	1	2	5	10	25	50
1	72095	54022	60596	81400	89568	100798	125200
2	3246	27694	5323	44244	70541	67582	83596
3	13559	28301	23488	19511	46353	34543	58981
4	42	30148	5265	46398	99323	44336	89338
5	32753	60298	7856	27429	43322	45622	97272
10	3319	36788	33	35271	46112	39926	69969
20	9847	7588	0	9214	650	12299	12525
30	0	0	0	3	14	0	5278
40	126	0	0	6	0	0	2692
50	0	0	0	3	0	0	0
60	0	0	0	4	0	0	0
70	0	0	0	1	0	0	0
80	0	0	0	4	0	0	0
90	0	0	0	1428	0	0	0
100	0	0	0	0	0	0	0
Overall	209705	562706	125463	486093	715248	643986	1351284

Figure 1: S&P 500 Data
Index Level and Trading Volumes

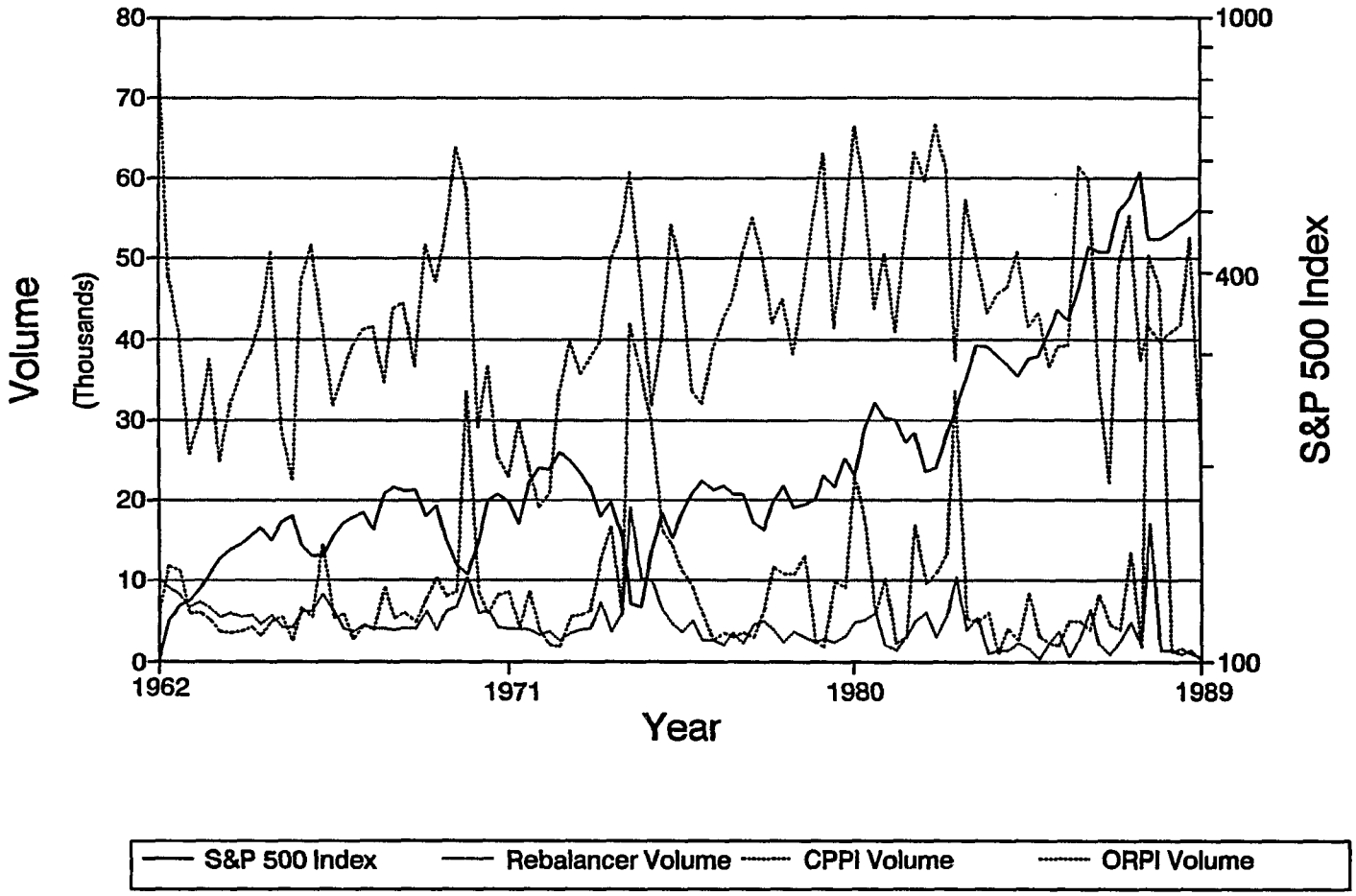


Figure 2: Portfolio Values
(Jan. 1962-Dec. 1988)

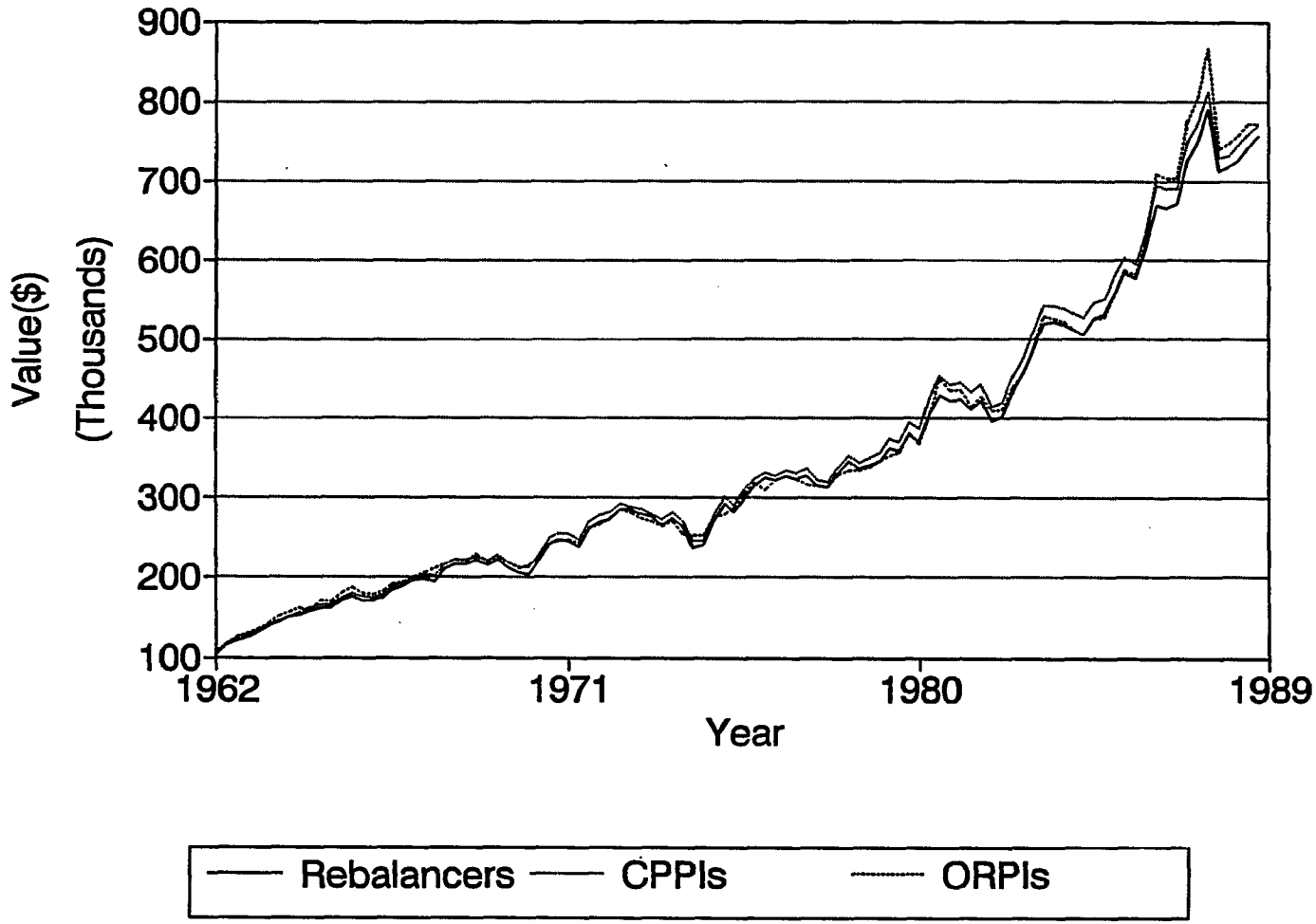


Figure 3:Quarterly Returns
(Jan. 1962-Dec. 1988)

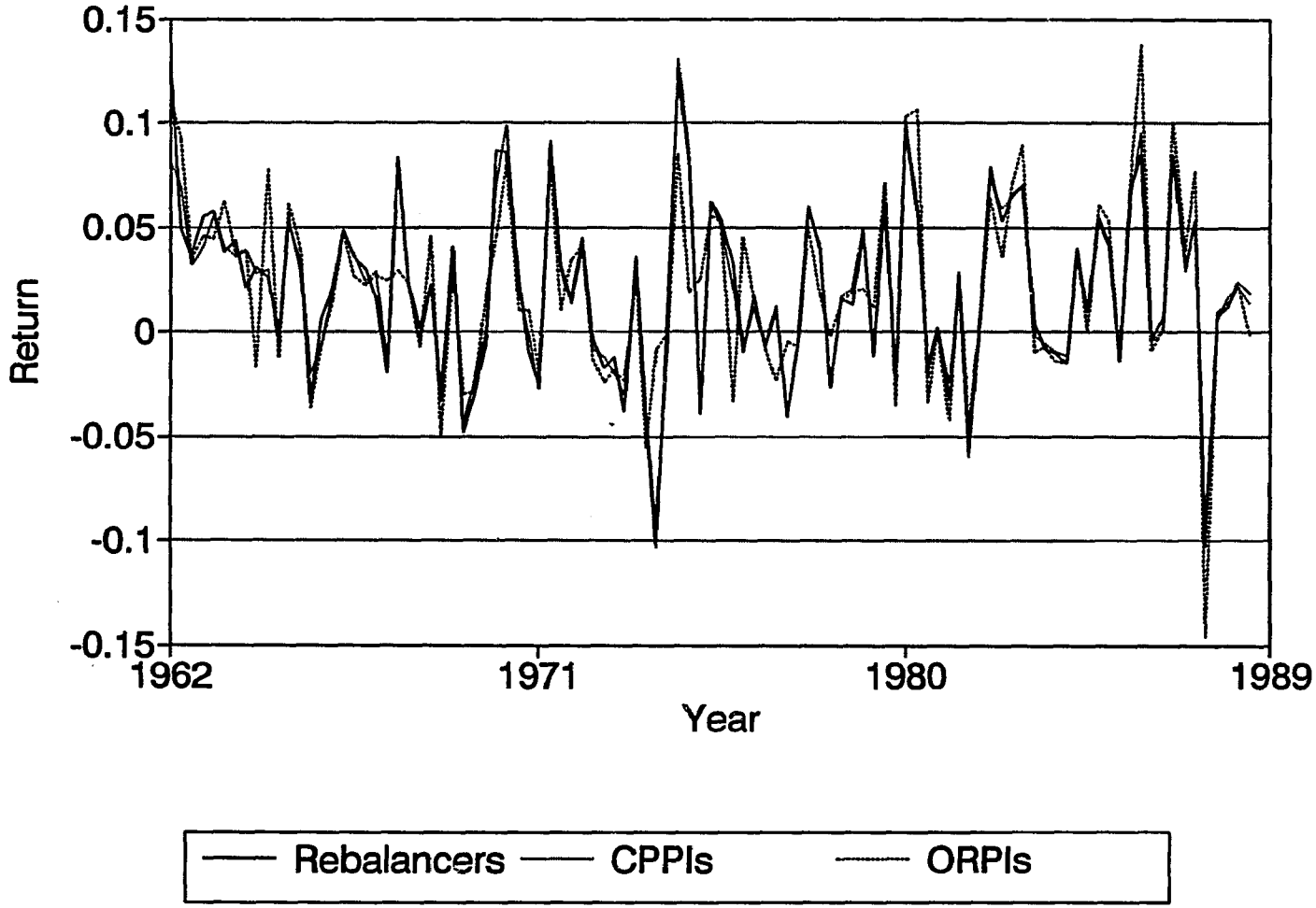


Figure 4: Period Rates of Return

150 Reb.s, 0 ORPIs and 10 Spec.s

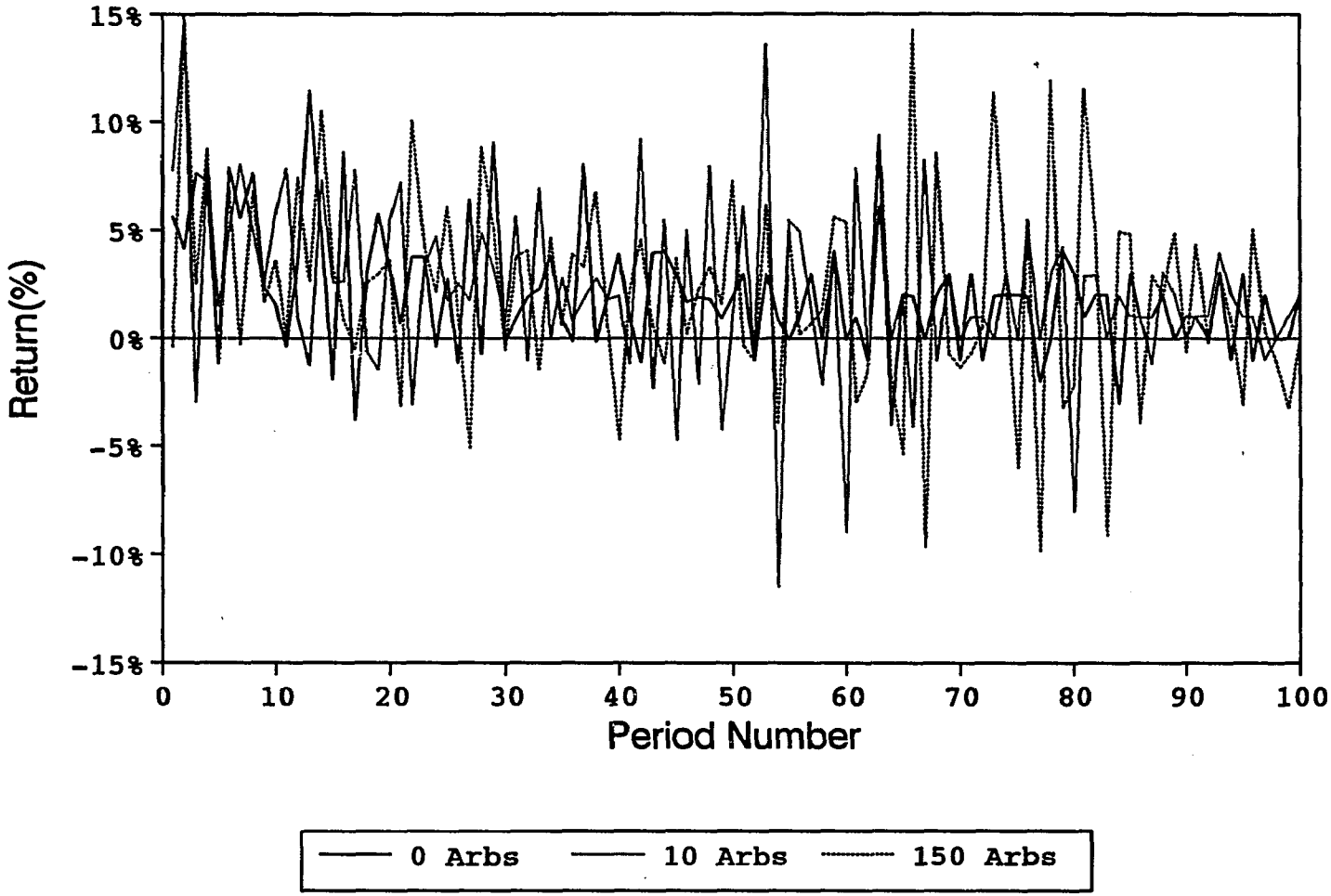


Figure 5: Period Rates of Return

145 Reb.s, 5 ORPIs and 10 Spec.s

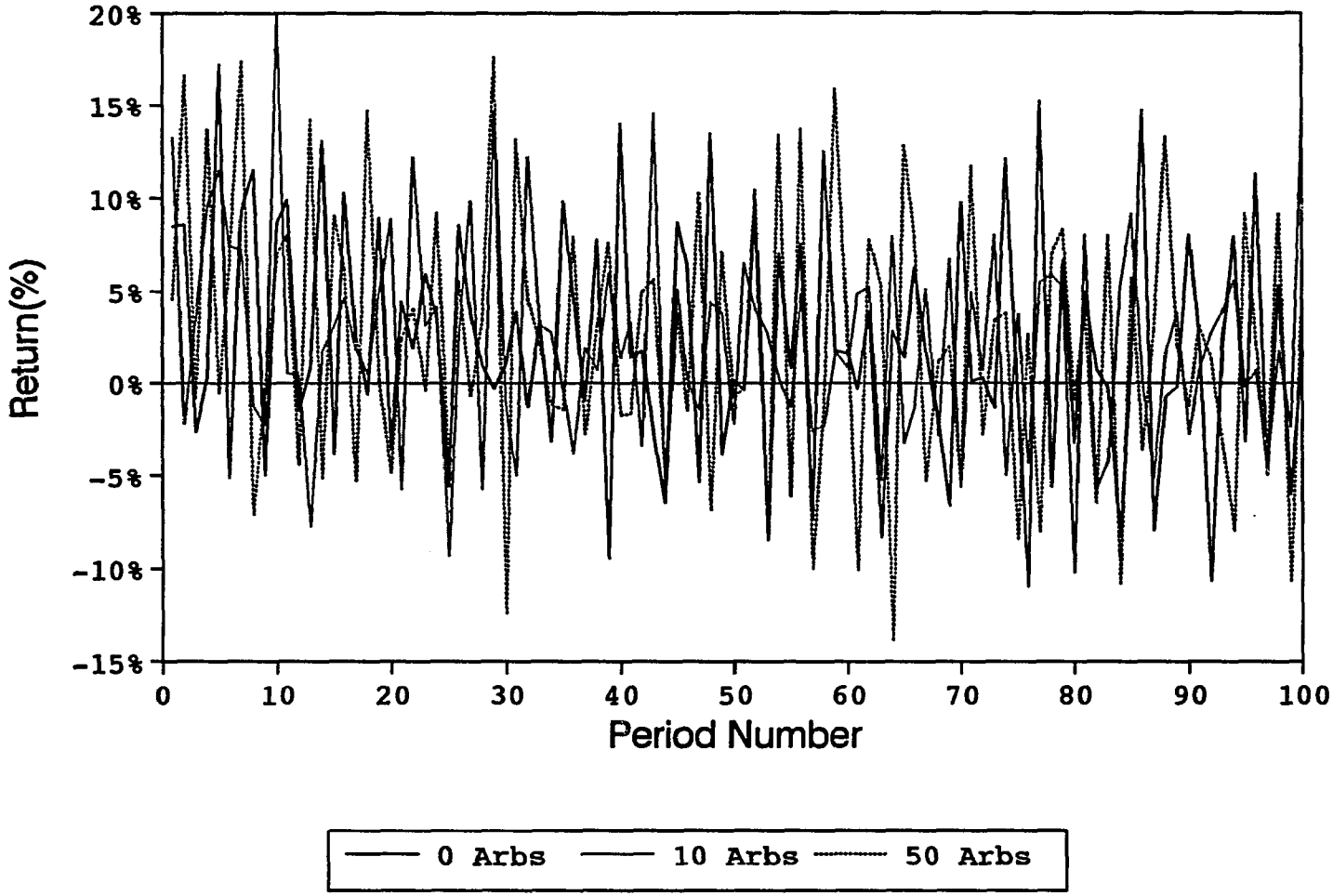


Figure 6: Period Rates of Return

140 Reb.s, 10 ORPIs and 10 Spec.s

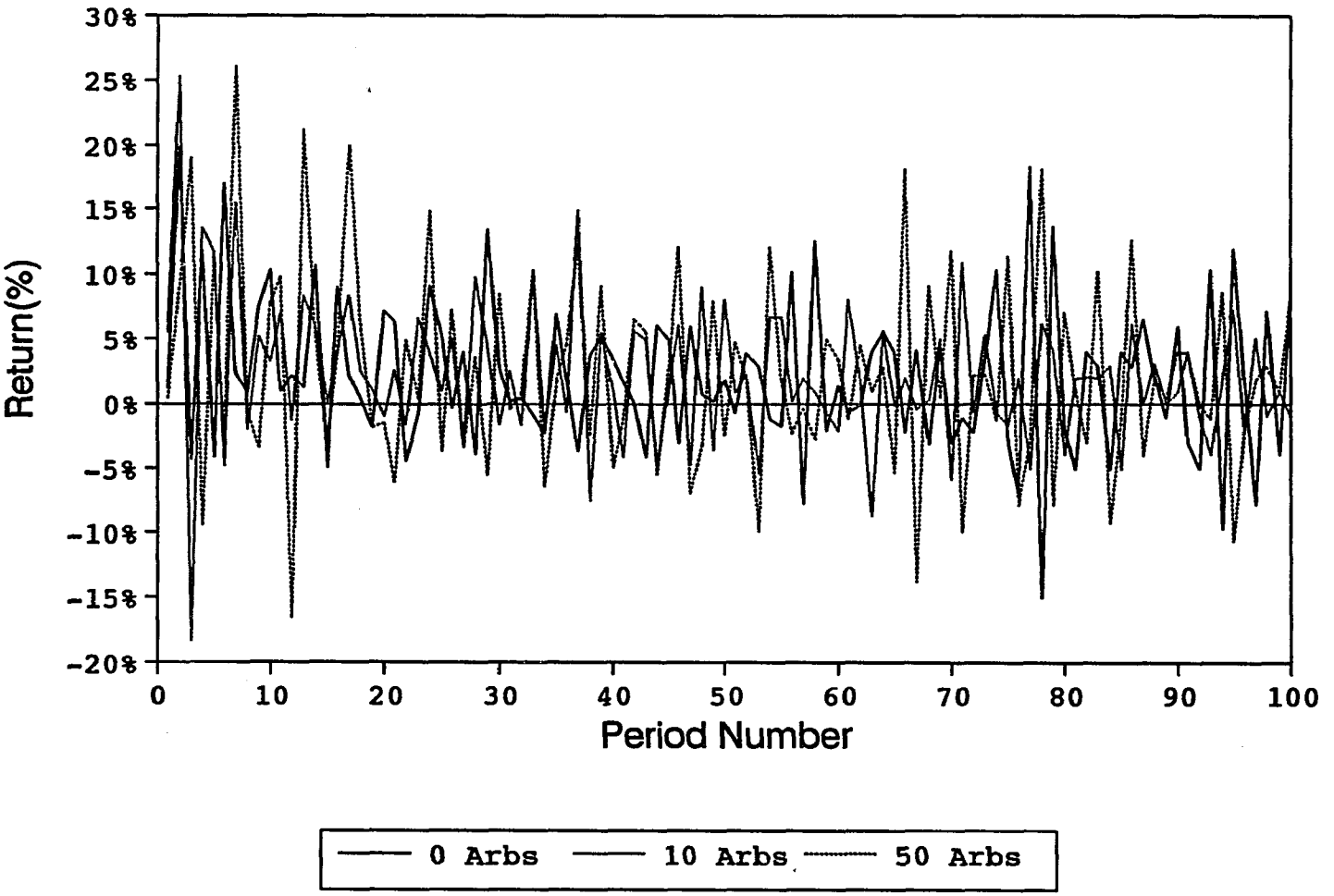


Figure 7: Period Rates of Return

125 Reb.s, 25 ORPIs and 10 Spec.s

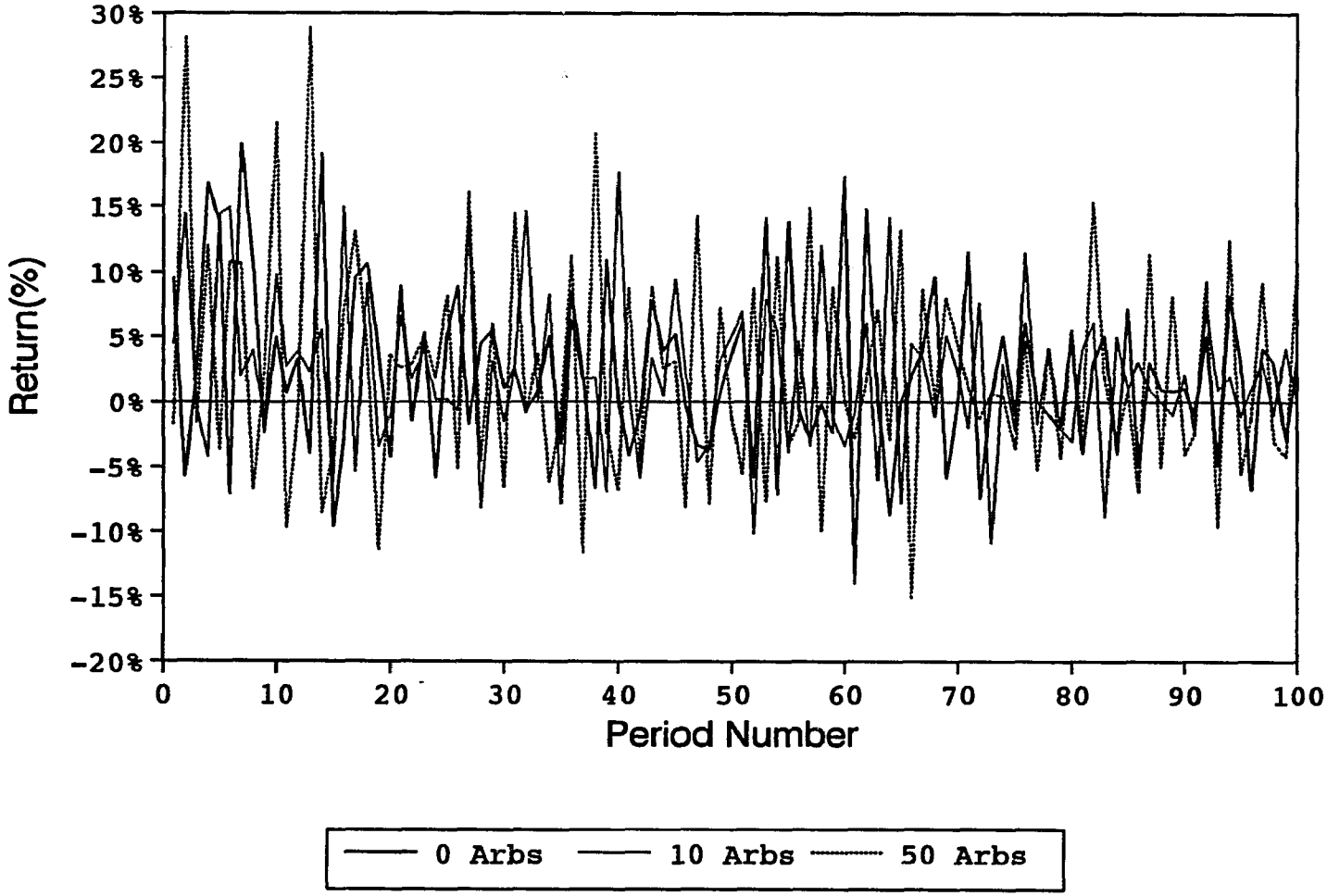


Figure 8: Period Rates of Return

100 Reb.s, 50 ORPIs and 10 Spec.s

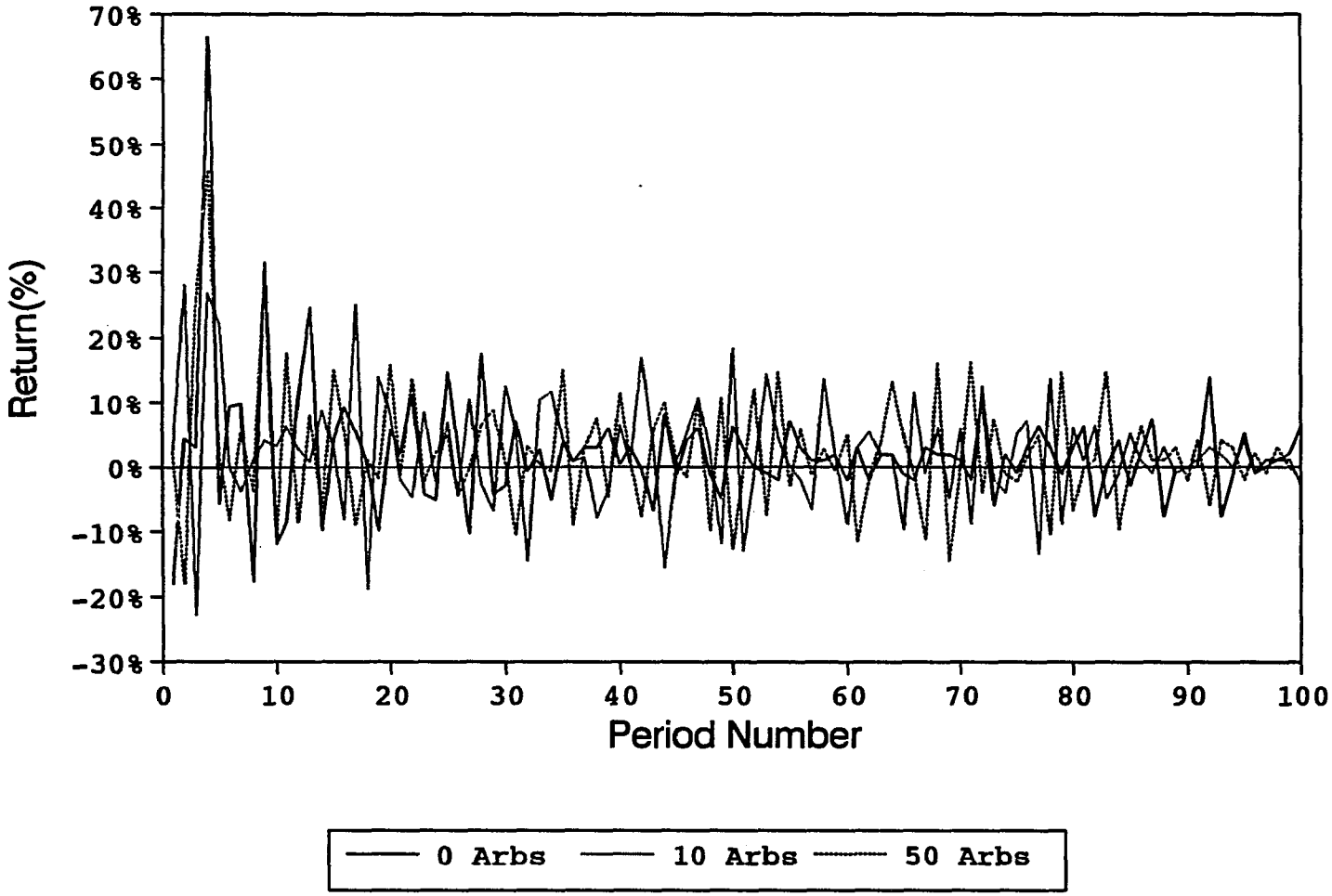


Figure 9: Period Rates of Return

75 Reb.s, 75 ORPIs and 10 Spec.s

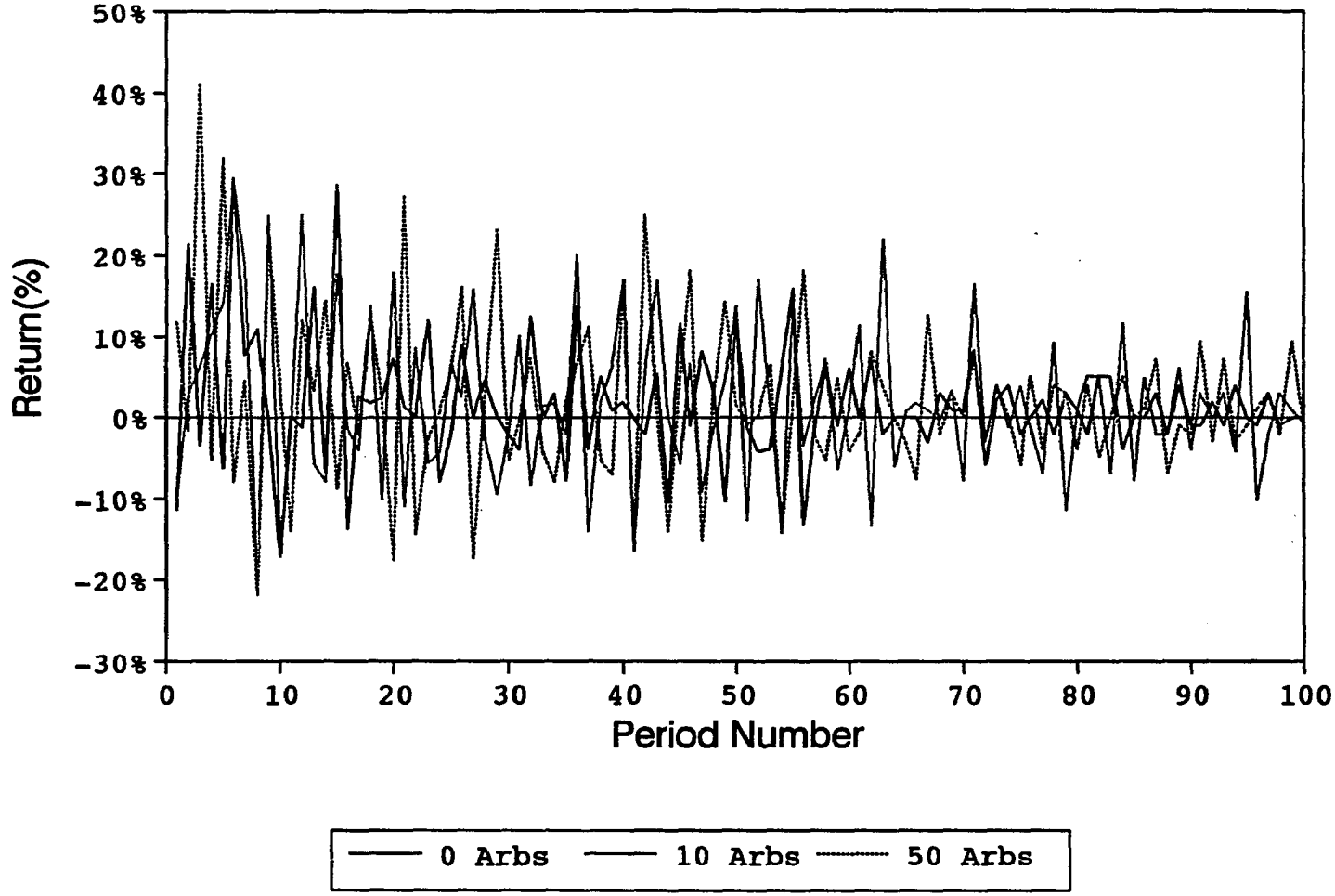


Figure 10: Period Rates of Return

50 Reb.s, 100 ORPIs and 10 Spec.s

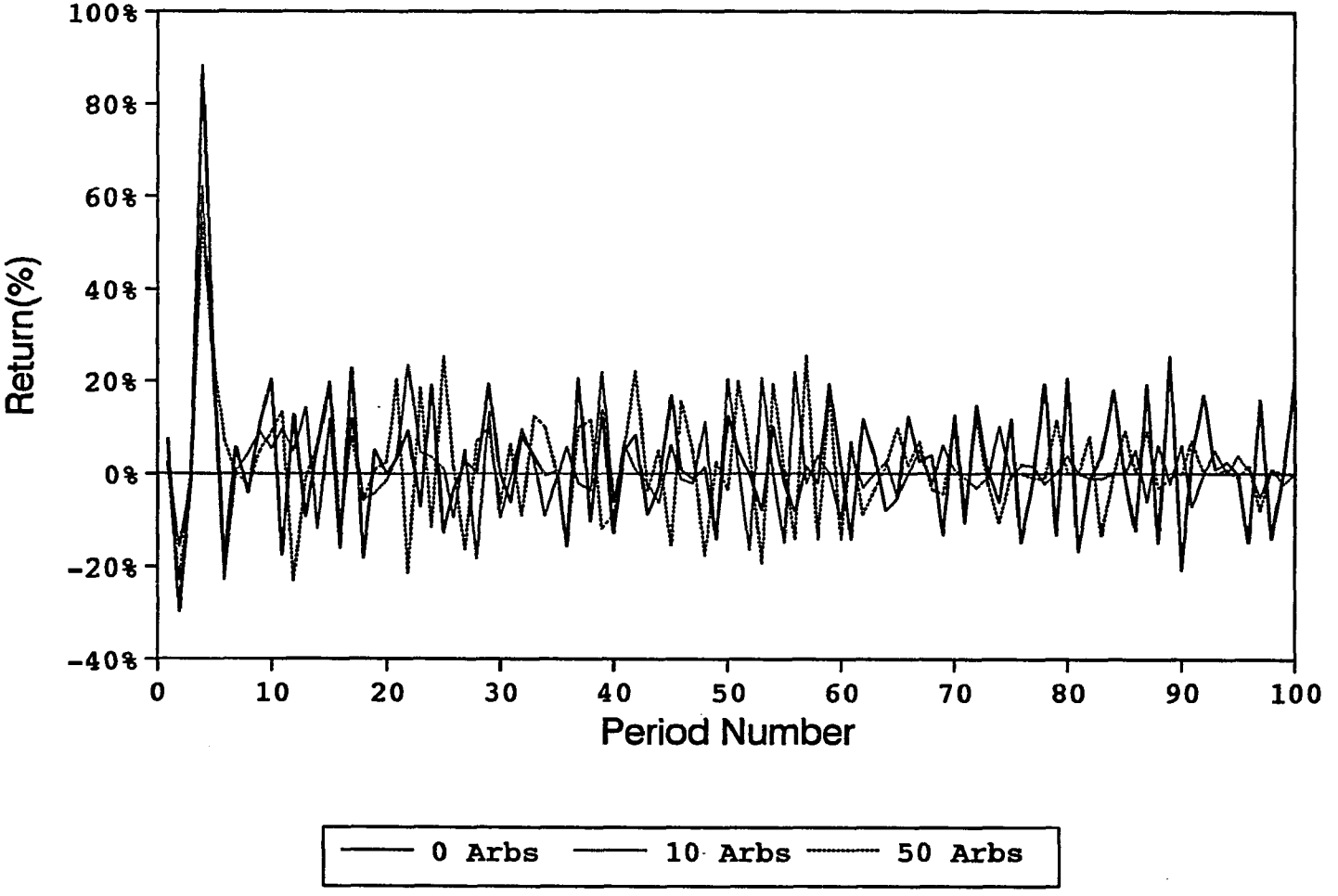


Figure 11: Period Rates of Return

25 Reb.s, 125 ORPIs and 10 Spec.s

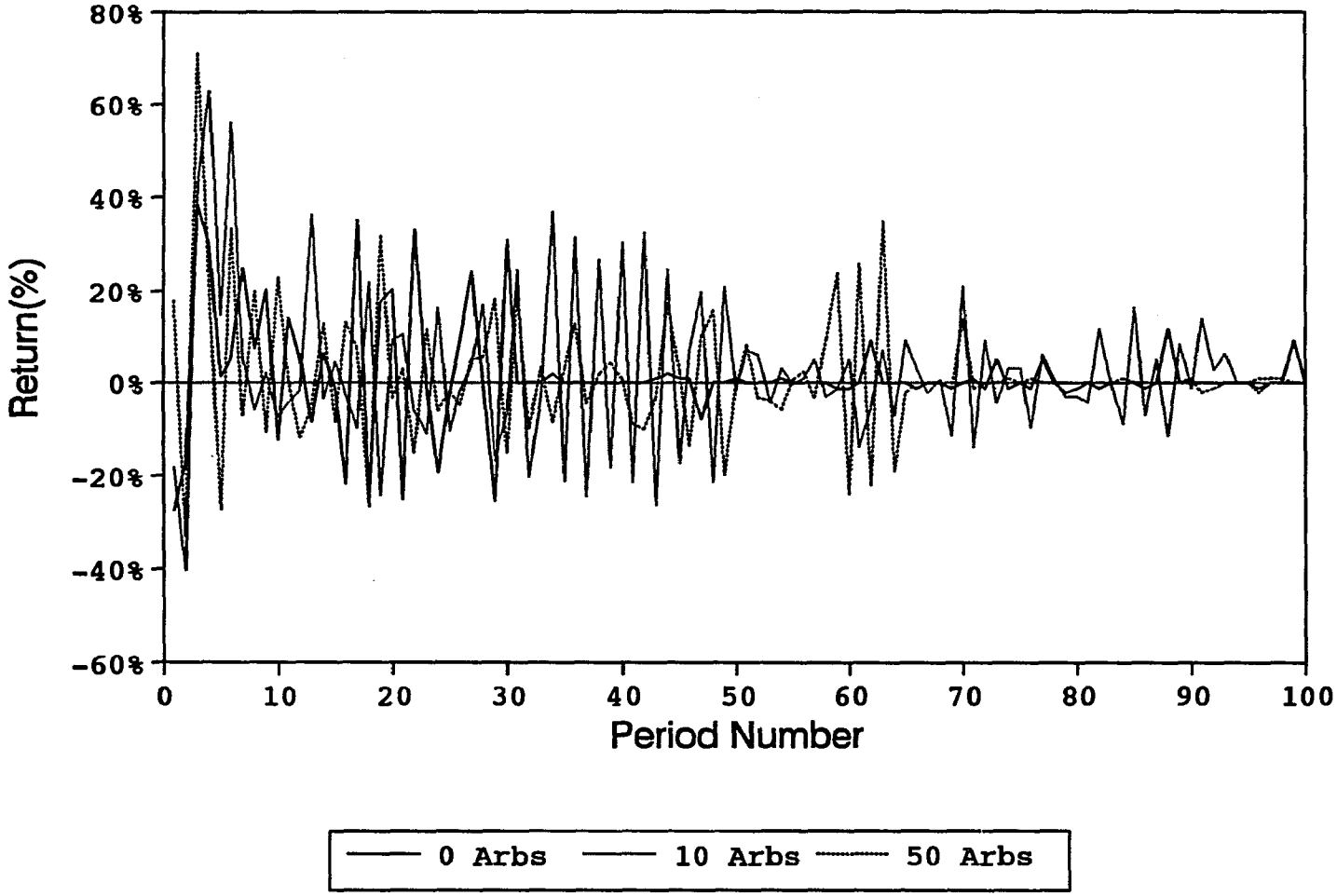


Figure 12: Period Rates of Return

5 Reb.s, 145 ORPIs and 10 Spec.s

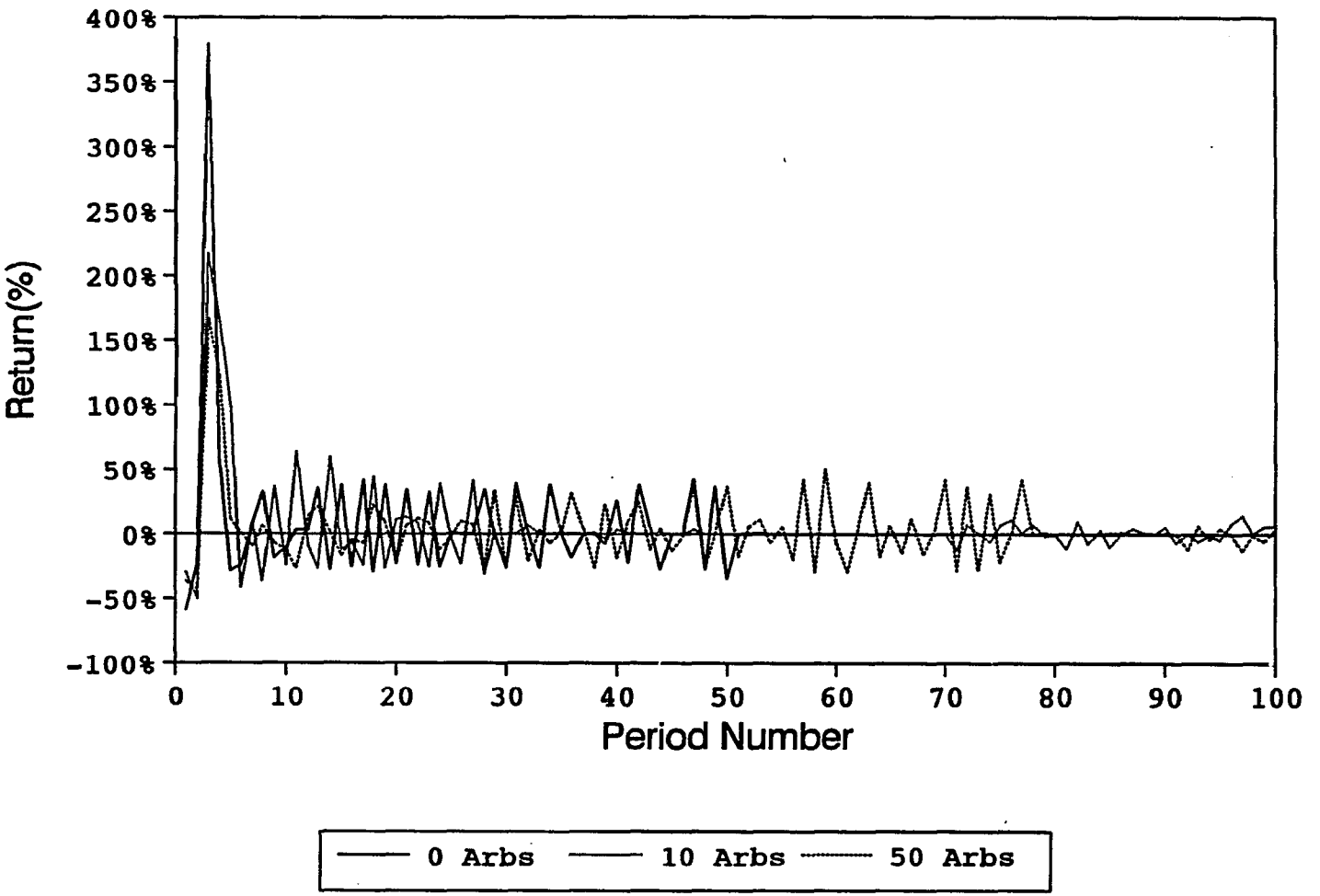


Figure 13: Period Rates of Return

0 Reb.s, 150 ORPIs and 10 Spec.s

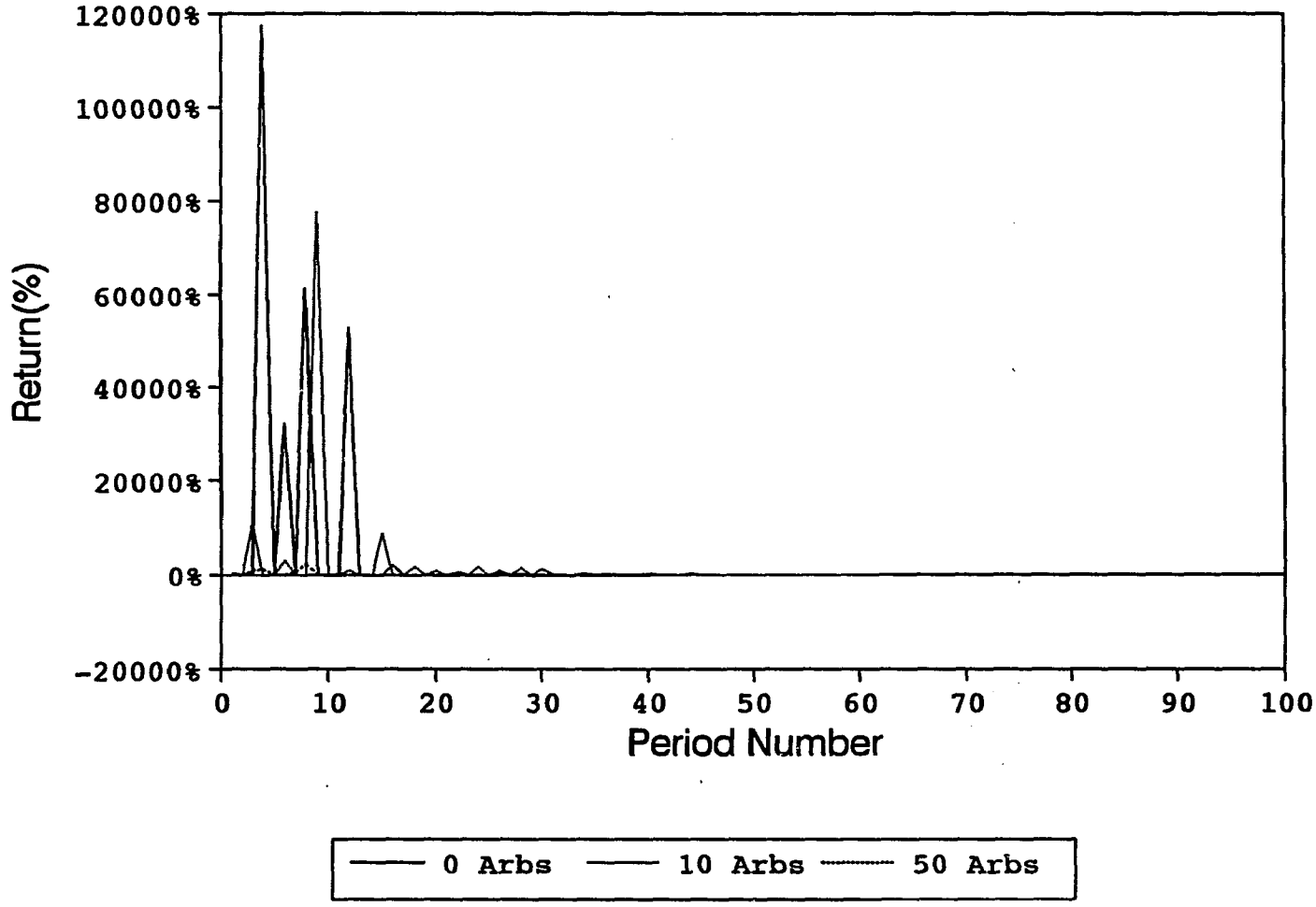


Figure 14: Portfolio Values
(125 Rebalancers and 25 ORPIs)

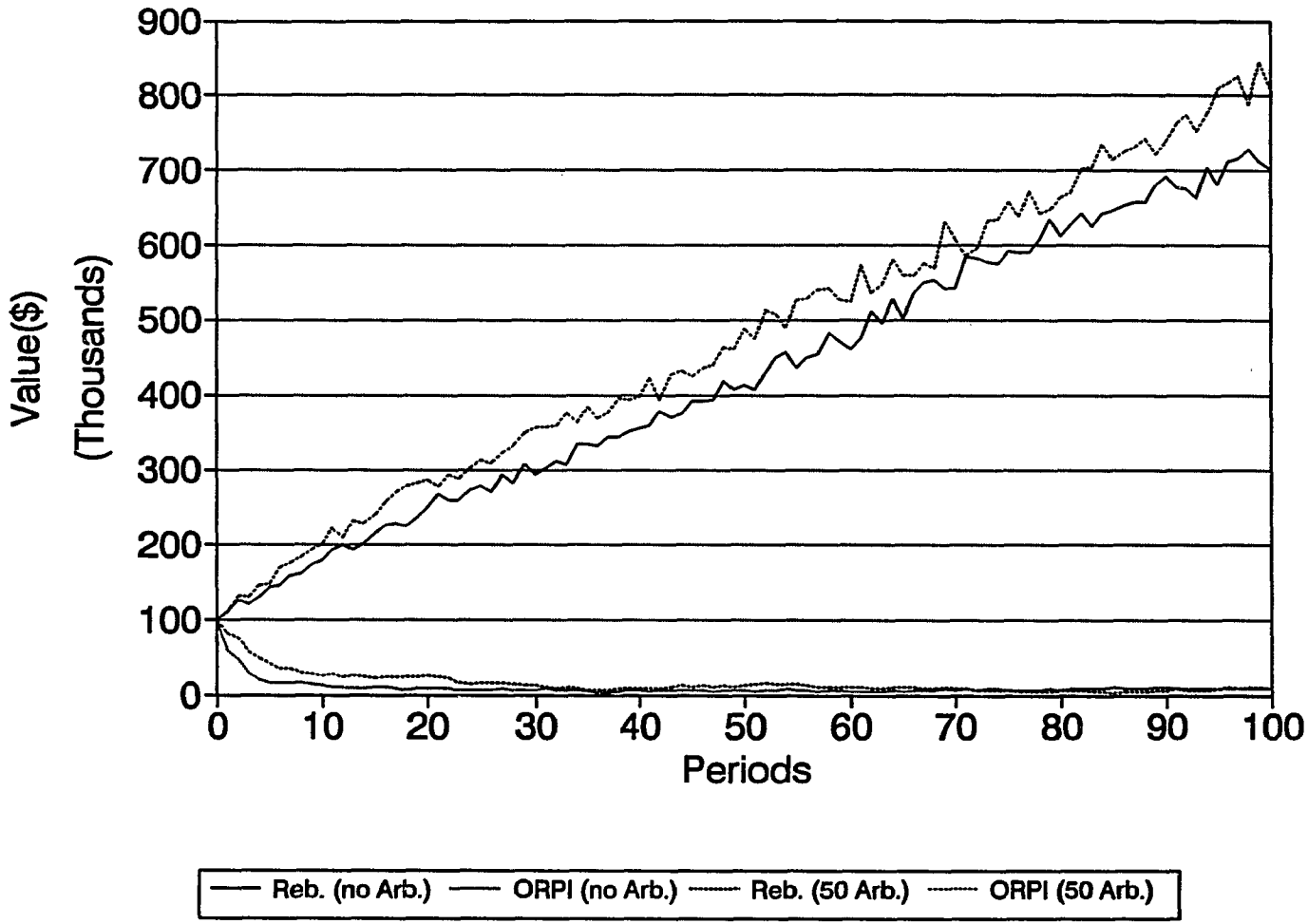


Figure 15: Portfolio Values
(75 Rebalancers and 75 ORPIs)

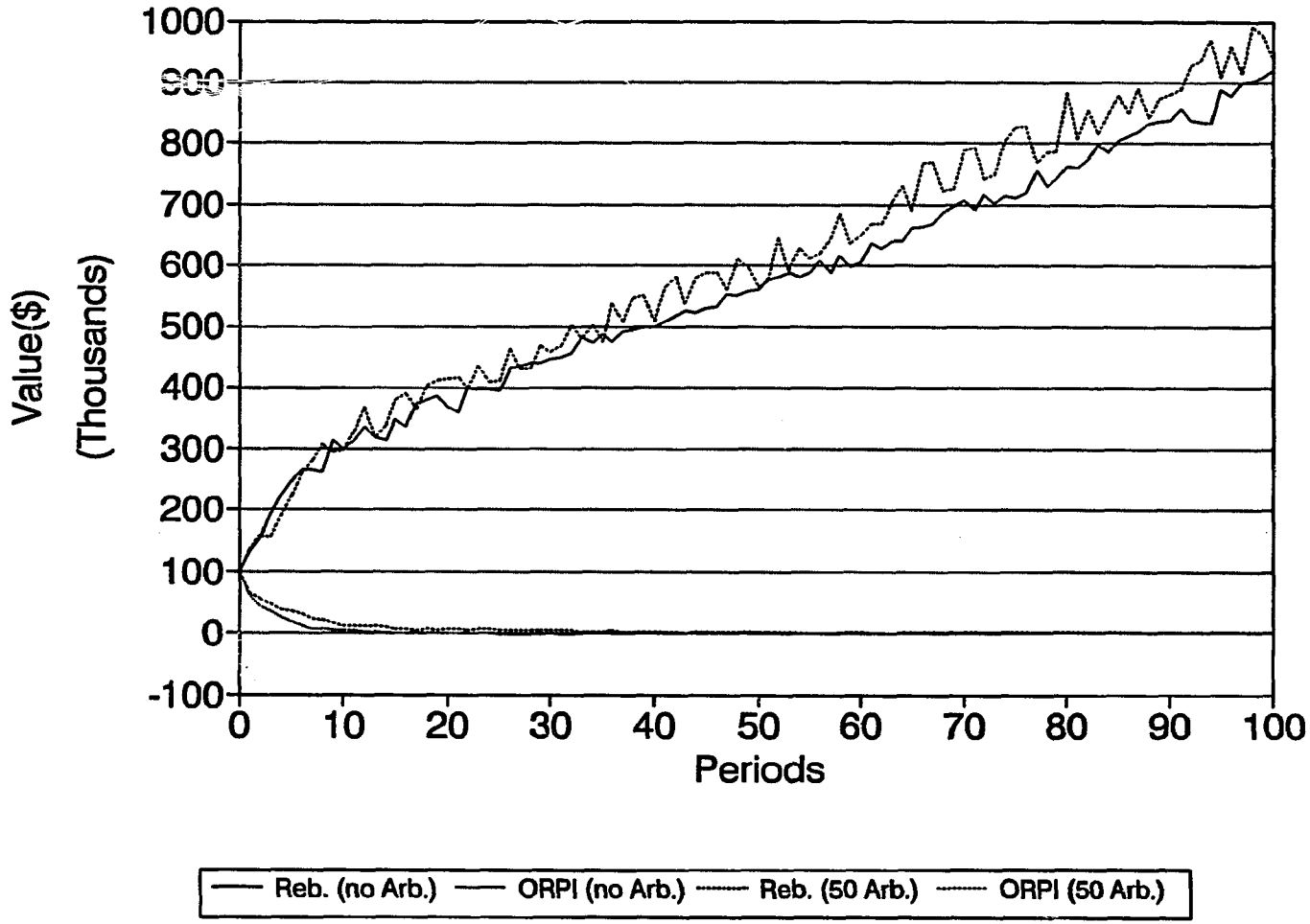


Figure 16: Portfolio Values
(25 Rebalancers and 125 ORPIs)

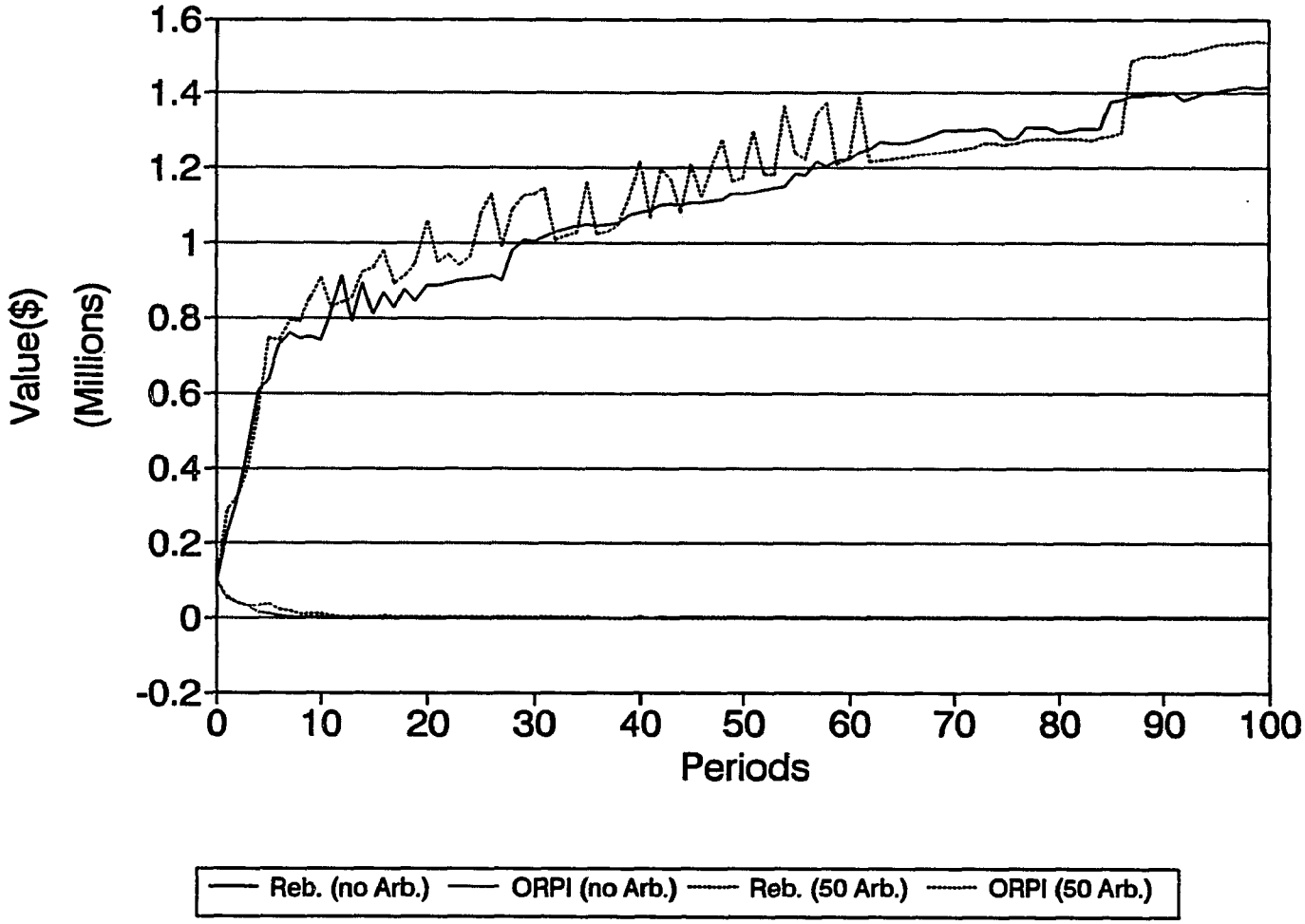


Figure 17: Portfolio Values
(Arbitragers and Speculators)

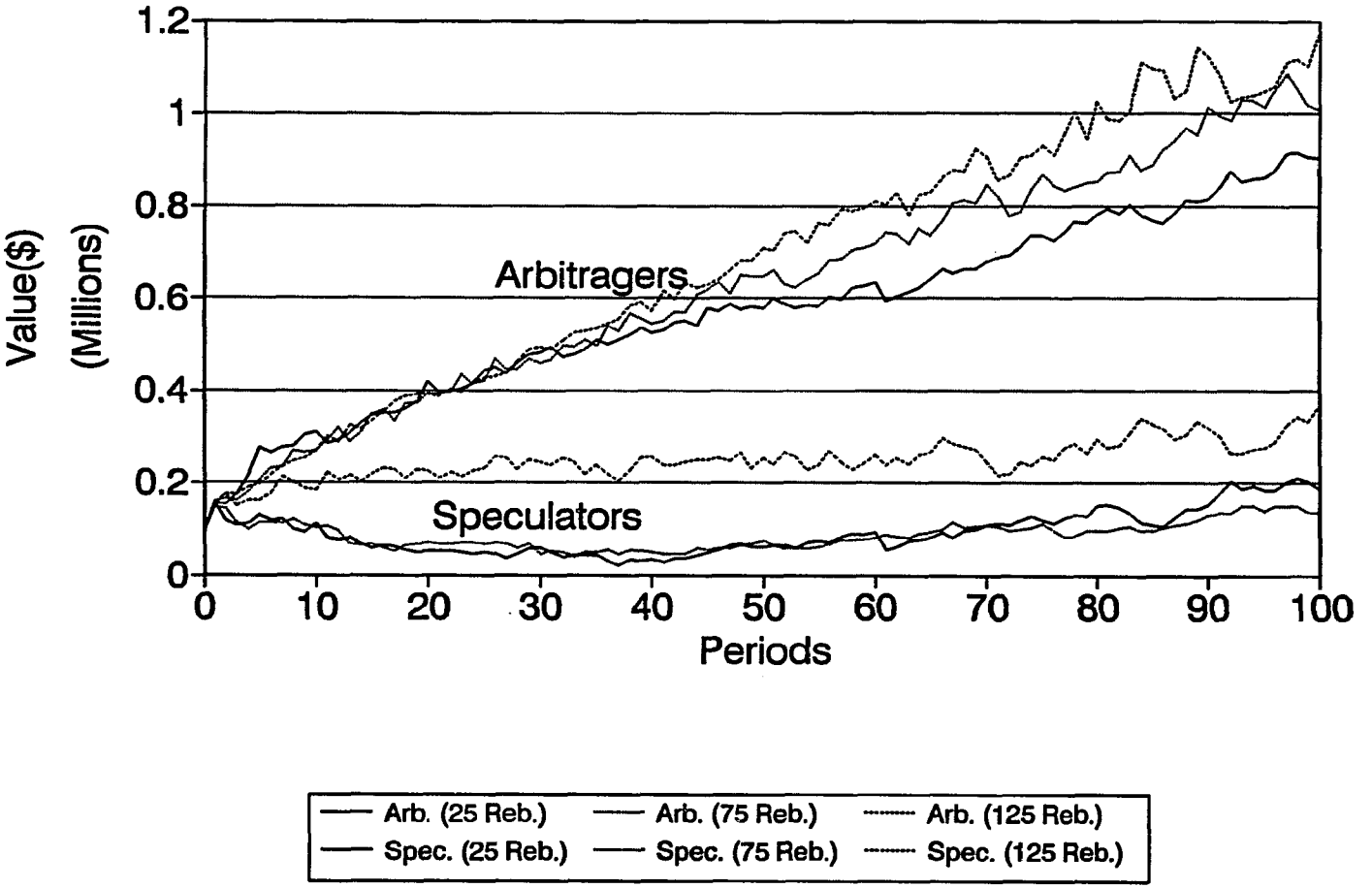


Figure 18: Volatility of Returns
(125 Reb. and 25 ORPI)

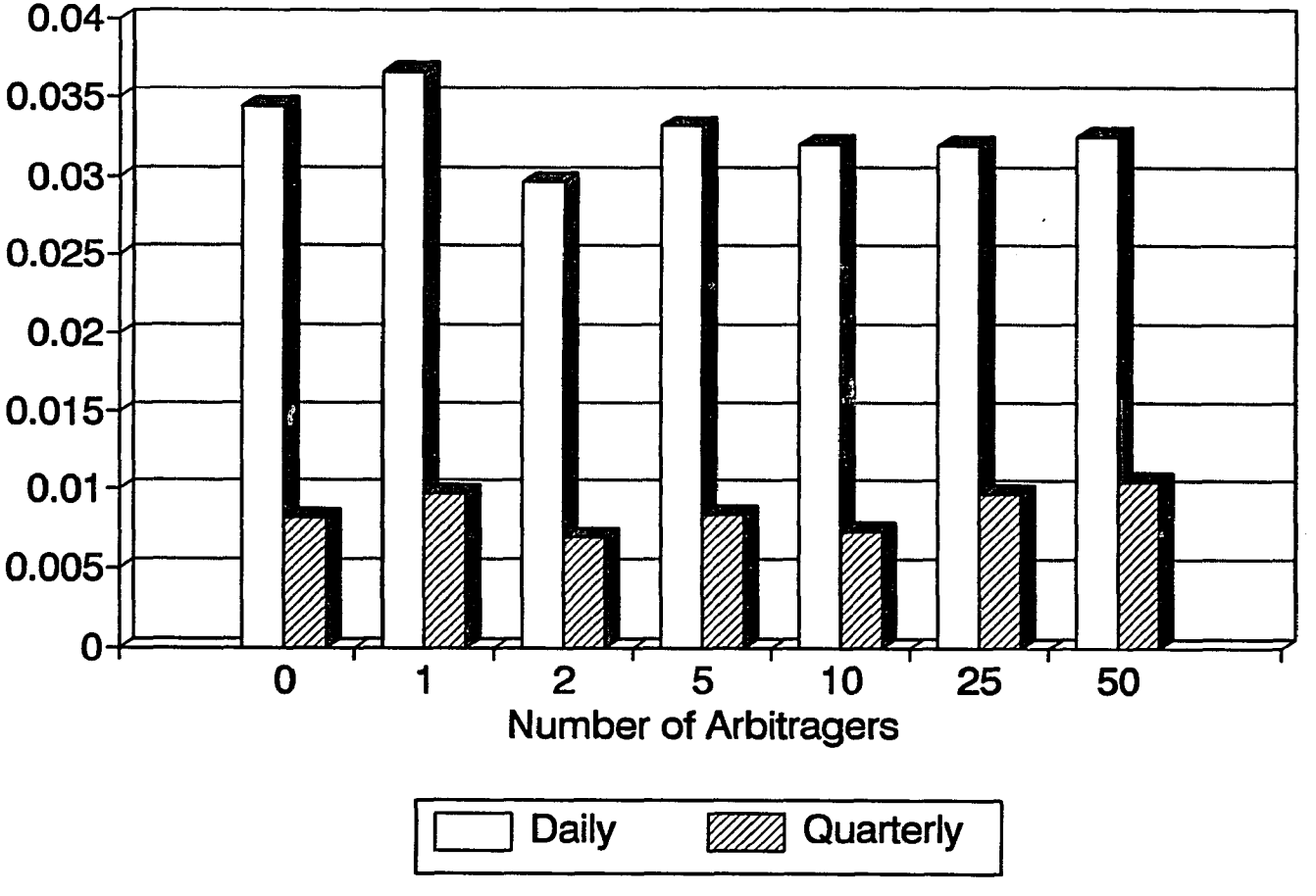


Figure 19: Volatility of Returns
(75 Reb. and 75 ORPI)

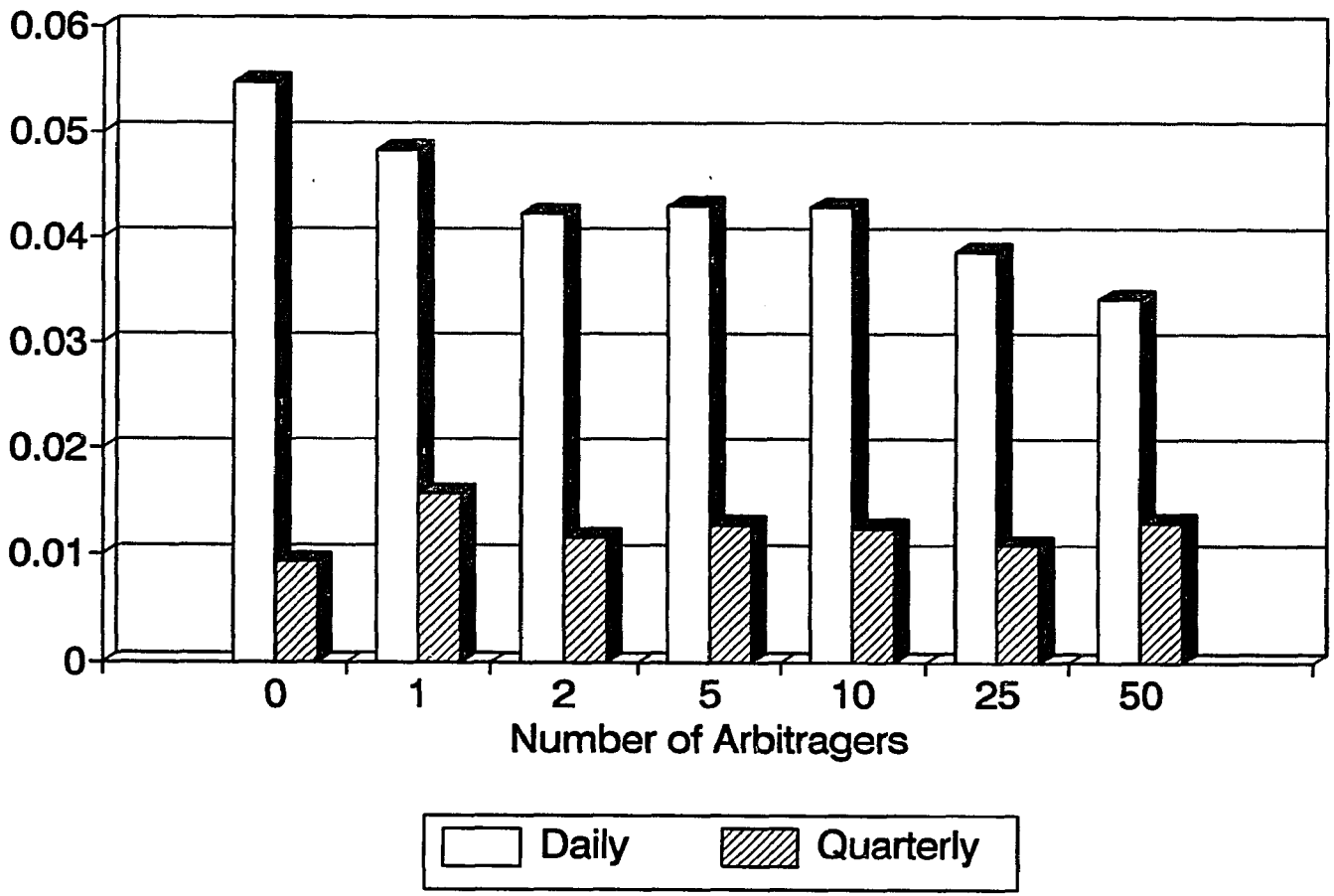
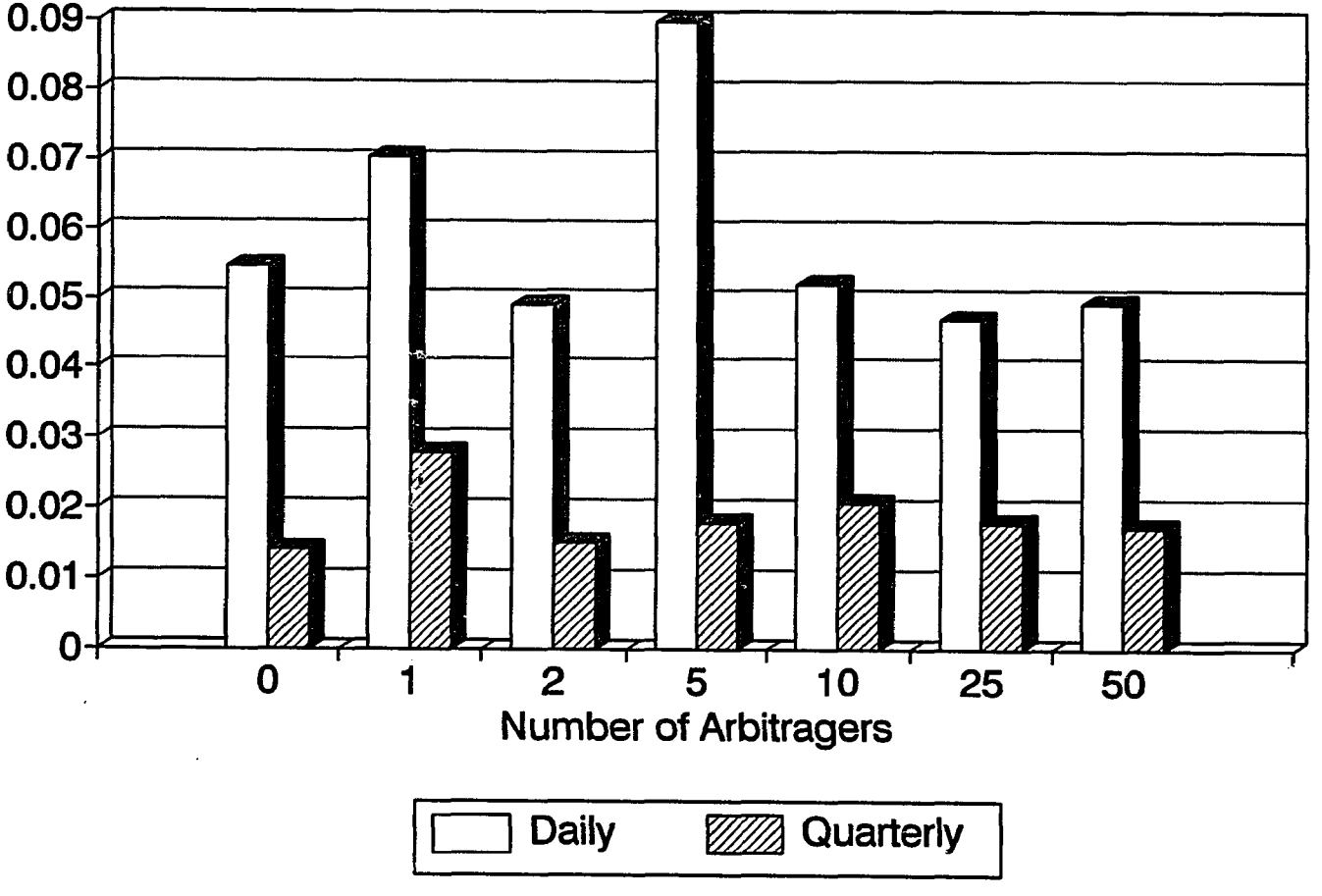


Figure 20: Volatility of Returns
(25 Reb. and 125 ORPI)



APPENDIX

This appendix consists of a brief discussion of how the market in the simulation model works, particularly the price determination mechanism and order execution, and then provides the three programs and datafiles used.

Price Determination and Order Execution

Why would an investor place an order? Assume all investors begin with 50 thousand cash and 50 thousand in stock. If an investor receives a cash inflow of 9 thousand, the proportion invested in stock is now 50 thousand divided by 109 thousand or 45.9%. If this investor is a rebalancer, he places an order since 45.9% is less than the lower bound of 46% (which can be chosen by the simulation user). Since the target ratio is 50% or 54.5 thousand, the investor must place a buy order for 4.5 thousand. The number of shares in the buy order would equal 4.5 thousand divided by the chosen bid price which will be described shortly.

Of course the stock to total portfolio value ratio is also affected by changing stock prices. Suppose a Rebalancer, beginning as above with half cash and half stock, reviews his portfolio after the stock price has fallen 20%. Thus, the stock portion is now 40 thousand divided by 90 thousand or 44.4% of the portfolio, again requiring a purchase. This time the purchase would be for 5 thousand dollars worth of stock.

A CPPI faced with the same situation above would act differently. Assume his parameters are as follows: Cushion is 0.25, target ratio is 2.0 (resulting in an initial position consisting of half cash and half stock), high ratio is 2.3 and low ratio is 1.7. Since the total portfolio is worth 90 thousand after a 20% drop in the stock price, and the initial cushion of 25 thousand leaves a floor of 75 thousand, the investor is left with a cushion of only 15 thousand (90 - 75). Combined with the high and low ratios, this yields an allowable range for the stock value of 25.5 to 34.5 thousand. Since the stock is worth 40 thousand, some must be sold since it is above the range. The target ratio yields a target stock amount of 30 thousand, thus 10 thousand dollars worth must be sold.

An ORPI facing the same situation as above would also want to sell. Assume the following parameters: Standard deviation is 20% per annum, insurance plans are one quarter of a year and adjustments are made to the portfolio only if the stock portion deviates more than 4% from the target ratio. After a 20% drop in stock price, his D1 is -2.1314 yielding a hedge ratio of 0.0165. Multiplying the hedge ratio by 90 thousand (which is the total value of the portfolio) yields a target stock amount of 1.485 thousand. This then requires a sale of 48.515 thousand dollars of stock.

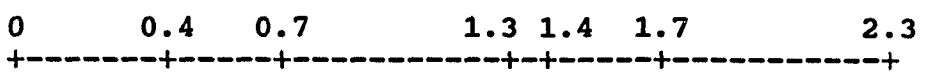
An Arbitrager trades based on the difference between the futures and the index. Assume that the futures contract

and Stock 2 remain at 100, but Stock 1 has gone down 20%. This causes the index to fall 10% so the arbitrager wants to buy stock and sell futures. Since the difference between the index and futures is 10%, the arbitrager trades 10% of his risk assets (set to 50% of total assets to yield half cash and half stock initially). Thus 10% of 50% is risked which is 5% of 100 thousand or 5 thousand. Buy orders of 2.5 thousand dollars for each stock are placed and a sell order for 5 thousand nominal value of futures is placed.

The Speculator has the simplest strategy. He receives signals of from 0 to 0.5 for each stock and then attempts to hold that proportion of his total assets in the stock. For example, if signals of 0.2 and 0.4 are received for Stocks 1 and 2 respectively, and assuming initial holdings of 25 thousand of each stock (totalling 50% of the whole portfolio), the first stock would be sold and the second bought. The 0.2 signal implies target holdings of 20 thousand and the 0.4 signal implies target holdings of 40 thousand. Thus, 5 thousand dollars of the first stock is sold and 15 thousand dollars of the second stock is bought.

In summary, orders may result from cash flows changing the relative amounts of cash to stock, stock price changes which change the relative amounts of cash to stock, privately received signals in the case of speculators, and finally, unexecuted orders requiring review which may result in order cancellation or revision.

All orders in the model consist of limit orders, thus there is no guarantee that the orders will be executed. Should an order remain unexecuted or only partially executed, it is placed in the order book until revised, cancelled or executed due to a subsequently placed order. Consider the timeline below:



At time 0, trading begins with the initial price set to 100. Suppose that between time 0 and time 0.4 investors have not yet found it necessary to trade. At time 0.4, investor 27 needs to buy 5 shares (shares must be bought and sold in whole units which may be thought of as round lots or even 10,000 share blocks). Since there have been no orders placed previously, the bid is set at 101 which is the last sale price of 100 times the bid.at.factor of 1.01. The order book would then look as follows:

Investor	Bid	Number	Investor	Ask	Number
27	101.00	5			

Now if at time 0.7 investor 53 needs to place a buy order, he sees a bid of 100 already in the book and thus estimates the current price to be 101 times the no.offers.est.factor (1.01) or 102.01. Then the bid is determined by taking the 102.01 and multiplying by the bid.at.factor (1.01) yielding 103.03. Assume that at this price, investor 53 needs 19 shares. The order book would then look as follows:

Investor Bid	Number	Investor Ask	Number
53 103.03	19		
27 101.00	5		

Note that since investor 53 has the highest bid, he is first in the list. Still there are no trades since there have not yet been any sell orders.

Now suppose that at time 1.3, investor 41 needs to place a sell order. He estimates the current price as 103.03 times 1.01 or 104.06. He then places his offer at 104.06 times the offer.at.factor (0.99) or 103.02. Assume that investor 41 needs 22 shares at this price. However, investor 41 realizes that investor 53 is willing to buy at 103.03 and thus offers the first 19 shares at that price with the remainder, 3 shares, at his reservation price of 103.02. Thus the order book would then look as follows:

Investor Bid	Number	Investor Ask	Number
27 101.00	5	41 103.02	3

At time 1.4, investor 27's buy order remains in the book. He first cancels the existing order and then determines if a new order is necessary. Suppose that he still needs to buy 5 shares. Since there is now a bid and an ask price, he averages the two to get a price estimate of 102.01. This is then multiplied by the bid.at.factor of 1.01 yielding a revised bid of 103.03. Since this is more than investor 41's ask price, investor 27 buys the 3 shares at 103.02, leaving an order for 2 shares at 103.03. Thus the order book would then look as follows:

Investor Bid	Number	Investor Ask	Number
27	103.03	3	

At time 1.7, investor 53 does not need to review his portfolio since his order was executed. Likewise, at time 2.3, investor 41 does no review. However, at time 2.4, investor 27 will again review his portfolio unless his order is executed prior to that time. Of course, this example has been greatly simplified. Time is actually broken into much smaller units than tenths of a day, price is not actually rounded to the nearest penny, rather it is continuous, and more traders would likely have placed orders.

Program Listings

KIM-MARKOWITZ WITH ORPIS

```
Preamble...

Normally mode is real.

The system has a last.sale.price, a pct.change.for.day, a sale.quantity,
  a report.interval, a remaining.report.periods, a period.number,
  a no.bids.price.est.factor, a no.offers.price.est.factor,
  owns the buyers, the sellers, the rebalancer.prototypes,
  the orpi.prototypes and the cppi.prototypes

Define sale.quantity and remaining.report.periods as integer variables

Tally daily.high as the daily max, daily.low as the daily min,
  period.high as the period max, period.low as the period min,
  grand.high as the max and grand.low as the min of last.sale.price

Tally daily.volume as the daily sum, period.volume as the
  period sum, grand.volume as the sum of sale.quantity

Tally max.ppc as the period max, max.gpc as the grand max,
  min.ppc as the period min,
  min.gpc as the grand min of pct.change.for.day

Tally pmpc as the period mean, gmpc as the grand mean,
  pspc as the period std.dev,
  gspc as the grand std.dev of pct.change.for.day

  Temporary entities....

Every rebalancer.prototype belongs to the rebalancer.prototypes
  and owns some investors,
  has a (prototype.number(1/2), pr.sequence.number(2/2)),
  a portfolio.review.interval, a dep.withdraw.interval,
  a trading.strategy.nr, an init.portfolio.value, an init.stock.fraction,
  a min.deposit, a max.deposit,
  a target.fraction, a high.fraction, and a low.fraction.
```

```
Define trading.strategy.nr as an integer variable  ''tr. str. pointer
Define prototype.number and pr.sequence.number as integer variables.

Define rebalancer.prototypes, cppi.prototypes, orpi.prototypes,
  and investors as fifo sets without p attributes,
  and without r, ff, fb, fa routines.

Every cppi.prototype belongs to the cppi.prototypes
  and owns some investors,
  has a (prototype.number(1/2), pr.sequence.number(2/2)),
  a portfolio.review.interval, a dep.withdraw.interval,
  a trading.strategy.nr, an init.portfolio.value, an init.stock.fraction,
  a min.deposit, a max.deposit, a max.stock.to.assets,
  a cppi.period, a cppi.ratio, a low.ratio, a high.ratio,
  a cppi.plan.start.cushion.pct, and a cppi.init.cushion.pct.

Every orpi.prototype belongs to the orpi.prototypes
  and owns some investors,
  has a (prototype.number(1/2), pr.sequence.number(2/2)),
  a portfolio.review.interval, a dep.withdraw.interval,
  a trading.strategy.nr, an init.portfolio.value, an init.stock.fraction,
  a min.deposit, a max.deposit, a sigma, an orpi.ratio, a buffer,
  and an orpi.period.

Every order has a price, a quantity, and a customer,
  may belong to the buyers, ''or'' the sellers.

Define buyers as a set ranked by high price.
Define sellers as a set ranked by high price.
Define quantity and customer as integer variables.


    Permanent entities...

Every trading.strategy has a change.bid.interval,
  a bid.at.factor, and an offer.at.factor

Define number.of.trading.strategies to mean n.trading.strategy

   Processes include ''the'' report and daily.close ''processes''

Every rebalancer has a my.prototype ''pointer'', an id.nr,
  a dollars.of.cash, a number.of.shares, a total.dep.withdraw,
  a next.portfolio.review.time, a next.dep.withdraw.time, a my.order,
  and belongs to an investors ''set

Every cppi.investor has a my.prototype ''pointer'', an id.nr,
  a dollars.of.cash, a number.of.shares, a total.dep.withdraw,
  a next.portfolio.review.time, a next.dep.withdraw.time, a my.order,
  and belongs to an investors ''set.
Every cppi.investor ''also'' has a cppi.floor and a next.plan.restart.time.

Every orpi.investor has a my.prototype ''pointer'', an id.nr,
  a dollars.of.cash, a number.of.shares, a total.dep.withdraw,
  a next.portfolio.review.time, a next.dep.withdraw.time, a my.order,
  and belongs to an investors ''set.
Every orpi.investor ''also'' has an exercise.price
  and a next.plan.restart.time.

Define my.order as an integer variable ''points to its buy or sell order.
Define my.prototype as an integer variable ''points to its prototype.
Define id.nr, and number.of.shares as integer variables.

Define current.price.estimate as a real function.
Define normal.cdf as a routine given 1 argument,
    yielding 1 argument.

End.

Main ''routine.
Define i and n as integer variables.
Define a as an alpha variable

Read a  read as /  ''finds and skips over a one line
Read report.interval and remaining.report.periods
Let last.sale.price = 100.0
Activate a report ''process'' now

Read a   read as /,/  ''finds and reads over a two line input heading
Read no.bids.price.est.factor, no.offers.price.est.factor

Call create.trading.strategies

Read a   read as /,/  ''find and skip over a two line
Read n   '' = number of rebalancer prototypes
```

```
Read a  read as /,/,/,/,/,/,/,/,/,/  ''finds and skips over 18 line
        read as /,/,/,/,/,/,/,/,/,/  ''heading in input file.
For i = 1 to n, call create.rebalancer.prototype(i)

Read a  read as /,/   ''finds and skips over a 2 line heading
Read n

Read a  read as /,/,/,/,/,/,/,/,/,/,/,/  ''finds and skips over a
        read as /,/,/,/,/,/,/,/,/,/,/,/  ''26 line heading in the
        read as /,/,/,/,/,/,/            ''input file.
For i = 1 to n, call create.cppi.prototype(i)

Read a  read as /,/   ''finds and skips over a 2 line heading
Read n

Read a  read as /,/,/,/,/,/,/,/,/,/  ''finds and skips over a
        read as /,/,/,/,/,/,/,/      ''22 line heading in the
        read as /,/,/,/,/,/,/,/      ''input file.
For i = 1 to n, call create.orpi.prototype(i)

Read a  read as /,/

Read n
If n = 0
Let seed.v(1) = 10
    else let seed.v(1) = n
Always...
Activate a daily.close now
Start simulation
End.

Routine buy.order(new.order)
Define new.order and old.order as integer variables ''order pointers.
If quantity(new.order) <= 0
    Print 1 line thus...
    Negative buy order placed by investor.
    return
Otherwise...
'top'
If sellers is empty or price(new.order) < price(l.sellers)
    file new.order in buyers
    return
Otherwise...

Let old.order = l.sellers
Let sale.quantity = min.f(quantity(new.order), quantity(old.order))
Subtract sale.quantity from quantity(new.order)
Subtract sale.quantity from quantity(old.order)
Let last.sale.price  = price(old.order)
Add sale.quantity to number.of.shares(customer(new.order))
Let amount = last.sale.price*sale.quantity
Subtract amount from dollars.of.cash(customer(new.order))
Subtract sale.quantity from number.of.shares(customer(old.order))
Add amount to dollars.of.cash(customer(old.order))

If quantity(old.order) = 0
   remove old.order from sellers
Always...
If quantity(new.order) = 0
   return
Otherwise go to top
End.

Routine to calc.cppi.order (p,i)
Define p and i as integer variables  ''pointers to prototype & investor
Define t as an integer variable  ''trading strategy number

Let t = trading.strategy.nr(p)
'top'
Let cpe = current.price.estimate
Let stock.value = number.of.shares(i) * cpe
Let total.value = stock.value + dollars.of.cash(i)
If total.value <= 0  return    else...
Let cushion = max.f(.01,total.value - cppi.floor(i))
If stock.value/cushion > high.ratio(p)
   let price(my.order(i)) = cpe * offer.at.factor(t)
   let quantity(my.order(i))=min.f((stock.value-cppi.ratio(p)*cushion)
                          / price(my.order(i)),number.of.shares(i))
   if quantity(my.order(i)) = 0  return   else...
   call sell.order(my.order(i))
   if my.order(i) is not in sellers
     or time.v+change.bid.interval(t) >
       min.f(next.portfolio.review.time(i),next.dep.withdraw.time(i),
       next.plan.restart.time(i))
       return
   else...
```

```
   wait change.bid.interval(t) days
   if my.order(i) is not in sellers  return    else...
   remove my.order(i) from sellers
   go to top

Otherwise
If stock.value/cushion < low.ratio(p)
   let price(my.order(i)) = cpe * bid.at.factor(t)
   let quantity(my.order(i))=trunc.f
    ((min.f(cppi.ratio(p)*cushion, max.stock.to.assets(p)*total.value)
                                        -stock.value) / price(my.order(i)))
   if quantity(my.order(i)) <= 0  return  else...
   call buy.order(my.order(i))
   if my.order(i) is not in buyers
     or time.v+change.bid.interval(t) >
       min.f(next.portfolio.review.time(i),next.dep.withdraw.time(i),
       next.plan.restart.time(i))
          return
   else...
   wait change.bid.interval(t) days
   if my.order(i) is not in buyers  return    else...
   remove my.order(i) from buyers
   go to top
Always return  end.

Routine to calc.orpi.order (p,i)
Define p and i as integer variables  ''pointers to prototype & investor
Define t as an integer variable  ''trading strategy number

Let t = trading.strategy.nr(p)
'top'
Let cpe = current.price.estimate
Let stock.value = number.of.shares(i) * cpe
Let total.value = stock.value + dollars.of.cash(i)
If total.value <= 0  return    else...
Let d.one = (log.e.f(cpe/exercise.price(i))+0.5*sigma**2*orpi.period(p)/260)
                                        /(sigma*sqrt.f(orpi.period(p)/260))
If d.one > 5.0
    let d.one = 5.0   go to compute always...
If d.one < -5.0
    let d.one = -5.0   go to compute always...
Now ''compute'' normal.cdf given d.one yielding nd.one
'compute'
Let no.trade.buffer=buffer(p)*10
If stock.value/total.value > nd.one+no.trade.buffer or
   stock.value/total.value < nd.one-no.trade.buffer  return  else...
If stock.value/total.value > nd.one+buffer(p)
   let price(my.order(i)) = cpe * offer.at.factor(t)
   let quantity(my.order(i))=(stock.value-total.value*nd.one)
                            / price(my.order(i))
   if quantity(my.order(i)) = 0  return   else...
   call sell.order(my.order(i))
   if my.order(i) is not in sellers
     or time.v+change.bid.interval(t) >
       min.f(next.portfolio.review.time(i),next.dep.withdraw.time(i),
       next.plan.restart.time(i))
       return
   else...
   wait change.bid.interval(t) days
   if my.order(i) is not in sellers  return    else...
   remove my.order(i) from sellers
   go to top

Otherwise
If stock.value/total.value < nd.one-buffer(p)
   let price(my.order(i)) = cpe * bid.at.factor(t)
   let quantity(my.order(i))=(total.value*nd.one-stock.value)/price(my.order(i))
   if quantity(my.order(i)) <= 0  return  else...
   call buy.order(my.order(i))
   if my.order(i) is not in buyers
     or time.v+change.bid.interval(t) >
       min.f(next.portfolio.review.time(i),next.dep.withdraw.time(i),
       next.plan.restart.time(i))
          return
   else...
   wait change.bid.interval(t) days
   if my.order(i) is not in buyers  return    else...
   remove my.order(i) from buyers
   go to top
Always return  end.

Routine to calc.rebalance.order (p,i)
Define p and i as integer variables
Define t as an integer variable  ''trading strategy number

Let t = trading.strategy.nr(p)
```

```
'top'
Let cpe = current.price.estimate
Let stock.value = number.of.shares(i) * cpe
Let total.value = stock.value + dollars.of.cash(i)
If total.value <= 0  return   else...
If stock.value/total.value > high.fraction(p)
   let price(my.order(i)) = cpe * offer.at.factor(t)
   let quantity(my.order(i))=(stock.value - target.fraction(p)
    *total.value) / price(my.order(i))
   call sell.order(my.order(i))
   if my.order(i) is not in sellers
     or time.v+change.bid.interval(t) >
     min.f(next.portfolio.review.time(i),next.dep.withdraw.time(i))
     return
   else...
   wait change.bid.interval(t) days
   if my.order(i) is not in sellers  return  - else...
   remove my.order(i) from sellers
   go to top

Otherwise
If stock.value/total.value < low.fraction(p)
   let price(my.order(i)) = cpe * bid.at.factor(t)
   let quantity(my.order(i))=(target.fraction(p)*total.value-stock.value)
                                          / price(my.order(i))
   call buy.order(my.order(i))
   if my.order(i) is not in buyers
     or time.v+change.bid.interval(t) >
     min.f(next.portfolio.review.time(i),next.dep.withdraw.time(i))
     return
   else...
   wait change.bid.interval(t) days
   if my.order(i) is not in buyers  return    else...
   remove my.order(i) from buyers
   go to top
Always return  end.

Routine to cancel.order(order).
Define order as an integer variable  ''order pointer

If order is in buyers
   remove order from buyers.
   return
Else
If order is in sellers
   remove order from sellers.
Always return end.

Process cppi.investor

Define i and p as integer variables  ''point to investor and prototype

Let i = cppi.investor
Let p = my.prototype(i)
File cppi.investor in investors(p)
Now initialize.investor given p and i
Let cppi.floor(i)=(1.0-cppi.init.cushion.pct(p))*(dollars.of.cash(i)
                          + number.of.shares(i)*last.sale.price)
Let next.plan.restart.time(i) = uniform.f(0,cppi.period(p),1)

'decide.next.event.type'
Let cpe = current.price.estimate
If dollars.of.cash(i)+number.of.shares(i)*cpe <= 0  suspend  always...
If next.plan.restart.time(i) < min.f(next.portfolio.review.time(i),
                                     next.dep.withdraw.time(i))
Go await_plan_restart

Else if next.portfolio.review.time(i) < next.dep.withdraw.time(i)
                                                  go await_review
  else go await_deposit_withdraw

'await_review'   ''then do the portfolio review next
Wait next.portfolio.review.time(i) - time.v days

'Do.review'
Now cancel.order(my.order(i))   ''if any''
Now calc.cppi.order given p and i
Add exponential.f(portfolio.review.interval(p),1)
                                 to next.portfolio.review.time(i)
Go decide.next.event.type


'await_deposit_withdraw'  ''then do the deposit/withdraw
Wait next.dep.withdraw.time(i) - time.v days
Now deposit.or.withdraw given p and i
```

```
Now calc.cppi.order given p and i  ''since ratio may now be too high or low.
Go decide.next.event.type

'await_plan_restart'
Wait next.plan.restart.time(i) - time.v days

'Restart.plan'
Let portfolio.value = dollars.of.cash(i) +
                      number.of.shares(i)*current.price.estimate
Let cppi.floor(i) = (1.0 - cppi.plan.start.cushion.pct(p))*portfolio.value
Add cppi.period(p) to next.plan.restart.time(i)
Let next.portfolio.review.time(i) = time.v
Go do.review
End.

Routine to create.cppi.prototype(pr.nr)
Define pr.nr, i and n as integer variables.
Define p to mean cppi.prototype

Create a cppi.prototype
File cppi.prototype in cppi.prototypes
Let prototype.number(p) = 2
Let pr.sequence.number(p) = pr.nr

Call read.gen.proto.attributes(cppi.prototype)

Read cppi.period(p), cppi.ratio(p), low.ratio(p), high.ratio(p),
 cppi.plan.start.cushion.pct(p), cppi.init.cushion.pct(p),
 max.stock.to.assets(p), and n

For i = 1 to n, activate a cppi.investor(cppi.prototype,i) now

Return   End.

Routine to create.orpi.prototype(pr.nr)
Define pr.nr, i and n as integer variables.
Define p to mean orpi.prototype

Create a orpi.prototype
File orpi.prototype in orpi.prototypes
Let prototype.number(p) = 3
Let pr.sequence.number(p) = pr.nr

Call read.gen.proto.attributes(orpi.prototype)

Read orpi.period(p), sigma(p), orpi.ratio(p), buffer(p), and n

For i = 1 to n, activate a orpi.investor(orpi.prototype,i) now

Return   End.

Routine to create.rebalancer.prototype(rp.nr)
Define rp.nr, i and n as integer variables.
Define p to mean rebalancer.prototype

Create a rebalancer.prototype
File rebalancer.prototype in rebalancer.prototypes
Let prototype.number(p) = 1
Let pr.sequence.number(p) = rp.nr

Call read.gen.proto.attributes(rebalancer.prototype)

Read target.fraction(p), low.fraction(p), high.fraction(p), n
For i = 1 to n, activate a rebalancer(rebalancer.prototype,i) now
Return   End.

Routine to create.trading.strategies
Define a as an alpha variable

Read a   read as /,/

Read number.of.trading.strategies
Create every trading.strategy

Read a   read as /,/,/,/,/

For every trading.strategy,
 read bid.at.factor, offer.at.factor, change.bid.interval

Return   End.

Routine for current.price.estimate.
If buyers is empty and sellers is empty
   return with last.sale.price
Else if buyers is empty and sellers is not empty
   return with price(1.sellers)*no.bids.price.est.factor
```

```
Else if sellers is empty and buyers is not empty
   return with price(f.buyers)*no.offers.price.est.factor
Else
   return with (price(f.buyers) + price(l.sellers))/2.0
End.

Process daily.close
call print.daily.heading
let last.closing.price=100
'top'
Let pct.change.for.day=(last.sale.price/last.closing.price)-1
Let last.closing.price=last.sale.price
If buyers is not empty let bid = price(f.buyers)
Else let bid = 0.0
Always...
If sellers is not empty let asked = price(L.sellers)
  else let asked = 0.0
Always...
print 1 line with time.v, daily.high, daily.low,last.sale.price,bid,
asked and daily.volume, pct.change.for.day thus
**** ******.** ******.** ******.** ******.** ******.** ******.** **.*****
If last.sale.price > 10**6
Stop
Else ...
Reset the daily totals of last.sale.price
Reset the daily totals of sale.quantity
Wait 1 day
Go to top
End.

Routine to deposit.or.withdraw ''for'' given p and i
Define p and i as integer variables ''pointers to an investor i with proto p.

Call cancel.order''if any''(my.order(i))

Let deposit.amount = uniform.f(min.deposit(p),max.deposit(p),1)

Add deposit.amount to dollars.of.cash(i)
Add deposit.amount to total.dep.withdraw(i)

If prototype.number(p) = 2  ''cppi investor
   add deposit.amount*(1.0 - cppi.plan.start.cushion.pct(p))
   to cppi.floor(i)
Always...

Add exponential.f(dep.withdraw.interval(p),1)
   to next.dep.withdraw.time(i)

Return    End.

Routine to initialize.investor ''for'' given p and i
Define p and i as integer variables ''prototype and investor pointer
Let number.of.shares(i) = init.portfolio.value(p)*init.stock.fraction(p)
                                                    /last.sale.price
Let dollars.of.cash(i) = init.portfolio.value(p) -
                          number.of.shares(i)*last.sale.price
Let next.portfolio.review.time(i) = time.v +
                        exponential.f(portfolio.review.interval(p),1)
Let next.dep.withdraw.time(i) = time.v +
                          exponential.f(dep.withdraw.interval(p),1)
Create an order called my.order(i)
Let customer(my.order(i)) = i
Return   End.

Routine to ''compute'' normal.cdf given d.one yielding nd.one

Let YY = 1/(1+.2316419*abs.f(d.one))
Let ZZ = .3989423*exp.f(-1*(d.one**2)/2)
Let nd.one = 1-ZZ*(1.330274*YY**5 - 1.821256*YY**4 + 1.781478*YY**3
                    - .356538*YY**2 + .3193815*YY)
If d.one > 0 go to bottom
    else...
    let nd.one = 1-nd.one
'bottom'
''Print 1 line with d.one, nd.one and current.price.estimate thus
''*.******** *.******** ****.**
''Above two lines were used as a check for d.one and nd.one
Return      end.

Process orpi.investor

Define i and p as integer variables  ''point to investor and prototype

Let i = orpi.investor
Let p = my.prototype(i)
File orpi.investor in investors(p)
```

```
Now initialize.investor given p and i
Let exercise.price(i) = current.price.estimate/orpi.ratio(p)
Let next.plan.restart.time(i) = uniform.f(0,orpi.period(p),1)

'decide.next.event.type'
Let cpe = current.price.estimate
If dollars.of.cash(i)+number.of.shares(i)*cpe <= 0  suspend  always...
If next.plan.restart.time(i) < min.f(next.portfolio.review.time(i),
                                        next.dep.withdraw.time(i))
Go await_plan_restart

Else if next.portfolio.review.time(i) < next.dep.withdraw.time(i)
                                                    go await_review
  else go await_deposit_withdraw

'await_review'   ''then do the portfolio review next
Wait next.portfolio.review.time(i) - time.v days

'Do.review'
Now cancel.order(my.order(i))   ''if any''
Now calc.orpi.order given p and i
Add exponential.f(portfolio.review.interval(p),1)
                                  to next.portfolio.review.time(i)
Go decide.next.event.type


'await_deposit_withdraw'  ''then do the deposit/withdraw
Wait next.dep.withdraw.time(i) - time.v days
Now deposit.or.withdraw given p and i
Now calc.orpi.order given p and i  ''since ratio may now be too high or low.
Go decide.next.event.type

'await_plan_restart'
Wait next.plan.restart.time(i) - time.v days

'Restart.plan'
''Let portfolio.value = dollars.of.cash(i) +
''                      number.of.shares(i)*current.price.estimate
Let exercise.price(i) = current.price.estimate/orpi.ratio(p)
Add orpi.period(p) to next.plan.restart.time(i)
Let next.portfolio.review.time(i) = time.v
Go do.review
End.

Routine print.daily.heading
Print 3 lines thus...

 Day    High      Low       Last      Bid     Asked   Volume    Return


Return   End.

routine to print.status
define i and p as integer variables
print 9 lines with time.v, last.sale.price, period.volume, grand.volume,
    max.ppc, min.ppc, pspc, gspc.
thus...


              Report for period ending day ***

   last      volume  accumulated  maximum  minimum  standard   standard
   sale        for     volume      price    price   deviation  deviation
   price     period               change   change  for period  so far
******.**   ******    *******    **.****  **.****   **.****    **.****

let period.volume = 0
Reset the period totals of pct.change.for.day
let curr.price = current.price.estimate
write as
"sell list (price, quant, p-type, p-nr., i-nr.,cash.val, shares.val): ",/
for each order in sellers, do...
    let i = customer(order)
    let p = my.prototype(i)
    write price, quantity, prototype.number(p), pr.sequence.number(p),
     id.nr(i), dollars.of.cash(i), number.of.shares(i)*curr.price
     as d(10,3), s 2, i 8, 3 i 6, 2 d(10,1),/
loop
write as /
write as
"buy list (price, quant, p-type, p-nr., i-nr., cash.val, shares.val): ",/
for each order in buyers, do...
    let i = customer(order)
    let p = my.prototype(i)
    write price, quantity, prototype.number(p), pr.sequence.number(p),
     id.nr(i), dollars.of.cash(i), number.of.shares(i)*curr.price
```

```
    as d(10,3), s 2, i 8, 3 i 6,2 d(10,1),/
loop
write as /

Define r as an integer variable
let lsp = last.sale.price

If rebalancer = 0 go to next1 always...
print 5 lines thus...
          Status of rebalancing investors

   id.    number of      share      dollars of       total        actual
   nr.     shares        value         cash          value       fraction

For each rebalancer in investors(my.prototype(rebalancer)), do...
Let r = rebalancer
  write id.nr(r),
  number.of.shares(r), number.of.shares(r)*lsp, dollars.of.cash(r),
  dollars.of.cash(r) + number.of.shares(r)*lsp,
''  number.of.shares(r)*lsp/(dollars.of.cash(r) + number.of.shares(r)*lsp)
  total.dep.withdraw(r)
as i 10, i 10, d(15,2), d(15,2), d(15,2), d(10,2),/
Let total.dep.withdraw(r) = 0.0
loop

'next1'
If cppi.investor = 0 go to next2 always...
print 5 lines thus...
          Status of CPPI investors

   id.    number of      share      dollars of       total        actual
   nr.     shares        value         cash          value       fraction

For each cppi.investor in investors(my.prototype(cppi.investor)), do...
Let r = cppi.investor
  write id.nr(r),
  number.of.shares(r), number.of.shares(r)*lsp, dollars.of.cash(r),
  dollars.of.cash(r) + number.of.shares(r)*lsp,
''  number.of.shares(r)*lsp/(dollars.of.cash(r) + number.of.shares(r)*lsp)
  total.dep.withdraw(r)
as i 10, i 10, d(15,2), d(15,2), d(15,2), d(10,2),/
Let total.dep.withdraw(r) = 0.0
loop

'next2'
If orpi.investor = 0 go to next3 always...
print 5 lines thus...
          Status of ORPI investors

   id.    number of      share      dollars of       total        actual
   nr.     shares        value         cash          value       fraction

For each orpi.investor in investors(my.prototype(orpi.investor)), do...
Let r = orpi.investor
  write id.nr(r),
  number.of.shares(r), number.of.shares(r)*lsp, dollars.of.cash(r),
  dollars.of.cash(r) + number.of.shares(r)*lsp,
''  number.of.shares(r)*lsp/(dollars.of.cash(r) + number.of.shares(r)*lsp)
  total.dep.withdraw(r)
as i 10, i 10, d(15,2), d(15,2), d(15,2), d(10,2),/
Let total.dep.withdraw(r) = 0.0
loop
'next3'
write as /,/
call print.daily.heading
return  end

Routine to read.gen.proto.attributes ''for prototype''(p)
Define p as an integer variable  ''prototype pointer

Read init.portfolio.value(p), init.stock.fraction(p),
 portfolio.review.interval(p), dep.withdraw.interval(p),
 min.deposit(p), max.deposit(p), and  trading.strategy.nr(p)

If 0 < trading.strategy.nr(p) <= number.of.trading.strategies is false
   write as "incorrect trading strategy specified",/   stop
Always return
End.

Process rebalancer

Define r and rp as integer variables ''point to rebalancing investor
                                     ''and prototype
Let r = rebalancer
Let rp = my.prototype(r)
File rebalancer in investors(rp)
```

```
Now initialize.investor ''for the'' given rp and r.

'decide.next.event.type'
Let cpe = current.price.estimate
If dollars.of.cash(r)+number.of.shares(r)*cpe <= .0001  suspend   always...
If next.portfolio.review.time(r) < next.dep.withdraw.time(r) go await_review
  else go await_deposit_withdraw

'await_review'   ''then do the portfolio review next
Wait next.portfolio.review.time(r) - time.v days
Call cancel.order''if any''(my.order(r))
Call calc.rebalance.order(rp,r)
Add exponential.f(portfolio.review.interval(rp),1) to
                                        next.portfolio.review.time(r)
Go decide.next.event.type

'await_deposit_withdraw'  ''then do the deposit/withdraw
Wait next.dep.withdraw.time(r) - time.v days
Call deposit.or.withdraw ''for'' (rp, r)
Call calc.rebalance.order(rp,r) ''since a rebalance may now be necessary
Go decide.next.event.type

End.

Process report

''initial report''

'top'
Wait report.interval days
Call print.status
Subtract 1 from remaining.report.periods
If remaining.report.periods = 0   stop
Else go to top
End.

Routine sell.order(new.order)
Define new.order and old.order as integer variables ''order pointers.
Define p to mean last.sale.price
Define q to mean sale.quantity

'top'
If buyers is empty or price(new.order) > price(f.buyers)
    file new.order in sellers
    return
Otherwise...

Let old.order = f.buyers
Let q = min.f(quantity(new.order), quantity(old.order))
Subtract q from quantity(new.order)
Subtract q from quantity(old.order)
Let p = price(old.order)
Subtract q from number.of.shares(customer(new.order))
Add p*q to dollars.of.cash(customer(new.order))
Add q to number.of.shares(customer(old.order))
Subtract p*q from dollars.of.cash(customer(old.order))

If quantity(old.order) = 0
   remove old.order from buyers
Always...
If quantity(new.order) = 0
   return
Otherwise go to top
End.
```

DATAFILE

```
   Days per report period          Report periods per simulation

         65                                100

   When there are no bids,                  When there are no offers
estimated price is best offer times...   estimated price is best bid times...

         0.99                              1.01

Number of
trading strategies...
          1

Enter the following information for each trading strategy:

When buying, bid at      When selling, offer at     Reconsider bid/offer
estimated price times    estimated price times      this often...
this factor...           this factor...             (decimal, days)

   1.01                        0.99                       1.0

Number of prototypes for
investors who REBALANCE...
 2

For each rebalancer prototype enter the following information,
on any number of lines...

     Value of starting portfolio
     Initial ratio of stock value to total value ($stock/$stock+cash)
     How often portfolio is reviewed for rebalancing (decimal, days)

     How often, on the average, do deposits or withdrawls occur (dec., days)
     Minimum deposit ( < 0 represents a withdrawl)
     Maximum deposit (program assumes uniform distribution between min and max)

     Trading strategy number (number between 1 and nr. of trading strategies)

     Target fraction of stock to total assets ($stock/$stock+cash)
     Low fraction: rebalance if actual is less
     High fraction: rebalance if actual is greater

     Number of investors with this prototype

      100000    .5      5
     10   -8000  9000
      1
      0.5  0.46  0.55
      100

       100000    .5      5
        10   -8000  9000
        1
       0.5  0.46  0.55
       0

How many CPPI (constant proportion portfolio insurer) prototypes
do you want?

1

For each CPPI investor prototype enter the following information,
on any number of lines...

     Value of starting portfolio
     Initial ratio of stock value to total value ($stock/$stock+cash)
     How often portfolio is reviewed to see if actual fraction is
      close to target fraction (decimal, days)

     How often, on the average, do deposits or withdrawls occur (dec., days)
     Minimum deposit ( < 0 represents a withdrawl)
     Maximum deposit (program assumes uniform distribution between min and max)
```

```
      Trading strategy number (number between 1 and nr. of trading strategies)

      Length of insurance plan (decimal, days)
      Target ratio of $stock to $cushion (= $assets - $floor)
      Low ratio:  buy stock if actual ratio is this low
      High ratio:  sell stock if actual ratio is this high

      Cushion as a fraction of portfolio value: at start of simulation
      Cushion as a fraction of portfolio value: start of new insurance plan

      Maximum ratio of stock to assets:  max $stock/$stock+$cash
       ($cash may be negative if this max ratio exceeds 1.0)

      Number of investors with this prototype

        100000      .5     5

          10      -8000    9000

          1

          65       2.0      1.7       2.3

          .25       .25

          1.0

          0

How many ORPI (option replicating portfolio insurer) prototypes
do you want?

1

For each ORPI investor prototype enter the following information,
on any number of lines...

      Value of starting portfolio
      Initial ratio of stock value to total value ($stock/$stock+cash)
      How often portfolio is reviewed to see if actual fraction is
       close to target fraction (decimal, days)

      How often, on the average, do deposits or withdrawls occur (dec., days)
      Minimum deposit ( < 0 represents a withdrawl)
      Maximum deposit (program assumes uniform distribution between min and max)
      Trading strategy number (number between 1 and nr. of trading strategies)

      Length of insurance plan (decimal, days)
      Expected standard deviation of returns (decimal)
      Ratio of stock market price to exercise price of call option being
         replicated (exp(-0.5*sigma_squared*plan_length_in_years) indicates
         half stock and half cash to start, higher ratio: less insurance)
      Buffer:  Percent deviation from target hedge ratio before a buy or sell
         trade is initiated.

      Number of investors with this prototype

      100000      .5       5

      10      -8000    9000

      1

      65      .20      1.00      0.01

      50

         Random Seed

         0
```

KIM-MARKOWITZ WITH S&P 500 DATA

```
Preamble...

Normally mode is real.

The system has a last.sale.price, a pct.change.for.day, a sale.quantity,
  a report.interval, a remaining.report.periods, a period.number,
''  a no.bids.price.est.factor, a no.offers.price.est.factor,''
  a last.period.price, a mkt.return, a sum.mkt.return,
  a sum.mkt.return.squared, a total.number.of.shares, a total.cash,
  a market.value, a period.deposits, a number.of.periods,
  a mkt.sigma, a mkt.sharpe, a mkt.treynor,
  and owns the buyers, the sellers, the rebalancer.prototypes,
  the orpi.prototypes and the cppi.prototypes.

Define sale.quantity, period.number and total.number.of.shares
   as integer variables

Tally daily.high as the daily max, daily.low as the daily min,
  period.high as the period max, period.low as the period min,
  grand.high as the max and grand.low as the min of last.sale.price

Tally daily.volume as the daily sum, period.volume as the
  period sum, grand.volume as the sum of sale.quantity

Tally max.ppc as the period max, max.gpc as the grand max,
  min.ppc as the period min,
  min.gpc as the grand min of pct.change.for.day

Tally pmpc as the period mean, gmpc as the grand mean,
  pspc as the period std.dev,
  gspc as the grand std.dev of pct.change.for.day

  Temporary entities....

Every rebalancer.prototype belongs to the rebalancer.prototypes
  and owns some investors,
  has a (prototype.number(1/2), pr.sequence.number(2/2)),
  a portfolio.review.interval, a dep.withdraw.interval,
  a trading.strategy.nr, an init.portfolio.value, an init.stock.fraction,
  a min.deposit, a max.deposit,
  a target.fraction, a high.fraction, and a low.fraction.

Define trading.strategy.nr as an integer variable  ''tr. str. pointer
Define prototype.number and pr.sequence.number as integer variables.

Define rebalancer.prototypes, cppi.prototypes, orpi.prototypes,
  and investors as fifo sets without p attributes,
  and without r, ff, fb, fa routines.

Every cppi.prototype belongs to the cppi.prototypes
  and owns some investors,
  has a (prototype.number(1/2), pr.sequence.number(2/2)),
  a portfolio.review.interval, a dep.withdraw.interval,
  a trading.strategy.nr, an init.portfolio.value, an init.stock.fraction,
  a min.deposit, a max.deposit, a max.stock.to.assets,
  a cppi.period, a cppi.ratio, a low.ratio, a high.ratio,
  a cppi.plan.start.cushion.pct, and a cppi.init.cushion.pct.

Every orpi.prototype belongs to the orpi.prototypes
  and owns some investors,
  has a (prototype.number(1/2), pr.sequence.number(2/2)),
  a portfolio.review.interval, a dep.withdraw.interval,
  a trading.strategy.nr, an init.portfolio.value, an init.stock.fraction,
  a min.deposit, a max.deposit, an est.sigma, an orpi.ratio, a buffer,
  and an orpi.period.

Every order has ''a price,'' a quantity, and a customer,
  may belong to the buyers, ''or'' the sellers.

''Define buyers as a set ranked by high price.
''Define sellers as a set ranked by high price.
Define quantity and customer as integer variables.


    Permanent entities...

Every trading.strategy has a change.bid.interval'',
  ''a bid.at.factor, and an offer.at.factor''

Define number.of.trading.strategies to mean n.trading.strategy

   Processes include ''the'' report and daily.close ''processes''

Every rebalancer has a my.prototype ''pointer'', an id.nr,
```

```
  a dollars.of.cash, a number.of.shares, a total.dep.withdraw,
  a next.portfolio.review.time, a next.dep.withdraw.time, a my.order,
  an init.value, an alpha, a beta, a t.value, a sum.of.returns,
  a sum.of.squares, a sum.of.xy, a sigma, a sharpe, a treynor,
  and belongs to an investors ''set

Every cppi.investor has a my.prototype ''pointer'', an id.nr,
  a dollars.of.cash, a number.of.shares, a total.dep.withdraw,
  a next.portfolio.review.time, a next.dep.withdraw.time, a my.order,
  an init.value, an alpha, a beta, a t.value, a sum.of.returns,
  a sum.of.squares, a sum.of.xy, a sigma, a sharpe, a treynor,
  and belongs to an investors ''set.
Every cppi.investor ''also'' has a cppi.floor and a next.plan.restart.time.

Every orpi.investor has a my.prototype ''pointer'', an id.nr,
  a dollars.of.cash, a number.of.shares, a total.dep.withdraw,
  a next.portfolio.review.time, a next.dep.withdraw.time, a my.order,
  an init.value, an alpha, a beta, a t.value, a sum.of.returns,
  a sum.of.squares, a sum.of.xy, a sigma, a sharpe, a treynor,
  and belongs to an investors ''set.
Every orpi.investor ''also'' has an exercise.price
  and a next.plan.restart.time.

Define my.order as an integer variable ''points to its buy or sell order.
Define my.prototype as an integer variable ''points to its prototype.
Define id.nr, and number.of.shares as integer variables.

''Define current.price.estimate as a real function.''
Define normal.cdf as a routine given 1 argument,
    yielding 1 argument.
Define compute.stats as a routine given 3 arguments,
    yielding 6 arguments.

End.

Main ''routine.
Define i and n as integer variables.
Define a as an alpha variable

Read a  read as /  ''finds and skips over a one line
Read report.interval and remaining.report.periods
Let last.sale.price = 100.0
Let last.period.price = last.sale.price
''Let number.of.periods = remaining.report.periods''
Activate a report ''process'' now

''Read a   read as /,/  finds and reads over a two line input heading
''Read no.bids.price.est.factor, no.offers.price.est.factor

Call create.trading.strategies

Read a   read as /,/  ''find and skip over a two line
Read n   '' = number of rebalancer prototypes

Read a  read as /,/,/,/,/,/,/,/,/,/,/  ''finds and skips over 18 line
        read as /,/,/,/,/,/,/,/,/,/,/  ''heading in input file.
For i = 1 to n, call create.rebalancer.prototype(i)

Read a  read as /,/   ''finds and skips over a 2 line heading
Read n

Read a  read as /,/,/,/,/,/,/,/,/,/,/,/,/  ''finds and skips over a
        read as /,/,/,/,/,/,/,/,/,/,/,/,/  ''26 line heading in the
        read as /,/,/,/,/,/,/              ''input file.
For i = 1 to n, call create.cppi.prototype(i)

Read a  read as /,/   ''finds and skips over a 2 line heading
Read n

Read a  read as /,/,/,/,/,/,/,/,/,/,/  ''finds and skips over a
        read as /,/,/,/,/,/,/,/,/,/     ''22 line heading in the
        read as /,/,/,/,/,/,/,/,/       ''input file.
For i = 1 to n, call create.orpi.prototype(i)

Read a  read as /,/

Read n
If n = 0
Let seed.v(1) = 10
    else let seed.v(1) = n
Always...
Let last.sale.price = 100
Activate a daily.close now
Start simulation
End.
```

```
Routine buy.order(new.order)
Define new.order as an integer variable ''order pointer.
If quantity(new.order) <= 0
    Print 1 line thus...
    Negative buy order placed by investor.
    return
Otherwise...
    Let sale.quantity = quantity(new.order)
    Add quantity(new.order) to number.of.shares(customer(new.order))
    Let amount = last.sale.price*quantity(new.order)
    Subtract amount from dollars.of.cash(customer(new.order))
    return
End.

Routine to calc.cppi.order (p,i)
Define p and i as integer variables  ''pointers to prototype & investor

Let cpe = last.sale.price
Let stock.value = number.of.shares(i) * cpe
Let total.value = stock.value + dollars.of.cash(i)
If total.value <= 0  return    else...
Let cushion = max.f(.01,total.value - cppi.floor(i))
If stock.value/cushion > high.ratio(p)
   let quantity(my.order(i))=min.f((stock.value-cppi.ratio(p)*cushion)
                                / cpe,number.of.shares(i))
   if quantity(my.order(i)) = 0  return   else...
   call sell.order(my.order(i))
Otherwise...
Always
If stock.value/cushion < low.ratio(p)
   let quantity(my.order(i))=trunc.f
    ((min.f(cppi.ratio(p)*cushion, max.stock.to.assets(p)*total.value)
                                        -stock.value) / cpe)
   if quantity(my.order(i)) <= 0  return  else...
   call buy.order(my.order(i))
Always return  end.

Routine to calc.orpi.order (p,i)
Define p and i as integer variables  ''pointers to prototype & investor

Let cpe = last.sale.price
Let stock.value = number.of.shares(i) * cpe
Let total.value = stock.value + dollars.of.cash(i)
If total.value <= 0  return    else...
Let d.one = (log.e.f(cpe/exercise.price(i))+0.5*est.sigma**2*orpi.period(p)/260)
                                        /(est.sigma*sqrt.f(orpi.period(p)/260))
If d.one > 5.0
    let d.one = 5.0   go to compute always...
If d.one < -5.0
    let d.one = -5.0   go to compute always...
Now ''compute'' normal.cdf given d.one yielding nd.one
'compute'
Let trade.buffer = buffer(p)*10.0
If stock.value/total.value > nd.one+trade.buffer or
   stock.value/total.value < nd.one-trade.buffer  return  else...
If stock.value/total.value > nd.one+buffer(p)
   let quantity(my.order(i))=(stock.value-total.value*nd.one) / cpe
   if quantity(my.order(i)) = 0  return   else...
   call sell.order(my.order(i))
Otherwise...
Always
If stock.value/total.value < nd.one-buffer(p)
   let quantity(my.order(i))=(total.value*nd.one-stock.value)/cpe
   if quantity(my.order(i)) <= 0  return  else...
   call buy.order(my.order(i))
Always return  end.

Routine to calc.rebalance.order (p,i)
Define p and i as integer variables

Let cpe = last.sale.price
Let stock.value = number.of.shares(i) * cpe
Let total.value = stock.value + dollars.of.cash(i)
If total.value <= 0  return   else...
If stock.value/total.value > high.fraction(p)
   let quantity(my.order(i))=(stock.value - target.fraction(p)
    *total.value) / cpe
   call sell.order(my.order(i))
Otherwise...
Always
If stock.value/total.value < low.fraction(p)
   let quantity(my.order(i))=(target.fraction(p)*total.value-stock.value)
                                        / cpe
   call buy.order(my.order(i))
Always return  end.
```

```
Routine to cancel.order(order).
Define order as an integer variable  ''order pointer

If order is in buyers
   remove order from buyers.
   return
Else
If order is in sellers
   remove order from sellers.
Always return end.

Routine to compute.stats given x, x2, xy
  yielding bet, alp, t, sharp, trey and sig
Define n as an integer variable
Let n = period.number + 1
Let y = sum.mkt.return
Let y2 = sum.mkt.return.squared

Let bet = (n*xy - x*y) / (n*y2 - y**2)
Let alp = (x - bet*y) / n
Let d2 = (x2/n-((x/n)**2)-(bet**2)*y2/n+((bet*y/n)**2))
                    *(1/n + ((y/n)**2)/y2)
Let d = sqrt.f(abs.f(d2))
Let t = alp / d

Let sig = sqrt.f(abs.f((n*x2 - x**2) / n**2))
Let sharp = x/n/sig
Let trey = x/n/bet

Return.
End.

Process cppi.investor

Define i and p as integer variables  ''point to investor and prototype

Let i = cppi.investor
Let p = my.prototype(i)
File cppi.investor in investors(p)
Now initialize.investor given p and i
Let cppi.floor(i)=(1.0-cppi.init.cushion.pct(p))*(dollars.of.cash(i)
                          + number.of.shares(i)*last.sale.price)
Let next.plan.restart.time(i) = uniform.f(0,cppi.period(p),1)

'decide.next.event.type'
Let cpe = last.sale.price
If dollars.of.cash(i)+number.of.shares(i)*cpe <= 0  suspend  always...
If next.plan.restart.time(i) < min.f(next.portfolio.review.time(i),
                                        next.dep.withdraw.time(i))
Go await_plan_restart

Else if next.portfolio.review.time(i) < next.dep.withdraw.time(i)
                                                    go await_review
  else go await_deposit_withdraw

'await_review'   ''then do the portfolio review next
Wait next.portfolio.review.time(i) - time.v days

'Do.review'
Now cancel.order(my.order(i))   ''if any''
Now calc.cppi.order given p and i
Add exponential.f(portfolio.review.interval(p),1)
                                  to next.portfolio.review.time(i)
Go decide.next.event.type


'await_deposit_withdraw'  ''then do the deposit/withdraw
Wait next.dep.withdraw.time(i) - time.v days
Now deposit.or.withdraw given p and i
Now calc.cppi.order given p and i  ''since ratio may now be too high or low.
Go decide.next.event.type

'await_plan_restart'
Wait next.plan.restart.time(i) - time.v days

'Restart.plan'
Let portfolio.value = dollars.of.cash(i) +
                      number.of.shares(i)*cpe
Let cppi.floor(i) = (1.0 - cppi.plan.start.cushion.pct(p))*portfolio.value
Add cppi.period(p) to next.plan.restart.time(i)
Let next.portfolio.review.time(i) = time.v
Go do.review
End.

Routine to create.cppi.prototype(pr.nr)
Define pr.nr, i and n as integer variables.
```

```
Define p to mean cppi.prototype

Create a cppi.prototype
File cppi.prototype in cppi.prototypes
Let prototype.number(p) = 2
Let pr.sequence.number(p) = pr.nr

Call read.gen.proto.attributes(cppi.prototype)

Read cppi.period(p), cppi.ratio(p), low.ratio(p), high.ratio(p),
 cppi.plan.start.cushion.pct(p), cppi.init.cushion.pct(p),
 max.stock.to.assets(p), and n

For i = 1 to n, activate a cppi.investor(cppi.prototype,i) now

Return   End.

Routine to create.orpi.prototype(pr.nr)
Define pr.nr, i and n as integer variables.
Define p to mean orpi.prototype

Create a orpi.prototype
File orpi.prototype in orpi.prototypes
Let prototype.number(p) = 3
Let pr.sequence.number(p) = pr.nr

Call read.gen.proto.attributes(orpi.prototype)

Read orpi.period(p), est.sigma(p), orpi.ratio(p), buffer(p), and n

For i = 1 to n, activate a orpi.investor(orpi.prototype,i) now

Return   End.

Routine to create.rebalancer.prototype(rp.nr)
Define rp.nr, i and n as integer variables.
Define p to mean rebalancer.prototype

Create a rebalancer.prototype
File rebalancer.prototype in rebalancer.prototypes
Let prototype.number(p) = 1
Let pr.sequence.number(p) = rp.nr

Call read.gen.proto.attributes(rebalancer.prototype)

Read target.fraction(p), low.fraction(p), high.fraction(p), n
For i = 1 to n, activate a rebalancer(rebalancer.prototype,i) now
Return   End.

Routine to create.trading.strategies
Define a as an alpha variable

Read a   read as /
Read a   read as /

Read number.of.trading.strategies
Create every trading.strategy

Return   End.

Process daily.close
''call print.daily.heading
'top'
Read pct.change.for.day
If time.v > remaining.report.periods - 1
    call print.status
Otherwise...
    Let last.sale.price=last.sale.price*(1+pct.change.for.day)
Always
''print 1 line with time.v, last.sale.price, daily.volume, pct.change.for.day
''thus...
''    **** ******.** ********** **.*****
Reset the daily totals of last.sale.price
Reset the daily totals of sale.quantity
Wait 1 day
Go to top
End.

Routine to deposit.or.withdraw ''for'' given p and i
Define p and i as integer variables ''pointers to an investor i with proto p.

Call cancel.order''if any''(my.order(i))

Let deposit.amount = uniform.f(min.deposit(p),max.deposit(p),1)

Add deposit.amount to dollars.of.cash(i)
```

```
Add deposit.amount to total.dep.withdraw(i)
Add deposit.amount to total.cash
Add deposit.amount to period.deposits

If prototype.number(p) = 2  ''cppi investor
   add deposit.amount*(1.0 - cppi.plan.start.cushion.pct(p))
   to cppi.floor(i)
Always...

Add exponential.f(dep.withdraw.interval(p),1)
   to next.dep.withdraw.time(i)

Return    End.

Routine to initialize.investor ''for'' given p and i
Define p and i as integer variables ''prototype and investor pointer
Let init.value(i) = init.portfolio.value(p)
Add init.value(i) to market.value
Let number.of.shares(i) = init.portfolio.value(p)*init.stock.fraction(p)
                                                    /last.sale.price
Add number.of.shares(i) to total.number.of.shares
Let dollars.of.cash(i) = init.portfolio.value(p) -
                         number.of.shares(i)*last.sale.price
Add dollars.of.cash(i) to total.cash
Let next.portfolio.review.time(i) = time.v +
                    exponential.f(portfolio.review.interval(p),1)
Let next.dep.withdraw.time(i) = time.v +
                      exponential.f(dep.withdraw.interval(p),1)
Create an order called my.order(i)
Let customer(my.order(i)) = i
Return   End.

Routine to ''compute'' normal.cdf given d.one yielding nd.one

Let YY = 1/(1+.2316419*abs.f(d.one))
Let ZZ = .3989423*exp.f(-1*(d.one**2)/2)
Let nd.one = 1-ZZ*(1.330274*YY**5 - 1.821256*YY**4 + 1.781478*YY**3
                 - .356538*YY**2 + .3193815*YY)
If d.one > 0 go to bottom
    else...
    let nd.one = 1-nd.one
'bottom'
''Print 1 line with d.one, nd.one and current.price.estimate thus
'*.*******    *.*******    ****.**
''Above two lines were used as a check for d.one and nd.one
Return      end.

Process orpi.investor

Define i and p as integer variables  ''point to investor and prototype

Let i = orpi.investor
Let p = my.prototype(i)
File orpi.investor in investors(p)
Now initialize.investor given p and i
Let exercise.price(i) = last.sale.price/orpi.ratio(p)
Let next.plan.restart.time(i) = uniform.f(0,orpi.period(p),1)

'decide.next.event.type'
Let cpe = last.sale.price
If dollars.of.cash(i)+number.of.shares(i)*cpe <= 0  suspend  always...
If next.plan.restart.time(i) < min.f(next.portfolio.review.time(i),
                                     next.dep.withdraw.time(i))
Go await_plan_restart

Else if next.portfolio.review.time(i) < next.dep.withdraw.time(i)
                                                    go await_review
  else go await_deposit_withdraw

'await_review'   ''then do the portfolio review next
Wait next.portfolio.review.time(i) - time.v days

'Do.review'
Now cancel.order(my.order(i))   ''if any''
Now calc.orpi.order given p and i
Add exponential.f(portfolio.review.interval(p),1)
                                  to next.portfolio.review.time(i)
Go decide.next.event.type


'await_deposit_withdraw'  ''then do the deposit/withdraw
Wait next.dep.withdraw.time(i) - time.v days
Now deposit.or.withdraw given p and i
Now calc.orpi.order given p and i  ''since ratio may now be too high or low.
Go decide.next.event.type
```

```
'await_plan_restart'
Wait next.plan.restart.time(i) - time.v days

'Restart.plan'
''Let portfolio.value = dollars.of.cash(i) +
''                      number.of.shares(i)*last.sale.price
Let exercise.price(i) = last.sale.price/orpi.ratio(p)
Add orpi.period(p) to next.plan.restart.time(i)
Let next.portfolio.review.time(i) = time.v
Go do.review
End.

Routine print.daily.heading
Print 3 lines thus...

Day      Last      Volume    Return

Return   End.

Routine to print.status
''define i and p as integer variables
''print 9 lines with time.v
Print 1 line with period.number, last.sale.price, period.volume,
grand.volume, max.ppc, min.ppc, pspc, gspc
thus...
*** ******.**   ******    *******       **.****  **.****   **.****    **.****
''
''
''               Report for period ending day ***
''
''   last     volume  accumulated  maximum  minimum  standard   standard
''   sale       for     volume      price    price   deviation  deviation
''   price    period               change   change  for period  so far
''*******.**   ******    *******    **.****  **.****  **.****    **.****
''
let period.volume = 0
Reset the period totals of pct.change.for.day

Define r and nn as integer variables
Let lsp = last.sale.price
Add 1 to period.number
Let nn = period.number
Let mkt.return = (lsp / last.period.price - 1.0) / 2.0
''Write lsp, total.number.of.shares, total.cash, mv, market.value,
''  period.deposits, last.period.price
''  as d(6,2), i 8, d(11,2), d(11,2), d(11,2), d(11,2), d(8,2),/
Let last.period.price = lsp
Add mkt.return to sum.mkt.return
Add (mkt.return**2) to sum.mkt.return.squared
If time.v > remaining.report.periods - 1
    Let mkt.sigma = sqrt.f(abs.f((nn*sum.mkt.return.squared
                        - sum.mkt.return**2) / nn**2))
    Let mkt.sharpe = sum.mkt.return/nn/mkt.sigma
    Let mkt.treynor = sum.mkt.return/nn
    Write mkt.return, sum.mkt.return, mkt.return**2, sum.mkt.return.squared,
      mkt.sigma, mkt.sharpe, and mkt.treynor as
      /, d(9,4), d(9,4), d(12,8), d(12,8), d(12,8), d(12,8), d(12,8),/
Always...
If rebalancer = 0 go to next1 always...
''print 5 lines thus...
''           Status of rebalancing investors
''
''   id.    number of      share      dollars of        total        period
''   nr.     shares        value        cash            value       deposits
''
For each rebalancer in investors(my.prototype(rebalancer)), do...
Let r = rebalancer
''  write id.nr(r),
''  number.of.shares(r), number.of.shares(r)*lsp, dollars.of.cash(r),
''    dollars.of.cash(r) + number.of.shares(r)*lsp,
''    total.dep.withdraw(r)
''as i 8, i 8, d(15,2), d(15,2), d(15,2), d(10,2),/

Let value = number.of.shares(r)*lsp + dollars.of.cash(r)
Let return = (value - init.value(r) - total.dep.withdraw(r))
               /(init.value(r) + 0.5*total.dep.withdraw(r))
Add return to sum.of.returns(r)
Add (return**2) to sum.of.squares(r)
Add (return*mkt.return) to sum.of.xy(r)
Let init.value(r) = value
''Write return, sum.of.returns(r), return**2, sum.of.squares(r), sum.of.xy(r)
''  as d(8,4), d(8,4), d(12,8), d(12,8), d(12,8),/

Let total.dep.withdraw(r) = 0.0
If time.v > remaining.report.periods - 1
  call compute.stats giving sum.of.returns(r), sum.of.squares(r), sum.of.xy(r)
```

```
  yielding beta(r), alpha(r), t.value(r), sharpe(r), treynor(r), sigma(r)
Write beta(r), alpha(r), t.value(r), sharpe(r), treynor(r), sigma(r)
  as d(9,4), d(9,4), d(9,4), d(9,6), d(9,6), d(9,6),/
Always...
loop

'next1'
If cppi.investor = 0 go to next2 always...
''print 5 lines thus...
''           Status of CPPI investors
''
''   id.    number of      share      dollars of        total        period
''   nr.     shares        value         cash           value       deposits
''
For each cppi.investor in investors(my.prototype(cppi.investor)), do...
Let r = cppi.investor
''  write id.nr(r),
''  number.of.shares(r), number.of.shares(r)*lsp, dollars.of.cash(r),
''  dollars.of.cash(r) + number.of.shares(r)*lsp,
''  total.dep.withdraw(r)
''as i 8, i 8, d(15,2), d(15,2), d(15,2), d(10,2),/

Let value = number.of.shares(r)*lsp + dollars.of.cash(r)
Let return = (value - init.value(r) - total.dep.withdraw(r))
                /(init.value(r) + 0.5*total.dep.withdraw(r))
Add return to sum.of.returns(r)
Add (return**2) to sum.of.squares(r)
Add (return*mkt.return) to sum.of.xy(r)
Let init.value(r) = value
''Write return, sum.of.returns(r), return**2, sum.of.squares(r), sum.of.xy(r)
''  as d(8,4), d(8,4), d(12,8), d(12,8), d(12,8),/

Let total.dep.withdraw(r) = 0.0
If time.v > remaining.report.periods - 1
  call compute.stats giving sum.of.returns(r), sum.of.squares(r), sum.of.xy(r)
  yielding beta(r), alpha(r), t.value(r), sharpe(r), treynor(r), sigma(r)
Write beta(r), alpha(r), t.value(r), sharpe(r), treynor(r), sigma(r)
  as d(9,4), d(9,4), d(9,4), d(9,6), d(9,6), d(9,6),/
Always...
loop

'next2'
If orpi.investor = 0 go to next3 always...
''print 5 lines thus...
''           Status of ORPI investors
''
''   id.    number of      share      dollars of        total        period
''   nr.     shares        value         cash           value       deposits
''
For each orpi.investor in investors(my.prototype(orpi.investor)), do...
Let r = orpi.investor
''  write id.nr(r),
''  number.of.shares(r), number.of.shares(r)*lsp, dollars.of.cash(r),
''  dollars.of.cash(r) + number.of.shares(r)*lsp,
''  total.dep.withdraw(r)
''as i 8, i 8, d(15,2), d(15,2), d(15,2), d(10,2),/

Let value = number.of.shares(r)*lsp + dollars.of.cash(r)
Let return = (value - init.value(r) - total.dep.withdraw(r))
                /(init.value(r) + 0.5*total.dep.withdraw(r))
Add return to sum.of.returns(r)
Add (return**2) to sum.of.squares(r)
Add (return*mkt.return) to sum.of.xy(r)
Let init.value(r) = value
''Write return, sum.of.returns(r), return**2, sum.of.squares(r), sum.of.xy(r)
''  as d(8,4), d(8,4), d(12,8), d(12,8), d(12,8),/

Let total.dep.withdraw(r) = 0.0
If time.v > remaining.report.periods - 1
  call compute.stats giving sum.of.returns(r), sum.of.squares(r), sum.of.xy(r)
  yielding beta(r), alpha(r), t.value(r), sharpe(r), treynor(r), sigma(r)
Write beta(r), alpha(r), t.value(r), sharpe(r), treynor(r), sigma(r)
  as d(9,4), d(9,4), d(9,4), d(9,6), d(9,6), d(9,6),/
Always...
loop

'next3'
''write as /,/
If time.v > remaining.report.periods - 1 stop
else...
''call print.daily.heading
return  end

Routine to read.gen.proto.attributes ''for prototype''(p)
Define p as an integer variable  ''prototype pointer
```

```
Read init.portfolio.value(p), init.stock.fraction(p),
 portfolio.review.interval(p), dep.withdraw.interval(p),
 min.deposit(p), max.deposit(p), and  trading.strategy.nr(p)

If 0 < trading.strategy.nr(p) <= number.of.trading.strategies is false
   write as "incorrect trading strategy specified",/   stop
Always return
End.

Process rebalancer

Define r and rp as integer variables ''point to rebalancing investor
                                     ''and prototype
Let r = rebalancer
Let rp = my.prototype(r)
File rebalancer in investors(rp)
Now initialize.investor ''for the'' given rp and r.

'decide.next.event.type'
Let cpe = last.sale.price
If dollars.of.cash(r)+number.of.shares(r)*cpe <= .0001  suspend   always...
If next.portfolio.review.time(r) < next.dep.withdraw.time(r) go await_review
  else go await_deposit_withdraw

'await_review'   ''then do the portfolio review next
Wait next.portfolio.review.time(r) - time.v days
Call cancel.order''if any''(my.order(r))
Call calc.rebalance.order(rp,r)
Add exponential.f(portfolio.review.interval(rp),1) to
                                          next.portfolio.review.time(r)
Go decide.next.event.type

'await_deposit_withdraw'  ''then do the deposit/withdraw
Wait next.dep.withdraw.time(r) - time.v days
Call deposit.or.withdraw ''for'' (rp, r)
Call calc.rebalance.order(rp,r) ''since a rebalance may now be necessary
Go decide.next.event.type

End.

Process report

''initial report''

'top'
Wait report.interval days
Call print.status
Go to top
End.

Routine sell.order(new.order)
Define new.order as an integer variable ''order pointer.
Define p to mean last.sale.price

If quantity(new.order) <= 0
    print 1 line thus...
    Negative sell order placed.
    return
Otherwise...
    Let sale.quantity = quantity(new.order)
    Subtract quantity(new.order) from number.of.shares(customer(new.order))
    Add p*quantity(new.order) to dollars.of.cash(customer(new.order))
return
End.
```

DATAFILE

Days per report period: 65

Report periods per simulation: 6662

Number of
trading strategies...
1

Number of prototypes for
investors who REBALANCE...
2

For each rebalancer prototype enter the following information,
on any number of lines...

Value of starting portfolio
Initial ratio of stock value to total value ($stock/$stock+cash)
How often portfolio is reviewed for rebalancing (decimal, days)

How often, on the average, do deposits or withdrawls occur (dec., days)
Minimum deposit (< 0 represents a withdrawl)
Maximum deposit (program assumes uniform distribution between min and max)

Trading strategy number (number between 1 and nr. of trading strategies)

Target fraction of stock to total assets ($stock/$stock+cash)
Low fraction: rebalance if actual is less
High fraction: rebalance if actual is greater

Number of investors with this prototype

```
 100000    .5      5
10   -8000  9000
 1
 0.5  0.48  0.52
 100

  100000    .5      5
   10   -8000  9000
   1
  0.5  0.46  0.55
  0
```

How many CPPI (constant proportion portfolio insurer) prototypes
do you want?

1

For each CPPI investor prototype enter the following information,
on any number of lines...

Value of starting portfolio
Initial ratio of stock value to total value ($stock/$stock+cash)
How often portfolio is reviewed to see if actual fraction is
close to target fraction (decimal, days)

How often, on the average, do deposits or withdrawls occur (dec., days)
Minimum deposit (< 0 represents a withdrawl)
Maximum deposit (program assumes uniform distribution between min and max)

Trading strategy number (number between 1 and nr. of trading strategies)

Length of insurance plan (decimal, days)
Target ratio of $stock to $cushion (= $assets - $floor)
Low ratio: buy stock if actual ratio is this low
High ratio: sell stock if actual ratio is this high

Cushion as a fraction of portfolio value: at start of simulation
Cushion as a fraction of portfolio value: start of new insurance plan

Maximum ratio of stock to assets: max $stock/$stock+$cash
($cash may be negative if this max ratio exceeds 1.0)

Number of investors with this prototype

```
      100000      .5     5
       10       -8000    9000
       1
       65        2.0      1.92       2.08
       .25       .25
       1.0
       100
How many ORPI (option replicating portfolio insurer) prototypes
do you want?

1

For each ORPI investor prototype enter the following information,
on any number of lines...

     Value of starting portfolio
     Initial ratio of stock value to total value ($stock/$stock+cash)
     How often portfolio is reviewed to see if actual fraction is
      close to target fraction (decimal, days)

     How often, on the average, do deposits or withdrawls occur (dec., days)
     Minimum deposit ( < 0 represents a withdrawl)
     Maximum deposit (program assumes uniform distribution between min and max)
     Trading strategy number (number between 1 and nr. of trading strategies)

     Length of insurance plan (decimal, days)
     Expected standard deviation of returns (decimal)
     Ratio of stock market price to exercise price of call option being
        replicated (exp(-0.5*sigma_squared*plan_length_in_years) indicates
        half stock and half cash to start, higher ratio: less insurance)
     Buffer:  Percent deviation from target hedge ratio before a buy or sell
        trade is initiated.

     Number of investors with this prototype


     100000      .5      5

     10      -8000    9000

     1

     65      .20      1.00      0.02

     100

        Random Seed

        0

    0.011278  0.005664  -0.01126  0.006765  0.011494  0.009265
    0.005196  -0.00344         0  -0.01815  -0.01021  0.003914
    0.006912  -0.00017  -0.00774  0.001774  0.005490  0.007574
    0.011014  0.006916  -0.00824  0.003982  0.002414  -0.00636
    -0.00675  0.002615  -0.00069  0.001392  0.001390  0.010758
    0.007038  -0.00034  0.006309  0.006100  -0.00421  0.011163
    -0.0013  -0.00201  -0.00050  -0.01276  -0.00221  0.000340
    0.007498  -0.00947  -0.00751  0.004129  0.000342  0.001198
    0.002395  0.004266  -0.00237  0.003236  0.003226  -0.00084
    -0.00135  -0.00695  -0.01451  -0.01837  0.005827  -0.01422
    -0.00676  0.008965  -0.01386  0.010993  0.001069  0.009615
    0.006525         0  0.002277  0.000699  -0.00331  -0.00175
    0.005619  -0.00331  -0.00332  -0.00966  -0.01331  -0.01133
    -0.02674  0.032155  -0.00941  -0.00274  0.021635  0.014716
    -0.00035  0.010615  0.011029  0.010389  0.006169  -0.00664
    0.007887  0.013780  -0.00218  0.011772  -0.00315  0.003168
    -0.00565  0.010531  0.005955  0.012004  -0.00292  0.006029
    0.006317  0.004668  -0.00240  -0.00513  0.011301  0.003991
    0.000635  0.002065  -0.01252  0.000803  0.004974  -0.00335  0.002403  -0.00319
    -0.00480  0.008216  0.003835  -0.00286  -0.00015  0.006227  -0.00142  0.000476
    0.002223  -0.00649   0.01643  0.006434  -0.00015  0.009669  -0.00231  0.001858
    0.002163  0.005396  -0.00138  -0.00675  0.007113  0.000767  0.001534  0.002451
    0.002750  0.001981  0.002585  0.004854  -0.00015  -0.00573  0.005315  0.001661
    -0.00211  -0.00090  0.004386  -0.00346         0  -0.00619  0.001064  0.004860
    0.003023  0.000904  0.001656  -0.00481  -0.00558  0.001367  -0.00697  0.000152
    -0.00702  -0.01107  -0.00295  0.009672  0.000309  0.001699  0.006322  0.001072
    0.002755  0.002442  0.003654  -0.00470  0.005030  -0.00485  -0.00213  0.007331
```

-0.00151	0.005163	0.000302	0.002869	0.004216	-0.00149	-0.00015	0.004206
-0.00014	0.007779	0.007274	0.006337	0.003515	-0.00102	-0.00233	0.007029
0.004653	0.000723	-0.00318	-0.00043	0.004935	0.001011	0.003318	0.002732
0.000573	-0.00086	-0.00071	0.002153	0.002435	0.002858	-0.00199	-0.00713
-0.00129	0.008208	0.004856	0.002416	-0.00056	-0.00383	0.003133	-0.00255
0.000569	-0.00469	0.002572	0	-0.00057	-0.00114	-0.00214	0.002003
0.004570	0.006682	-0.00155	0.000141	-0.00240	0.000709	-0.00240	-0.00667
0.001286	0.005426	-0.00255	0.000284	-0.00441	0.001143	0.000999	-0.00114
0.003428	-0.00071	-0.00227	-0.00899	-0.00489	0.004343	-0.00735	0.008713
0.006910	0.004003	-0.00683	0.004301	-0.00214	-0.00186	-0.00172	-0.00631
-0.00086	-0.00303	-0.00638	-0.00204	-0.00658	0.000147	0.005448	-0.00029
0.004101	0.001896	0.008300	-0.00158	-0.00086	0.003329	0.005916	0.006598
-0.00299	0.000857	0.006569	0.001560	0.002833	0.003955	0.004361	0.001541
-0.00069	-0.00083	-0.00126	0.003506	0.003075	0.002090	-0.00542	0.007270
0.001665	0.004711	0.002206	-0.00027	0.004956	-0.00219	-0.00356	0.005649
0.002877	-0.00068	0.000273	-0.00136	0.000684	-0.00437	0.005769	0.001092
-0.00463	0.00466	-0.00559	-0.00850	-0.00193	-0.00596	0.007252	0.001107
0.007330	0.000274	-0.00205	-0.00137	-0.00578	0.000277	0.000969	0.000415
0.001383	0.007873	0.003974	0.000819	0.000818	-0.00572	0.000548	0.003835
0.009961	0.006350	-0.00026	-0.00886	0.002845	-0.00243	-0.00514	-0.00871
0.003433	0.004106	0.002181	-0.00394	0.000819	-0.00463	-0.00822	-0.00718
0.000974	0.009179	-0.01295	-0.02806	0.039793	-0.00179	0.013563	0.005872
-0.00054	0.002445	0.006504	-0.00376	-0.00054	0.000405	-0.00121	0.000135
0.002029	0.003240	0.005922	-0.00147	-0.00308	-0.00161	-0.00632	0.002167
0.004731	0.001614	0.001612	0.006169	0.005465	0.000928	0.002251	0.000264
0.004095	0.003684	-0.00052	-0.00026	0.001836	0.003666	-0.00117	0.000130
-0.00195	0.002748	0.005351	0.000778	0.000259	-0.00038	0.000259	-0.00609
0.000913	0.004432	-0.00090	-0.00116	-0.00169	0.002345	0.003249	-0.00168
0.003634	0.003103	-0.00064	-0.00051	-0.00025	0.000129	0.001032	0.000902
0.000773	0	0.002446	-0.00321	0.002319	0.002185	0.003206	-0.00191
-0.00012	0.003202	0.000255	0.003319	0.004580	0.001646	0.000758	0
0.002274	0.000756	-0.00100	-0.00479	0.000126	-0.00177	0.002284	0.002785
-0.00063	-0.00202	0.003292	0.005805	0.003011	0.001000	-0.00349	0.000125
-0.00062	0.001882	-0.00100	0.002757	0.00125	0.001373	0.004364	-0.00062
0.000496	-0.00062	-0.00136	-0.00783	-0.00501	0.006931	-0.00250	-0.00301
0.008935	0.003742	0.005095	0.002225	0.001110	-0.00184	-0.00123	0.003214
-0.00234	-0.00135	0.001855	-0.00357	-0.00520	0.004483	0.000495	0.000371
-0.00210	-0.00211	-0.00161	0.001370	-0.00323	-0.00511	-0.00263	-0.01031
0.004449	-0.00480	0.006358	0.003790	0.003650	-0.00163	0.004648	0.005376
0.005099	-0.00024	0.001237	0.002719	-0.00419	0.003590	0.001850	0.003078
0.002209	0.000612	0.0071	0.004011	0.004600	0.001687	0	0.001203
0.001682	-0.00059	-0.00300	0.003371	0.003599	0.004423	-0.00321	-0.00238
-0.00023	-0.00047	-0.00023	-0.00455	-0.00276	0.000844	0.002050	0.001083
-0.00216	-0.01253	0.001586	-0.00913	0.006393	-0.00097	-0.00024	0.005014
0.002920	-0.00072	0.000121	0.000485	-0.00097	-0.00461	0.001586	-0.00194
-0.00573	-0.00147	0.004672	0.003549	-0.00195	0.004277	0.001582	0.003037
0.002422	0.001329	0.002172	0.000602	0.004211	-0.00275	-0.00264	0.002891
0.006607	-0.00369	0.004552	0.000357	0.000238	0.001072	0.002500	0.000831
-0.00047	-0.00071	-0.00118	0.003330	0.004504	0.000589	0.000118	0.002830
0.002116	0.000234	-0.00328	-0.00200	-0.00636	0.006884	0.001178	0.002943
-0.00093	-0.00188	0.002354	-0.00164	0	-0.00364	0.000472	0.001534
0.003770	-0.00046	0.000235	0.000821	-0.00046	-0.00410	0.002828	0.001292
0.000234	0.005163	0.004436	0.002208	-0.00046	0.001160	-0.00324	-0.00313
-0.00338	-0.00327	-0.00868	-0.01030	0.002872	0.004654	0.002019	-0.00023
-0.00391	-0.00642	-0.00011	0.002516	-0.00251	-0.00275	0.003965	0.004189
0.004648	0.001067	-0.00059	-0.00213	0	-0.00095	-0.00309	0.005846
0.005338	-0.00613	0.004749	0.003072	0.004358	0.001290	0.000351	0.002459
0.002686	0	0.004310	0.003247	0.001618	-0.0003	-0.00092	0.002542
0.001383	0.000921	0.003335	0.002866	0.000914	0.000228	-0.00034	0.000913
-0.00068	-0.00319	-0.00389	0.003335	-0.00894	-0.01064	0.007365	-0.00116
-0.00464	0.001167	0.003264	0.001859	0.004987	0.006117	0.000344	0.002637
-0.00205	0.001719	-0.00160	-0.00320	-0.00206	0.000345	-0.00161	-0.00173
0.004159	0.003567	0.000344	-0.00126	-0.00126	-0.00241	0.000345	-0.00011
0.001151	0.001840	-0.00287	-0.00736	-0.00197	0.001976	-0.00046	0.001857
0.002432	0	-0.00034	0.000578	0.005661	0.005974	0.004339	0.001137
0.002271	-0.00102	0.004084	-0.00056	-0.00180	0.005436	0.001126	0.000112
0.001687	-0.00044	-0.00078	0.002024	0.001346	0.003137	0.002234	0.002340
-0.00077	-0.00211	-0.00122	0.004355	0.003669	-0.00188	-0.00621	-0.00089
0.002347	-0.00546	-0.00482	-0.00743	0.005789	-0.00338	-0.00520	0.006602
-0.00791	-0.00718	-0.00218	0.002416	-0.00264	-0.01093	-0.01035	-0.00364
0.004602	-0.01304	0.005713	0.008403	0.006338	-0.00466	-0.00339	0.001881
-0.00633	-0.01310	-0.00598	-0.01757	0.009926	0.020749	0.004279	0.008049
-0.00199	-0.00376	0.008503	0.003747	-0.00023	-0.00116	0.003271	-0.00174
-0.00035	-0.00070	-0.01261	-0.00567	-0.00261	0.002623	-0.00023	-0.00214
0.001907	0.007735	0.006731	0.001994	0.000468	0.003861	0	0.003263
-0.00244	0.000116	0.003027	0.002902	0.004514	0.001152	0.001956	-0.00057
-0.00229	-0.00115	-0.00149	0.001733	0.001153	0.003801	0.000688	0.000114
-0.00045	0	0.005506	0.004677	0.003406	0.003395	0.002594	0.002587
0.002917	-0.00391	0.005503	0.005585	0.000333	0.000333	-0.00299	0.004565
-0.00399	0.001780	0.006998	-0.00242	-0.00453	-0.00066	-0.00066	0.002002
0.006105	-0.00099	-0.00077	0.004200	0.005723	-0.00021	-0.00010	-0.00164
0.002083	0.003282	0.001309	-0.00021	0.001743	0.000435	-0.00337	0.005781
0.003362	-0.00324	0.002277	-0.00205	0.000867	0.001625	-0.00097	-0.00151
-0.00325	-0.00108	0.003049	0.004776	0.000864	-0.00237	0.002056	-0.00410
0.000216	-0.00650	0.001527	0.001743	0.000978	-0.00249	-0.00206	-0.00120
-0.00316	0.000657	-0.00745	0.008831	-0.00120	0.003067	0.002621	0.000326

0.000544	0.001523	0.001086	-0.00043	-0.00466	0.003928	0.003043	-0.00108
-0.00726	0.000109	0.003059	0.004247	0.002494	-0.00270	0.000867	0.006395
0.002261	0.000859	0.00204	0.000857	-0.00235	0.001824	0.001499	0.002887
0.001919	-0.00276	-0.00352	0.001178	0.002567	0.001494	-0.00159	-0.00032
-0.00384	-0.00460	-0.00775	0.004014	0.001296	0.006583	0.003538	-0.00042
0.005451	-0.00244	-0.00021	-0.00170	-0.00512	-0.00010	-0.00536	-0.00269
-0.00584	-0.00424	-0.00644	0.002750	0.000877	-0.01271	-0.01010	0.003589
-0.00257	-0.01344	0.001590	0.008845	0	-0.00123	-0.01125	-0.00569
0.005838	0.003528	0.004083	0.007568	0.002914	-0.00368	0.001795	0.002799
0.000893	-0.00390	-0.00548	0.005068	0.007957	0.009117	0.00606	0.002737
0.002184	0.000326	-0.00370	0.000984	0.003605	0.001306	-0.00445	-0.00010
0.005569	0.003692	-0.00162	-0.00205	-0.00097	-0.00250	-0.00686	-0.00076
-0.00175	-0.01155	-0.00511	-0.01633	-0.00102	-0.01730	0.008804	0.001722
-0.01146	-0.00881	-0.01240	-0.00924	0.017816	-0.00117	0.004822	0.009013
0.006612	0.003457	0	0.002986	-0.01374	-0.00034	-0.00162	0.001163
-0.00743	-0.00690	0.001178	0.006711	0.010994	0.004511	0.002764	-0.00390
-0.00299	0.000462	-0.00034	0.002659	0.001614	-0.00403	0.000924	-0.00577
-0.00476	-0.00945	-0.00141	0.010266	0.002453	0.014448	0.003675	0.002632
-0.00182	-0.00651	-0.00667	0.006025	0.002994	-0.00103	-0.00758	-0.00949
0.000117	-0.00128	-0.01849	-0.00155	0.004779	-0.00392	-0.00202	-0.01543
0.000243	0.009959	0.009380	0.000834	-0.00297	-0.00310	-0.00455	-0.00108
0.001806	-0.00517	-0.01341	-0.00551	-0.01256	-0.00673	-0.01733	-0.00166
0.012290	-0.01277	-0.02113	-0.02460	0.017845	0.016346	0.007782	-0.00360
-0.00594	-0.00805	-0.00379	0.003155	0.021234	0.005262	0.010342	0.012005
-0.00112	-0.00500	-0.00691	-0.01682	0.002959	-0.00346	0.002446	0.003082
-0.01267	-0.01037	0.003276	-0.02168	0.002670	-0.00545	-0.00856	-0.01147
0.018169	0.005098	0.028434	-0.00194	-0.00377	0.011357	0.015618	-0.00800
-0.00269	0.004496	0.002941	0.006120	0.008618	0.008167	0.000123	-0.00049
0.007606	0.000866	-0.00395	0.003103	-0.00099	0.008051	0.006266	0.000610
-0.00695	0.003932	0.008324	-0.00691	-0.00660	-0.01439	-0.00524	0.006778
0.007979	-0.00173	-0.00359	0.000373	-0.00459	0.000624	0.001372	0.007477
0.010885	0.004038	0.001096	0.010469	-0.00325	-0.00108	-0.01210	-0.00073
-0.00379	-0.00381	0.005187	0.003809	-0.00269	-0.00576	-0.00481	-0.00297
-0.00049	0.000622	0.002114	0.013035	0.007107	0.007666	0	0.007970
0.005271	0.007388	-0.00260	0.011030	0.006452	0.000349	0.002913	0.003717
0.001389	-0.00762	-0.00046	0.004078	0.005803	-0.00057	-0.00207	0.003471
0.007264	-0.00206	-0.00263	0.008855	-0.00410	0.001488	0.001028	0.006736
0.001134	-0.00464	0.000341	-0.00557	-0.00068	0.001259	-0.00045	-0.01086
0.003701	0.010371	0.005474	0.001474	-0.00215	0.000681	0.001247	0.002945
0.004066	-0.00517	-0.00090	0.009507	0.010090	0.001776	-0.00055	-0.00221
0.002777	0.007645	-0.00076	0.000440	-0.00198	-0.00033	-0.00551	-0.01064
-0.00022	0.006388	0.001670	-0.00644	-0.01253	0.007252	-0.00112	0.007659
0.010842	0.007077	0.008674	0.000870	0.001849	0.002062	0.003466	0.005290
-0.00096	0.008492	0.002131	-0.00180	-0.00181	0.002562	0.004365	0.001272
0.001482	-0.01036	-0.00267	0.004284	-0.00288	-0.00823	0.004638	-0.00386
-0.00269	-0.00497	-0.00434	-0.00479	-0.01150	0.011199	-0.00230	-0.00538
-0.01558	0.012909	-0.00487	-0.01514	0.020355	0.007536	0.005389	0.001750
0.005242	0.006301	-0.00237	0.000974	0.000540	-0.00032	-0.00075	-0.00259
-0.00249	0.000326	-0.00391	-0.00371	0.000109	-0.00503	-0.00231	0.002978
0.004949	-0.00043	0.004051	0.003926	0.004671	-0.00086	0.000216	0.003462
0.000107	0.008086	0.001604	0.002135	0.002024	-0.00329	0.000106	0.003413
0.003083	0.001483	0.002751	0.006543	0.004299	-0.00125	0.001777	-0.00260
0.001150	0.000940	-0.00261	-0.00397	-0.00535	0.001373	-0.00232	0.000846
0.001585	-0.00559	-0.00541	-0.00138	-0.00555	-0.00418	-0.00064	0.002590
0.002045	0.006124	0.000427	0.005657	0.001910	-0.00063	0.000318	0.001907
0.00476	0.010527	0.002187	0.000727	0.002700	-0.00372	-0.00041	0.006449
0.002584	0.006082	-0.00850	0.00031	0	-0.00082	-0.00403	0.003425
-0.00227	0.002488	0.006103	0.002570	-0.00687	-0.00485	-0.00643	0.002611
-0.00781	-0.00262	0.002631	0.001889	-0.00052	-0.00440	-0.00568	0.001059
0.004443	0.000210	-0.00179	-0.00938	-0.01267	-0.00399	-0.00606	-0.00326
-0.00371	0.004937	0.006769	-0.00260	-0.00630	0.004048	0.009154	0.002375
-0.01260	0.015821	0.005907	0.002669	0.002875	0.003185	0	-0.00497
0.005319	0.006349	0.001366	0.004305	-0.00115	-0.00115	-0.00314	-0.00115
0.003473	0.001363	-0.00460	-0.00273	-0.00147	0.005495	0.002417	-0.00188
0.000630	0.006823	-0.00020	0.006048	-0.00373	-0.00457	-0.00324	0.006082
0.007087	-0.00124	0.000207	0.001036	0.001035	-0.00310	-0.00622	-0.00187
-0.00083	-0.00334	-0.01270	-0.00393	-0.00523	0.001395	0.001607	-0.00107
-0.00492	-0.00699	0.003469	-0.00313	-0.00433	0.000326	0.001741	-0.01260
-0.01144	-0.00879	0.012012	0.001775	-0.00376	0.003890	0.005758	0.004514
-0.00383	-0.00781	0.003881	-0.00497	-0.00799	-0.00279	-0.01335	-0.00227
0.017555	-0.00179	-0.00078	0.012355	0.001109	-0.00221	-0.01899	0.008831
0.005499	-0.00669	-0.00011	-0.00730	0.001018	0.000226	0.005540	0.008208
-0.00100	0.007033	0.025277	0.001730	0.008959	0.003958	-0.00586	0.017794
0.007583	0.008989	0.000621	0.000310	0.001966	0.002789	-0.01266	-0.00552
0.012169	0.001451	0.003105	0.002992	0.002571	0.001333	0.003893	0.006328
0.000710	-0.00314	0.005592	0.000101	-0.00525	0.001118	-0.00314	-0.00071
-0.00050	-0.00479	-0.00717	-0.00464	0.004976	0.002579	-0.00216	0.001856
-0.00164	0.006495	0.003073	0.007761	0.013275	0.003900	-0.00488	0.007608
0.006160	0.001382	0.002465	-0.00403	-0.00118	-0.00988	-0.00139	0.005200
0.001492	-0.00268	-0.00308	-0.00099	-0.00400	-0.00180	0.003420	0.011730
0.010207	0.002844	0.001565	-0.00048	-0.00078	-0.00547	-0.00255	-0.00966
-0.01124	-0.00120	-0.01280	0.004084	-0.00701	0.000921	-0.00470	-0.00668
0.002276	0.00413	-0.00215	-0.00030	0.010308	0.005305	-0.00466	0.006219
0.003242	-0.00040	-0.00262	-0.00010	0.002533	-0.00131	-0.00070	0.001215
0.004653	0.007047	0.007198	0.004566	0.000296	-0.00493	-0.00208	0.003382
0.003767	0.002568	0.000886	0.000689	0.005705	0.003423	-0.00224	-0.00048

0.003518	0.001850	0.004666	0.003580	-0.00009	0.000385	-0.00433	-0.00106
0.001357	0.002032	0.004636	0.007787	0.001621	-0.00400	-0.00698	0.003466
-0.00287	-0.00577	0.001064	-0.00338	0.000388	0.001648	0.002227	0.004347
0.006445	0.004874	0.000665	0.005513	0.001323	0.002077	-0.00160	0.003114
0.001693	0.007325	0.004661	0.005660	-0.00230	-0.00092	-0.00324	0.002414
-0.00250	-0.00250	-0.00065	0.002422	-0.00446	-0.00410	0.002906	-0.00588
-0.01062	-0.00161	0.001047	-0.00389	-0.00897	0.000578	0.000673	0.000577
-0.01461	-0.01219	-0.00414	0.004166	-0.00286	-0.00525	0.007271	0.004845
0.005510	-0.00146	-0.00333	-0.00059	0.003443	0.004412	-0.00048	0.000195
0.000097	0.000976	0.000390	0.004485	-0.00116	0.000291	0.002720	0.003294
-0.00009	0.001159	-0.00019	0.000772	-0.00096	-0.01129	-0.01015	-0.00739
-0.00854	-0.01192	-0.00628	0.004796	-0.00314	-0.00010	0.002547	0.009554
0.003926	-0.01012	-0.00050	0.003446	0.003333	-0.00271	-0.00666	-0.00396
0.002551	0.002442	0.007310	0.006350	-0.00210	-0.00130	0.001608	0.007324
0.007072	0.004055	-0.00088	-0.00631	-0.00099	-0.00784	0.002502	0.008787
0.005246	0.000984	-0.00078	-0.00039	-0.00886	0.001490	0.004564	-0.00671
0.002187	0.000198	0.004662	0.004443	0.003047	0.007448	0.008755	-0.00173
0.004733	0.003557	0.004694	-0.00181	0.004108	-0.00047	-0.00152	0.004290
0.007784	-0.00292	0.000850	-0.00915	-0.00885	0.004133	0.001244	-0.00009
-0.00219	-0.00756	-0.00299	0.001937	-0.00502	-0.00301	-0.00038	0.001657
-0.00622	-0.00900	-0.00770	-0.00866	-0.01295	0.003969	-0.00334	-0.00376
-0.00142	-0.00582	-0.00586	-0.00455	0.011327	-0.00318	0.002474	0.000822
0.003904	0.003786	0.008768	0.006771	-0.00582	-0.01413	-0.00768	-0.01548
0.004088	-0.01273	-0.00327	0.009974	0.006093	-0.00845	-0.01506	-0.00427
-0.00343	-0.00905	-0.01903	-0.00809	0.005029	0.021127	0.017859	-0.00513
0.004516	0.005459	0.000745	-0.00053	-0.00617	-0.00781	0.000755	0.006904
0.007071	0.006063	0.005287	0	0.002945	0.005977	-0.01032	-0.00663
0.002014	0.004233	0.006533	0.000314	-0.00586	-0.00821	-0.00594	-0.01003
0.007335	0.016812	-0.00768	0.001379	0.005511	0.000843	-0.00337	0.002853
0.003055	0.004622	0	-0.00135	-0.00764	-0.00643	-0.00796	-0.00310
-0.00644	0.007782	-0.00053	0.001824	-0.00289	0.006230	-0.00683	0.005697
0.010581	0.012162	0.000209	0.006790	-0.00114	0.002077	0.007671	0.006481
-0.00378	0.006771	-0.00183	-0.00285	-0.00870	0.001239	0.003198	-0.00092
0.000617	0.004423	0.000307	0.006040	0.000712	-0.00264	-0.00183	-0.00480
-0.00359	-0.00679	-0.00020	-0.00508	-0.01032	-0.00621	-0.01145	-0.00321
0.003550	0.005789	-0.00628	-0.00611	-0.01079	0.003273	-0.00239	-0.01199
-0.00088	-0.00077	0.000442	0.003203	-0.00297	-0.00905	-0.00579	0.015807
0.008498	-0.00875	-0.00386	0.010528	0.007786	-0.00696	0.003835	0.005021
0.010210	0.004946	-0.00684	-0.00204	0.000539	-0.00302	-0.00757	0.000981
-0.00152	0.000327	-0.00828	-0.01396	0.002007	0.000890	0.000444	-0.00978
-0.01010	-0.00623	-0.00947	-0.01267	-0.00781	0.008586	0.011895	-0.00610
-0.00394	0.005005	0.007876	-0.01045	0.009756	-0.00241	-0.00219	-0.00080
-0.00115	0.012388	0.003659	0.003076	-0.00045	0.015456	-0.00503	0.006749
0.002346	0.005796	-0.00210	-0.00044	-0.00622	-0.01039	0.002711	-0.00067
-0.00405	-0.00532	-0.01081	0.004372	0.002864	-0.00137	-0.00411	-0.00080
0.011380	0.020345	0.001670	-0.00322	0	0.004909	-0.00310	-0.00445
-0.00704	-0.00270	-0.00033	0.000452	-0.00327	-0.00679	-0.00855	-0.00184
-0.00980	-0.00244	0.001867	-0.00524	-0.01300	-0.01459	-0.00325	-0.01582
-0.01460	0.019185	-0.00354	-0.00098	-0.02541	-0.00970	0.011068	0.00453
-0.00488	-0.01057	-0.00954	-0.01695	-0.01424	0.019353	0.000780	-0.01949
-0.02570	-0.01849	0.001247	-0.02768	-0.01366	0.050223	0.025285	0.026001
0.016851	0	0.008736	-0.01477	-0.01538	0.001575	-0.00052	-0.01009
-0.01364	-0.00322	0.002290	0.023796	-0.00196	0.006710	0.007058	-0.00532
-0.02453	-0.01056	0.000676	-0.00743	-0.00789	-0.00233	0.004400	-0.00164
-0.01563	-0.00766	0.024849	0.014520	0.006886	-0.00120	-0.00080	0.010884
0.014754	0.017684	0.000386	-0.00952	0.000649	0.012592	-0.00230	-0.00218
0.001545	0.003471	0.000384	-0.00025	-0.01319	0.002207	-0.00012	-0.00129
0.002594	-0.01397	-0.00498	-0.00527	-0.00875	0.005618	0.001995	0.011549
0.009973	0.008835	0.020608	0.022084	0.001605	0.001109	-0.00160	0.009620
-0.00415	-0.00699	0.000123	0.013957	0.009014	0.002535	-0.00301	-0.00591
0.002673	-0.00545	-0.00865	0.005285	0.006113	0.004010	-0.00859	-0.00305
0.014327	0.013038	0.000715	-0.00131	0.005246	-0.00106	0.001306	0.009962
0.015382	0.004394	0.000460	-0.01081	-0.01012	-0.01069	-0.00130	0.001546
0.005463	-0.00437	-0.01340	0.005893	0.000239	-0.00334	0.004677	-0.00549
-0.00228	0.003729	-0.00083	-0.00131	0.003123	0.008502	0.002018	-0.00343
0.001426	0.005343	0.001417	0.002830	-0.01034	-0.00926	-0.00155	0.001201
-0.00659	0.001449	0.009769	0.006211	0.006410	0.003656	0.009871	0.014779
0.003096	0.011546	0.004746	0.006299	0.005365	-0.00522	0.000782	0.004244
0.003781	-0.00509	-0.00155	0.000669	0.003566	0.001999	-0.00310	0.001111
0.000666	0.005660	0.005297	0.010868	0.002063	-0.00130	-0.01085	0.007131
0.005991	0.000324	-0.00205	-0.00227	0.008045	-0.00172	0.002593	0.002478
0.004084	0.003746	0.000213	0.004372	0.007325	0.004215	0.003253	-0.00732
0.003372	0.007037	0.005632	0.000103	0.002074	-0.00010	0.003208	0.005364
0.000615	-0.00123	0.005339	0.005310	0.002336	-0.00466	-0.00651	-0.00840
-0.01054	0.003865	0.006660	0.001964	-0.00175	0.002584	-0.00020	-0.00030
0.010005	0.010621	0.004244	0.000805	-0.00160	0.000906	0.001811	0.011449
0.004964	-0.00088	0.000692	-0.00177	-0.00386	-0.00337	-0.00658	-0.00010
0.003413	0.000800	0.002299	0.000498	0.000797	0.001693	0.002287	0.007143
0.004630	0.001176	0.007639	0.000971	0.003787	0.001451	-0.00028	0.005024
-0.00384	-0.00241	0.001935	0.004731	-0.00105	0.004329	0.003640	-0.00133
-0.00649	-0.00634	0.004840	-0.00009	-0.00529	-0.00348	-0.00495	0.00254
0.002728	-0.00204	-0.00467	-0.01487	0.001390	0.002380	0.002374	-0.00315
-0.00851	-0.00659	0.001206	-0.00190	0.002314	0.005721	0.007584	0.000495
0.002871	-0.00157	-0.00810	-0.00029	0.003489	0.004272	-0.00841	0.000997
0.001993	-0.00019	-0.01522	-0.01111	-0.00286	0.008402	-0.00243	-0.00183
-0.00255	0.011049	0.008905	0.000802	0	-0.00020	0.002806	0.002998

0.003488	0.001291	-0.01309	-0.00281	0.000604	-0.00171	-0.00181	0.003942
-0.00040	-0.00171	-0.00171	-0.00272	-0.00902	-0.00726	-0.01081	-0.00458
0.003975	-0.01511	-0.00656	0.002130	0.001700	-0.00763	0.000106	0.011973
0.014155	-0.00322	0.032082	0.012454	-0.01390	-0.00446	0.001731	0.010271
0.010670	0.000099	-0.00169	0.002394	-0.00955	-0.00492	0.000404	0.002220
0.014100	0.004568	0.001878	-0.00532	-0.00376	-0.00348	-0.00729	0.004328
-0.00110	0.003010	-0.00280	-0.00341	-0.00875	-0.00091	-0.00233	-0.00539
0.002663	0.000204	0.004494	0.005999	0.002830	-0.00100	0.007163	0.002003
-0.00659	-0.00201	0.004134	-0.00542	-0.00908	-0.00346	-0.00449	-0.00359
-0.01391	-0.00052	-0.00094	-0.00429	-0.00378	-0.01002	0.001812	0.002873
-0.01517	0.004094	0.018566	-0.00126	-0.00348	-0.00074	0.000741	-0.01111
-0.01381	0	-0.00336	0.009803	0.001510	-0.00775	-0.00564	-0.00895
-0.00693	0.001885	0.017823	0.015988	0.006209	0.016491	-0.00732	0.023407
-0.00566	0.003730	0.000516	0.000412	0.007528	0.002866	-0.00306	0.008907
0.012177	0.005213	0.012866	0.002461	-0.00609	-0.00434	0.002084	0.009905
0.002550	-0.00420	0.003045	-0.00411	0.004131	0.009599	0.004268	-0.00038
-0.00144	0.003193	-0.00057	-0.00579	0.003883	0.002998	0.003375	-0.00163
0	-0.00221	-0.01041	0.002047	-0.00272	0.009756	0.006376	-0.00211
0.000673	0.006441	-0.00038	0.002102	0.006484	-0.00757	0.007733	0.000378
-0.00485	-0.00466	0.004206	0.005617	-0.00028	-0.00293	0.000094	0.000854
0.000664	0.006922	0.000094	0.003578	0.007319	-0.00027	0.005777	0.007689
0.000919	0.000826	-0.00018	-0.00523	-0.00959	0.002608	0.001301	-0.00232
0.003907	-0.00305	-0.00836	0.001405	0.008517	-0.00213	-0.00204	-0.00121
-0.00634	0.006667	0.002611	0.005954	0.008139	0.003853	0.001827	-0.00155
0.002832	0.003826	-0.00245	-0.00063	-0.00300	0.002374	-0.00519	-0.00146
-0.00137	-0.00642	-0.00989	-0.00214	0.001496	0.005791	-0.00910	-0.00571
-0.00084	0.002453	0.003576	-0.00459	-0.01319	0.006492	0.003320	0.005767
0.004512	-0.00187	0.002156	0.009823	0.009634	0.006515	0.000820	0.004827
0.001359	0.001810	-0.00280	-0.00743	0.001460	0.000364	-0.00829	-0.00560
-0.00517	-0.00343	-0.00391	0.001403	0.005046	0.007810	0.000461	-0.00073
-0.00230	0.004162	0.002118	-0.00101	-0.00377	-0.00729	-0.00102	-0.00325
-0.00186	0.002995	0.003266	0.005675	0.008695	-0.00320	-0.00533	-0.00730
-0.00400	-0.00570	0.004892	-0.00861	-0.00047	0.002929	-0.00310	0.008033
0.011813	-0.00296	-0.00065	-0.00232	0.000932	0.000093	0.009404	0.008210
0.007777	0.002633	0.001630	0.000723	0.001535	0.001713	0.008104	0.005359
-0.00435	-0.00356	-0.00286	0.003772	-0.00035	0.006176	-0.00133	-0.01104
-0.00315	-0.00397	0.001633	0.001449	0.004702	0.003780	-0.00251	-0.00611
-0.00235	-0.00126	-0.00581	-0.00949	0.003964	0.000275	-0.00110	-0.00183
-0.00055	0.000460	-0.00156	0.000830	-0.00433	0.001110	0.013774	0.006292
0.001812	-0.00352	0.001270	-0.00190	-0.01090	0.006704	0.002554	0.000819
-0.00445	-0.00821	-0.00626	-0.01065	0.006837	0.006418	-0.00129	0.011013
0.010161	0.004168	-0.00081	0.002438	-0.00333	-0.00027	0.008952	0.009768
0.004970	0.008743	-0.00210	-0.00552	0.001323	0.002026	0.001494	0.009218
-0.00391	0.005502	0.003126	0.000346	0.005886	0.005937	0.003165	-0.00469
-0.00214	0.000429	0.001287	0.006085	0.003322	-0.00161	0.003657	0.004999
0.002192	0.002187	-0.00386	-0.00084	-0.00269	0.000169	-0.01150	-0.00479
-0.00335	-0.00724	0.006254	0.004057	0.005416	0.009578	0.008894	0.003946
-0.00142	0.003936	-0.00016	-0.00100	-0.00250	0.006782	-0.00781	-0.00720
-0.00253	0.004570	0.001432	-0.00058	-0.00479	0.000084	-0.01260	-0.00239
-0.00377	-0.00155	0.001726	-0.01094	-0.00357	-0.00104	0.001925	-0.00690
-0.00439	0.013432	0.012033	0.006203	-0.01438	-0.00564	0.004630	0.003652
-0.00615	-0.00217	-0.01118	-0.00857	-0.01149	0.007033	-0.00564	0.011076
0.003562	0.012602	0.003067	-0.00192	-0.00385	0.000615	0.005445	0.004367
-0.00747	-0.00508	-0.01206	-0.00196	-0.01304	-0.01493	0.000367	0.008817
0.015659	0.000537	0.009765	-0.01055	-0.01201	-0.00853	-0.00430	-0.00229
0.007003	0.014458	0.012177	0.004188	-0.00088	-0.00444	-0.00571	-0.00448
0.005408	0.005648	-0.00534	-0.01416	-0.01500	0.005076	-0.01524	-0.00242
0.001215	0.012418	0.016508	0.007076	-0.00423	0.006514	-0.00728	-0.00814
-0.01250	-0.02098	0.006326	-0.00131	-0.00817	-0.01610	-0.01088	0.008274
0.004730	0.029499	0.007466	-0.00398	-0.01488	-0.00906	-0.00971	-0.00923
0.016024	-0.00296	0.014667	0.011243	-0.00308	0.014901	-0.00637	-0.01115
-0.01221	-0.01427	0.003764	0.004327	-0.01177	0.004747	-0.01398	0.010269
0.003097	0.010326	-0.00410	-0.01304	-0.01000	-0.00088	-0.00491	0.008491
0.013510	0.022024	-0.00283	-0.01336	0.015179	0.000473	0.005959	0.001880
0.005537	0.003546	0.005766	0.013870	0.001915	-0.00236	-0.00310	-0.00942
-0.01284	-0.00149	-0.00168	0.002253	-0.00168	-0.00938	0.000568	-0.00795
-0.01011	-0.00964	0.002920	-0.00698	0.000195	-0.00684	-0.00708	-0.00356
0.013727	-0.00284	0.007872	0.005858	0.009803	-0.00144	0.003561	0.002494
0.001243	0.004873	-0.00370	-0.00868	-0.00606	-0.00155	0.002911	0.010448
-0.00277	-0.00364	0.020333	0.008311	0.004121	0.001492	0.006427	0.007218
0.002297	-0.00595	-0.00202	0.005359	-0.00009	-0.00340	0.013282	0.003459
-0.00090	-0.00826	0.017121	0.003150	-0.01247	0.001272	-0.00199	0.000363
0.001909	-0.00961	0.005404	0.004738	0.002085	0.007963	-0.00206	-0.01637
-0.00951	-0.00554	-0.00575	-0.01447	-0.00530	0.008003	0.011531	-0.01607
-0.00816	-0.00076	-0.01830	-0.00019	0.014156	-0.03051	-0.02035	0.011149
-0.00320	-0.02876	-0.00911	0.020376	-0.00348	-0.01387	-0.02146	-0.00330
-0.01527	0.024522	0.022135	0.014920	-0.01949	-0.02571	-0.01271	0.009850
-0.00578	0.021455	0.000844	-0.00284	-0.01068	-0.00684	0.030570	0.020889
-0.00204	0.000102	0.001332	0.021703	-0.00901	-0.00839	-0.01988	-0.02808
-0.01102	0.013746	-0.00256	0.008670	0.015281	0.017037	-0.01788	-0.00167
0.012054	0.005385	-0.00257	-0.00196	-0.00558	-0.00083	0.010936	-0.00504
-0.01294	-0.02129	-0.00310	0.002795	0.000429	-0.01039	-0.01808	0.003088
0.000439	-0.00032	0.014513	-0.00162	0.014329	0.013591	0.007179	-0.00377
0.010207	0.004166	-0.00186	-0.00717	0	0.018737	0.006781	-0.01061
0.008665	0.011249	0.002730	0.005950	-0.00090	-0.00371	-0.01238	-0.00836
0.003497	-0.00235	-0.00071	0.003803	0.003174	-0.01388	-0.01832	-0.00885

-0.00776	0.001072	0.010498	0	-0.01399	-0.01053	0.006302	-0.00226
-0.00303	-0.00075	0.017490	0.007473	0.004451	-0.01086	-0.00394	-0.01681
-0.01644	-0.00808	0.006810	-0.00199	0.003444	0.021149	-0.00140	-0.00868
-0.00186	0.003731	0.001968	0.014404	-0.01602	-0.00885	0.000330	-0.00264
-0.00807	-0.01683	-0.00396	0.000569	-0.00932	0.002296	0.014778	-0.00237
-0.01674	0.006214	-0.00171	0.020852	0.011672	0.001885	0.018270	0.006415
0.005942	-0.00880	-0.00238	0.003041	-0.01126	-0.01380	-0.00655	-0.00681
-0.00709	-0.00850	0.002629	0.014710	-0.01472	-0.01551	-0.00359	0.000232
-0.01999	-0.00059	-0.00700	-0.03071	0.004809	-0.01828	-0.00125	0.040806
0.007576	-0.01157	0.010747	0.000955	-0.00286	0.003232	0.010022	0.004016
-0.01188	-0.01881	-0.01771	-0.00543	-0.01478	-0.00706	-0.00203	0.008906
0.015512	0.026452	-0.01306	-0.00870	-0.01372	-0.01579	-0.02242	-0.00560
-0.00825	-0.01453	0.005095	-0.01921	-0.00965	-0.01717	0.008525	-0.01690
-0.00253	-0.01088	0.030861	-0.02259	-0.02595	0.031736	0.007760	-0.02380
-0.00688	-0.00996	-0.02684	-0.02263	0.016257	0.016903	0.005046	0.034997
0.000713	-0.01026	-0.02016	-0.00661	-0.01642	-0.02287	-0.02155	-0.00236
-0.00015	-0.01735	0.000963	0.041867	-0.00169	0.045959	0.029047	0.019343
0.022209	-0.01760	-0.01553	0.011243	0.015596	0.016878	-0.00503	-0.02871
-0.01140	0	-0.00185	0.039092	0.020321	-0.00551	-0.00027	-0.01082
0.027777	-0.00479	0.006153	-0.00398	0.003203	-0.01969	-0.00434	-0.00395
-0.01574	-0.03671	-0.01544	-0.00439	0.004123	0.010560	-0.00101	0.009298
0.006765	0.000428	-0.02658	-0.01380	0.003573	-0.01898	-0.01693	0.009075
0.025609	0.005796	-0.00325	-0.00563	-0.00909	0.016852	0.004735	-0.00368
-0.01093	-0.01419	0.013947	0.008373	-0.00444	0.000297	0.020845	0.024358
0.006834	0.005091	-0.00070	-0.01379	0.016133	0.020233	-0.00413	-0.00871
0.006417	-0.00124	-0.01512	0.001691	-0.00534	0.014710	0.0046	0.012626
0.032748	0.008756	0.016177	-0.01359	0.010103	0.010912	-0.00269	0.017265
-0.00493	0.000891	-0.00343	0.002807	0.017052	0.013638	0.006048	-0.00699
0.006301	0.009454	0.004987	-0.01428	-0.02345	0.010562	0.004977	0.010152
0.017649	0.006383	-0.00789	0.009529	0.007288	0.007710	-0.00694	-0.00912
0.001794	0.012180	0.014747	-0.01023	-0.00928	-0.00865	-0.00263	-0.02362
0.007860	0.018644	0.003110	-0.00584	-0.00863	0	-0.01367	-0.00772
-0.00655	0.007965	0.022842	0.011226	0.004894	0.016868	0.008177	0.003476
0.007505	-0.01088	0.010776	-0.00160	-0.01113	-0.00092	0.006741	-0.00450
-0.00684	0.019383	0.009163	0.012712	0.009639	-0.01598	0.004963	0.005388
0.010830	0.000883	0.010705	0.007534	-0.00932	-0.01072	0.001105	-0.00508
-0.01121	0.003705	0.013312	-0.00264	-0.00697	-0.00033	0.016391	0.015688
0.003348	-0.00312	0.000971	-0.00226	-0.01373	-0.00844	0.001216	-0.00519
0.004884	0.010384	-0.00962	-0.00209	0.018033	0.006411	0.010905	0.006088
0.004565	0.002008	0	0.004008	-0.00357	-0.00706	0.001911	-0.00869
-0.00160	0.015098	0.000105	-0.00158	0.005599	0.004412	-0.01045	-0.01035
-0.00459	-0.00815	-0.01070	-0.01388	-0.00121	-0.00866	-0.00671	-0.00563
0.007257	-0.00090	-0.00856	-0.00954	-0.01055	0.000232	0.000579	-0.00324
0.006161	0.006585	-0.01320	-0.00430	0.008878	-0.00185	-0.01450	-0.02036
-0.00180	0.014565	0.009254	-0.01293	0.005597	0.023333	0.005555	-0.01611
0.006434	0.001976	-0.00672	0.003153	-0.01501	-0.00957	-0.00405	-0.00179
-0.00504	-0.00953	0.003410	0.020517	0.021651	-0.00943	-0.00152	0.009418
-0.00116	0.006422	-0.01345	-0.01364	-0.01120	0.010732	0.025411	0.010820
-0.00126	0.013483	0.004889	-0.00181	0.014170	-0.00201	-0.00056	0.001569
-0.00570	0.010803	0.008238	0.001656	0.005842	-0.01545	-0.00111	0.008692
-0.01237	-0.00089	-0.00302	-0.01066	0.004767	0.007230	0.004486	-0.00245
0.000111	0.005932	0.014688	-0.00164	-0.00076	0.005386	-0.00502	-0.01120
-0.00377	-0.00122	0.001898	0.011259	0.002535	0.003298	-0.00624	-0.01477
-0.01936	0.002739	-0.01161	0.002879	0.002641	0.008934	-0.00317	0.000341
0.002960	0.009535	0.002473	0.003140	-0.00704	-0.00743	0.006693	0.008227
0.008830	-0.00132	-0.00399	0.004678	0.007872	0.018481	0.010261	0.004490
0.006705	0.003912	0.014534	-0.00788	0.016323	-0.00535	0.004036	0.013608
0.005492	-0.00627	-0.00203	0.011934	0.004737	-0.00611	-0.00545	0.016035
0.007491	0.000099	0.003073	0.007313	-0.01501	-0.00926	0.001608	0.008532
0.002986	-0.00516	-0.00578	-0.00622	0.008076	0.015623	0.006804	-0.00479
0.004133	-0.00333	-0.01553	-0.00399	0.003109	0.00539	-0.00576	-0.01060
0.001920	0.010897	0.003892	0.003579	0.009411	-0.01010	-0.01050	0.011222
-0.00059	-0.00406	0.001294	0.001292	0.015192	0.011541	-0.00551	0
-0.00427	-0.00390	0.007450	-0.00515	0.000097	0.012322	-0.00144	-0.01112
-0.00909	-0.00918	-0.00149	0.008483	-0.00732	0.003588	0.007648	0.014096
0.004374	-0.00329	-0.00670	0.001368	-0.00556	0.002650	0	-0.00479
-0.00708	0.005350	-0.00571	0.002775	0.007117	0.011974	-0.00145	-0.00174
-0.00593	-0.00802	-0.00246	0.001681	-0.00079	0.008104	-0.00725	-0.01797
0.000502	-0.00150	0.000402	0.008049	-0.00329	0.003705	-0.00089	-0.00978
-0.00524	0.001723	-0.00060	0.008304	0.013660	0.010206	-0.00480	0.005420
0.015684	0.001447	0.005011	-0.00776	-0.00212	0.00523	-0.00067	-0.00279
0.004157	0.004043	-0.00661	0.005019	-0.00547	0.002800	0.001444	0.009617
0.008763	-0.00217	0.002649	-0.00707	-0.00494	-0.00372	-0.00546	0.000964
0.001059	0.001250	0.000096	-0.00566	-0.00415	-0.00116	0.004954	-0.00241
0.009206	0.002784	-0.00555	-0.00057	-0.00289	0.008889	-0.00335	0.001537
0.000287	0.001726	0.003543	-0.00229	-0.01118	-0.00986	-0.00400	-0.00676
0.007504	-0.00695	0.001579	0.005814	0.008229	0.011174	-0.00134	0.003656
0.006999	-0.00085	-0.00514	0.002394	-0.00344	-0.00335	0.002597	0.010843
0.008828	0.000470	0.014202	-0.00343	-0.00502	-0.00112	0.004307	-0.01249
-0.00519	-0.00123	-0.01016	-0.00134	-0.00769	-0.00251	0.005535	-0.00946
-0.00897	-0.00816	0.012994	-0.01243	0.000297	0.005848	-0.00019	0.002858
-0.00953	-0.00803	0.001100	0.009893	0.006926	-0.00147	0.012695	0.001943
-0.01144	0.004807	-0.01552	-0.01210	-0.00240	-0.00553	0.0084	-0.00401
0.006650	0.001401	0.005697	0.012722	0.000294	0.006573	-0.00614	0.004413
0.007225	-0.00688	-0.00331	0.003819	-0.00361	0.006267	0.007785	-0.00067
0.005701	0.004131	0.001818	-0.00066	0.004205	0.000666	-0.00323	-0.00515

-0.00585	0.005499	0.004701	0.001241	0.011636	0.006694	-0.00402	0.005078
0.005426	-0.00428	-0.01214	-0.00889	0.002482	-0.00009	0.001809	-0.01026
-0.00691	0.007737	-0.00182	-0.00269	-0.00395	0.005129	-0.00847	0.003399
-0.00067	-0.00116	-0.00766	-0.00537	0.001375	0.000981	0.004998	-0.00175
-0.00498	0.000294	0.000098	-0.00284	-0.00856	0.000893	-0.00595	0.005188
0.002977	0.004552	-0.00571	-0.00426	0	-0.00298	-0.00588	-0.00120
0.003417	0.008415	-0.00268	0.004881	0.003172	0.000494	-0.00375	-0.00852
0.006599	-0.00019	0.007650	0.005521	0.001863	-0.00088	-0.00215	-0.00539
-0.00305	-0.00792	-0.00499	-0.00641	-0.00060	0.006969	-0.01153	-0.00121
0.008026	-0.00987	-0.00223	-0.00102	0.004493	0.005388	0.012843	0.000099
0.008386	0.000396	-0.00494	-0.00467	0.003297	-0.00647	-0.01313	-0.01310
-0.00041	0.008753	0.002449	0.002444	0.004977	0.005054	0.005330	0.001500
-0.00619	-0.00311	0.002924	-0.00693	-0.00050	0.003038	0.004443	0.003016
0.005312	-0.00418	-0.00430	-0.01307	-0.00489	-0.00921	0.00248	-0.00762
-0.00155	0.008426	-0.00196	0.009820	-0.00470	0.005142	0.004809	-0.00061
0.003260	0.002843	0.011342	-0.00240	0.002308	0.001201	0.004501	0.003186
-0.00277	0.001592	0.005665	-0.00207	-0.00831	-0.00029	0.003695	-0.00378
-0.0001	-0.00509	0.003514	-0.00140	-0.00240	-0.00100	0.001407	0.005924
0.007686	0.008320	-0.00058	-0.00137	0.000787	-0.00806	-0.00575	-0.01625
0.001520	0.000607	0.002731	-0.00625	-0.00131	0.003761	0.000202	-0.00779
0.000612	0.008873	-0.00768	-0.00285	0.003064	-0.00458	0.000102	-0.00061
-0.00174	0.002871	-0.00173	-0.00399	-0.01110	-0.00093	0.008952	-0.00557
0.004046	0.00062	0.006402	0.002668	0.003070	-0.00744	-0.00935	-0.00352
0.000624	0.004787	0.002589	-0.00330	-0.00652	0.000417	-0.00823	-0.00010
-0.00052	0.003577	-0.00146	0.000734	0.005665	0.007094	0.002175	-0.00733
-0.00364	0.003867	-0.00083	-0.00229	-0.00856	-0.00937	-0.00616	0.001069
-0.00096	-0.00010	-0.01155	0.003139	-0.00377	-0.00747	-0.00687	0.012088
0.002605	0.003032	-0.00302	-0.01072	-0.00700	0.000551	0.009034	0.007752
0.001842	0.005623	0.018606	0.013409	-0.00687	0.006399	-0.00500	-0.00303
0.001786	-0.00083	0.008818	0.004162	0.002072	-0.00672	-0.01551	0.002961
-0.00147	-0.00021	-0.00422	-0.01527	-0.00053	0.001940	0.007422	-0.00021
-0.00074	0.005023	-0.00510	-0.00160	-0.00760	-0.00204	0.005946	0.008060
0.009488	0	0.000633	0.002005	0.001685	-0.01345	-0.00319	-0.00834
-0.01207	-0.01015	-0.00573	-0.00476	0.000891	-0.00144	-0.00289	0.005032
0.007565	-0.00518	-0.00222	-0.00723	0.000112	0.001568	-0.00906	0
0.008579	-0.00100	0.007619	0.002224	-0.00565	-0.00133	0.009273	0.005535
-0.00583	-0.00243	-0.00244	-0.00912	-0.00235	-0.00844	-0.00136	-0.00420
-0.00034	0.000913	0.009698	-0.00870	-0.00775	0.001723	0.001491	0.001488
-0.00628	0.005293	0.005494	0.000569	0.011264	0.000787	0.004497	-0.00257
0.004376	0.007708	0.006873	-0.01134	-0.00356	-0.00122	-0.00548	0.007089
0.001564	-0.00256	-0.00223	-0.00840	0.004521	0.008777	0.001673	0.004232
0.003548	-0.00265	-0.00155	0.009654	0.021323	0.016465	-0.01079	0.004602
0.007244	-0.00211	0.015158	0.009084	0.001862	-0.00991	0.010118	0.008675
-0.00430	-0.01018	-0.00342	0.006254	-0.00352	-0.00301	0.000208	0.013344
0.008950	0.007035	0.005974	0.002516	-0.00983	-0.00507	0.009885	-0.01049
-0.00989	-0.00288	-0.00227	0.002899	0.003923	0.001131	0.008115	0.018443
0.003702	-0.00199	0.000899	-0.00279	-0.00380	0.000200	-0.00090	-0.01145
-0.00935	0.000718	-0.01005	-0.00518	0.002395	-0.00405	-0.01304	0.004016
0.004422	0.001782	-0.00041	-0.00460	-0.00862	0.000530	0.006043	0.004004
0.006927	0.003231	0.000103	0.013818	0.002049	-0.00930	0.012903	-0.00091
-0.00285	-0.00030	0.007368	0.006501	0.004642	0.004621	0.006799	-0.00019
0.022451	0.005732	0.003961	-0.00356	0.004442	0.004711	-0.00803	0.002894
0.000096	-0.00115	0.007703	0.004109	-0.00333	-0.00802	0.004042	0.005752
0.001620	-0.00171	-0.00896	-0.00548	0.000773	-0.00173	0.003775	0.007812
0.008517	0.000379	0.012995	0.001779	0.000093	-0.00607	-0.01166	-0.00932
-0.00873	-0.00658	-0.00780	0.001671	-0.00058	0.000196	0.007461	-0.00935
0.002951	0.005688	0.004096	-0.00349	0.004483	0.002037	0.002420	0.010336
-0.00124	0.008902	-0.00483	-0.00209	-0.01958	-0.01315	-0.00760	-0.01154
-0.01389	0.002348	-0.00702	-0.00184	-0.01315	-0.01499	0.004968	-0.02009
0.039721	-0.01280	0.005961	-0.01029	-0.01407	0.006393	-0.00031	0.003706
-0.01730	-0.00687	0.002378	0.010786	0.007576	0.008790	-0.00251	0.004946
0.003246	0.002088	-0.00875	-0.01471	0.010133	0.016684	-0.00135	0.013416
0.000513	-0.00420	-0.00463	0.004967	-0.00535	-0.00548	-0.00020	-0.00739
-0.01982	0.008561	0.004668	0.000317	0.016893	0.012563	-0.00881	-0.00393
-0.00176	0.006450	0.011061	0.007975	0.005579	-0.00332	0.005364	-0.00563
0.003341	0.008375	0.007605	-0.01221	0.000201	0.002412	0.000300	0.001503
0.007007	-0.00437	0.010283	0.006621	-0.00304	-0.00492	-0.01108	0.000300
-0.00460	-0.01417	-0.00040	-0.00907	0.005043	0.002253	0.003371	0.007433
-0.00060	-0.00141	-0.00060	0.007601	-0.00352	-0.00746	-0.00559	-0.00112
-0.01576	0.001560	0.006439	0.000722	0.011240	-0.00193	0.005824	0.011580
-0.00040	0.001306	0.001705	-0.00130	0.001504	0.008311	0.003674	-0.00554
0.007462	0.004148	-0.00068	-0.00551	0.014251	-0.00351	-0.00088	-0.00431
-0.00679	0.014866	0.002441	0.005942	-0.00077	-0.00300	0.004568	-0.00996
-0.00303	-0.00862	0.001186	0.004543	-0.00412	-0.00049	0.003358	0.006202
0.002935	-0.00478	-0.00205	-0.00039	-0.00078	0.000393	0.000884	-0.01100
-0.01658	0.001514	0.002924	-0.00945	0	-0.00466	0.000815	0.002853
0.015444	-0.00010	0.002101	0.003694	-0.00626	0.000500	0.002902	-0.00169
-0.00939	-0.00030	0.000908	0.001512	0.013088	0.006758	0.004837	-0.00294
0.004138	0.009223	-0.00525	-0.00107	-0.00107	-0.00519	0.000197	0.000492
0.004526	0.005387	-0.00535	-0.00421	0.006000	0.005182	0.00107	-0.00893
0.000980	0.003330	0.011617	0.008203	-0.00258	-0.00537	-0.00916	-0.00360
0.004104	-0.00885	-0.00137	-0.00078	0.002066	-0.00225	0.003740	0.010885
0.000194	0	0.000484	0.006398	0.003467	-0.00067	-0.00057	0.002499
0.012943	0.003123	-0.00462	0.008626	0.009586	0.000931	0.006789	-0.00147
0.001942	0.004893	0.000735	0.000734	-0.00330	-0.00027	0.004972	-0.00109
0	0	0.002751	-0.01719	-0.00967	0.004229	0.007580	0.004737

-0.00610	0.002883	0.000278	0.008437	0.000735	-0.00771	0.002592	0.020594
-0.00036	-0.00778	0.000638	0.002553	0.002273	-0.00807	-0.00695	0.009487
0	0.005292	0.009984	-0.01249	-0.02957	-0.01247	-0.00237	-0.00533
-0.01081	-0.00164	0.001938	0.002127	-0.01939	-0.00875	-0.00426	0.001595
-0.00438	0.005700	0.001392	0.019461	-0.00827	0.007365	-0.00058	-0.00673
-0.00608	-0.01314	0.004305	0.012063	0.019702	-0.00550	0.004371	0.007157
-0.00326	0.004239	-0.00518	0.001928	0.007507	0.020349	-0.00393	0.003666
0.000374	-0.00608	-0.00310	0.009071	0.004307	0.006993	-0.00444	0.001395
-0.00167	0.000279	0.001395	0.011609	0.003764	-0.00942	-0.00092	0.000554
-0.00618	0.000650	0.001114	0.001670	-0.00111	0.000927	-0.02019	-0.00510
0.012355	0.002722	0.020035	0.000917	0.007702	0.000273	0.004184	0.006885
-0.00080	-0.00315	0.003342	0.009273	-0.00526	0.017307	0.002291	-0.00079
0.010914	-0.00679	0.009906	-0.00902	0.008409	-0.00651	0.002535	0.009244
0.004839	0.014361	-0.00703	0.006659	0.004580	-0.01452	-0.01122	-0.00701
0.016317	-0.01021	-0.00208	-0.01486	0.005735	-0.01403	-0.00026	0.01166
-0.01020	0.002488	-0.01463	-0.02231	-0.01610	-0.00364	0.011923	-0.00844
-0.01169	-0.00179	-0.03006	0.017993	0.002017	-0.01140	-0.00785	-0.02961
-0.00090	-0.00514	-0.00466	0.025045	0.014004	0.000881	0.004893	-0.00516
-0.01918	0.010080	0.018873	0.009407	-0.00278	-0.00915	-0.00204	-0.01062
-0.00482	-0.00494	-0.00745	0.036372	0.002900	0.006459	0.007279	0.004564
0.002082	0.004061	-0.00780	0.001137	0.007577	-0.00122	0.008752	-0.00979
-0.01328	0.000572	0.014506	0.005174	0.001310	0.003364	0.002980	-0.00046
0.000929	0.011975	0.014769	0.007051	0.005924	-0.01597	0.008796	-0.00431
-0.00225	0.019002	0.001509	0.003724	0.004505	0.008354	0.011861	-0.00430
0.002510	0.002417	-0.00051	0.001982	-0.01376	-0.00523	0.003945	0.005501
0.013722	-0.00454	-0.00163	-0.01517	0.006039	0.006525	0.015387	0.007066
-0.00380	0.001188	-0.00873	0.007610	0.018414	-0.00591	0.002766	0.01513
0.004940	0.003851	-0.00261	-0.00212	-0.00114	-0.00829	0.005381	0.007988
-0.00138	-0.00458	-0.00378	-0.00189	-0.00198	0.006708	0.014397	0.002514
0.009465	-0.00793	-0.00411	0.015979	0.003752	-0.01853	-0.00640	0.009543
0.013654	0.004463	-0.00682	-0.00255	-0.01057	-0.01165	0.002457	0.011112
0.019233	-0.00555	-0.00430	-0.01257	0.006163	0.005964	0.006810	-0.00095
0.001035	0.008514	0.016806	-0.00364	0.00662	0.008897	-0.00743	0.007262
-0.01265	-0.01841	-0.02223	0.015541	0.013311	0.007551	0.009680	0.018557
-0.00554	0.004961	-0.00463	-0.00572	0.013354	-0.00007	0.012725	-0.01106
-0.00529	0.008287	-0.00580	0.000606	-0.01811	0.002470	-0.01517	0.001329
-0.00109	-0.01266	0.009343	0.012316	0.017746	-0.01842	0.002094	0.002322
0.013747	0.025369	0.014117	0.004835	0.004374	0.014156	-0.00458	0.009636
-0.00918	-0.00575	0.007374	0.006028	0.002497	-0.02355	-0.00174	-0.00189
-0.00168	-0.01795	-0.02551	-0.00099	-0.01701	-0.00701	0.014682	0.001702
0.008883	0.017534	0.000827	0.005263	0.015557	-0.00353	0.004286	0.005078
-0.01127	0.002221	0.003177	0.004272	0.011955	0.001087	-0.02200	-0.01495
0.003156	0.000299	-0.00172	0.001350	0.005619	0.004097	-0.00296	-0.02024
-0.00220	-0.00837	-0.00023	-0.00299	0.009858	-0.00594	-0.00076	-0.00529
-0.02037	0.012213	0.001011	0.008087	0.007482	-0.01018	-0.00023	-0.00773
-0.00592	-0.00392	0.006536	0.005242	-0.01455	-0.00023	0.006083	0.000314
0.008870	0.012293	0.008993	0.005637	-0.01098	0.002297	-0.00710	-0.00061
0.009780	-0.00503	-0.00390	0.024932	-0.00060	0.011794	-0.00564	0.002240
-0.00566	0.004645	0.012007	-0.00751	0.018118	-0.00612	-0.01188	-0.00274
0.012809	0.004191	-0.00183	-0.00608	-0.01151	-0.00014	0.002987	0.002680
-0.00118	-0.01011	-0.00352	0.011230	0.003950	0.005567	-0.00900	-0.00067
-0.00149	0.008959	0.002515	-0.00848	-0.00952	-0.00180	-0.00067	-0.01544
-0.00267	0.003529	0.006805	-0.00007	-0.01481	0.007786	-0.00130	0.005591
0.006779	0.002799	-0.00339	-0.00068	-0.00189	-0.00318	0.010964	0.007531
-0.00239	-0.00644	-0.00135	-0.01351	0.000689	0.001912	0.009621	0.000151
-0.00204	0.002652	0.010807	-0.00194	0.000898	-0.01092	0.008853	-0.01260
0.004785	-0.00241	0.010610	-0.00517	0.001130	-0.00188	-0.00505	-0.00515
-0.01097	-0.00870	-0.00987	0.006830	0.000623	0.007637	0.000541	0.002087
0.000077	0.004473	0.000844	0.003222	-0.01560	-0.00295	-0.00942	0.002123
0.008320	0.011209	-0.00585	0.000154	0.006580	0.006999	-0.00336	0.005364
0.011358	-0.00022	-0.00670	0.005996	0.009883	-0.00336	0.000824	-0.00763
-0.00958	-0.00845	0.002920	0.001532	-0.01117	-0.02886	-0.00294	-0.00135
-0.01160	0.004615	-0.01039	0.001873	0.003820	-0.01822	-0.00965	-0.01740
0.003559	0.014696	0.012235	-0.00781	-0.00737	-0.00751	-0.01446	-0.00759
0.008429	-0.00477	-0.00882	-0.00553	-0.01947	0.024474	0.003548	0.00207
0.007746	0.019473	0.001256	-0.00100	0.016081	0.008243	-0.00703	-0.00197
-0.00354	-0.01639	0.00766	-0.00434	-0.00176	0.010926	-0.00149	-0.00383
-0.00869	-0.00371	0.009563	0.001341	-0.00326	0.023769	0.018951	0.004831
-0.00048	-0.00962	-0.00704	0.005054	-0.00478	0.001793	0.002196	-0.01233
-0.01175	0.007568	-0.00734	0.003742	0.008284	-0.00090	0.015707	0.004372
0.008383	0.010072	-0.00197	-0.01118	0.003448	0.009111	-0.00847	-0.00295
0.005287	0.001833	-0.00620	-0.01720	0.001710	-0.00463	0.005718	0.007147
-0.00532	-0.00372	-0.00463	0.001880	-0.00220	-0.00490	0.005178	0.002044
0.001550	-0.02191	-0.00724	-0.00209	0.005213	-0.02317	-0.00411	-0.01220
0.005745	0.006837	0.007650	-0.01066	-0.00603	0.004164	-0.00319	0.00026
-0.00190	0.004774	0.027475	0.012445	-0.02176	0.001952	-0.01296	-0.00051
0.007215	-0.02242	-0.00828	0.008620	-0.00200	-0.00043	-0.00279	-0.00324
0.001143	-0.00527	-0.01439	-0.00071	0.017577	-0.00229	-0.00088	0.001768
-0.00556	-0.01561	-0.00937	-0.00491	-0.01829	0.013881	0.005329	-0.00045
-0.00685	0.007734	-0.00155	-0.00183	0.011184	0.002810	0.019528	0.006916
-0.00510	0.002124	-0.01121	0.003216	-0.00026	-0.00276	0.016345	0.011688
-0.00338	0.005491	0.000866	0.006582	-0.00189	-0.00008	-0.00137	0.004489
0.003953	-0.00094	-0.01079	0.002425	0.012703	0.012373	0.005225	-0.01056
-0.00627	-0.00963	0.002669	0.003177	0.005564	0.001787	0.008583	0.006656
-0.00912	0.008785	-0.00209	-0.00797	-0.00177	-0.01101	-0.00745	-0.00820
-0.00261	0.002618	-0.00087	-0.00339	-0.01127	-0.00397	-0.00692	-0.00178

0.003223	-0.00160	-0.01582	0.000272	-0.00444	-0.00583	0.005688	0.014870
-0.01150	-0.00245	-0.00747	-0.01166	-0.00297	-0.00074	0.010261	0.016989
-0.00281	-0.00628	0.010262	-0.00045	-0.00544	-0.00821	-0.00975	-0.00334
-0.00065	0.002891	0.012089	0.006799	-0.00109	0.009045	0.000271	0.005431
-0.00306	0.007315	-0.00107	0.000448	-0.00269	-0.00728	-0.00842	-0.01553
-0.00009	-0.00584	0.017648	-0.01055	-0.01567	-0.00923	-0.01378	-0.00607
-0.00232	-0.00233	-0.00175	0.013962	0.002310	0.047555	-0.00467	0.005804
0.035360	0.027340	-0.00663	0.019420	0.008249	-0.01214	0.004696	0.015723
-0.01054	0.017167	0.019953	-0.01067	0.006838	-0.00188	-0.00819	0.010498
0.007035	0.009666	-0.00418	-0.00985	-0.00032	0.019345	-0.00712	-0.00145
-0.00395	0.002432	-0.00307	-0.01306	-0.00994	0.012871	-0.00377	0.003868
0.032710	0.022465	0.017468	0.026096	-0.00022	0.016884	-0.01565	-0.00743
0.023657	-0.00109	0.019402	-0.00122	-0.00136	-0.03996	0.008700	0.005948
-0.01249	0.000898	0.013162	0.014911	0.03913	-0.00713	0.002185	-0.01209
0.018370	-0.01300	0.004250	-0.01573	-0.01791	-0.01174	0.018534	0.002972
-0.00954	-0.02043	-0.00961	0.007146	0.007469	-0.00504	0.032339	0.001299
0.000720	-0.00093	0.022207	0.006701	-0.00637	-0.01276	-0.00307	0.002722
-0.01829	-0.01564	0.000443	0.016186	-0.00894	0.017246	0.001587	0.006410
0.017606	-0.00991	0.003338	-0.00644	0.002209	-0.01635	0.021758	0.004315
0.023316	-0.00061	0.011020	-0.00681	0.006242	-0.00654	0.006313	0.000409
-0.00211	-0.00771	0.007021	-0.01667	-0.02697	0.012717	-0.00148	0.019287
0.001663	0.005466	-0.01610	0.001888	0.007191	0.013032	0.005405	-0.00837
-0.00480	0.017241	0.000881	0.008738	-0.00416	-0.00586	0.000067	0.003798
-0.01702	0.009004	0.019143	0.000935	-0.01121	0.019046	0.009411	0.007747
0.001238	0	-0.01568	0.010643	-0.00699	-0.00368	-0.00271	0.003513
-0.01030	-0.00140	0.002072	0.008605	-0.00350	0.014270	0.003664	-0.00456
-0.00537	-0.00171	0.011874	-0.00280	0.000392	-0.00738	-0.00559	0.004767
0.007182	0.014982	0.004383	0.006096	0.008547	0.004047	0.006236	-0.00644
0.012601	-0.00410	0.002311	-0.01003	0.018890	-0.00228	0.009353	0.009021
-0.01404	0.001418	0.005975	0.005939	0.011078	-0.00174	0.000844	-0.00596
-0.00430	0.003957	-0.00909	0.001897	-0.00268	-0.00783	0.000925	0.007956
0.012910	0.004047	-0.00439	-0.00616	-0.01258	0.000985	0.008797	0.002683
0.002493	-0.01249	-0.00866	0.002912	0.005252	0.013277	0.004185	0.009605
0.012087	-0.00005	-0.00065	0.008933	0.002697	-0.00245	-0.00099	-0.01138
-0.01650	0.005794	0.008821	0.004758	-0.01367	0.011284	-0.00546	-0.00286
0.006164	-0.01534	-0.00042	0.003324	-0.01036	-0.00206	0.005367	0.027058
-0.00135	-0.00100	0.003789	0.004836	-0.01620	-0.01521	-0.01502	-0.00319
-0.00018	0.008826	-0.01290	0.002541	-0.01582	0.005968	0.008805	0.000062
0.003775	0.009558	-0.00183	0.011504	-0.01052	0.002629	0.002195	-0.00955
-0.00933	-0.00254	0.008082	0.000678	0.002033	0.011194	-0.00103	0.004688
0.017515	0.000417	-0.00113	-0.00506	-0.00862	-0.00410	0.003398	-0.00592
0.011315	0.008301	0.009724	-0.00496	0.008016	-0.00147	0.003303	-0.00964
-0.00249	-0.00464	-0.00693	-0.00162	0.002834	0.008841	0.015142	0.003053
0.010831	-0.01337	-0.00416	0.001473	-0.00011	0.003355	-0.01537	-0.00643
0.001499	-0.00616	0.000241	0.002891	-0.00654	-0.00326	-0.00891	0.001101
0.000550	0.007333	-0.00843	-0.00617	-0.00326	-0.00092	0.013662	0.002683
0.011434	0.001744	-0.00732	0.004354	0.000301	-0.00626	0.005815	0.004757
0.000719	0.001317	-0.00382	0.008226	-0.00899	0.000540	-0.00630	0.001994
-0.00180	0.002659	-0.00427	-0.00066	0.003210	-0.00416	-0.00970	-0.01016
0.004453	-0.00043	-0.00197	0.009629	-0.00177	-0.00030	0.009435	0.003520
-0.00290	0.000424	-0.00539	0.016703	0.012171	0.002784	-0.00224	-0.00562
-0.00095	-0.00023	-0.00435	0.000957	0.003888	-0.00166	-0.00304	-0.00496
-0.00806	0.00649	-0.00662	-0.00363	-0.00182	-0.00652	0.003315	-0.00410
0.003809	-0.01499	-0.01758	0.004175	-0.01820	-0.00275	0.005662	-0.00863
0.010713	-0.00229	-0.00076	-0.00249	-0.00706	-0.00213	-0.00012	0.020869
0.011364	-0.01556	0.001530	0.007194	0.006637	-0.00847	-0.01038	-0.01075
0.004011	-0.00541	0.012892	0.002814	-0.00006	0.004082	0.011816	-0.00935
0.006845	-0.00125	-0.01241	0.001084	-0.00121	0.004021	0.016401	-0.00225
-0.00213	-0.00753	-0.00202	-0.00076	-0.01586	0.002838	-0.00019	0.002701
-0.00558	0.017612	-0.00266	0.006420	0.004105	-0.00673	0.00076	-0.00772
0.007971	0.003796	0.010400	-0.00255	0.001000	0.010184	0.001360	-0.00432
-0.01296	0.002262	0.006584	-0.00255	-0.00068	-0.00943	-0.00624	0.003174
-0.00006	-0.00898	-0.00504	-0.00674	-0.00549	-0.00474	-0.01253	0.002578
-0.00877	0.000399	0.001330	0.017867	0.007178	-0.00447	0.008851	-0.00058
0.001613	-0.01359	-0.00568	-0.00039	-0.01143	-0.00904	0.018117	0.005799
0.014612	-0.00213	-0.00032	-0.00317	-0.00818	-0.00700	0.007913	0.002224
0.000130	0.003263	-0.00611	-0.00340	0.007356	-0.00306	-0.01523	-0.00352
0.005665	0.004772	0.005145	-0.00643	-0.00680	-0.00545	-0.00401	-0.00758
0.006832	0.008398	0.007396	-0.00661	0.003129	0.022700	0.025376	0.027596
0.001539	0.000676	-0.00590	0.023431	-0.00072	0.000060	-0.00610	-0.00985
0.005958	0.002259	0.004873	0.017521	-0.00458	0.000359	0.002333	-0.00638
0.005767	-0.00179	-0.00299	0.000480	-0.01079	-0.00357	0.008278	-0.00772
-0.00066	0.001156	0.001398	0.019796	0.005001	0.000533	-0.00722	-0.00423
0.003174	-0.01074	-0.00235	0.002057	0.003985	0.004089	-0.00515	-0.00891
-0.00625	-0.00702	0.002954	-0.00147	-0.00338	-0.00283	0.002721	0.004133
0.008600	0.009684	-0.00597	-0.00388	0.024125	-0.00083	-0.00357	-0.00161
0.000658	-0.00532	-0.00613	-0.00308	0.012501	-0.00449	0.008429	-0.00041
0.006928	0.010855	-0.00727	-0.00289	-0.00640	-0.00143	-0.00830	0.000120
-0.00060	-0.01079	-0.00609	0.006621	0.002071	0.014587	-0.00820	0.004469
-0.00763	-0.00672	-0.00201	-0.00464	0.003439	-0.00783	0.004071	-0.00307
0.003512	0.001474	-0.00269	-0.00504	0.005438	0.005654	0.027504	-0.00565
-0.00466	-0.00522	0.007552	-0.00173	-0.00432	0.003076	0.005894	-0.01118
-0.00483	-0.00540	0.003421	-0.00152	0.007256	0.018949	-0.00237	0.015484
0.001700	0.002283	-0.00268	0.003455	0.022822	0.001426	0.010371	-0.00332
0.003621	0.000225	0.010090	0.001172	0.001337	-0.00556	0.009628	0.001441
-0.00099	0.007703	0.002035	-0.00922	0.000277	0.015452	-0.00512	-0.00444

-0.00148	-0.00082	-0.00546	-0.00460	-0.00072	0.010824	-0.00253	0.002656
0.011258	-0.00638	0.000933	-0.00867	-0.00631	-0.00228	-0.00173	0.004866
-0.00818	-0.00196	-0.00736	0.001982	0.015038	-0.00256	0.001507	-0.00172
-0.00597	0.002584	0.006220	0	0.006238	0.003376	-0.00408	-0.00786
-0.00044	-0.00558	0.001011	0.006789	0.004291	0.001942	0.002104	0.001547
0.002649	-0.00462	0.001493	-0.00226	0.006530	0.002089	0.006419	-0.00681
-0.00850	-0.00442	-0.00811	0.003588	0.005977	-0.00049	0.004278	-0.00077
0.007197	0.012972	0.001790	-0.00400	0.003643	0.006069	0.009479	0.012271
-0.00042	-0.00569	-0.00509	0.003678	-0.00228	-0.00095	0.000373	0.009587
-0.00121	0.003803	0.000631	0.004732	-0.00722	-0.00089	-0.00248	-0.00756
-0.01215	0.009550	-0.00304	0.004342	-0.00378	0.000535	0.015423	-0.00242
0.003119	0.001686	0.006155	0.003242	0.003023	-0.00218	-0.00291	0.005589
-0.00306	-0.00458	0.006909	0.002963	0.001814	-0.00294	0.010377	0.004776
-0.00649	0.003858	-0.00399	-0.00926	-0.00503	0.002505	0.001770	-0.01455
0.001740	0.005212	0.006233	-0.00327	-0.00449	-0.01411	-0.00133	0.006766
-0.00333	-0.00366	-0.00175	0.000587	-0.00080	-0.00619	0.001504	0.009121
0.005742	-0.00951	-0.00101	0.000748	0.004217	0.003880	0.000529	-0.00158
-0.00381	-0.00287	-0.00053	0.005179	0.000053	-0.00717	-0.01000	-0.00724
-0.00424	-0.00016	-0.00831	0.001929	0.009245	-0.00730	0.012359	-0.00911
-0.01073	0.003487	0.004357	0.016421	-0.00545	0.001629	-0.00618	-0.00736
0	0.003574	0.001424	0.008206	0.011341	-0.00155	0.010210	-0.00170
-0.00330	-0.00042	0.005776	0.005583	-0.00312	-0.00519	0.001279	0.007829
0.004439	-0.00131	0.009008	-0.00146	0.005856	0.002027	-0.00072	0.005710
0.018428	0.004004	-0.00494	0.009944	-0.00477	0.003028	-0.00020	0.001610
0.012161	0.000546	-0.00580	0.001597	0.009318	-0.00182	-0.00845	0.001995
0.016777	-0.00171	-0.00436	0.006207	0.000685	0.009393	0.002035	0.015527
0.009907	-0.00646	-0.00398	0.001000	0.004380	-0.01123	-0.00685	0.002462
0.009439	0.005104	0.002847	-0.00799	0.006154	-0.00109	0.014953	-0.02726
-0.00894	-0.00072	0.00369	-0.00038	0.007839	0.004369	-0.00353	-0.00431
-0.00838	-0.01117	0.003734	0.010673	0.004650	0.011668	0.002287	-0.00456
0.011704	0.010293	-0.00546	0.000799	0.002394	0.005106	0.00783	-0.00147
0.000231	0.006621	0.010855	0.012240	-0.01209	0.011194	0.010800	-0.00124
-0.00245	0.001117	0.012185	0.000661	-0.00661	-0.00461	-0.00017	0.003521
0.001954	0.004477	0.022552	0.003668	0.002795	0.014408	-0.00794	0.00473
-0.00076	0.003989	-0.01352	0.008528	-0.00259	0.010991	0.007037	-0.00029
-0.01573	0.002424	-0.01374	-0.01626	-0.00026	0.021388	0.000984	0.011508
-0.00198	0.005551	0.001896	0.018887	0.003344	-0.00267	0.009736	-0.00947
-0.00276	0.001116	0.001115	0.003260	-0.01057	-0.02074	-0.00152	-0.00157
0.012521	-0.00206	-0.00488	0.004447	0.003036	-0.00113	-0.00492	0.004779
-0.01309	-0.00712	0.001890	0.012478	-0.00279	0.019834	0.005122	0.014087
0.007681	0.005473	-0.00254	-0.00933	0.001918	-0.00639	0.007009	0.000081
-0.02324	-0.00158	0.006469	0.001493	0.017557	0.001627	-0.00723	0.002619
-0.00379	0.014422	-0.00937	0.007216	0.007691	-0.00076	0.003457	0.004967
0.004783	0.002618	-0.00360	-0.03073	-0.01007	0.005091	0.000782	-0.00325
-0.01696	-0.01868	0.005777	0.004510	0.001228	-0.00050	0.008211	0.002057
-0.00301	0.009539	-0.01752	-0.00618	0.008697	-0.00198	-0.00512	0.004597
0.004406	-0.00080	0.000844	-0.00067	0.016041	0.011052	0.009575	0.002360
0.003654	0.000930	-0.00351	0.013224	-0.00040	0.002082	-0.00951	0.020297
0.001819	-0.00181	0.000355	-0.01743	0.006277	0.014995	-0.01323	-0.00930
-0.00189	-0.00246	-0.04808	-0.01917	0.005505	-0.00094	-0.00017	0.002719
-0.00043	0.011713	0.003149	0.002588	-0.01883	0.001725	-0.00999	0.006132
0.009856	0.001369	-0.00089	0.004578	-0.00157	0.009683	-0.00350	-0.00156
0.001826	-0.00228	0.014572	0.003056	-0.00288	-0.01201	-0.00038	0.001610
0.012782	-0.00426	0.002140	0.002052	0.007021	0.011496	0.001107	0.007459
0.001627	0.001543	-0.00287	-0.00040	0.001464	0.003859	-0.00178	-0.01467
0.00609	-0.00527	-0.02643	0.003716	0.018471	0.015740	0.006467	0.002909
0.002417	0.001808	-0.00068	0.019875	-0.00059	-0.00319	-0.00739	-0.00003
-0.00748	0.006739	-0.01111	-0.00330	0.003476	0.007372	-0.00991	-0.00315
0.011953	-0.00392	-0.00968	0.001664	0.000688	-0.00911	-0.00531	-0.00493
0.017673	0.023290	0.002339	0.010087	0.007637	0.005635	0.006068	-0.00134
0.010347	0.010851	0.002975	0.011491	-0.00111	-0.00446	0.022662	-0.01390
-0.00181	0.015355	0.006027	-0.00421	-0.00058	0.008646	-0.00166	0.013224
0.005435	-0.00398	-0.00671	-0.01110	0.008979	-0.00691	0.014802	0.020700
-0.00024	0.000526	-0.00031	-0.01085	0.001770	0.003959	-0.00366	0.004382
-0.00422	0.003958	0.015838	0.006582	0.000481	-0.00811	0.008879	-0.00189
0.003134	-0.00456	-0.00572	0.014710	0.001984	0.004440	0.013907	0.010027
0.001593	-0.00417	0.001831	-0.01595	-0.02340	0.008644	0.002365	0.004240
0.023090	0.005126	-0.01742	0.001921	-0.01480	-0.00126	-0.02348	-0.02261
0.018914	0.008683	-0.00285	0.024397	-0.02006	-0.00128	-0.01847	0.001101
0.002412	0.007292	0.013318	-0.00114	0.004617	0.020666	0.000439	-0.00257
-0.00454	-0.00613	0.005933	0.002318	0.000884	-0.02314	-0.00271	-0.02452
-0.00504	0.007044	0.007102	0.024632	-0.00131	0.007030	-0.00226	-0.00093
-0.00472	0.017368	0.005520	-0.00555	0.011143	0.001887	0.000638	0.004235
0.009674	0.005038	0.005146	0.000361	0.002886	0.004187	0.008730	-0.00393
-0.00508	0.006843	-0.00582	0.002409	-0.01266	-0.00348	0.008879	-0.00232
0.008133	0.002895	-0.00249	0.002828	-0.00246	0.009914	-0.00083	0.007345
0.006044	-0.01017	-0.00911	-0.00025	-0.00213	0.004743	0.004461	0.005408
0.010629	0.007603	0.001917	-0.00342	-0.00421	0.007020	0.011430	0.002825
0.015479	0.016250	-0.00282	0.006799	-0.00197	0.000359	-0.01454	0.001761
0.015189	0.003165	-0.00765	0.010319	-0.00653	-0.00953	-0.01309	0.008439
-0.01940	-0.00531	-0.00456	-0.01096	-0.00991	0.001147	0.010225	0.015293
0.003416	-0.01652	-0.00906	0.000221	-0.00022	-0.01372	0.028852	0.005289
-0.00457	0.001376	0.009495	-0.00467	0.000435	0.017089	0.002260	0.000030
-0.02700	-0.00213	-0.01375	-0.00983	-0.00540	0.016581	-0.02953	-0.02342
-0.05159	-0.20466	0.053326	0.090993	-0.03920	-0.00012	-0.08278	0.024245
0.000385	0.049254	0.028679	0.015727	-0.01927	-0.00741	0.022172	-0.01599

-0.02891	-0.01714	0.012133	0.027366	-0.01158	0.004559	-0.01507	0.010327
-0.02239	0.008123	0.004090	0.013992	-0.00929	-0.01540	-0.04177	0.007381
0.00625	-0.03529	-0.00572	0.021614	0.026884	0.016942	-0.02226	0.007492
0.029194	0.002518	0.021746	-0.02055	0.025434	0.001525	0.001643	0.012842
-0.00450	-0.02559	-0.00399	0.013369	-0.00314	0.035858	0.010509	0.001005
0.008421	-0.06768	0.016803	-0.00836	0.001589	0.000284	0.025093	-0.00067
-0.01016	-0.02683	0.002101	0.013819	0.023002	-0.01031	-0.00076	0.015678
0.014923	-0.00789	0.002078	-0.01314	0	-0.00495	-0.00741	0.010517
0.019624	-0.00276	0.006563	0.008540	-0.00238	-0.00501	0.014346	0.015403
-0.00233	-0.00222	-0.01077	0.003363	0.020422	-0.00037	0.000971	-0.00037
-0.00216	0.000298	0.007667	-0.00137	-0.01940	0.004168	0.005398	-0.00090
0.009469	0.009566	-0.00036	-0.00877	0.000372	0.000259	-0.02067	-0.01837
-0.00174	0.007788	-0.00769	0.003176	-0.01081	0.009449	0.027000	0.002523
0.012285	0.002709	0.004479	0.000736	-0.04352	0.000077	-0.00215	-0.00497
-0.00694	0.001132	0.014507	0.008918	0.005600	-0.00049	-0.00451	-0.00487
0.00088	0.005505	-0.01018	-0.00587	-0.00506	-0.00365	0.004210	-0.01673
0.002131	0.011542	0.007515	-0.01302	-0.01562	0.004853	0.001781	-0.00865
0.010684	0.000986	0.003428	-0.00475	0.034487	0.017279	-0.00509	0.004220
0.002252	-0.00704	0.023947	-0.00486	0.003923	0.000626	0.010574	0.000546
-0.01705	0.003372	-0.00642	0.010150	0.014686	-0.00304	-0.00378	-0.01724
0.012079	-0.00488	0.009299	-0.00628	0.014828	-0.01374	-0.00088	-0.00647
0.001962	-0.00997	0.005488	0.003490	0.006623	-0.00566	-0.00754	0.005699
-0.01237	-0.01185	0.004477	0.001926	-0.01014	0.013409	0.022554	0.000698
-0.00055	0.003381	-0.00384	-0.00286	-0.00431	-0.01292	-0.01722	0.003245
-0.00076	-0.01470	0.007229	0.000805	0.000997	-0.00302	-0.01252	0.000428
0.015713	-0.00746	0.001929	0.010205	0.000685	-0.00377	-0.01212	0.023727
0.004197	0.001054	0.000036	0.003611	-0.00138	0.003602	0.007030	-0.00438
0.009398	-0.00676	0.003384	0.001593	-0.00362	0.002155	-0.00326	-0.00230
0.003057	0.013044	-0.00249	-0.00194	-0.00279	0.004582	0.001948	0.020853
0.000611	-0.00111	-0.01421	0.004525	0.001017	0.003302	0.010744	-0.00862
0.021337	0.002757	-0.00486	0.000353	-0.00354	-0.01457	0.004508	0.001579
0.000322	0	0.000501	-0.01035	-0.00861	0.004453	-0.00661	0.001316
-0.02108	-0.00074	0.002316	-0.01684	0.002956	0.007066	-0.00093	0.003718
0.006699	-0.00658	0.005276	0.00845	0.010298	-0.00442	-0.00249	0.011478
0.009675	0.001944	-0.00560	0.001663	-0.00184	-0.00075	-0.00361	-0.00374
0.007328	0.009482	-0.00516	-0.00032	-0.00183	0.003611	-0.00374	0.000903
0.008372	-0.00601	0.0 0.0	0.0 0.0	0.0 0.0	0.0 0.0	0.0	

KIM-MARKOWITZ WITH FUTURES, ARBITRAGERS AND SPECULATORS

```
Preamble...

Normally mode is real.

The system has a report.interval, a remaining.report.periods,
   a period.number, a no.bids.price.est.factor,
   a no.offers.price.est.factor, a mkt.return, a sum.mkt.return,
   a sum.mkt.return.squared, a total.cash, a market.value,
   a period.deposits, a number.of.periods,
   a mkt.sigma, a mkt.sharpe, a mkt.treynor,
   a number.of.stocks, a number.of.futures,
  and owns ''the buyers, the sellers,
   the rebalancer.prototypes, the orpi.prototypes,
   the cppi.prototypes, the arb.prototypes and the spec.prototypes.

Define remaining.report.periods as an integer variable.

Tally daily.high as the daily max, daily.low as the daily min,
  period.high as the period max, period.low as the period min,
  grand.high as the max and grand.low as the min of last.sale.price

Tally daily.volume as the daily sum, period.volume as the
  period sum, grand.volume as the sum of sale.quantity

Tally max.ppc as the period max, max.gpc as the grand max,
  min.ppc as the period min,
  min.gpc as the grand min of pct.change.for.day

Tally pmpc as the period mean, gmpc as the grand mean,
  pspc as the period std.dev,
  gspc as the grand std.dev of pct.change.for.day

  Temporary entities....

Every rebalancer.prototype belongs to the rebalancer.prototypes
  and owns some investors,
  has a (prototype.number(1/2), pr.sequence.number(2/2)),
  a portfolio.review.interval, a dep.withdraw.interval,
  a trading.strategy.nr, an init.portfolio.value, an init.stock.fraction,
  a min.deposit, a max.deposit,
  a target.fraction, a high.fraction, and a low.fraction.

Every cppi.prototype belongs to the cppi.prototypes
  and owns some investors,
  has a (prototype.number(1/2), pr.sequence.number(2/2)),
  a portfolio.review.interval, a dep.withdraw.interval,
  a trading.strategy.nr, an init.portfolio.value, an init.stock.fraction,
  a min.deposit, a max.deposit, a max.stock.to.assets,
  a cppi.period, a cppi.ratio, a low.ratio, a high.ratio,
  a cppi.plan.start.cushion.pct, and a cppi.init.cushion.pct.

Every orpi.prototype belongs to the orpi.prototypes
  and owns some investors,
  has a (prototype.number(1/2), pr.sequence.number(2/2)),
  a portfolio.review.interval, a dep.withdraw.interval,
  a trading.strategy.nr, an init.portfolio.value, an init.stock.fraction,
  a min.deposit, a max.deposit, an est.sigma, an orpi.ratio, a buffer,
  and an orpi.period.

Every arb.prototype belongs to the arb.prototypes
  and owns some investors,
  has a (prototype.number(1/2), pr.sequence.number(2/2)),
  a portfolio.review.interval, a dep.withdraw.interval,
  a trading.strategy.nr, an init.portfolio.value, an init.stock.fraction,
  a gap, a risk.level, a min.deposit, a max.deposit.

Every spec.prototype belongs to the spec.prototypes
  and owns some investors,
  has a (prototype.number(1/2), pr.sequence.number(2/2)),
  a portfolio.review.interval, a dep.withdraw.interval,
  a trading.strategy.nr, an init.portfolio.value, an init.stock.fraction,
  a gap, a risk.level, a min.deposit, a max.deposit.

Define trading.strategy.nr as an integer variable  ''tr. str. pointer
Define prototype.number and pr.sequence.number as integer variables.

Define rebalancer.prototypes, cppi.prototypes, orpi.prototypes,
  and investors as fifo sets without p attributes,
  and without r, ff, fb, fa routines.

Every order has a (sec.nr(1/2), mkt.nr(2/2)),
    a price, a quantity, and a customer,
    may belong to the buyers, ''or'' the sellers.
```

```
Define buyers as a set ranked by high price.
Define sellers as a set ranked by low price.
Define quantity and customer as integer variables.
Define mkt.nr and sec.nr as an integer variable.


    Permanent entities...

Every security owns the buyers, the sellers,
  has a last.sale.price, a last.period.price, a pct.change.for.day,
  a sale.quantity, a total.number.of.shares, and a last.closing.price.

Define sale.quantity and total.number.of.shares as integer variables.

Every trading.strategy has a change.bid.interval,
  a bid.at.factor, and an offer.at.factor

Define number.of.trading.strategies to mean n.trading.strategy

   Processes include ''the'' report and daily.close ''processes''

Every rebalancer has a my.prototype ''pointer'', an id.nr,
  a dollars.of.cash, a number.of.shares, a total.dep.withdraw,
  a next.portfolio.review.time, a next.dep.withdraw.time, a my.order,
  an init.value, an alpha, a beta, a t.value, a sum.of.returns,
  a sum.of.squares, a sum.of.xy, a sigma, a sharpe, a treynor,
  and belongs to an investors ''set

Every cppi.investor has a my.prototype ''pointer'', an id.nr,
  a dollars.of.cash, a number.of.shares, a total.dep.withdraw,
  a next.portfolio.review.time, a next.dep.withdraw.time, a my.order,
  an init.value, an alpha, a beta, a t.value, a sum.of.returns,
  a sum.of.squares, a sum.of.xy, a sigma, a sharpe, a treynor,
  and belongs to an investors ''set.
Every cppi.investor ''also'' has a cppi.floor and a next.plan.restart.time.

Every orpi.investor has a my.prototype ''pointer'', an id.nr,
  a dollars.of.cash, a number.of.shares, a total.dep.withdraw,
  a next.portfolio.review.time, a next.dep.withdraw.time, a my.order,
  an init.value, an alpha, a beta, a t.value, a sum.of.returns,
  a sum.of.squares, a sum.of.xy, a sigma, a sharpe, a treynor,
  and belongs to an investors ''set.
Every orpi.investor ''also'' has an exercise.price
  and a next.plan.restart.time.

Every arbitrager has a my.prototype ''pointer'', an id.nr,
  a dollars.of.cash, a number.of.shares, a total.dep.withdraw,
  a next.portfolio.review.time, a next.dep.withdraw.time, a my.order,
  a my.order1, a my.order2, a number.of.shares1, a number.of.shares2,
  an init.value, an alpha, a beta, a t.value, a sum.of.returns,
  a sum.of.squares, a sum.of.xy, a sigma, a sharpe, a treynor,
  and belongs to an investors ''set.

Every speculator has a my.prototype ''pointer'', an id.nr,
  a dollars.of.cash, a number.of.shares, a total.dep.withdraw,
  a next.portfolio.review.time, a next.dep.withdraw.time, a my.order,
  a my.order1, a my.order2, a number.of.shares1, a number.of.shares2,
  an init.value, an alpha, a beta, a t.value, a sum.of.returns,
  a sum.of.squares, a sum.of.xy, a sigma, a sharpe, a treynor,
  and belongs to an investors ''set.

Define my.order as an integer variable ''points to its buy or sell order.
Define my.prototype as an integer variable ''points to its prototype.
Define id.nr, and number.of.shares as integer variables.
Define my.order1 as an integer variable ''points to its buy or sell order.
Define my.order2 as an integer variable ''points to its buy or sell order.
Define number.of.shares1 and number.of.shares2 as integer variables.

Define current.price.estimate as a real function.
Define normal.cdf as a routine given 1 argument,
    yielding 1 argument.
Define compute.stats as a routine given 3 arguments,
    yielding 6 arguments.

End.

Main ''routine.
Define f, s, i, and n as integer variables.
Define a as an alpha variable

Read a  read as /  ''finds and skips over a one line
Read report.interval and remaining.report.periods
Let number.of.periods = remaining.report.periods

Read a  read as /
Read f and s ''number of futures and stocks
```

```
Let number.of.futures = f
Let number.of.stocks = s
Let n.security = f + s
Create every security(n.security)
For every security, do...
    Let last.sale.price(security) = 100.0
    Let last.period.price(security) = 100.0
    Let last.closing.price(security) = 100.0
Loop

Activate a report ''process'' now

Read a   read as /,/  ''finds and reads over a two line input heading
Read no.bids.price.est.factor, no.offers.price.est.factor

Call create.trading.strategies

Read a   read as /,/  ''find and skip over a two line
Read n   '' = number of rebalancer prototypes

Read a  read as /,/,/,/,/,/,/,/,/,/  ''finds and skips over 18 line
        read as /,/,/,/,/,/,/,/,/,/  ''heading in input file.
For i = 1 to n, call create.rebalancer.prototype(i)

Read a  read as /,/   ''finds and skips over a 2 line heading
Read n

Read a  read as /,/,/,/,/,/,/,/,/,/,/  ''finds and skips over a
        read as /,/,/,/,/,/,/,/,/,/,/  ''26 line heading in the
        read as /,/,/,/,/,/           ''input file.
For i = 1 to n, call create.cppi.prototype(i)

Read a  read as /,/   ''finds and skips over a 2 line heading
Read n

Read a  read as /,/,/,/,/,/,/,/,/  ''finds and skips over a
        read as /,/,/,/,/,/,/,/    ''22 line heading in the
        read as /,/,/,/,/,/,/,/    ''input file.
For i = 1 to n, call create.orpi.prototype(i)

Read a  read as /,/   ''finds and skips over a 2 line heading
Read n

Read a  read as /,/,/,/,/,/,/  ''finds and skips over a
        read as /,/,/,/,/,/    ''16 line heading in the
        read as /,/,/,/,/,/    ''input file.
For i = 1 to n, call create.arb.prototype(i)

Read a  read as /,/   ''finds and skips over a 2 line heading
Read n

Read a  read as /,/,/,/,/,/,/  ''finds and skips over a
        read as /,/,/,/,/,/    ''16 line heading in the
        read as /,/,/,/,/,/    ''input file.
For i = 1 to n, call create.spec.prototype(i)

Read a  read as /,/

Read n
If n = 0
Let seed.v(1) = 10
    else let seed.v(1) = n
Always...
Activate a daily.close now
Start simulation
End.

Process arbitrager

Define s, s1, s2, r and rp as integer variables ''point to arbitrager
                                  ''and prototype
Let s = 1
Let s1 = 2
Let s2 = 3
Let r = arbitrager
Let rp = my.prototype(r)
File arbitrager in investors(rp)
Now initialize.investor ''for the'' given rp and r.

'decide.next.event.type'
Let cpe = current.price.estimate(s)
Let cpe1 = current.price.estimate(s1)
Let cpe2 = current.price.estimate(s2)
If dollars.of.cash(r)+number.of.shares(r)*cpe+number.of.shares1(r)*cpe1
       +number.of.shares2(r)*cpe2 <= .0001  suspend  always...
If next.portfolio.review.time(r) < next.dep.withdraw.time(r) go await_review
```

```
  else go await_deposit_withdraw

'await_review'   ''then do the portfolio review next
Wait next.portfolio.review.time(r) - time.v days
Call cancel.order''if any''(my.order(r),s)
Call cancel.order''if any''(my.order1(r),s1)
Call cancel.order''if any''(my.order2(r),s2)
Call calc.arb.order(rp,r)
Add exponential.f(portfolio.review.interval(rp),1) to
                                          next.portfolio.review.time(r)
Go decide.next.event.type

'await_deposit_withdraw'  ''then do the deposit/withdraw
Wait next.dep.withdraw.time(r) - time.v days
Call deposit.or.withdraw ''for'' (rp, r)
Call cancel.order''if any''(my.order(r),s)
Call cancel.order''if any''(my.order1(r),s1)
Call cancel.order''if any''(my.order2(r),s2)
Call calc.arb.order(rp,r) ''since a rebalance may now be necessary
Go decide.next.event.type

End.

Routine buy.order(new.order,s)
Define cust, i, s, q, qs, and new.order as integer variables
    ''order pointers.

'top'
If quantity(new.order) <= 0
    Print 1 line thus...
    Negative buy order placed by investor.
    return   Otherwise...
If n.sellers(s) = 0   file new.order in buyers(s)   return
Otherwise...
  Let i = f.sellers(s)
  Let p = price(i)
  Let qs = quantity(i)
  Let cust = customer(i)
If price(new.order) < p,
      file new.order in buyers(s)
      return
Otherwise...
  Let q = min.f(quantity(new.order), qs)
  Let sale.quantity(s) = q
  Subtract q from quantity(new.order)
  Subtract q from qs
  Let last.sale.price(s) = p
  Let amount = p*q
  Subtract amount from dollars.of.cash(customer(new.order))
  Add amount to dollars.of.cash(cust)
If s = 1
  Add q to number.of.shares(customer(new.order))
  Subtract q from number.of.shares(cust)
Always... if s = 2
  Add q to number.of.shares1(customer(new.order))
  Subtract q from number.of.shares1(cust)
Always... if s = 3
  Add q to number.of.shares2(customer(new.order))
  Subtract q from number.of.shares2(cust)
Always...

If qs <= 0
   remove i from sellers(s)
Always...   Let quantity(i) = qs
If quantity(new.order) <= 0
   return
Otherwise go to top
End.

Routine to calc.arb.order (p,i)
Define q, s, s1, s2, p and i as integer variables
Define t as an integer variable  ''trading strategy number

Let t = trading.strategy.nr(p)
Let s = 1
Let s1 = 2
Let s2 = 3
'top'
Let cpe = current.price.estimate(s)
Let cpe1 = current.price.estimate(s1)
Let cpe2 = current.price.estimate(s2)
Let stock.value = number.of.shares1(i)*cpe1+number.of.shares2(i)*cpe2
Let futures.value = number.of.shares(i) * cpe
Let total.value = stock.value + futures.value + dollars.of.cash(i)
If total.value <= 0  return   else...
If cpe > (cpe1 + cpe2)/2 * bid.at.factor(t) / offer.at.factor(t) * gap(p)
```

```
   let q = min.f(total.value*abs.f(((cpe-(cpe1+cpe2)/2)/cpe)/cpe),
           total.value*risk.level(p)/cpe)
If q <= 0   return   else...
   let price(my.order(i)) = cpe * offer.at.factor(t)
   let quantity(my.order(i)) = min.f(q, number.of.shares(i))
If quantity(my.order(i)) <= 0   return   else...
   let price(my.order1(i)) = cpe1 * bid.at.factor(t)
   let quantity(my.order1(i)) = q/2
   let price(my.order2(i)) = cpe2 * bid.at.factor(t)
   let quantity(my.order2(i)) = q/2
If q/2*(price(my.order1(i))+price(my.order2(i))) > dollars.of.cash(i)
    return   else...
   call buy.order(my.order1(i),s1)
   call buy.order(my.order2(i),s2)
   call sell.order(my.order(i),s)
   if (m.sellers(my.order(i)) = 0 and m.buyers(my.order1(i)) = 0 and
     m.buyers(my.order2(i)) = 0) or time.v+change.bid.interval(t) >
     min.f(next.portfolio.review.time(i),next.dep.withdraw.time(i))
     return
   else...
   wait change.bid.interval(t) days
   if m.sellers(my.order(i)) = 0 and m.buyers(my.order1(i)) = 0
      and m.buyers(my.order2(i)) = 0
   return    else...
Call cancel.order(my.order(i),s)
Call cancel.order(my.order1(i),s1)
Call cancel.order(my.order2(i),s2)
   go to top

Otherwise...
If cpe < (cpe1 + cpe2)/2 / bid.at.factor(t) * offer.at.factor(t) / gap(p)
   let q = min.f(total.value*abs.f(((cpe-(cpe1+cpe2)/2)/cpe)/cpe),
           total.value*risk.level(p)/cpe)
If q <= 0   return   else...
   let price(my.order(i)) = cpe * bid.at.factor(t)
   let quantity(my.order(i)) = q
   let price(my.order1(i)) = cpe1 * offer.at.factor(t)
   let quantity(my.order1(i)) = min.f(q/2, number.of.shares1(i))
If quantity(my.order1(i)) <= 0   return   else...
   let quantity(my.order2(i)) = min.f(q/2, number.of.shares2(i))
If quantity(my.order2(i)) <= 0   return   else...
   let price(my.order2(i)) = cpe2 * offer.at.factor(t)
   call buy.order(my.order(i),s)
   call sell.order(my.order1(i),s1)
   call sell.order(my.order2(i),s2)
   if (m.buyers(my.order(i)) = 0 and m.sellers(my.order1(i)) = 0 and
     m.sellers(my.order2(i)) = 0) or time.v+change.bid.interval(t) >
     min.f(next.portfolio.review.time(i),next.dep.withdraw.time(i))
     return
   else...
   wait change.bid.interval(t) days
   if m.buyers(my.order(i)) = 0 and m.sellers(my.order1(i)) = 0
      and m.sellers(my.order2(i)) = 0
   return    else...
Call cancel.order(my.order(i),s)
Call cancel.order(my.order1(i),s1)
Call cancel.order(my.order2(i),s2)
   go to top
Always return  end.

Routine to calc.cppi.order (p,i)
Define s, p and i as integer variables  ''pointers to prototype & investor
Define t as an integer variable  ''trading strategy number

Let t = trading.strategy.nr(p)
Let s = 1
'top'
Let cpe = current.price.estimate(s)
Let stock.value = number.of.shares(i) * cpe
Let total.value = stock.value + dollars.of.cash(i)
If total.value <= 0  return    else...
Let cushion = max.f(.01,total.value - cppi.floor(i))
If stock.value/cushion > high.ratio(p)
   let price(my.order(i)) = cpe * offer.at.factor(t)
   let quantity(my.order(i))=abs.f(min.f((stock.value-cppi.ratio(p)*cushion)
                            / price(my.order(i)),number.of.shares(i)))
   if quantity(my.order(i)) <= 0  return   else...
   call sell.order(my.order(i),s)
   if m.sellers(my.order(i)) = 0 or sec.nr(my.order(i)) NE s
     or time.v+change.bid.interval(t) >
       min.f(next.portfolio.review.time(i),next.dep.withdraw.time(i),
       next.plan.restart.time(i))
       return
   else...
   wait change.bid.interval(t) days
   if m.sellers(my.order(i)) = 0 or sec.nr(my.order(i)) NE s
```

```
   return    else...
   remove my.order(i) from sellers(s)
   go to top

Otherwise...
If stock.value/cushion < low.ratio(p)
   let price(my.order(i)) = cpe * bid.at.factor(t)
   let quantity(my.order(i))=abs.f
    ((min.f(cppi.ratio(p)*cushion, max.stock.to.assets(p)*total.value)
                                        -stock.value) / price(my.order(i)))
   if quantity(my.order(i)) <= 0  return  else...
   call buy.order(my.order(i),s)
   if m.buyers(my.order(i)) = 0 or sec.nr(my.order(i)) NE s
     or time.v+change.bid.interval(t) >
       min.f(next.portfolio.review.time(i),next.dep.withdraw.time(i),
       next.plan.restart.time(i))
          return
   else...
   wait change.bid.interval(t) days
   if m.buyers(my.order(i)) = 0 or sec.nr(my.order(i)) NE s
   return    else...
   remove my.order(i) from buyers(s)
   go to top
Always return  end.

Routine to calc.orpi.order (p,i)
Define s, p and i as integer variables  ''pointers to prototype & investor
Define t as an integer variable  ''trading strategy number

Let t = trading.strategy.nr(p)
Let s = 1
'top'
Let cpe = current.price.estimate(s)
Let stock.value = number.of.shares(i) * cpe
Let total.value = stock.value + dollars.of.cash(i)
If total.value <= 0  return    else...
Let d.one = (log.e.f(cpe/exercise.price(i))+0.5*est.sigma**2*orpi.period(p)/260)
                                        /(est.sigma*sqrt.f(orpi.period(p)/260))
If d.one > 5.0
    let nd.one = 1.0   go to compute always...
If d.one < -5.0
    let nd.one = 0.0   go to compute always...
Now ''compute'' normal.cdf given d.one yielding nd.one
'compute'
''Let trade.buffer = buffer(p)*10.0
''If stock.value/total.value > nd.one+trade.buffer or
''   stock.value/total.value < nd.one-trade.buffer  return  else...
If stock.value/total.value > nd.one+buffer(p)
   let price(my.order(i)) = cpe * offer.at.factor(t)
   let quantity(my.order(i))=min.f(abs.f(stock.value-total.value*nd.one)
                        / price(my.order(i)), number.of.shares(i))
   if quantity(my.order(i)) <= 0  return   else...
   call sell.order(my.order(i),s)
   if m.sellers(my.order(i)) = 0 or sec.nr(my.order(i)) NE s
     or time.v+change.bid.interval(t) >
       min.f(next.portfolio.review.time(i),next.dep.withdraw.time(i),
       next.plan.restart.time(i))
       return
   else...
   wait change.bid.interval(t) days
   if m.sellers(my.order(i)) = 0 or sec.nr(my.order(i)) NE s
   return    else...
   remove my.order(i) from sellers(s)
   go to top

Otherwise
If stock.value/total.value < nd.one-buffer(p)
   let price(my.order(i)) = cpe * bid.at.factor(t)
   let quantity(my.order(i))=min.f(abs.f(total.value*nd.one-stock.value),
                        abs.f(dollars.of.cash(i)))/price(my.order(i))
   if quantity(my.order(i)) <= 0  return  else...
   call buy.order(my.order(i),s)
   if m.buyers(my.order(i)) = 0 or sec.nr(my.order(i)) NE s
     or time.v+change.bid.interval(t) >
       min.f(next.portfolio.review.time(i),next.dep.withdraw.time(i),
       next.plan.restart.time(i))
          return
   else...
   wait change.bid.interval(t) days
   if m.buyers(my.order(i)) = 0 or sec.nr(my.order(i)) NE s
   return    else...
   remove my.order(i) from buyers(s)
   go to top
Always return  end.

Routine to calc.rebalance.order (p,i)
```

```
Define s, p and i as integer variables
Define t as an integer variable  ''trading strategy number

Let t = trading.strategy.nr(p)
Let s = 1  ''security number
'top'
Let cpe = current.price.estimate(s)
Let sec.nr(my.order(i)) = s
Let stock.value = number.of.shares(i) * cpe
Let total.value = stock.value + dollars.of.cash(i)
If total.value <= 0  return   else...
If stock.value/total.value > high.fraction(p)
   let price(my.order(i)) = cpe * offer.at.factor(t)
   let quantity(my.order(i))=(stock.value - target.fraction(p)
    *total.value) / price(my.order(i))
If quantity(my.order(i)) <= 0   return   else...
   call sell.order(my.order(i),s)
   if m.sellers(my.order(i)) = 0 or sec.nr(my.order(i)) NE s
     or time.v+change.bid.interval(t) >
     min.f(next.portfolio.review.time(i),next.dep.withdraw.time(i))
     return
   else...
   wait change.bid.interval(t) days
   if m.sellers(my.order(i)) = 0 or sec.nr(my.order(i)) NE s
   return    else...
   remove my.order(i) from sellers(s)
   go to top

Otherwise...
If stock.value/total.value < low.fraction(p)
   let price(my.order(i)) = cpe * bid.at.factor(t)
   let quantity(my.order(i))=(target.fraction(p)*total.value-stock.value)
                                          / price(my.order(i))
If quantity(my.order(i)) <= 0   return   else...
   call buy.order(my.order(i),s)
   if m.buyers(my.order(i)) = 0 or sec.nr(my.order(i)) NE s
     or time.v+change.bid.interval(t) >
     min.f(next.portfolio.review.time(i),next.dep.withdraw.time(i))
     return
   else...
   wait change.bid.interval(t) days
   if m.buyers(my.order(i)) = 0 or sec.nr(my.order(i)) NE s
   return    else...
   remove my.order(i) from buyers(s)
   go to top
Always return  end.

Routine to calc.spec.order (p,i)
Define s, s1, s2, p and i as integer variables
Define t as an integer variable  ''trading strategy number

Let t = trading.strategy.nr(p)
Let s = 1
Let s1 = 2
Let s2 = 3
'top'
Let cpe = current.price.estimate(s)
Let cpe1 = current.price.estimate(s1)
Let cpe2 = current.price.estimate(s2)
Let value = number.of.shares(i)*cpe
Let value1 = number.of.shares1(i)*cpe1
Let value2 = number.of.shares2(i)*cpe2
Let stock.value = value + value1 + value2
Let total.value = stock.value + dollars.of.cash(i)
If total.value <= 0  return   else...
Let signal1 = uniform.f(0.0, 0.5, 1)
Let signal2 = uniform.f(0.0, 0.5, 1)

If cpe1 > 0.01
 If value1 > total.value * (signal1 + gap(p))
   let price(my.order1(i)) = cpe1 * offer.at.factor(t)
   let quantity(my.order1(i))=min.f((value1-total.value*signal1)/
                          price(my.order1(i)), number.of.shares1(i))
   if quantity(my.order1(i)) <= 0   return   otherwise...
   call sell.order(my.order1(i),s1)
 Else... If value1 < total.value * (signal1 - gap(p))
   let price(my.order1(i)) = cpe1 * bid.at.factor(t)
   let quantity(my.order1(i))=(total.value*signal1-value1)/cpe1
   if dollars.of.cash(i) < price(my.order1(i))*quantity(my.order1(i))
   return    otherwise...
   if quantity(my.order1(i)) <= 0   return   otherwise...
   call buy.order(my.order1(i),s1)   always...
Always...   always...
If cpe2 > 0.01
 If value2 > total.value * (signal2 + gap(p))
   let price(my.order2(i)) = cpe2 * offer.at.factor(t)
```

```
   let quantity(my.order2(i))=min.f((value2-total.value*signal2)/
                          price(my.order2(i)), number.of.shares2(i))
   if quantity(my.order2(i)) <= 0   return   otherwise...
   call sell.order(my.order2(i),s2)
 Else... If value2 < total.value * (signal2 - gap(p))
   let price(my.order2(i)) = cpe2 * bid.at.factor(t)
   let quantity(my.order2(i))=min.f(abs.f(value2-total.value*signal2),
                              abs.f(dollars.of.cash(i)))/cpe2
   if dollars.of.cash(i) < price(my.order2(i))*quantity(my.order2(i))
   return    otherwise...
   if quantity(my.order2(i)) <= 0   return   otherwise...
   call buy.order(my.order2(i),s2)   always...
Always...   always...

If (m.buyers(my.order1(i)) = 0 and m.buyers(my.order2(i)) = 0
     and m.sellers(my.order1(i)) = 0 and m.sellers(my.order2(i)) = 0)
     or time.v+change.bid.interval(t) >
     min.f(next.portfolio.review.time(i),next.dep.withdraw.time(i))
     return
   else...
   wait change.bid.interval(t) days
   if m.buyers(my.order1(i)) = 0 and m.buyers(my.order2(i)) = 0
     and m.sellers(my.order1(i)) = 0 and m.sellers(my.order2(i)) = 0
     return    else...
   Call cancel.order(my.order1(i),s1)
   Call cancel.order(my.order2(i),s2)
   go to top
end.

Routine to cancel.order(order,s).
Define order and s as integer variables
    ''order and security pointers

If m.buyers(order) = 1
      remove order from buyers(s)
      return
Else...
If m.sellers(order) = 1
   remove order from sellers(s)

Always  return  end.

Routine to compute.stats given x, x2, xy
  yielding bet, alp, t, sharp, trey and sig
Define n as an integer variable
Let n = number.of.periods - remaining.report.periods + 1
Let y = sum.mkt.return
Let y2 = sum.mkt.return.squared

Let bet = (n*xy - x*y) / (n*y2 - y**2)
Let alp = (x - bet*y) / n
Let d2 = (x2/n-((x/n)**2)-(bet**2)*y2/n+((bet*y/n)**2))
                  *(1/n + ((y/n)**2)/y2)
Let d = sqrt.f(abs.f(d2))
Let t = alp / d

Let sig = sqrt.f(abs.f((n*x2 - x**2) / n**2))
Let sharp = x/n/sig
Let trey = x/n/bet

Return.
End.

Process cppi.investor

Define s, i and p as integer variables  ''point to investor and prototype
Let s = 1
Let i = cppi.investor
Let p = my.prototype(i)
File cppi.investor in investors(p)
Now initialize.investor given p and i

Let cppi.floor(i)=(1.0-cppi.init.cushion.pct(p))*(dollars.of.cash(i)
                          + number.of.shares(i)*last.sale.price(s))
Let next.plan.restart.time(i) = uniform.f(0,cppi.period(p),1)

'decide.next.event.type'
Let cpe = current.price.estimate(s)
If dollars.of.cash(i)+number.of.shares(i)*cpe <= 0  suspend  always...
If next.plan.restart.time(i) < min.f(next.portfolio.review.time(i),
                                     next.dep.withdraw.time(i))
Go await_plan_restart

Else if next.portfolio.review.time(i) < next.dep.withdraw.time(i)
                                                  go await_review
  else go await_deposit_withdraw
```

```
'await_review'   ''then do the portfolio review next
Wait next.portfolio.review.time(i) - time.v days

'Do.review'
Now cancel.order(my.order(i),s)   ''if any''
Now calc.cppi.order given p and i
Add exponential.f(portfolio.review.interval(p),1)
                                    to next.portfolio.review.time(i)
Go decide.next.event.type

'await_deposit_withdraw'  ''then do the deposit/withdraw
Wait next.dep.withdraw.time(i) - time.v days
Now deposit.or.withdraw given p and i
Now cancel.order(my.order(i),s)   ''if any''
Now calc.cppi.order given p and i  ''since ratio may now be too high or low.
Go decide.next.event.type

'await_plan_restart'
Wait next.plan.restart.time(i) - time.v days

'Restart.plan'
Let portfolio.value = dollars.of.cash(i) +
                      number.of.shares(i)*current.price.estimate(s)
Let cppi.floor(i) = (1.0 - cppi.plan.start.cushion.pct(p))*portfolio.value
Add cppi.period(p) to next.plan.restart.time(i)
Let next.portfolio.review.time(i) = time.v
Go do.review
End.

Routine to create.arb.prototype(rp.nr)
Define rp.nr, i and n as integer variables.
Define p to mean arb.prototype

Create a arb.prototype
File arb.prototype in arb.prototypes
Let prototype.number(p) = 4
Let pr.sequence.number(p) = rp.nr

Call read.gen.proto.attributes(arb.prototype)

Read g, risk.level(p), n
Let gap(p) = g + 1.0
For i = 1 to n, activate a arbitrager(arb.prototype,i) now
Return   End.

Routine to create.cppi.prototype(pr.nr)
Define pr.nr, i and n as integer variables.
Define p to mean cppi.prototype

Create a cppi.prototype
File cppi.prototype in cppi.prototypes
Let prototype.number(p) = 2
Let pr.sequence.number(p) = pr.nr

Call read.gen.proto.attributes(cppi.prototype)

Read cppi.period(p), cppi.ratio(p), low.ratio(p), high.ratio(p),
 cppi.plan.start.cushion.pct(p), cppi.init.cushion.pct(p),
 max.stock.to.assets(p), and n

For i = 1 to n, activate a cppi.investor(cppi.prototype,i) now

Return   End.

Routine to create.orpi.prototype(pr.nr)
Define pr.nr, i and n as integer variables.
Define p to mean orpi.prototype

Create a orpi.prototype
File orpi.prototype in orpi.prototypes
Let prototype.number(p) = 3
Let pr.sequence.number(p) = pr.nr

Call read.gen.proto.attributes(orpi.prototype)

Read orpi.period(p), est.sigma(p), orpi.ratio(p), buffer(p), and n

For i = 1 to n, activate a orpi.investor(orpi.prototype,i) now

Return   End.

Routine to create.rebalancer.prototype(rp.nr)
Define rp.nr, i and n as integer variables.
Define p to mean rebalancer.prototype
```

```
Create a rebalancer.prototype
File rebalancer.prototype in rebalancer.prototypes
Let prototype.number(p) = 1
Let pr.sequence.number(p) = rp.nr

Call read.gen.proto.attributes(rebalancer.prototype)

Read target.fraction(p), low.fraction(p), high.fraction(p), n
For i = 1 to n, activate a rebalancer(rebalancer.prototype,i) now
Return   End.

Routine to create.spec.prototype(rp.nr)
Define rp.nr, i and n as integer variables.
Define p to mean spec.prototype

Create a spec.prototype
File spec.prototype in spec.prototypes
Let prototype.number(p) = 5
Let pr.sequence.number(p) = rp.nr

Call read.gen.proto.attributes(spec.prototype)

Read gap(p), n
For i = 1 to n, activate a speculator(spec.prototype,i) now
Return   End.

Routine to create.trading.strategies
Define a as an alpha variable

Read a   read as /,/

Read number.of.trading.strategies
Create every trading.strategy

Read a   read as /,/,/,/,/

For every trading.strategy,
 read bid.at.factor, offer.at.factor, change.bid.interval

Return   End.

Routine for current.price.estimate(s).
Define s as an integer variable ''security pointer

If buyers(s) is empty and sellers(s) is empty
   return with last.sale.price(s))
Else if buyers(s) is empty and sellers(s) is not empty
   return with price(f.sellers(s))*no.bids.price.est.factor
Else if sellers(s) is empty and buyers(s) is not empty
   return with price(f.buyers(s))*no.offers.price.est.factor
Else
   return with (price(f.buyers(s)) + price(f.sellers(s)))/2.0
End.

Process daily.close
Define s as an integer variable ''security pointer
call print.daily.heading
'top'
Wait 1 day

For every security, do...
  let s = security
  Let pct.change.for.day(s)=(last.sale.price(s)/last.closing.price(s))-1
  Let last.closing.price(s)=last.sale.price(s)
  If buyers(s) is not empty let bid = price(f.buyers(s))
  Else let bid = 0.0
  Always...
  If sellers(s) is not empty let asked = price(l.sellers(s))
  else let asked = 0.0
  Always...
''  print 1 lines with time.v, daily.high(s), daily.low(s),last.sale.price(s),
''  bid, asked and daily.volume(s), pct.change.for.day(s), s thus...
''  **** *****.** *****.** *****.** *****.** *****.** ****** **.***** ***
    If last.sale.price(s) > 10**6
  Stop
  Else ...
  Reset the daily totals of last.sale.price(s)
  Reset the daily totals of sale.quantity(s)
Loop
  Go to top
End.

Routine to deposit.or.withdraw ''for'' given p and i
Define p and i as integer variables ''pointers to an investor i with proto p.

Let deposit.amount = uniform.f(min.deposit(p),max.deposit(p),1)
```

```
Add deposit.amount to dollars.of.cash(i)
Add deposit.amount to total.dep.withdraw(i)
Add deposit.amount to total.cash
Add deposit.amount to period.deposits

If prototype.number(p) = 2  ''cppi investor
   add deposit.amount*(1.0 - cppi.plan.start.cushion.pct(p))
   to cppi.floor(i)
Always...

Add exponential.f(dep.withdraw.interval(p),1)
   to next.dep.withdraw.time(i)

Return    End.
Routine to initialize.investor ''for'' given p and i
Define p and i as integer variables ''prototype and investor pointers

Let init.value(i) = init.portfolio.value(p)
Add init.value(i) to market.value
If prototype.number(p) = 4 or prototype.number(p) = 5
Let number.of.shares1(i) = init.portfolio.value(p)*init.stock.fraction(p)
                                                  / last.sale.price(2) / 2
Let number.of.shares2(i) = init.portfolio.value(p)*init.stock.fraction(p)
                                                  / last.sale.price(3) / 2
Add number.of.shares1(i) to total.number.of.shares(2)
Add number.of.shares2(i) to total.number.of.shares(3)
Let dollars.of.cash(i) = init.portfolio.value(p)
                          - number.of.shares1(i)*last.sale.price(2)
                          - number.of.shares2(i)*last.sale.price(3)
Else...
Let number.of.shares(i) = init.portfolio.value(p)*init.stock.fraction(p)
                                                  /last.sale.price(1)
Add number.of.shares(i) to total.number.of.shares(1)
Let dollars.of.cash(i) = init.portfolio.value(p) -
                          number.of.shares(i)*last.sale.price(1)
Always...
Add dollars.of.cash(i) to total.cash
Let next.portfolio.review.time(i) = time.v +
                      exponential.f(portfolio.review.interval(p),1)
Let next.dep.withdraw.time(i) = time.v +
                     exponential.f(dep.withdraw.interval(p),1)
Create an order called my.order(i)
Let customer(my.order(i)) = i
Let sec.nr(my.order(i)) = 1
If prototype.number(p) = 4 or prototype.number(p) = 5
  Create an order called my.order1(i)
  Let customer(my.order1(i)) = i
  Let sec.nr(my.order1(i)) = 2
  Create an order called my.order2(i)
  Let customer(my.order2(i)) = i
  Let sec.nr(my.order2(i)) = 3
Always...
Return   End.

Routine to ''compute'' normal.cdf given d.one yielding nd.one

Let YY = 1/(1+.2316419*abs.f(d.one))
Let ZZ = .3989423*exp.f(-1*(d.one**2)/2)
Let nd.one = 1-ZZ*(1.330274*YY**5 - 1.821256*YY**4 + 1.781478*YY**3
                 - .356538*YY**2 + .3193815*YY)
If d.one > 0 go to bottom
    else...
    let nd.one = 1-nd.one
'bottom'
''Print 1 line with d.one, nd.one and current.price.estimate thus
''**.********     *.********      ****.**
''Above two lines were used as a check for d.one and nd.one
Return      end.

Process orpi.investor

Define s, i and p as integer variables  ''point to investor and prototype
Let s = 1
Let i = orpi.investor
Let p = my.prototype(i)
File orpi.investor in investors(p)
Now initialize.investor given p and i
Let exercise.price(i) = current.price.estimate(s)/orpi.ratio(p)
Let next.plan.restart.time(i) = uniform.f(0,orpi.period(p),1)

'decide.next.event.type'
Let cpe = current.price.estimate(s)
If dollars.of.cash(i)+number.of.shares(i)*cpe <= 0  suspend  always...
If next.plan.restart.time(i) < min.f(next.portfolio.review.time(i),
                                     next.dep.withdraw.time(i))
```

```
Go await_plan_restart

Else if next.portfolio.review.time(i) < next.dep.withdraw.time(i)
                                                    go await_review
  else go await_deposit_withdraw

'await_review'   ''then do the portfolio review next
Wait next.portfolio.review.time(i) - time.v days

'Do.review'
Now cancel.order(my.order(i),s)   ''if any''
Now calc.orpi.order given p and i
Add exponential.f(portfolio.review.interval(p),1)
                                    to next.portfolio.review.time(i)
Go decide.next.event.type


'await_deposit_withdraw'  ''then do the deposit/withdraw
Wait next.dep.withdraw.time(i) - time.v days
Now deposit.or.withdraw given p and i
Now cancel.order(my.order(i),s)   ''if any''
Now calc.orpi.order given p and i  ''since ratio may now be too high or low.
Go decide.next.event.type

'await_plan_restart'
Wait next.plan.restart.time(i) - time.v days

'Restart.plan'
Let exercise.price(i) = current.price.estimate(s)/orpi.ratio(p)
Add orpi.period(p) to next.plan.restart.time(i)
Let next.portfolio.review.time(i) = time.v
Go do.review
End.

Routine print.daily.heading
''Print 3 lines thus...
''
''Day    High      Low       Last      Bid     Asked   Volume    Return
''
Return   End.

Routine to print.status
Define s as an integer variable
Let lines.v = 10000
''Print 6 lines with time.v thus...
''
''              Report for period ending day ***
''
''sec   last     volume  accumulated  maximum  minimum  standard   standard
''no.   sale       for     volume      price    price   deviation  deviation
''      price    period               change   change  for period  so far
''
For every security, do...
  Let s = security
  Let lsp = last.sale.price(s)
  Let value = value + lsp*total.number.of.shares(s)
''print 1 line with s, last.sale.price(s),
''period.volume(s), grand.volume(s), max.ppc(s), min.ppc(s), pspc(s), gspc(s).
''thus...
''***  *****.**  ********  *********  **.****  **.****   **.****    **.****

Reset the period totals of sale.quantity(s)
Reset the period totals of pct.change.for.day(s)
''let curr.price = current.price.estimate(s)
''write as
''"sell list (price, quant, p-type, p-nr., i-nr.,cash.val, shares.val): ",/
''for each order in sellers(s), do...
''    let i = customer(order)
''    let p = my.prototype(i)
''    write price(order), quantity(order), prototype.number(p),
''      pr.sequence.number(p),
''      id.nr(i), dollars.of.cash(i), number.of.shares(i)*curr.price
''      as d(10,3), s 2, i 8, 3 i 6, 2 d(10,1),/
''loop
''write as /
''write as
''"buy list (price, quant, p-type, p-nr., i-nr., cash.val, shares.val): ",/
''for each order in buyers(s), do...
''    let i = customer(order)
''    let p = my.prototype(i)
''    write price(order), quantity(order), prototype.number(p),
''      pr.sequence.number(p),
''      id.nr(i), dollars.of.cash(i), number.of.shares(i)*curr.price
''      as d(10,3), s 2, i 8, 3 i 6,2 d(10,1),/
''loop
Loop
```

```
Define r and nn as integer variables
Let mv = value + total.cash
Let nn = number.of.periods - remaining.report.periods + 1
Let mkt.return = (mv - market.value - period.deposits)
                 /(market.value + 0.5*period.deposits)
''Write lsp, total.number.of.shares, total.cash, mv, market.value,
''  period.deposits, last.period.price
''  as d(6,2), i 8, d(11,2), d(11,2), d(11,2), d(11,2), d(8,2),/
Let market.value = mv
Let period.deposits = 0.0
Add mkt.return to sum.mkt.return
Add (mkt.return**2) to sum.mkt.return.squared
Let mkt.sigma = sqrt.f(abs.f((nn*sum.mkt.return.squared
                - sum.mkt.return**2) / nn**2))
Let mkt.sharpe = sum.mkt.return/nn/mkt.sigma
Let mkt.treynor = sum.mkt.return/nn
If remaining.report.periods = 1
Write mkt.return, sum.mkt.return, mkt.return**2, sum.mkt.return.squared,
  mkt.sigma, mkt.sharpe, and mkt.treynor as
  d(9,4), d(9,4), d(12,8), d(12,8), d(12,8), d(12,8), d(12,8),/
Always...
Let lsp = last.sale.price(1)
Let lsp1 = last.sale.price(2)
Let lsp2 = last.sale.price(3)
If rebalancer = 0 go to next1 always...
''print 5 lines thus...
''            Status of rebalancing investors
''
''   id.    number of      share      dollars of        total        period
''   nr.     shares        value         cash           value       deposits
''
For each rebalancer in investors(my.prototype(rebalancer)), do...
Let r = rebalancer
  write ''id.nr(r),
  ''  number.of.shares(r), number.of.shares(r)*lsp, dollars.of.cash(r),
    dollars.of.cash(r) + number.of.shares(r)*lsp'',
  ''  total.dep.withdraw(r)
as ''i 8, i 8, d(15,2), d(15,2), ''d(15,2)'', d(10,2),/

Let value = number.of.shares(r)*lsp + dollars.of.cash(r)
Let return = (value - init.value(r) - total.dep.withdraw(r))
             /(init.value(r) + 0.5*total.dep.withdraw(r))
Add return to sum.of.returns(r)
Add (return**2) to sum.of.squares(r)
Add (return*mkt.return) to sum.of.xy(r)
Let init.value(r) = value
''Write return, sum.of.returns(r), return**2, sum.of.squares(r), sum.of.xy(r)
''  as d(8,4), d(8,4), d(12,8), d(12,8), d(12,8),/

Let total.dep.withdraw(r) = 0.0
If remaining.report.periods = 1
  call compute.stats giving sum.of.returns(r), sum.of.squares(r), sum.of.xy(r)
  yielding beta(r), alpha(r), t.value(r), sharpe(r), treynor(r), sigma(r)
Write beta(r), alpha(r), t.value(r), sharpe(r), treynor(r), sigma(r),
  prototype.number(my.prototype(r))
  as d(9,4), d(9,4), d(9,4), d(9,6), d(9,6), d(9,6), i 3,/
Always...
loop
write as /
'next1'
If cppi.investor = 0 go to next2 always...
''print 5 lines thus...
''            Status of CPPI investors
''
''   id.    number of      share      dollars of        total        period
''   nr.     shares        value         cash           value       deposits
''
For each cppi.investor in investors(my.prototype(cppi.investor)), do...
Let r = cppi.investor
  write ''id.nr(r),
  ''number.of.shares(r), number.of.shares(r)*lsp, dollars.of.cash(r),
  dollars.of.cash(r) + number.of.shares(r)*lsp'',
  ''total.dep.withdraw(r)
as ''i 8, i 8, d(15,2), d(15,2), ''d(15,2)'', d(10,2),/

Let value = number.of.shares(r)*lsp + dollars.of.cash(r)
Let return = (value - init.value(r) - total.dep.withdraw(r))
             /(init.value(r) + 0.5*total.dep.withdraw(r))
Add return to sum.of.returns(r)
Add (return**2) to sum.of.squares(r)
Add (return*mkt.return) to sum.of.xy(r)
Let init.value(r) = value
''Write return, sum.of.returns(r), return**2, sum.of.squares(r), sum.of.xy(r)
''  as d(8,4), d(8,4), d(12,8), d(12,8), d(12,8),/

Let total.dep.withdraw(r) = 0.0
```

```
If remaining.report.periods = 1
  call compute.stats giving sum.of.returns(r), sum.of.squares(r), sum.of.xy(r)
  yielding beta(r), alpha(r), t.value(r), sharpe(r), treynor(r), sigma(r)
Write beta(r), alpha(r), t.value(r), sharpe(r), treynor(r), sigma(r),
  prototype.number(my.prototype(r))
  as d(9,4), d(9,4), d(9,4), d(9,6), d(9,6), d(9,6), i 3,/
Always...
loop
write as /
'next2'
If orpi.investor = 0 go to next3 always...
''print 5 lines thus...
''            Status of ORPI investors
''
''   id.    number of      share      dollars of        total        period
''   nr.     shares        value         cash           value       deposits

For each orpi.investor in investors(my.prototype(orpi.investor)), do...
Let r = orpi.investor
  write ''id.nr(r),
  ''number.of.shares(r), number.of.shares(r)*lsp, dollars.of.cash(r),
  dollars.of.cash(r) + number.of.shares(r)*lsp'',
  ''total.dep.withdraw(r)
as ''i 8, i 8, d(15,2), d(15,2), ''d(15,2)'', d(10,2),/

Let value = number.of.shares(r)*lsp + dollars.of.cash(r)
Let return = (value - init.value(r) - total.dep.withdraw(r))
                /(init.value(r) + 0.5*total.dep.withdraw(r))
Add return to sum.of.returns(r)
Add (return**2) to sum.of.squares(r)
Add (return*mkt.return) to sum.of.xy(r)
Let init.value(r) = value
''Write return, sum.of.returns(r), return**2, sum.of.squares(r), sum.of.xy(r)
''  as d(8,4), d(8,4), d(12,8), d(12,8), d(12,8),/

Let total.dep.withdraw(r) = 0.0
If remaining.report.periods = 1
  call compute.stats giving sum.of.returns(r), sum.of.squares(r), sum.of.xy(r)
  yielding beta(r), alpha(r), t.value(r), sharpe(r), treynor(r), sigma(r)
Write beta(r), alpha(r), t.value(r), sharpe(r), treynor(r), sigma(r),
  prototype.number(my.prototype(r))
  as d(9,4), d(9,4), d(9,4), d(9,6), d(9,6), d(9,6), i 3,/
Always...
loop
write as /
'next3'
If arbitrager = 0 go to next4 always...
''print 5 lines thus...
''            Status of Arbitragers
''
''   id.    number of      share      dollars of        total        period
''   nr.     shares        value         cash           value       deposits

For each arbitrager in investors(my.prototype(arbitrager)), do...
Let r = arbitrager

Let value = number.of.shares(r)*lsp + number.of.shares1(r)*lsp1
          + number.of.shares2(r)*lsp2 + dollars.of.cash(r)
  write ''id.nr(r),
  ''number.of.shares(r), value, dollars.of.cash(r),
  dollars.of.cash(r) + value'',
  ''total.dep.withdraw(r)
as ''i 8, i 8, d(15,2), d(15,2), ''d(15,2)'', d(10,2),/
Let return = (value - init.value(r) - total.dep.withdraw(r))
                /(init.value(r) + 0.5*total.dep.withdraw(r))
Add return to sum.of.returns(r)
Add (return**2) to sum.of.squares(r)
Add (return*mkt.return) to sum.of.xy(r)
Let init.value(r) = value
''Write return, sum.of.returns(r), return**2, sum.of.squares(r), sum.of.xy(r)
''  as d(8,4), d(8,4), d(12,8), d(12,8), d(12,8),/

Let total.dep.withdraw(r) = 0.0
If remaining.report.periods = 1
  call compute.stats giving sum.of.returns(r), sum.of.squares(r), sum.of.xy(r)
  yielding beta(r), alpha(r), t.value(r), sharpe(r), treynor(r), sigma(r)
Write beta(r), alpha(r), t.value(r), sharpe(r), treynor(r), sigma(r),
  prototype.number(my.prototype(r))
  as d(9,4), d(9,4), d(9,4), d(9,6), d(9,6), d(9,6), i 3,/
Always...
loop
write as /
'next4'
If speculator = 0 go to next5 always...
''print 5 lines thus...
''            Status of Speculators
```

```
''
''   id.    number of      share      dollars of        total        period
''   nr.    shares(1)      value         cash           value      deposits

For each speculator in investors(my.prototype(speculator)), do...
Let r = speculator

Let value = number.of.shares(r)*lsp + number.of.shares1(r)*lsp1
          + number.of.shares2(r)*lsp2 + dollars.of.cash(r)
  write ''id.nr(r),
  ''number.of.shares1(r), value, dollars.of.cash(r),
  dollars.of.cash(r) + value'',
  ''total.dep.withdraw(r)
as ''i 8, i 8, d(15,2), d(15,2), ''d(15,2)'', d(10,2),/
Let return = (value - init.value(r) - total.dep.withdraw(r))
             /(init.value(r) + 0.5*total.dep.withdraw(r))
Add return to sum.of.returns(r)
Add (return**2) to sum.of.squares(r)
Add (return*mkt.return) to sum.of.xy(r)
Let init.value(r) = value
''Write return, sum.of.returns(r), return**2, sum.of.squares(r), sum.of.xy(r)
''  as d(8,4), d(8,4), d(12,8), d(12,8), d(12,8),/

Let total.dep.withdraw(r) = 0.0
If remaining.report.periods = 1
  call compute.stats giving sum.of.returns(r), sum.of.squares(r), sum.of.xy(r)
  yielding beta(r), alpha(r), t.value(r), sharpe(r), treynor(r), sigma(r)
Write beta(r), alpha(r), t.value(r), sharpe(r), treynor(r), sigma(r),
  prototype.number(my.prototype(r))
  as d(9,4), d(9,4), d(9,4), d(9,6), d(9,6), d(9,6), i 3,/
Always...
loop
write as /
'next5'
''write as /,/
call print.daily.heading
return  end
Routine to read.gen.proto.attributes ''for prototype''(p)
Define p as an integer variable  ''prototype pointer

Read init.portfolio.value(p), init.stock.fraction(p),
 portfolio.review.interval(p), dep.withdraw.interval(p),
 min.deposit(p), max.deposit(p), and  trading.strategy.nr(p)

If 0 < trading.strategy.nr(p) <= number.of.trading.strategies is false
   write as "incorrect trading strategy specified",/   stop
Always return
End.

Process rebalancer

Define s, r and rp as integer variables ''point to rebalancing investor
                                        ''and prototype
Let s = 1
Let r = rebalancer
Let rp = my.prototype(r)
File rebalancer in investors(rp)
Now initialize.investor ''for the'' given rp and r.

'decide.next.event.type'
Let cpe = current.price.estimate(s)
If dollars.of.cash(r)+number.of.shares(r)*cpe <= .0001  suspend   always...
If next.portfolio.review.time(r) < next.dep.withdraw.time(r) go await_review
  else go await_deposit_withdraw

'await_review'   ''then do the portfolio review next
Wait next.portfolio.review.time(r) - time.v days
Call cancel.order(my.order(r),s)
Call calc.rebalance.order(rp,r)
Add exponential.f(portfolio.review.interval(rp),1) to
                                        next.portfolio.review.time(r)
Go decide.next.event.type

'await_deposit_withdraw'  ''then do the deposit/withdraw
Wait next.dep.withdraw.time(r) - time.v days
Call deposit.or.withdraw ''for'' (rp, r)
Call cancel.order(my.order(r),s)
Call calc.rebalance.order(rp,r) ''since a rebalance may now be necessary
Go decide.next.event.type

End.

Process report

''initial report''
```

```
'top'
Wait report.interval days
Call print.status
Subtract 1 from remaining.report.periods
If remaining.report.periods = 0   stop
Else go to top
End.

Routine sell.order(new.order,s)
Define cust, i, s, q, qb, and new.order as integer variables
    ''order pointers.

'top'
If quantity(new.order) <= 0
Print 1 line thus...
Negative sell order placed.
return   Otherwise...
If n.buyers(s) = 0   file new.order in sellers(s)   return
Always...
  Let i = f.buyers(s)
  Let p = price(i)
  Let qb = quantity(i)
  Let cust = customer(i)
If price(new.order) > p,
      file new.order in sellers(s)
      return
Otherwise...

  Let q = min.f(quantity(new.order), qb)
  Let sale.quantity(s) = q
  Subtract q from quantity(new.order)
  Subtract q from qb
  Let last.sale.price(s) = p
  Add p*q to dollars.of.cash(customer(new.order))
  Subtract p*q from dollars.of.cash(cust)
If s = 1
  Add q to number.of.shares(cust)
  Subtract q from number.of.shares(customer(new.order))
Always... if s = 2
  Add q to number.of.shares1(cust)
  Subtract q from number.of.shares1(customer(new.order))
Always... if s = 3
  Add q to number.of.shares2(cust)
  Subtract q from number.of.shares2(customer(new.order))
Always...

If qb <= 0
   remove i from buyers(s)
Always...  let quantity(i) = qb
If quantity(new.order) <= 0
   return
Otherwise... go to top
End.

Process speculator

Define s, s1, s2, r and rp as integer variables ''point to speculator
                                        ''and prototype
Let s = 1
Let s1 = 2
Let s2 = 3
Let r = speculator
Let rp = my.prototype(r)
File speculator in investors(rp)
Now initialize.investor ''for the'' given rp and r.

'decide.next.event.type'
Let cpe = current.price.estimate(s)
Let cpe1 = current.price.estimate(s1)
Let cpe2 = current.price.estimate(s2)
If dollars.of.cash(r)+number.of.shares1(r)*cpe1+number.of.shares2(r)*cpe2
     + number.of.shares(r)*cpe <= .0001    suspend    always...
If next.portfolio.review.time(r) < next.dep.withdraw.time(r) go await_review
  else go await_deposit_withdraw

'await_review'   ''then do the portfolio review next
Wait next.portfolio.review.time(r) - time.v days
Call cancel.order''if any''(my.order1(r),s1)
Call cancel.order''if any''(my.order2(r),s2)
Call calc.spec.order(rp,r)
Add exponential.f(portfolio.review.interval(rp),1) to
                                        next.portfolio.review.time(r)
Go decide.next.event.type

'await_deposit_withdraw'  ''then do the deposit/withdraw
Wait next.dep.withdraw.time(r) - time.v days
```

```
Call deposit.or.withdraw ''for'' (rp, r)
Call cancel.order''if any''(my.order1(r),s1)
Call cancel.order''if any''(my.order2(r),s2)
Call calc.spec.order(rp,r) ''since a rebalance may now be necessary
Go decide.next.event.type

End.
```

DATAFILE

Days per report period: 65

Report periods per simulation: 100

Number of futures and number of stocks are

1 2

When there are no bids, estimated price is best offer times...: 0.99

When there are no offers estimated price is best bid times...: 1.01

Number of trading strategies...
1

Enter the following information for each trading strategy:

When buying, bid at estimated price times this factor...	When selling, offer at estimated price times this factor...	Reconsider bid/offer this often... (decimal, days)
1.01	0.99	1.0

Number of prototypes for investors who REBALANCE...
1

For each rebalancer prototype enter the following information, on any number of lines...

Value of starting portfolio
Initial ratio of stock value to total value ($stock/$stock+cash)
How often portfolio is reviewed for rebalancing (decimal, days)

How often, on the average, do deposits or withdrawls occur (dec., days)
Minimum deposit (< 0 represents a withdrawl)
Maximum deposit (program assumes uniform distribution between min and max)

Trading strategy number (number between 1 and nr. of trading strategies)

Target fraction of stock to total assets ($stock/$stock+cash)
Low fraction: rebalance if actual is less
High fraction: rebalance if actual is greater

Number of investors with this prototype

```
 100000    .5      5
10   -8000  9000
 1
 0.5  0.46  0.55
 75
```

How many CPPI (constant proportion portfolio insurer) prototypes do you want?

1

For each CPPI investor prototype enter the following information, on any number of lines...

Value of starting portfolio
Initial ratio of stock value to total value ($stock/$stock+cash)
How often portfolio is reviewed to see if actual fraction is close to target fraction (decimal, days)

How often, on the average, do deposits or withdrawls occur (dec., days)
Minimum deposit (< 0 represents a withdrawl)
Maximum deposit (program assumes uniform distribution between min and max)

Trading strategy number (number between 1 and nr. of trading strategies)

Length of insurance plan (decimal, days)
Target ratio of $stock to $cushion (= $assets - $floor)
Low ratio: buy stock if actual ratio is this low
High ratio: sell stock if actual ratio is this high

```
      Cushion as a fraction of portfolio value: at start of simulation
      Cushion as a fraction of portfolio value: start of new insurance plan

      Maximum ratio of stock to assets:  max $stock/$stock+$cash
       ($cash may be negative if this max ratio exceeds 1.0)

      Number of investors with this prototype


       100000      .5     5

        10       -8000    9000

        1

       65        2.0       1.7       2.3

       .25        .25

       1.0

        0

How many ORPI (option replicating portfolio insurer) prototypes
do you want?

1

For each ORPI investor prototype enter the following information,
on any number of lines...

      Value of starting portfolio
      Initial ratio of stock value to total value ($stock/$stock+cash)
      How often portfolio is reviewed to see if actual fraction is
       close to target fraction (decimal, days)

      How often, on the average, do deposits or withdrawls occur (dec., days)
      Minimum deposit ( < 0 represents a withdrawl)
      Maximum deposit (program assumes uniform distribution between min and max)
      Trading strategy number (number between 1 and nr. of trading strategies)

      Length of insurance plan (decimal, days)
      Expected standard deviation of returns (decimal)
      Ratio of stock market price to exercise price of call option being
         replicated (exp(-0.5*sigma_squared*plan_length_in_years) indicates
         half stock and half cash to start, higher ratio: less insurance)
      Buffer:  Percent deviation from target hedge ratio before a buy or sell
         trade is initiated.

      Number of investors with this prototype


      100000      .5      5

      10      -8000    9000

      1

      65      .20      1.00      0.04

      75

How many Arbitrager prototypes
do you want?

1

For each Arbitrage investor prototype enter the following information,
on any number of lines...

      Value of starting portfolio
      Initial ratio of stock value to total value ($stock/$stock+cash)
      How often portfolio is reviewed to see if actual fraction is
       close to target fraction (decimal, days)
      How often, on the average, do deposits or withdrawals occur (dec., days)

      Minimum deposit ( < 0 represents a withdrawal)
      Maximum deposit (program assumes uniform distribution between min and max)
      Trading strategy number (number between 1 and nr. of trading strategies)
      Gap -  one plus % deviation before deciding to trade
      Risk level - % of portfolio risked on each arbitrage
      Number of investors with this prototype


       100000      .5     1
```

```
         10       -8000     9000

          1

        0.01         1

          50

How many Speculator prototypes
do you want?

1

For each Speculate investor prototype enter the following information,
on any number of lines...

     Value of starting portfolio
     Initial ratio of stock value to total value ($stock/$stock+cash)
     How often portfolio is reviewed to see if actual fraction is
      close to target fraction (decimal, days)
     How often, on the average, do deposits or withdrawals occur (dec., days)

     Minimum deposit ( < 0 represents a withdrawal)
     Maximum deposit (program assumes uniform distribution between min and max)

     Trading strategy number (number between 1 and nr. of trading strategies)

     Gap
     Number of investors with this prototype

       100000       .5      5

         10       -8000     9000

          1

          .01

          10

          Random Seed

          1
```

REFERENCES

Anderson, R. and Tutuncu, M. (1988). The Simple Price Dynamics of Portfolio Insurance and Program Trading. *Center for Studies in Futures Markets-Columbia University*, Working Paper #173.

Benninga, S. and Blume, M. (1985). On the Optimality of Portfolio Insurance. *Journal of Finance*, 40, December, 1341-1352.

Black, F. (1986). Noise. *Journal of Finance*, 41, July, 529-543.

Black, F. (1988). An Equilibrium Model of the Crash. *NBER Macroeconomics Annual 1988*, Stanley Fischer, ed., MIT Press, 269-275.

Black, F. (1990). Bluffing. *Goldman Sachs Asset Management*, New York, NY, Unpublished manuscript.

Black, F. and Jones, R. (1987). Simplifying Portfolio Insurance. *Journal of Portfolio Management*, Fall, 48-51.

Black, F. and Scholes, M. (1973). The Pricing of Options and Corporate Liabilities. *Journal of Political Economy*, 81, May-June, 637-659.

Bookstaber, R. and Langsam, J. (1988). Portfolio Insurance Trading Rules. *Journal of Futures Markets*, 8, February, 15-31.

Edwards, F. (1988a). Futures Trading and Cash Market Volatility: Stock Index and Interest Rate Futures. *Journal of Futures Markets*, 8, August, 421-439.

Edwards, F. (1988b). Studies of the 1987 Stock Market Crash: Review and Appraisal. *Journal of Financial Services Research*, 1, June, 231-251.

Elton, E., Gruber, M. and Rentzler, J. (1987). Professionally Managed, Publicly Traded Commodity Funds. *Journal of Business*, 60, April, 175-199.

Elton, E., Gruber, M. and Rentzler, J. (1989). New Public Offerings, Information and Investor Rationality: The Case of Publicly Offered Commodity Funds. *Journal of Business*, 62, January, 1-15.

Grossman, S. (1988a). An Analysis of the Implications for Stock and Futures Price Volatility of Program Trading and Dynamic Hedging Strategies. *Journal of Business*, 61, July, 275-298.

Grossman, S. (1988b). Program Trading and Stock and Futures Price Volatility. *Journal of Futures Markets*, 8, August, 413-419.

Grossman, S. and Miller, M. (1988). Liquidity and Market Structure. *Journal of Finance*, 43, July, 617-633.

Grossman S. and Shiller, R. (1981). The determinants of the Variability of Stock Market Prices. *American Economic Review*, 71, May, 222-227.

Hardouvelis, G. (1988a). Evidence on Stock Market Speculative Bubbles: Japan, the United States, and Great Britain. *Federal Reserve Bank of New York Quarterly Review*, Summer, 4-16.

Hardouvelis, G. (1988b). Margin Requirements and Stock Market Volatility. *Federal Reserve Bank of New York Quarterly Review*, Summer, 80-89.

Kim, G. and Markowitz, H. (1989). Investment rules, margin, and market volatility. *Journal of Portfolio Management*, 16, Fall, 45-52.

Kiviat, P., Villanueva, R. and Markowitz, H. (1983). The SIMSCRIPT II.5 Programming Language, ed. E. Russell, CACI.

Kleidon, A. (1988) Bubbles, Fads, and Stock Price Volatility Tests: A Partial Evaluation-Discussion. *Journal of Finance*, 43, July, 656-660.

Leland, H. (1980). Who Should Buy Portfolio Insurance? *Journal of Finance*, 35, May, 581-594.

Leland, H. (1985). Option Pricing and Replication with Transaction Costs. *Journal of Finance*, 40, December, 1283-1301.

Markowitz, H. (1988). Stock Market Simulator SMS1: Program Description. *Baruch College, CUNY*, Unpublished manuscript.

Merrick, J. Jr. (1988). Portfolio Insurance with Stock Index Futures. *Journal of Futures Markets*, 8, August, 441-455.

Perold, A. (1986). Constant Proportion Portfolio Insurance. *Harvard Business School*, Cambridge, Mass., August, Unpublished Manuscript.

Perold, A. and Sharpe, W. (1988). Dynamic Strategies for Asset Allocation. *Financial Analysts Journal*, January-February, 16-27.

Poterba, J. and Summers, L. (1986). The Persistence of Volatility and Stock Market Fluctuations. *American Economic Review*, 76, December, 1142-1151.

Rubinstein, M. (1985). Alternative Paths to Portfolio Insurance. *Financial Analysts Journal*, July-August, 42-52.

Rubinstein, M. and Leland, H. (1981). Replicating Options with Positions in Stock and Cash. *Financial Analysts Journal*, 37, No. 4, 63-72.

Shiller, R. (1981). Do Stock Prices Move Too Much to be Justified by Subsequent Changes in Dividends? *American Economic Review*, 71, June, 421-435.

Summers, L. (1986). Does the Stock Market Rationally Reflect Fundamental Values? *Journal of Finance*, 41, July, 591-601.

West, K. (1988). Bubbles, Fads, and Stock Price Volatility Tests: A Partial Evaluation. *Journal of Finance*, 43, July, 639-656.