



Minimizing the Probability of Ruin in Retirement

CHRISTOPHER J. ROOK*

ABSTRACT

Retirees who exhaust their savings while still alive are said to experience financial ruin. The savings are typically grown during life's accumulation phase then spent during the retirement decumulation phase. Extensive research into invest and harvest decumulation strategies has been conducted, but recommendations differ markedly. This has likely been a source of concern and confusion for the retiree. The goal of this research is to find what has heretofore been elusive, namely an optimal decumulation strategy. Optimality here implies that no alternate strategy exists or can be constructed that delivers a lower probability of ruin, given a fixed inflation-adjusted withdrawal rate.

Change Log	
Date	SSRN Version/Modification
4/5/2014	Original version.
5/17/2014	Added Section 4.6.1 and corresponding Appendix G for optimal model when time of final withdrawal is random. Added Section 4.8 which discusses model extensions/enhancements.
8/7/2014	Extended the random time of final withdrawal model to a group of retirees in Section 4.6.2. Added full C++ implementation in Appendix H.

*The author is a statistical programmer in the pharmaceutical industry and studies in the Dept. of Systems Engineering at Stevens Institute of Technology. The intent is to publish this research after subjecting it to scrutiny from the academic community. Please forward comments/suggestions to: rookchrj@gmail.com.

Table of Contents

1.0 Introduction.....	1
2.0 Literature Review.....	2
3.0 Formulating and Solving the Model	8
3.1 Definitions and Notation.....	9
3.2 The Retirement Sequence	11
3.3 The Ruin Factor	11
3.4 The Ruin Paths of Two Retirees	14
3.5 The Event of Financial Ruin	16
3.6 An Initial Solution Attempt	18
3.7 A Better Approach	20
3.7.1 Induction at Time $t=T_D$	22
3.7.2 Induction at Time $t=T_D-1$	22
3.7.3 Value Function at Time t	23
3.7.4 Discretizing Along α and $RF(t)$ Dimensions	25
4.0 An Implementation	28
4.1 Distributional Assumptions	29
4.2 Discretized DP Results	29
4.3 Numerical Accuracy of the Solution.....	32
4.4 Consequences of Suboptimal Strategies	32
4.5 Algorithm Scalability.....	33
4.6 Incorporating Random Time of Final Withdrawal (T_D)	33
4.6.1 Optimal Model for Random T_D	34
4.6.1.1 Induction at Time $t=S_{Max}$	36
4.6.1.2 Induction at Time $t=S_{Max}-1$	36
4.6.1.3 Value Function at Time t	37
4.6.2 Extension of Random T_D Model to Multiple Retirees.....	39
4.7 Making Adjustments over Time	40
4.8 Model Extensions/Enhancements	41
5.0 Summary/Conclusion.....	42
Appendix A: Derivation of the Ruin Factor	44
Appendix B: Retirees 1 and 2 (Full Details).....	50
Appendix C: Induction at Times T_D-2 and T_D-3 for Fixed T_D	52
C.1 Induction at Time $t=T_D-2$	52
C.2 Induction at Time $t=T_D-3$	54
Appendix D: Derivation of Ruin Factor Bucket Probabilities.....	57
Appendix E: Limiting Behavior of $V(t, RF(t))$ as # Buckets Increases.....	59
Appendix F: Derivation of Historical Stock and Bond Distributions.....	60
Appendix G: Induction at Times $S_{Max}-2$ and $S_{Max}-3$ for Random T_D	66
G.1 Induction at Time $t=S_{Max}-2$	66
G.2 Induction at Time $t=S_{Max}-3$	68
Appendix H: Full C++ Implementation.....	71
References.....	96

1.0 Introduction

The retiree with savings faces important decisions such as how much to withdraw annually and what asset allocation to use during retirement. Withdraw too little and face a reduced standard of living; withdraw too much and face running out of money, termed *financial ruin*. Likewise, if the retiree invests the wrong percentage in stocks they face ruin from excessive risk or not enough risk. Unfortunately, there is no consensus regarding a precise safe withdrawal rate or the optimal asset allocation for retirees. Generally, a safe withdrawal rate is posited somewhere between 3% and 5%, adjusted annually for inflation, but the probabilities of success when using these rates varies considerably. In retirement, the term *glide-path* refers to asset allocations that either shed or acquire equities systematically over time. Some advocate for a declining, and others for a rising glide-path during retirement. With such a large cohort (> 79 million U.S. Baby Boomers) on the precipice of spending their savings, finding optimal decumulation strategies is increasingly important. The first Baby Boomers to retire at age 65 began doing so in 2011, and will continue at a rate of 10,000/day until 2030.

Much research has been conducted in the area of safe withdrawal rates and a sampling is reviewed here. The general approach has been to develop an intuition-based heuristic, then test it using historical data, Monte Carlo simulation, or bootstrapping. The heuristic often applies to withdrawals and limits them strategically based on decision rules. It can also apply to the retiree's asset allocation and glide-path. This paper differs because it is an attempt to avoid heuristics, and let probabilities drive the analysis. At each point in time, as the retiree ages, there is a minimum probability of ruin based on an optimal asset allocation, and it is the goal of this research to find those values. A model is formulated that, when solved, reveals precisely these measures as a function of time. The solution can then serve as a roadmap for the retiree during

decumulation. It is optimal in the sense that no other asset allocation strategy can deliver a lower probability of ruin. Unfortunately, the model proves intractable under basic market assumptions and must be approximated by discretization. The user, however, controls the approximation's accuracy by setting the discretization precision, which is limited only by available processing power. The user thus has a means to achieve near optimality with this model and the result applies to all retirees using any initial withdrawal rate.

2.0 Literature Review

Bengen (1994) determined the safe initial withdrawal rate for retirees by analyzing historical sequences of returns on portfolios consisting of large company stocks and intermediate term government bonds. He concluded that a 4% initial withdrawal rate, adjusted annually for inflation, would not have been exhausted over any prior 30 year period using 50% stocks and 50% bonds. A 4% initial withdrawal rate was therefore considered safe. In terms of asset allocation, Bengen (1994) concluded that the retiree using a 4% initial withdrawal rate should invest between 50% and 75% of the portfolio in stocks and not change the allocation unless the retiree's objectives change. Bengen advised that the stock allocation be based on the retiree's risk tolerance with 75% being optimal due to the potential for a larger terminal account balance. It was this research that led to what is now commonly referred to as the "4% rule", which is widely used today. Bengen (2006) followed up on his initial research by considering the addition of small cap stocks to the portfolio. Here, the term SAFEMAX was coined and refers to the highest initial withdrawal rate that has a positive ending account balance after 30 years for all historical sequences of returns analyzed. When at least 20% of the portfolio's stock was invested in small cap equities, the SAFEMAX was found to be 4.5%.

Cooley, Hubbard, and Walz (1998) also applied historical sequences of returns to retirement portfolios of stocks and bonds, using the S&P 500 Index and long-term high-grade corporate bonds to represent each. Their findings coincide with Bengen (1994) in concluding that an initial 4% withdrawal rate on an account having 75% stocks and 25% bonds had a 98% success rate using a 30-year retirement horizon. Success here refers to a terminal portfolio with positive balance, and the withdrawal rate is adjusted annually for inflation. They also found that an account consisting of 50% stocks and 50% bonds had a 95% success rate under the same assumptions. As with Bengen (1994), the asset allocations were held constant over time. Cooley, Hubbard, and Walz (2011) updated their initial analysis to include 14 years of new data and found slightly improved results. A portfolio with 75% stocks and 25% bonds realized success rates over 99%, and a portfolio of 50% stocks and 50% bonds saw a success rate of 96%.

Pye (2000) analyzed sustainable withdrawals from retirement portfolios by making distributional assumptions about investment returns and used Monte Carlo simulation to test various withdrawal strategies. Stocks and TIPS were taken as investment alternatives and withdrawals were adjusted annually for inflation, but were reduced once certain thresholds were violated. If the portfolio approached a ruin event, the limits restrict withdrawals and reduce the probability of a shortfall. Instead of measuring the probability of ruin directly, Pye (2000) measured the probability of hitting the limits and thereby facing a reduced standard of living. Success is defined as not hitting the limits during the retirement horizon. For an initial 4% inflation-adjusted withdrawal rate using a portfolio of 100% stocks, the probability of success over a 35 year retirement horizon was 81%. Pye (2000) also found that a 4.5% withdrawal rate had the same 35 year probability of success using a 40% stock and 60% TIPS asset allocation. Note that the asset allocations did not change over the retirement horizon.

Guyton (2004) developed decision rules for making withdrawals and tested these rules on the 30 year period between 1973 and 2003, which is considered the most challenging period for retirees due to high inflation and 2 severe market downturns. Portfolios were allocated to stocks and bonds using either 65% or 80% equities. The rules call for an initial withdrawal rate that was adjusted annually for inflation only when the total account balance did not lose money in the previous year. The retiree would not face reduced withdrawals in absolute terms, but would forego the inflation adjustment during years of poor returns. Further, withdrawals were made from stocks and bonds in a systematic manner that drew from the accounts with prior year gains.

Using these rules, Guyton (2004) found that a portfolio of 65% stocks could sustain an initial withdrawal rate of 5.4% for the entire 30 year period studied, plus 10 additional years (future returns were estimated). A portfolio consisting of 80% equities could sustain an initial withdrawal rate of 5.8% for the same period. These rates are higher than prior research, due to the foregoing of an inflation-adjustment during years when the account performed poorly.

Milevsky and Robinson (2005) introduced a quantity called the stochastic present value (SPV) to derive the distribution for the probability of ruin over the retiree's life expectancy, which was assumed to be exponentially distributed. The SPV is the present value of the retiree's future withdrawals discounted using random market returns. If the SPV exceeds the retiree's account balance at the start of retirement ruin is experienced. Milevsky and Robinson (2005) found that under the assumption of lognormal portfolio returns and exponential life expectancy, the SPV random variable follows a reciprocal gamma distribution of known form. The parameters of the reciprocal gamma distribution are directly related to the parameters of the lognormal returns (mean and variance) and the exponential life expectancy (rate). The solution is closed form in nature, but serves an approximation to the true probability of ruin when the

exponential distribution is used. Milevsky and Robinson (2005) found that an all stock portfolio generated a probability of success of 87.73% using an initial 4% inflation-adjusted withdrawal rate. The probability increased to 91% with a balanced portfolio of 50% stocks and 50% bonds. The asset allocations remained constant over the retirement horizon.

Blanchett (2007) analyzed safe withdrawal rates across various asset allocations by bootstrap sampling historical returns of stocks and bonds. Note that the bond portion of the portfolio consisted of half cash, and stocks included both domestic and international equities. Fixing the success rate at 95%, Blanchett (2007) found that a portfolio using 40% equities and 60% bonds/cash had the highest inflation-adjusted initial withdrawal rate of 4.2% over a 30 year retirement horizon. Blanchet also analyzed 5 glide-path scenarios that either held the stock-bond ratio constant, or reduced equities in a systematic manner over time. The systematic equity reductions followed either a linear, stair, concave, or convex pattern and each glide-path was combined with 8 different starting equity allocations. Overall, including the constant allocation, 43 different glide-paths were tested on retirement horizons ranging between 20 and 40 years. Blanchet (2007) found that, for most initial withdrawal rates, a constant allocation (that is, no glide-path) provided the lowest probability of ruin, or highest probability of success. Specifically, over a 30 year retirement horizon, a 4% initial withdrawal rate (adjusted for inflation) with constant allocation of 30% stocks and 70% bonds would generate the lowest probability of ruin at 2.06% (success probability = 97.94%).

Stout (2008) formulated retirement ruin as the objective of a stochastic optimization problem and solved it using Monte Carlo simulation. Asset allocations to stocks and bonds changed systematically until the minimum probability of ruin was found, which is stochastic since it depends on random returns and random age of death. Unlike other studies that use fixed

retirement horizons, Stout (2008) incorporated survival probabilities from the U.S. Centers for Disease Control. The optimal asset allocation that minimizes the probability of ruin remains constant over time. Stout (2008) found that a 65 year old retiree using a 4.25% inflation-adjusted withdrawal rate would achieve a success rate of 96.75%, assuming an optimal allocation of 50.25% to stocks and 49.75% to bonds. Further, as the withdrawal rate increases above 4.25% the portfolio's equity percentage must also increase to maintain optimality.

Target-date funds are investments sold by financial firms that adjust the equity allocation over time. These investments are useful for retirees as well for those who are saving for retirement. Generally, the glide-path in target-date funds reduces the equity percentage as the investor ages. Some target-date funds cease reallocation at the retirement date, while others continue to reallocate through retirement. Morningstar, Inc. (2009) surveyed financial firms and summarized the glide-paths across a myriad target-date funds. They found that a retiree from 1995 currently has an average of 20% in equities, whereas a retiree from 2000 has an average of 34.5% in equities. A person who retired in 2010 presently has on average 50.2% of their account in equities, and someone who plans to retire in 2015 (within the next year) has on average 62.1% in equities. This survey provides a great deal of insight into the retirement glide-paths used by large financial firms in their target-date funds. On average the retiree starts with about 62% equities and this decreases to a 20% equity allocation 2 decades into retirement.

There is not universal agreement, however, that downward sloping equity glide-paths outperform fixed equity allocations. Similar to Blanchett (2007), Cohen, Gardner, and Fan (2010) found that a constant equity allocation will outperform a sloping equity glide-path almost universally, calling into question the rationale behind the divestment of equities during retirement. Their research is based on the retiree's expected ending wealth which is simulated

when comparing asset allocations. Cohen, Gardner, and Fan (2010) also found that retirees should have a lower starting equity percentage than is recommended in much of the research noted above. Namely, a beginning 32% equity allocation was found to yield a 94% probability of success using an initial 4% withdrawal rate, adjusted annually for inflation. If the starting equity allocation is raised to 60% the probability of success declines to 88%, and a starting equity allocation of 80% reduces the probability of success to 84%. Cohen, Gardner, and Fan's (2010) rationale for a smaller equity allocation is to prevent against sequence risk, which is the risk of poor returns early in retirement. Early account losses coupled with constant, inflation-adjusted withdrawals are known to put the retiree at danger of experiencing financial ruin.

Fan, Murray, and Pittman (2013) developed an adaptive allocation model that monitors the retiree's account balance over time and adjusts the asset allocation dynamically to maximize expected wealth at the end of a 20-year horizon, prior to purchasing an annuity. A decision tree is built using various combinations of market returns and their impact on different asset allocations between time points. By folding back the tree an optimal asset allocation policy reveals itself at the start of retirement. Further, by eliminating sub-optimal nodes, the final tree serves as a decision guide for the retiree over time. At each node in the tree an optimization occurs over various asset allocations using an objective function that punishes shortfalls quadratically. The adaptive model results in higher expected surpluses and lower expected losses when compared with fixed-mix strategies. An interesting feature of the Fan, Murray, and Pittman (2013) model is that it becomes most conservative when current assets match discounted future spending. The adaptive model adjusts the initial withdrawal rate for inflation annually. When the retiree experiences a strong sequence of returns, shortfall risk declines and the penalty approaches zero. This feature reveals a feedback mechanism that encourages the building of

more wealth as wealth grows. Therefore the adaptive model allocates a higher percentage to stocks when the market underperforms, and when it outperforms.

Pfau and Kitces (2014) researched various equity glide-paths for periods of typical and atypical (sub-standard) market returns using Monte Carlo simulation. A 4% initial withdrawal rate adjusted for inflation annually had the highest probability of success using equity glide-paths that start between 20% and 40%, and end between 60% and 80%, which held true for typical and atypical return periods. Rising equity glide-paths thus outperformed both static and declining equity allocations. Pfau and Kitces (2014) offer the intuition behind these findings. Namely, that the retiree's portfolio assumes the greatest sequence of returns risk at the start of retirement when the account balance is largest, at which time it is prudent to limit equity exposure. As the retiree ages the sequence of returns risk diminishes and a higher equity allocation is warranted. This holds true whether the market has performed well or poorly. Pfau and Kitces (2014) note that a rising equity glide-path during a poor market allows the retiree to accumulate equities when prices are low, reaping the benefits of a market upturn. Even if the market has performed well, however, and shortfall risk is small, accumulating stock during the latter part of retirement is viewed as an opportunity to build a legacy for heirs.

3.0 Formulating and Solving the Model

In this section a model is formulated which satisfies the condition of being an optimal decumulation strategy. It also serves as a roadmap to guide the retiree/advisor how to maintain optimality over time. Subsequently, in Section 4, the model is implemented and the results are presented. Some conclusions are justified in appendices to this report, and the reader is encouraged to review those sections as well.

3.1 Definitions and Notation

It is assumed that the retiree owns a diversified investment account consisting of stocks and bonds. Financial ruin is defined as the event of emptying the retirement account while still alive. Define:

- t = time point ($t=0, 1, \dots, T_D$)
- T_D = time of last withdrawal prior to death
- $T_D - k$ = k time points before T_D after making the withdrawal (there are k withdrawals remaining at time $t=T_D - k$)
- $\text{Ruin}(t)$ = the event of ruin is experienced at time t
- $\text{Ruin}^C(t)$ = the event of ruin is not experienced at time t
- $\$A$ = value of retirement account at time $t=0$
- α = proportion of account in stocks ($0 \leq \alpha \leq 1$)
- $(1-\alpha)$ = proportion of account in bonds ($0 \leq \alpha \leq 1$)
- W_R = initial withdrawal rate adjusted for inflation at each time point
- I_t = inflation rate between times $t-1$ and t
- $R_{(t,\alpha)}$ = total rate of return for the investment portfolio with $100*\alpha\%$ stock and $100*(1-\alpha)\%$ bonds between times $t-1$ and t
- $r_{(t,\alpha)} | I_t, R_{(t,\alpha)}$ = inflation-adjusted rate of return for the investment portfolio with $100*\alpha\%$ stock and $100*(1-\alpha)\%$ bonds between times $t-1$ and t , given I_t and $R_{(t,\alpha)}$
- E_R = expense ratio charged by the financial institution per time t
- $\hat{r}_{(t,\alpha)} = (1+r_{(t,\alpha)})(1-E_R)$ inflation/expense adjusted compounded return on the portfolio with $100*\alpha\%$ stock and $100*(1-\alpha)\%$ bonds between times $t-1$ and t

The quantities I_t , $R_{(t,\alpha)}$, $(r_{(t,\alpha)} | I_t, R_{(t,\alpha)})$, and $\hat{r}_{(t,\alpha)}$ are assumed to be continuous R.V.s with known (or estimated) probability distributions, and T_D is also random. Both $\$A$ and W_R are

fixed constants, as is E_R . Note that W_R is based on $\$A$, and will be deducted at time t after adjusting for inflation. For example, if $\$A=\$100k$ and $W_R=0.04$ then $\$100k*(W_R)*(1+I_1)$ is withdrawn at time $t=1$ for spending between times $t=1$ and $t=2$. The terms α and t represent variables under study with α controlled by the retiree/advisor, and t not. The inflation-adjusted rate of return on the portfolio having $100*\alpha\%$ stock and $100*(1-\alpha)\%$ bonds at time t is $(r_{(t,\alpha)} | I_t, R_{(t,\alpha)})$. Since this quantity is calculated after $R_{(t,\alpha)}$ and I_t are observed it is defined conditionally on those random variables. The quantity $\hat{r}_{(t,\alpha)}$ reflects the inflation/expense-adjusted compounded return for the portfolio defined by α at time t . The random variable $\hat{r}_{(t,\alpha)}$ thus depends on 5 quantities: $t, I_t, R_{(t,\alpha)}, \alpha, E_R$, where t, α, E_R are deterministic and I_t and $R_{(t,\alpha)}$ are random. To determine the real account balance at time t , the balance at time $t-1$ is multiplied by $\hat{r}_{(t,\alpha)}$. Finally, $Ruin(t)$ and $Ruin^C(t)$ represent random events that occur with some probability.

The event $Ruin(t)$ has important features. First, from a practical sense, it can only be experienced once.¹ If $Ruin(t)$ occurs, ruin cannot occur after time t . Thus when referring to $P(Ruin(t))$ it would be proper to condition on ruin not being experienced prior to time t . For example, the probability of ruin at time $t=3$ is $P(Ruin(3) | Ruin^C(1) \cap Ruin^C(2))$. Further, the conditioning component can be expressed as $P(Ruin^C(1) \cap Ruin^C(2)) = P(Ruin^C(1))*P(Ruin^C(2) | Ruin^C(1))$. Therefore, the probability of experiencing the event $Ruin(3)$ is formally expressed as:

$$P(Ruin(3) | Ruin^C(1) \cap Ruin^C(2)) = \frac{P(Ruin^C(1) \cap Ruin^C(2) \cap Ruin(3))}{P(Ruin^C(1))*P(Ruin^C(2) | Ruin^C(1))} \quad (1)$$

Second, the probability of experiencing ruin is not static, but changes over time. In fact, it changes continuously with time when markets are open. To see this, suppose a retiree owns an $\alpha=1$ portfolio at time $t=0$ and the real account balance doubles by time $t=1$. Having fixed the initial withdrawal rate at W_R and survived ruin at time $t=1$, a very different probability of ruin

¹ From a probabilistic sense, however, it may be possible to experience ruin more than once.

exists at time $t=1$, than did at time $t=0$. The exact opposite holds true if the portfolio is halved by time $t=1$. At time $t=0$, the probability of ruin can be viewed as a *prior* probability that should be updated once returns are observed in what can loosely be called the *posterior* probability of ruin. It is appropriate for the retiree/advisor to re-evaluate ruin probabilities at scheduled intervals. This analysis assumes that the retiree's objective is to minimize the probability of ruin, not build wealth, given a withdrawal rate W_R . Probabilities of ruin can thus perish quickly and an account that is half the size it was at the previous time point has a higher ruin probability attached to it.

3.2 The Retirement Sequence

Events are assumed to unfold as follows. The retiree begins retirement by withdrawing enough cash for spending between times $t=0$ and $t=1$ leaving a balance of $\$A$ at time $t=0$. The withdrawal rate W_R is then determined based off $\$A$ and adjusted for inflation before each withdrawal. At time $t=1$ the retiree attempts to withdraw the amount of $(\$A) \cdot (W_R) \cdot (1+I_1)$ from the account which has been compounding returns for one time point. If the account balance is insufficient to make this withdrawal the event $\text{Ruin}(1)$ occurs, otherwise $\text{Ruin}^C(1)$ occurs. Note that the account value at time $t=1$ before attempting the withdrawal is $(\$A) \cdot (1+R_{(1,0)}) \cdot (1-E_R)$, where $R_{(1,0)}$ represents the portfolio's random total rate of return between times $t=0$ and $t=1$. Under an optimal strategy, at time $t=1$ the retiree/advisor should reassess the probability of ruin at any future time point ($t=2, 3, \dots, T_D$), and rebalance the portfolio accordingly.

3.3 The Ruin Factor

There is a quantity which encapsulates much information needed to assess the probability of ruin at time t . This quantity, termed the *ruin factor*, can and should be updated and reassessed at periodic intervals by the retiree/advisor. It is also actionable in the sense that it helps to reveal the precise asset allocation that minimizes the probability of ruin for the remainder of retirement.

To see this note that the account value at time $t=1$ is given by $(\$A)*(1+R_{(1,\alpha)})*(1-E_R)$, whereas the retiree requires $(\$A)*(W_R)*(1+I_1)$ for spending between times $t=1$ and $t=2$. Thus,²

$$\text{Ruin}(1) \text{ occurs} \quad \leftrightarrow \quad \$A*(1+R_{(1,\alpha)})*(1-E_R) \leq \$A*(W_R)*(1+I_1) \quad (2a)$$

$$\leftrightarrow \quad \$A*(1+r_{(1,\alpha)})*(1+I_1)*(1-E_R) \leq \$A*(W_R)*(1+I_1) \quad (2b)$$

$$\leftrightarrow \quad (1+r_{(1,\alpha)})*(1-E_R) \leq W_R \quad (2c)$$

$$\leftrightarrow \quad \hat{r}_{(1,\alpha)} \leq W_R \quad (2d)$$

It follows that $P(\text{Ruin}(1)) = F_{\hat{r}_{(1,\alpha)}}(W_R)$ (where $F_X(x) = P(X \leq x)$ represents the CDF of the random variable X). Note that the above assumes $(1+I_1) > 0$, or that the value of money is always greater than zero. Therefore, at time $t=0$, $P(\text{Ruin}(1))$ can be measured using various α . Define the *ruin factor*, $RF(t)$, at time t as the following recursive quantity:

$$\text{Let } RF(t) = RF(t-1) / [(1+r_{(t,\alpha)})(1-E_R) - RF(t-1)], \text{ for } t=1, 2, \dots, T_D \quad (3a)$$

$$= RF(t-1) / [\hat{r}_{(t,\alpha)} - RF(t-1)], \text{ for } t=1, 2, \dots, T_D \quad (3b)$$

$$\text{And assign, } RF(0) = W_R \quad (3c)$$

Then given no ruin prior to time t , it can be shown that the event $\text{Ruin}(t)$ occurs as:

$$\text{Ruin}(t) \mid (\text{Ruin}^C(1) \cap \dots \cap \text{Ruin}^C(t-1)) \leftrightarrow (1+r_{(t,\alpha)})*(1-E_R) \leq RF(t-1) \quad (4a)$$

$$\leftrightarrow \hat{r}_{(t,\alpha)} \leq RF(t-1) \quad (4b)$$

(See Appendix A.) Note that at time $t=t-1$, $RF(t-1)$ is known. Ruin is then experienced at time t if the random variable $\hat{r}_{(t,\alpha)} \leq RF(t-1)$. Assuming the CDF of $\hat{r}_{(t,\alpha)}$ is known or estimated, $RF(t-1)$ can be calculated and $P(\text{Ruin}(t))$ evaluated across various α at time $t=t-1$.

The ruin factor can be defined using any unit of time, assuming $\hat{r}_{(t,\alpha)}$ is the appropriate return. $RF(t)$ is not continuous with respect to time, however, since it depends on $\hat{r}_{(t,\alpha)}$. The ruin factor is positive until ruin occurs (if it occurs) when it becomes negative (or ∞). To see this note that at any time t , $\text{Ruin}(t)$ is avoided \leftrightarrow :

² The notation “ \leftrightarrow ” used here indicates an if-and-only-if condition.

$$\hat{r}_{(t,\alpha)} > RF(t-1) \quad (5)$$

This condition states that the upcoming time point's inflation/expense-adjusted return, $\hat{r}_{(t,\alpha)}$, must exceed $RF(t-1)$ to make the required end-of-year withdrawal. If this fails to hold, $\hat{r}_{(t,\alpha)} \leq RF(t-1)$ and $RF(t) = RF(t-1) / [\hat{r}_{(t,\alpha)} - RF(t-1)]$ is either negative (when $\hat{r}_{(t,\alpha)} < RF(t-1)$) or undefined (when $\hat{r}_{(t,\alpha)} = RF(t-1)$). If $RF(t)$ declines between time points the retiree has improved their position, and vice-versa. The ruin factor declines when the denominator exceeds 1. Namely,

$$RF(t) < RF(t-1) \quad (6a)$$

$$\leftrightarrow \left[RF(t-1) / [\hat{r}_{(t,\alpha)} - RF(t-1)] < RF(t-1) \right] \quad (6b)$$

$$\leftrightarrow [\hat{r}_{(t,\alpha)} - RF(t-1)] > 1 \quad (6c)$$

$$\leftrightarrow \hat{r}_{(t,\alpha)} > 1 + RF(t-1) \quad (6d)$$

Therefore, an inflation/expense-adjusted market return that exceeds the prior year's ruin factor (plus 1) will improve the retiree's position with respect to ruin.³ If the retiree is able to generate returns (specifically, $r_{(t,\alpha)} - E_R - r_{(t,\alpha)} * E_R$) that are exactly equal to $W_R = RF(0)$, then their position with respect to ruin does not change. In this case, the ruin factor starts at $RF(0) = W_R$, and remains constant at each successive time point (that is, $RF(1) = W_R$, $RF(2) = W_R$, ...). Thus, the retiree can avoid ruin by generating returns exactly equal to W_R at each time t .

$RF(t)$ also reveals in exact terms the real balance at time t , namely, $(\$A) * RF(0) / RF(t)$. (Proof omitted.) To see this simply express each $\hat{r}_{(t,\alpha)}$ in terms of $RF(t)$. For example, $\hat{r}_{(1,\alpha)} = RF(0) * [1 + 1 / RF(1)]$, $\hat{r}_{(2,\alpha)} = RF(1) * [1 + 1 / RF(2)]$, and so on. Start at $t=1$ and compound the initial balance $(\$A)$ by $\hat{r}_{(1,\alpha)}$, then remove the real withdrawal $(\$A) * (W_R)$. Repeat at each t to obtain the real account balance at time t . Finally, start with the inner most term and express each quantity as ruin factors. The expression collapses leaving a real account balance of $(\$A) * RF(0) / RF(t)$ at time t . The reciprocal of $RF(t)$ then equals the number of real withdrawals remaining at time t .

³ Note that $\hat{r}_{(t,\alpha)}$ can be compared with $1 + RF(t-1)$, or equivalently $(r_{(t,\alpha)} - E_R - r_{(t,\alpha)} * E_R)$ with $RF(t-1)$.

Note that at time t , $RF(t)$ is calculated and known. Future $RF(t)$ values are not known because they are functions of market returns yet to be observed. Therefore, future ruin factors are random variables, and at the start of retirement, $RF(0)=W_R$ is the only known ruin factor. The retiree/advisor does exercise some control over future $RF(t)$ values via the choice of α . To see this recall that $RF(t)=RF(t-1)/[\hat{r}_{(t,\alpha)} - RF(t-1)]$. If a low- α portfolio is used between times $t-1$ and t , the retiree/advisor is more confident of $\hat{r}_{(t,\alpha)}$, and thus $RF(t)$. Conversely, if a high- α portfolio is used between times $t-1$ and t , $\hat{r}_{(t,\alpha)}$ has greater uncertainty and so too does $RF(t)$.

3.4 The Ruin Paths of Two Retirees

Consider two retirees with common $\alpha=0.5$ portfolios. Rebalancing is performed during retirement and the unit of time is years, with $T_D=30$. Both begin time $t=0$ with initial account balances of $\$A=\$100k$ after removing funds to cover first year spending. The initial withdrawal rate for each is $W_R=0.04$ (=4%), adjusted annually for inflation. Both pay an expense ratio of $E_R=0.005$ (=0.5%) to their financial institution. To avoid ruin, the retirees must have a non-zero account balance after making the withdrawal at time $T_D=30$. Table 1 details the experiences of these retirees. Retiree 1 experiences the event Ruin(20) when insufficient funds exist to make the withdrawal at time $t=20$, whereas Retiree 2 does not experience a ruin event. The ruin factor appears in column 1 and is calculated at the start of each year. It decreases when the inflation/expense-adjusted rate of return exceeds it, and increases otherwise.

Retiree 1 begins with a favorable sequence of returns and the ruin factor changes little. By the end of year 3, Retiree 1 has a real account balance larger than the initial balance of $\$100k$. Between years 3 and 17, however, the market underperforms and Retiree 1 begins year 17 with a ruin factor $RF(17)=0.369$. To improve their position Retiree 1 must realize an inflation/expense-adjusted rate of return in excess of $RF(17)$ (that is, $(1+r_{(18,0.5)})(1-.005) > 1.369$). Such a real rate

of return (approximately 37%) is unlikely in one year, and does not happen. As shown the financial state of Retiree 1 declines quickly and $RF(t)$ begins an exponential climb ending with the event, $Ruin(20)$. Lastly, it was noted above that the real account balance at any time t prior to ruin is $(\$A)*RF(0)/RF(t)$, which can be confirmed using Table 1, and unrounded $RF(t)$ values.

Table 1. Experience of Two Retirees with Common $\alpha=0.5$ Portfolios

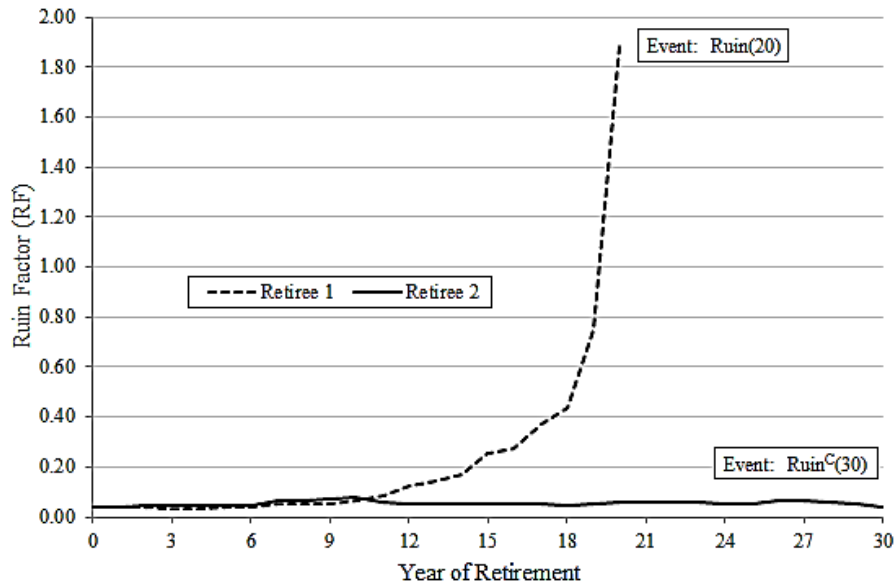
Retiree 1										
Year	Ruin Factor $RF(t)$	Start Year Total Balance	Total Return (1 + Rt)	Inflation (1 + It)	Inflation-Adjusted Return (1 + rt)	Expense Ratio (1 - ER)	Inflation-Adjusted EoY Withdrawal Rate	End Year Total Balance	End Year Real Balance	Terminal Ruin Factor
0	0.040	\$100,000	1.044	1.0250	1.019	0.995	0.0410	\$99,815	\$97,380	
1	0.041	\$99,815	1.190	1.0245	1.161	0.995	0.0420	\$113,961	\$108,524	
2	0.037	\$113,961	1.155	1.0239	1.128	0.995	0.0430	\$126,637	\$117,775	
3	0.034	\$126,637	1.040	1.0237	1.016	0.995	0.0440	\$126,699	\$115,106	
⋮										
17	0.369	\$15,995	1.249	1.0225	1.222	0.995	0.0604	\$13,843	\$9,167	
18	0.436	\$13,843	1.045	1.0225	1.022	0.995	0.0618	\$8,214	\$5,319	
19	0.752	\$8,214	1.180	1.0217	1.155	0.995	0.0631	\$3,333	\$2,113	
20	1.893	\$3,333	0.835	1.0216	0.818	0.995	0.0645	Ruin(20)	Ruin(20)	-1.753
21										
⋮										
⋮										
30										

Retiree 2										
Year	Ruin Factor $RF(t)$	Start Year Total Balance	Total Return (1 + Rt)	Inflation (1 + It)	Inflation-Adjusted Return (1 + rt)	Expense Ratio (1 - ER)	Inflation-Adjusted EoY Withdrawal Rate	End Year Total Balance	End Year Real Balance	Terminal Ruin Factor
0	0.040	\$100,000	1.048	1.0250	1.022	1.000	0.0410	\$100,657	\$98,202	
1	0.041	\$100,657	0.912	1.0261	0.889	0.995	0.0421	\$87,165	\$82,874	
2	0.048	\$87,165	1.098	1.0255	1.071	0.995	0.0431	\$90,912	\$84,289	
3	0.047	\$90,912	1.179	1.0248	1.151	0.995	0.0442	\$102,243	\$92,501	
⋮										
⋮										
26	0.063	\$118,437	1.013	1.0233	0.990	0.995	0.0758	\$111,753	\$58,943	
27	0.068	\$111,753	1.302	1.0237	1.272	0.995	0.0776	\$137,023	\$70,595	
28	0.057	\$137,023	1.214	1.0250	1.184	0.995	0.0796	\$157,544	\$79,187	
29	0.051	\$157,544	1.215	1.0240	1.187	0.995	0.0815	\$182,351	\$89,504	
30	0.045	\$182,351	1.138	1.0232	1.112	0.995	0.0834	\$198,143	\$95,049	0.042

Retiree 2 experiences better returns and starts their 31st year of retirement (time $t=30$) with a real account balance slightly below the initial time $t=0$ balance of \$100k. Retiree 2 has avoided ruin, making 30 annual withdrawals for spending. The ruin paths of both retirees are plotted in Figure 1 (lines are smoothed). At year 10, Retiree 1 is in a better financial position than Retiree 2. (See Appendix B for full details.) A question to ask is whether or not something

could have been done on behalf of Retiree 1 to prevent the rapid decline of their financial state after year 10. Was an $\alpha=0.5$ portfolio too risky 10 years into retirement, or not risky enough?

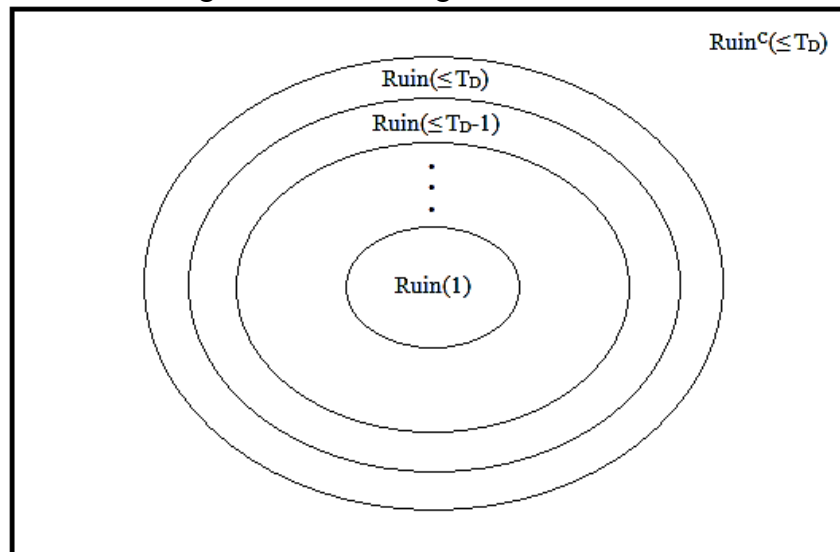
Figure 1. The Ruin Paths of Retirees 1 and 2



3.5 The Event of Financial Ruin

The event of financial ruin at time t was defined previously as $\text{Ruin}(t)$. Let $\text{Ruin}(\leq t)$ represent the event of ruin up to and including time t , $t=1, 2, \dots, T_D$. Using this definition a Venn diagram can be constructed as shown in Figure 2.

Figure 2. Venn Diagram of Ruin Event



It immediately follows that,

$$\text{Ruin}(t) \equiv \text{Ruin}(\leq t) \cap \text{Ruin}^C(\leq t-1), \quad (7)$$

which represents a single ring of the oval in Figure 2. Further,

$$\text{Ruin} \equiv \text{Ruin}(1) \cup \text{Ruin}(2) \cup \dots \cup \text{Ruin}(T_D) \quad (8a)$$

$$\equiv \text{Ruin}(\leq T_D), \quad (8b)$$

$$\rightarrow P(\text{Ruin}) = P(\text{Ruin}(\leq T_D)). \quad (8c)$$

The event of ruin can also be expressed in terms of conditional probabilities. From Figure 2,

$$\begin{aligned} \text{Ruin} &\equiv \text{Ruin}(1) \cup (\text{Ruin}(\leq 2) \cap \text{Ruin}^C(1)) \cup \dots \cup \\ &\quad (\text{Ruin}(\leq T_D-1) \cap \text{Ruin}^C(\leq T_D-2)) \cup \\ &\quad (\text{Ruin}(\leq T_D) \cap \text{Ruin}^C(\leq T_D-1)) \end{aligned} \quad (9a)$$

$$\begin{aligned} \rightarrow P(\text{Ruin}) &= P(\text{Ruin}(1)) + P(\text{Ruin}^C(1)) * P(\text{Ruin}(\leq 2) | \text{Ruin}^C(1)) + \dots + \\ &\quad P(\text{Ruin}^C(\leq T_D-2)) * P(\text{Ruin}(\leq T_D-1) | \text{Ruin}^C(\leq T_D-2)) + \\ &\quad P(\text{Ruin}^C(\leq T_D-1)) * P(\text{Ruin}(\leq T_D) | \text{Ruin}^C(\leq T_D-1)) \end{aligned} \quad (9b)$$

where the last expression holds since $P(A \cap B) = P(B) * P(A | B)$, for any events A and B. Each of the above expresses $P(\text{Ruin})$ directly, but it can also be expressed indirectly:

$$P(\text{Ruin}) = 1 - P(\text{Ruin}^C(\leq T_D)) \quad (10a)$$

$$= 1 - P(\text{Ruin}^C(1) \cap \text{Ruin}^C(2) \cap \dots \cap \text{Ruin}^C(T_D)) \quad (10b)$$

$$\begin{aligned} &= 1 - P(\text{Ruin}^C(1)) * P(\text{Ruin}^C(2) | \text{Ruin}^C(1)) * \dots \\ &\quad * P(\text{Ruin}^C(T_D) | \text{Ruin}^C(1) \cap \text{Ruin}^C(2) \cap \dots \cap \text{Ruin}^C(T_D-1)) \end{aligned} \quad (10c)$$

where the last expression holds since,

$$P(A \cap B \cap C) = P(A \cap B) * P(C | A \cap B) \quad (11a)$$

$$= P(A) * P(B | A) * P(C | A \cap B), \quad (11b)$$

for any events A, B, and C. The expressions in (9) and (10) will be used in upcoming sections.

The objective for the retiree is to minimize $P(\text{Ruin})=P(\text{Ruin}(\leq T_D))$, given W_R . As noted, the debate regarding the true value of $P(\text{Ruin})$ is unsettled. Further, the preferred glide-path is subject to much disagreement. It would be helpful for the retiree to know the exact optimal (minimum) $P(\text{Ruin})$ for each possible W_R at every time t , along with the α required to achieve this optimum. The retiree would benefit from a roadmap that informs them how their $P(\text{Ruin})$ is changing over time, and the actions needed to maintain optimality. Ultimately, such a roadmap should be available for the retiree at time $t=0$, guiding them through retirement decumulation.

3.6 An Initial Solution Attempt

In (9b), $P(\text{Ruin})$ is expressed as a sum of conditional probabilities which is analogous to summing the probabilities of the mutually exclusive rings from the Venn diagram in Figure 2. Since the objective is to minimize $P(\text{Ruin})$ over all α at each time t , a natural first attempt would take the minimum of both sides in (9b). Namely,

$$\begin{aligned} \text{Min}_{(\alpha)}\{P(\text{Ruin})\} &= \text{Min}_{(\alpha)}\{P(\text{Ruin}(1)) + P(\text{Ruin}^C(1))*P(\text{Ruin}(\leq 2) | \text{Ruin}^C(1)) + \\ &+ \dots + P(\text{Ruin}^C(\leq T_D-2))*P(\text{Ruin}(\leq T_D-1) | \text{Ruin}^C(\leq T_D-2)) + \\ &P(\text{Ruin}^C(\leq T_D-1))*P(\text{Ruin}(\leq T_D) | \text{Ruin}^C(\leq T_D-1))\}, \end{aligned} \quad (12)$$

Since each RHS probability is a value between 0 and 1 the minimum is distributed as:

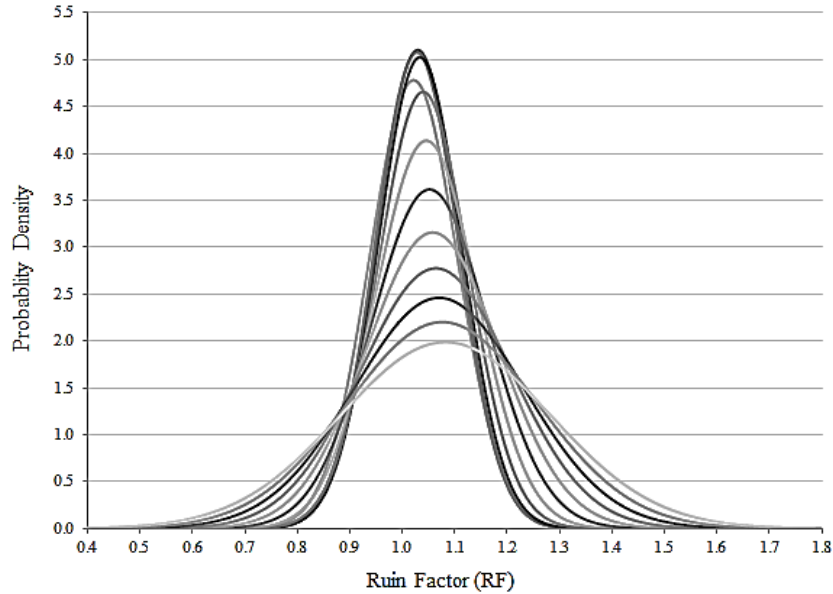
$$\begin{aligned} \text{Min}_{(\alpha)}\{P(\text{Ruin})\} &\stackrel{?}{=} \text{Min}_{(\alpha)}\{P(\text{Ruin}(1))\} + \\ &\text{Min}_{(\alpha)}\{P(\text{Ruin}^C(1))*P(\text{Ruin}(\leq 2) | \text{Ruin}^C(1))\} + \\ &\text{Min}_{(\alpha)}\{P(\text{Ruin}^C(\leq T_D-2))*P(\text{Ruin}(\leq T_D-1) | \text{Ruin}^C(\leq T_D-2))\} \quad (13) \\ &+ \dots + \text{Min}_{(\alpha)}\{P(\text{Ruin}^C(\leq T_D-1))*P(\text{Ruin}(\leq T_D) | \text{Ruin}^C(\leq T_D-1))\}, \end{aligned}$$

where $\stackrel{?}{=}$ refers to a hypothesis requiring verification. The RHS probabilities in (13) condition on ruin not being experienced prior to each time point, thus if proceeding sequentially in time, $P(\text{Ruin}^C(t))=1$ if ruin has been avoided at time t , resulting in:

$$\begin{aligned} \text{Min}_{\alpha}\{P(\text{Ruin})\} &\stackrel{?}{=} \text{Min}_{(\alpha, t=0)}\{P(\text{Ruin}(1))\} + \text{Min}_{(\alpha, t=1)}\{P(\text{Ruin}(2))\} + \dots + \\ &\quad \text{Min}_{(\alpha, t=T_D-2)}\{P(\text{Ruin}(T_D-1))\} + \text{Min}_{(\alpha, t=T_D-1)}\{P(\text{Ruin}(T_D))\}, \end{aligned} \quad (14)$$

where the notation $(\alpha, t=x)$ indicates that the minimization is over all α at time x . This leads to the following strategy: At time $t=0$, use the α that minimizes $P(\text{Ruin}(1))$. If $\text{Ruin}^C(1)$ occurs then rebalance the portfolio at time $t=1$ in a manner that minimizes $P(\text{Ruin}(2))$. The optimal asset allocation α at each time point can be determined by inspecting the tail behavior of the $\hat{f}_{(t,\alpha)}$ PDFs for a collection of α where $\text{Ruin}(t) \leftrightarrow \hat{f}_{(t,\alpha)} \leq \text{RF}(t-1)$.

Figure 3. Generic Probability Densities of $\hat{f}_{(t,\alpha)}$ for Various Asset Allocations (α)



Generic $\hat{f}_{(t,\alpha)}$ PDFs for various α are shown in Figure 3, where the density with low peak and fat tail may represent an $\alpha=1$ portfolio, and the density with high peak and smallest variance may, for example, represent the minimum volatility portfolio described by Markowitz.⁴ Given $\text{Ruin}^C(t-1)$, the asset allocation for time t can be determined by choosing the PDF with smallest tail area to the left of $\text{RF}(t-1)$. Using this α allows the retiree to act optimally with respect to the next time point, having an asset allocation that minimizes $P(\text{Ruin}(t))$.

⁴ See, Markowitz, H. (1952), PORTFOLIO SELECTION. *The Journal of Finance*, 7: 77–91. doi: 10.1111/j.1540-6261.1952.tb01525.x.

This approach, while simple, is flawed and would likely exhibit an unacceptable failure rate compared to other strategies. The reason is obvious, namely, that it acts optimally on a local basis and ignores the long-term impact of each short-term decision. The outcome of such a strategy is predictable. At early time points ($t=0, 1, 2, 3, \dots$) it would place the retiree into a low volatility portfolio attempting to minimize $P(\text{Ruin})$ before the next withdrawal. The inflation/-expense-adjusted returns with such a portfolio will struggle to outperform the ruin factor, causing it to increase over time. Eventually, the ruin factor may spike and the strategy would respond by shifting the retiree into stocks reflecting desperation. While suboptimal, the strategy will still outperform an $\alpha=0$ portfolio, since it eventually transfers into stocks which could delay ruin for a period of time, see (15) below.

$$\begin{aligned} \text{Min}_{\alpha}\{P(\text{Ruin})\} \neq & \text{Min}_{(\alpha, t=0)}\{P(\text{Ruin}(1))\} + \leftarrow \\ & \text{Min}_{(\alpha, t=1)}\{P(\text{Ruin}(2))\} + \dots + \\ & \text{Min}_{(\alpha, t=T_D-2)}\{P(\text{Ruin}(T_D-1))\} \\ & \text{Min}_{(\alpha, t=T_D-1)}\{P(\text{Ruin}(T_D))\}, \leftarrow \end{aligned} \quad (15)$$

Minimizing $P(\text{Ruin})$ by using sequential optimization fails because the α at time $t=0$ has an impact on the probability of ruin at later time points.

3.7 A Better Approach

The attempt to minimize $P(\text{Ruin})$ in Section 3.6 used the expressions in (9) and was unsuccessful because a collection of local optimums does not necessarily aggregate to a global optimum. The key to solving this problem successfully is not to attack it directly but indirectly, namely using backward induction. The induction process begins at time $t=T_D$ and each step employs the expressions in (10) and unfolds similarly. Namely, it is assumed that the retiree arrives at that time point and successfully makes their withdrawal. Having absorbed the previous time point's returns, they calculate the ruin factor which must be > 0 since ruin has been avoided. At $t=0$ the process ends and reveals the exact optimal probability of ruin for all withdrawal rates,

along with the dynamic asset allocation strategy required to maintain optimality throughout decumulation. Each step in the process is presented separately after some additional notation is defined and assumptions made (see also Appendix C). Let,

$$\begin{aligned} V(t, RF(t)) &= \text{Optimal (minimum) probability of ruin after time } t \text{ given } RF(t) > 0 \\ \alpha(t, RF(t)) &= \text{Value of } \alpha \text{ required to achieve } V(t, RF(t)). \end{aligned}$$

When the context $(t, RF(t))$ is unambiguous, the abbreviation $\tilde{\alpha} = \alpha(t, RF(t))$ is used. Note that in terms of ruin events, the value function $V(t, RF(t))$ is defined as,

$$V(t, RF(t)) = \text{Min}_{(\alpha)} \left\{ P(\text{Ruin}(t+1) \cup \text{Ruin}(t+2) \cup \dots \cup \text{Ruin}(T_D)) \mid \text{Ruin}^C(\leq t) \right\} \quad (16a)$$

$$\rightarrow V(t, RF(t)) = \text{Min}_{(\alpha)} \left\{ P(\text{Ruin}(t+1) \cup \text{Ruin}(t+2) \cup \dots \cup \text{Ruin}(T_D)) \right\} \quad (16b)$$

$$\rightarrow V(t, RF(t)) = \text{Min}_{(\alpha)} \left\{ 1 - P(\text{Ruin}^C(t+1) \cap \text{Ruin}^C(t+2) \cap \dots \cap \text{Ruin}^C(T_D)) \right\}, \quad (16c)$$

where the conditioning is dropped in (16b) since $\text{Ruin}^C(\leq t)$ is assumed to have occurred (during induction), thus conditioning on it has no effect. It has been established that,

$$\text{Ruin}^C(t) \text{ occurs} \leftrightarrow \hat{r}_{(t,\alpha)} > RF(t-1). \quad (17)$$

Let $f(\hat{r}_{(t,\alpha)})$ represent the PDF of $\hat{r}_{(t,\alpha)}$, then the conditional R.V. $(\hat{r}_{(t,\alpha)} \mid \hat{r}_{(t,\alpha)} > RF(t-1))$ has PDF:

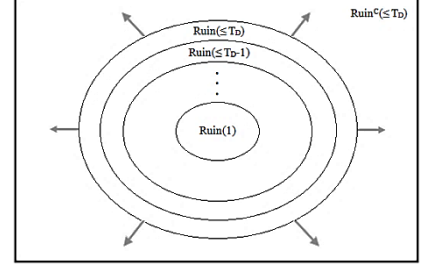
$$\hat{r}_{(t,\alpha)}^+ = (\hat{r}_{(t,\alpha)} \mid \hat{r}_{(t,\alpha)} > RF(t-1)) = \begin{cases} \frac{f(\hat{r}_{(t,\alpha)})}{\int_{RF(t-1)}^{\infty} f(\hat{r}_{(t,\alpha)}) d\hat{r}_{(t,\alpha)}} & , \text{ for } \hat{r}_{(t,\alpha)} > RF(t-1) \\ 0 & , \text{ O.W.} \end{cases} \quad (18)$$

In this analysis, returns from diversified α -based portfolios at different time points are assumed to be independent, which is consistent with the theory that markets are efficient. That is, any predictive capacity contained in past patterns of returns is instantly accounted for thereby removing its effect.⁵

⁵ See for example, Brigham, E.F. & Ehrhardt, M.C. Financial Management Theory and Practice 12th Edition. South-Western CENGAGE Learning (2008) for a discussion of the various forms of market efficiency. Individual analyses for evidence of serial correlation within stock and bond returns is conducted in Appendix F of this report.

3.7.1 Induction at Time $t=T_D$

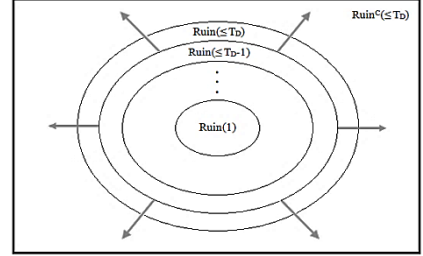
Assume the retiree arrives at time $t=T_D$ and makes their last withdrawal. The restricted sample space $S=\{\text{Ruin}^C(\leq T_D)\}$ includes a single event, shown at right. The retiree need not compute $\text{RF}(T_D)$ as there are no more withdrawals, however,



$\text{RF}(T_D) > 0$ (if computed) since $\text{Ruin}^C(\leq T_D)$ has occurred. At $t=T_D$, $P(\text{Ruin})=0$, and a boundary condition (B.C.) for the value function is $V(T_D, \text{RF}(T_D)) = 0, \forall \text{RF}(T_D) > 0$.

3.7.2 Induction at Time $t=T_D - 1$

Assume the retiree arrives at time $t=T_D-1$, makes their 2nd last withdrawal and has one remaining. The ruin factor $\text{RF}(T_D-1) (> 0)$ is calculated based on the portfolio's return just observed, $\hat{r}_{(T_D-1, \alpha)}$. The retiree now faces the restricted sample



space $S=\{\text{Ruin}(T_D), \text{Ruin}^C(\leq T_D)\}$, as shown, and seeks the α that minimizes $P(\text{Ruin}(T_D))$. This straight forward decision is made using the framework presented in Section 3.6. Namely, the retiree/advisor compares the tail probabilities for various asset allocations and selects the one which minimizes $P(\text{Ruin}(T_D))$. This optimization can be expressed as,

$$V(T_D-1, \text{RF}(T_D-1)) = \text{Min}_{(0 \leq \alpha \leq 1)} \left\{ P(\text{Ruin}(T_D)) \right\} \quad (19a)$$

$$\rightarrow V(T_D-1, \text{RF}(T_D-1)) = \text{Min}_{(0 \leq \alpha \leq 1)} \left\{ 1 - P(\text{Ruin}^C(T_D)) \right\} \quad (19b)$$

$$\rightarrow V(T_D-1, \text{RF}(T_D-1)) = \text{Min}_{(0 \leq \alpha \leq 1)} \left\{ 1 - P(\hat{r}_{(T_D, \alpha)} > \text{RF}(T_D-1)) \right\} \quad (19c)$$

$$\rightarrow V(T_D-1, \text{RF}(T_D-1)) = \text{Min}_{(0 \leq \alpha \leq 1)} \left\{ 1 - (1 - F_{\hat{r}_{(T_D, \alpha)}}(\text{RF}(T_D-1))) \right\}, \quad (19d)$$

given the known ruin factor $\text{RF}(T_D-1)$. Here, $F_{\hat{r}_{(T_D, \alpha)}}(\cdot)$ denotes the known/estimated CDF of $\hat{r}_{(T_D, \alpha)}$. Note that $V(T_D-1, \text{RF}(T_D-1))$ can be equivalently expressed as:

$$V(T_D-1, RF(T_D-1)) = \text{Min}_{(0 \leq \alpha \leq 1)} \left\{ \underbrace{1 - (1 - F_{\hat{r}(T_D, \alpha)}(RF(T_D-1)))}_{\text{Probability of no ruin at time } t=T_D} * \underbrace{(1 - E_{\hat{r}(T_D, \alpha)}^+[V(T_D, RF(T_D))])}_{\text{Expected prob. of no ruin after time } t=T_D, \text{ given } \text{Ruin}^C(T_D)} \right\}, \quad (20)$$

with optimal $\alpha = \alpha(T_D-1, RF(T_D-1))$.⁶ (See Appendix C for induction steps $t=T_D-2$ and $t=T_D-3$.)

3.7.3 Value Function at Time t

Assume the retiree arrives a time t , makes their withdrawal and updates the ruin factor $RF(t)$ (> 0) based on the portfolio return just observed, $\hat{r}_{(t, \alpha)}$. The retiree is facing the restricted sample space $S = \{\text{Ruin}(t+1), \text{Ruin}(t+2), \dots, \text{Ruin}(T_D), \text{Ruin}^C(\leq T_D)\}$ and seeks α to minimize:

$$P(\text{Ruin}) = P(\text{Ruin}(t+1) \cup \text{Ruin}(t+2) \cup \dots \cup \text{Ruin}(T_D)) \quad (21a)$$

$$= 1 - P(\text{Ruin}^C(t+1) \cap \text{Ruin}^C(t+2) \cap \dots \cap \text{Ruin}^C(T_D)) \quad (21b)$$

$$= 1 - P(\text{Ruin}^C(t+1)) * P(\text{Ruin}^C(t+2) \cap \dots \cap \text{Ruin}^C(T_D) \mid \text{Ruin}^C(t+1)). \quad (21c)$$

By continuing the induction process (see Appendix C), it follows that minimizing $P(\text{Ruin})$ over $(0 \leq \alpha \leq 1)$ results in the following value function at time t , with ruin factor $RF(t)$:

$$V(t, RF(t)) = \left\{ \begin{array}{l} \text{Min}_{(0 \leq \alpha \leq 1)} \left\{ \begin{array}{l} 1 - (1 - F_{\hat{r}(t+1, \alpha)}(RF(t)))^* \\ (1 - E_{\hat{r}(t+1, \alpha)}^+[V(t+1, \frac{RF(t)}{\hat{r}_{(t+1, \alpha)} - RF(t)})]) \end{array} \right\} \\ \text{For } 0 \leq t \leq T_D-1, \\ RF(t) > 0, \\ \text{and B.C., } V(T_D, RF(T_D)) = 0. \end{array} \right. \quad (22)$$

The choice of α at time t becomes a balancing act between minimizing $P(\text{Ruin})$ at time $t+1$, and after time $t+1$.

Optimality is achieved at $\alpha = \alpha(t, RF(t))$ and the expectation is over the conditional random variable $\hat{r}_{(t+1, \alpha)}^+$. Choosing α is now a balancing act between minimizing $P(\text{Ruin})$ at the next time

⁶ The right-most term is added for notational convenience. Recall that $V(T_D, RF(T_D)) = 0 \forall RF(T_D) > 0$ and it is therefore the expected value of zero. Further, $\hat{r}_{(T_D, \alpha)}^+ = (\hat{r}_{(T_D, \alpha)} \mid \hat{r}_{(T_D, \alpha)} > RF(T_D-1))$ as derived in Section 3.7.

point, and after the next time point. The local optimization problem encountered with the solution attempt of Section 3.6 has been addressed. Selecting a low volatility portfolio (small α) early in retirement to avoid ruin at the next time point now comes with a price because that portfolio results in a higher value for the next ruin factor (in expectation, relative to a high α portfolio). A larger ruin factor at the next time point increases the probability of ruin after the next time point and a balance between these competing forces is found at $\alpha(t, RF(t))$.

The value function $V(t, RF(t))$ represents a dynamic program (DP)⁷ with $V(0, RF(0))$ being the optimal (minimum) $P(\text{Ruin})$ at any future time point given the withdrawal rate $W_R=RF(0)$, at time $t=0$. Time t is considered the DP's *stage*, and $RF(t)$ the *state*. Since the solution finds the optimal $V(t, RF(t)) \forall (t, RF(t))$ it is also a roadmap for maintaining optimality over time using a dynamic asset allocation strategy. Note that $V(0, RF(0))$ is exact, not simulated, and optimal meaning it is not possible to achieve a lower probability of ruin under any alternate strategy using an inflation-adjusted $W_R=RF(0)$, and the same stocks and bonds.

The challenge with this DP is that it must be solved for all $RF(t) > 0$, at each t . Further, in practice the time of last withdrawal, T_D , is random. The nature of $V(t, RF(t))$ will depend on the distributional assumption made for inflation/expense-adjusted returns. If the expression being minimized has a known functional form then finding α for each t could be handled using calculus, with $\alpha=\alpha(t, RF(t))$ being the minimization's critical value. Closed form expressions for $V(t, RF(t))$ are unlikely, however. Accounting for the randomness in T_D is less of a concern, since the DP could be solved for various T_D , then incorporated with the probability of living T_D years in retirement (see Section 4.6). To overcome these challenges the α and $RF(t)$ dimensions will be discretized for $0 \leq \alpha \leq 1$ and $RF(t) > 0$. Further, a known T_D will be assumed initially,

⁷ See for example, Bradley, Stephen P., Arnoldo C. Hax, and Thomas L. Magnanti. *Applied Mathematical Programming*. Reading, MA: Addison-Wesley Publishing Company, 1977.

then relaxed in Section 4.6. The result is a numerical approximation to the exact optimal solution, not based on simulation. Further, since the level of granularization along the α and $RF(t)$ dimensions is determined by the user, an approximation to any desired degree of accuracy is within reach, limited only by the user's available processing power.

3.7.4 Discretizing Along α and $RF(t)$ Dimensions

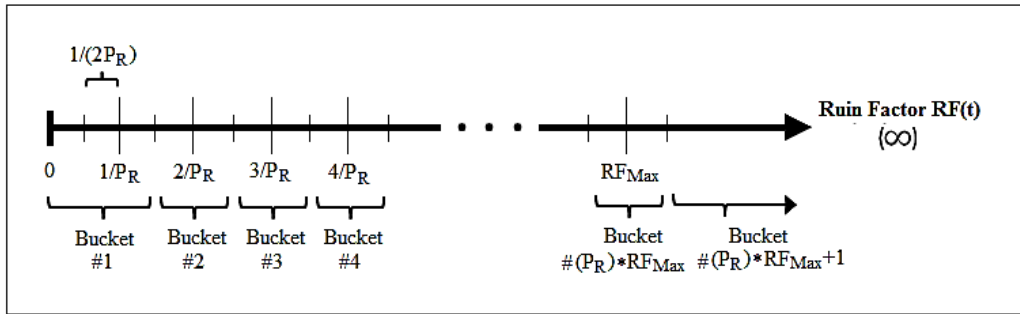
Let $\alpha_{\{\cdot\}}$ represent the set of values resulting from the discretization precision P_α along the α dimension and let $R_{\{\cdot\}}$ represent the set of values determined by the discretization precision P_R along the ruin factor dimension with maximum ruin factor RF_{Max} . That is,

$$\alpha_{\{\cdot\}} = \{0/P_\alpha, 1/P_\alpha, 2/P_\alpha, \dots, 1-(1/P_\alpha), 1\} \quad (23)$$

$$R_{\{\cdot\}} = \{1/P_R, 2/P_R, 3/P_R, \dots, RF_{Max}-(1/P_R), RF_{Max}\}. \quad (24)$$

Along the ruin factor dimension each value in the set $R_{\{\cdot\}}$ will serve as the midpoint of a bucket constructed around it. Every $RF(t) > 0$ will then map to one of the $(P_R)*RF_{Max}+1$ buckets, with the last bucket containing all ruin factors $> RF_{Max}+1/(2P_R)$. See Figure 4 below.

Figure 4. Discretization of Ruin Factor ($RF(t) > 0$) Dimension



The discrete DP implementation will use the bucket midpoints when a single $RF(t)$ value is required. Since the buckets are contiguous (no gaps), the solution will provide coverage for all $RF(t) > 0$. The value function is derived for $V(t, i/P_R)$, $i=1, 2, 3, \dots, (P_R)*RF_{Max}$, with $V(t, x)=1$ for $x > RF_{Max} + 1/(2P_R)$. As $P_R \rightarrow \infty$ the buckets collapse to the midpoint, and the discretization along the $RF(t)$ dimension approaches the continuous solution. The conditional expectation in

$V(t, RF(t))$ will be handled by discretizing the probability distribution of $\hat{r}_{(t+1, \alpha)}^+$ in a manner that coincides with the bucket boundaries. This is accomplished by recalling that,

$$RF(t+1) = \frac{RF(t)}{\hat{r}_{(t+1, \alpha)} - RF(t)}. \quad (25)$$

At time t the probability that $RF(t+1)$ falls in bucket i , $i = 1, 2, \dots, (P_R)*RF_{Max}$, $(P_R)*RF_{Max}+1$ given that $RF(t+1) > 0$ is defined by the probability mass function (PMF) (see Appendix D):

$$P(RF(t+1) \text{ in Bucket } i \mid \hat{r}_{(t+1, \alpha)} > RF(t)) =$$

$$\left\{ \begin{array}{ll} \frac{1 - F\hat{r}_{(t+1, \alpha)}(RF(t) * (1 + \frac{1}{(1.5) * (\frac{1}{P_R})}))}{1 - F\hat{r}_{(t+1, \alpha)}(RF(t))}, & i = 1 \\ \frac{F\hat{r}_{(t+1, \alpha)}(RF(t) * ((1 + \frac{1}{(\frac{1}{P_R}) * (i - \frac{1}{2})})) - F\hat{r}_{(t+1, \alpha)}(RF(t) * ((1 + \frac{1}{(\frac{1}{P_R}) * (i + \frac{1}{2})}))}{1 - F\hat{r}_{(t+1, \alpha)}(RF(t))}, & i = 2, 3, \dots, (P_R)*RF_{Max} \\ \frac{F\hat{r}_{(t+1, \alpha)}(RF(t) * (1 + \frac{1}{RF_{Max} + 1/(2P_R)})) - F\hat{r}_{(t+1, \alpha)}(RF(t))}{1 - F\hat{r}_{(t+1, \alpha)}(RF(t))}, & i = (P_R)*RF_{Max} + 1 \end{array} \right. \quad (26)$$

The expectation over $\hat{r}_{(t+1, \alpha)}^+$ in (22) is now a sum over the discrete values of $V(t+1, RF(t+1))$ multiplied by the probabilities in (26), which depend on $\alpha = \alpha(t, RF(t))$. Letting $V_d(t, RF(t))$ and $\alpha_d(t, RF(t))$ denote the discretized versions of V and $\tilde{\alpha}$, respectively, then:

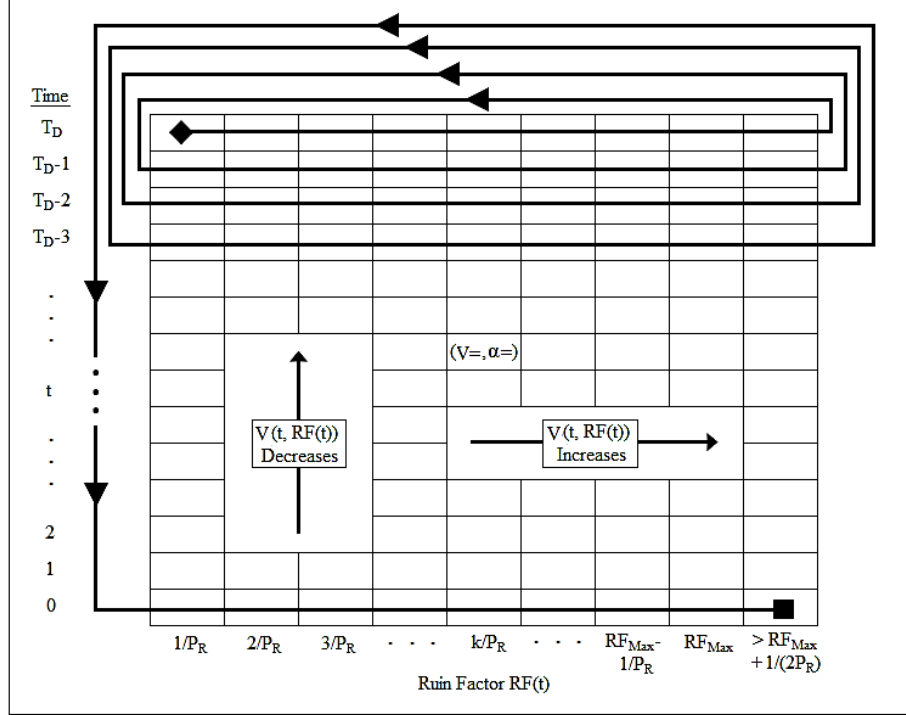
$$V_d(t, i/P_R) = \left\{ \begin{array}{l} \text{Min}_{(\alpha \in \alpha_{\{ \cdot \}})} \left\{ \begin{array}{l} 1 - (1 - F\hat{r}_{(t+1, \alpha)}(i/P_R))^* \\ (1 - \sum_{j=1}^{(P_R)*RF_{Max}+1} \{P(j) * [V_d(t+1, j/P_R)]\}) \end{array} \right\} \\ \text{For } 0 \leq t \leq T_D-1, i=1, 2, 3, \dots, (P_R)*RF_{Max}, \\ \text{w/B.C.'s} \quad V_d(T_D, i/P_R) = 0, \\ V_d(t, RF(t)) = 1 \text{ for } RF(t) > RF_{Max} + 1/(2P_R). \end{array} \right. \quad (27)$$

Here, $P(j)$ denotes $P(RF(t+1) \text{ in Bucket } \# j \mid \hat{r}_{(t+1, \alpha)} > RF(t))$ from (26) with optimal asset allocation $\alpha_d(t, RF(t))$. This DP can be solved using any programming language with the user's

choice of P_α and P_R . The code operates on a grid over the t and $RF(t)$ dimensions, see Figure 5.

A basic way to code this DP is to start at the diamond, follow the arrows, and end at the square.

Figure 5. Discretized DP Coded as a Time by Ruin Factor Grid



Values of $V_d(t, RF(t))$ and $\alpha_d(t, RF(t))$ are derived and retained at each cell in the grid. As indicated in Figure 5, $V(t, RF(t))$ from (22) exhibits some structure across the t and $RF(t)$ dimensions. For a fixed $RF(t)$, $V(t, RF(t))$ decreases with t , and for a fixed t , $V(t, RF(t))$ increases with $RF(t)$. Justifications are by indirection. For the former, assume that one unit of time elapses and $RF(t)$ does not change yet $V(t, RF(t))$ increases. As noted earlier, the real account balance at time t can be derived from $RF(t)$ as $(\$A)*RF(0)/RF(t)$. Therefore, the real account balance at time $t+1$ equals the real account balance at time t , but $P(\text{Ruin})$ has increased, which is a contradiction. Similarly, to prove the latter assume that at time t , $RF_2(t) > RF_1(t)$ but $P(\text{Ruin})$ is larger for $RF_1(t)$. This is also a contradiction since $(\$A)*RF(0)/RF_1(t) > (\$A)*RF(0)/RF_2(t)$ and $W_R=RF(0)$ is the same for both portfolios. Further, note that

$$\lim_{P_R \rightarrow \infty} \left[\lim_{P_\alpha \rightarrow \infty} \left[V_d(t, RF(t)) \right] \right] = V(t, RF(t)). \quad (28)$$

That is, the discrete value function approaches the continuous value function as the precision increases to infinity. This may appear obvious for α , since evaluating all values at each time point is necessary for optimality, but it is also true in the $RF(t)$ dimension. (See Appendix E.)

Recall that $V(t, RF(t))$ is optimal in the sense that no alternate strategy exists, or can be constructed, which produces a lower $P(\text{Ruin})$, given W_R and portfolios using the same stocks and bonds. The discrete DP thus represents a numerical approximation to the optimal decumulation strategy. The optimal $V(t, RF(t))$ is a continuous function along 4 dimensions ($t, RF(t), \alpha, V$) and discretizing it is CPU intensive, especially for large P_α and P_R .

4.0 An Implementation

The DP in (27) was coded using the technique described.⁸ Since $P(\text{Ruin})$ is highly sensitive to α for small $RF(t)$, a hybrid strategy was employed for discretizing along the α dimension with $P_\alpha=1,000$ for $RF(t) < 0.20$ and $P_\alpha=100$ for $RF(t) > 0.20$. The $RF(t)$ dimension was discretized using $P_R=1,000$ and $RF_{\text{Max}}=2.75$ resulting in a total of 2,751 $RF(t)$ buckets for each time t . The unit of time was set to years with final withdrawal at $T_D=30$. Further, an expense ratio of $E_R=0.5\%$ was assumed, then relaxed to facilitate comparisons with published research. Under this discretization, the grid shown in Figure 5 has 82,530 cells which are populated for V and α . Each year the retiree/advisor calculates $RF(t)$, rounds it to the nearest $1/P_R$, and consults the grid. The value of V reflects the optimal (minimum) $P(\text{Ruin})$ at any future time point up to $t=T_D$, and α yields the asset allocation necessary for achieving optimality. Note that DPs require an optimal policy be followed at the current, and all future time points reflecting the *principle of optimality*. Before coding the DP, distributional assumptions for stock and bond returns must be made.

⁸ See Appendix H for a full C++ implementation.

4.1 Distributional Assumptions

Historical annual S&P 500 total returns, 10-year Treasury Bond total returns, and the CPI-U from 1928 – 2013 will be used to represent stock returns, bond returns, and the inflation rate, respectively in this analysis. It will be assumed that future stock and bond returns originate from the same underlying probability distributions as past returns, and $N=86$ years of historical returns are considered.⁹ After adjusting historical total returns for inflation it was found that real total stock and bond returns originate as random samples from underlying normal distributions with (μ, σ) of $(0.0825, 0.2007)$ and $(0.0214, 0.0834)$, respectively. Further, a small positive correlation of $\rho=0.04387$ was measured between these returns at the same time point and this will be used throughout. (See Appendix F for a justification of this conclusion.)

Note that the model developed in (22) makes no assumption regarding the underlying distributions of asset class returns. The only requirement is that tail probabilities for convex combinations of correlated stock and bond returns are known or can be simulated/approximated at the bucket boundaries over the set $\alpha_{\{i,j\}}$. The bucket boundaries are fixed and known once P_R has been decided, therefore the tail probabilities can be preprocessed for any distribution and stored in memory, replacing CDF function calls with array references in the code. Further, the CDF call at the bucket midpoint can be replaced by the average of the CDF probabilities at the boundaries without impacting convergence of the discrete DP to its continuous counterpart.

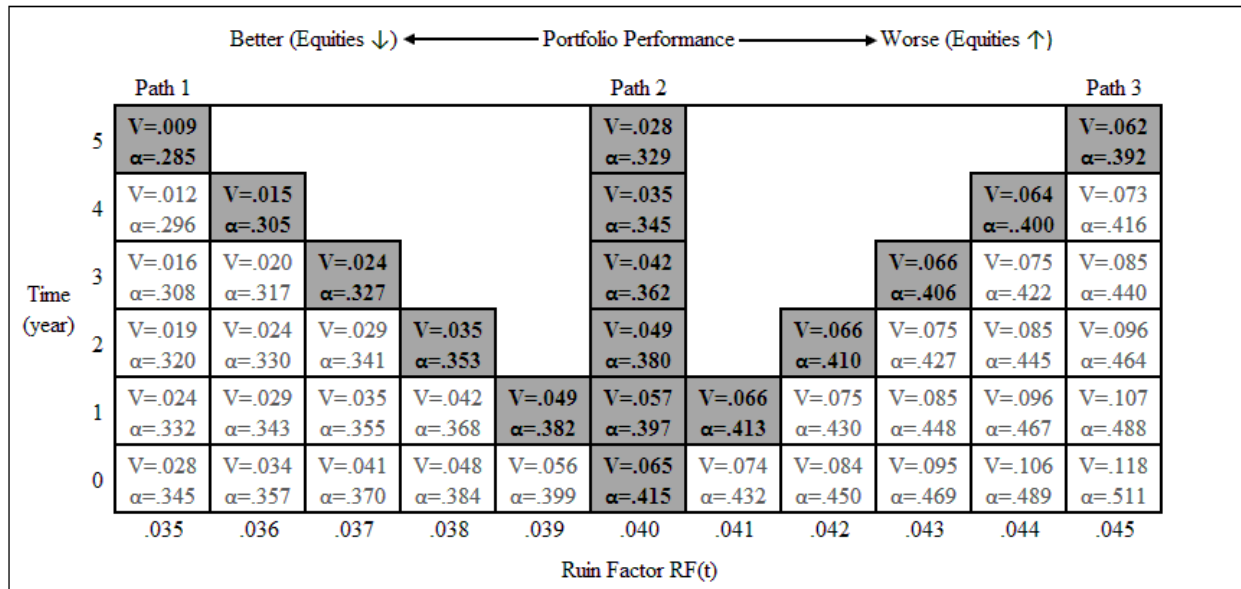
4.2 Discretized DP Results

The discrete DP implementation from Section 4.0 populates V and α for a grid of 82,530 cells and 46 are shown in Figure 6. An expense ratio of $E_R=0.5\%$ was used in this analysis. Here, V =minimum $P(\text{Ruin})$ at any future time point, and α =proportion of the portfolio in stocks.

⁹ The author would like to thank Professor Aswath Damodaran, of NYU for making the financial databases used in this research publicly available. (<http://pages.stern.nyu.edu/~adamodar/>) CPI-U data was taken from The Federal Reserve's Bank of Minneapolis. (http://www.minneapolisfed.org/community_education/teacher/calc/hist1913.cfm)

Since $RF(0)=W_R$, the time $t=0$ row of Figure 6 reveals the optimal $P(\text{Ruin})$ for initial withdrawal rates between 0.035 and 0.045, along with the asset allocation required for optimality during year 1. Noteworthy is that a 4% inflation-adjusted initial withdrawal rate yields an optimal $P(\text{Ruin})$ of 0.065 (success probability=0.935) and starting asset allocation of 41.5% stocks.

Figure 6. Select Discrete Implementation Cells using Expense Ratio $E_R=0.5\%$



Three paths have been shaded within Figure 6 for illustration. If the retiree uses an initial 4% withdrawal rate and experiences favorable market returns the ruin factor decreases over time and the retiree will trace a leftward path perhaps similar to Path 1. If the retiree experiences stable returns that match the withdrawal rate, the ruin factor remains constant and the retiree's path would resemble Path 2. If the retiree faces an unfavorable market the ruin factor increases and their path could resemble the rightward drifting Path 3. As shown in Paths 1 and 2, $P(\text{Ruin})$ decreases with time in a favorable/stable market and the optimal asset allocation shifts out of stocks. Thus Paths 1 and 2 follow the classic glide-path strategy used in target-date funds that extend through the retirement date. Path 3, however, reveals that the optimal equity allocation remains roughly constant or increases in an unfavorable market. This is consistent with findings

by researchers that warn of poor sequences of returns early in retirement. See Fan, Murray, and Pittman (2013), and Pfau and Kitces (2014). Note that a year 1 real (expense-adjusted) return of 2.76% would result in a rounded $RF(1)=0.041$ when $RF(0)=0.040$.

Figure 7. Select Discrete Implementation Cells using No Expense Ratio ($E_R=0.0\%$)

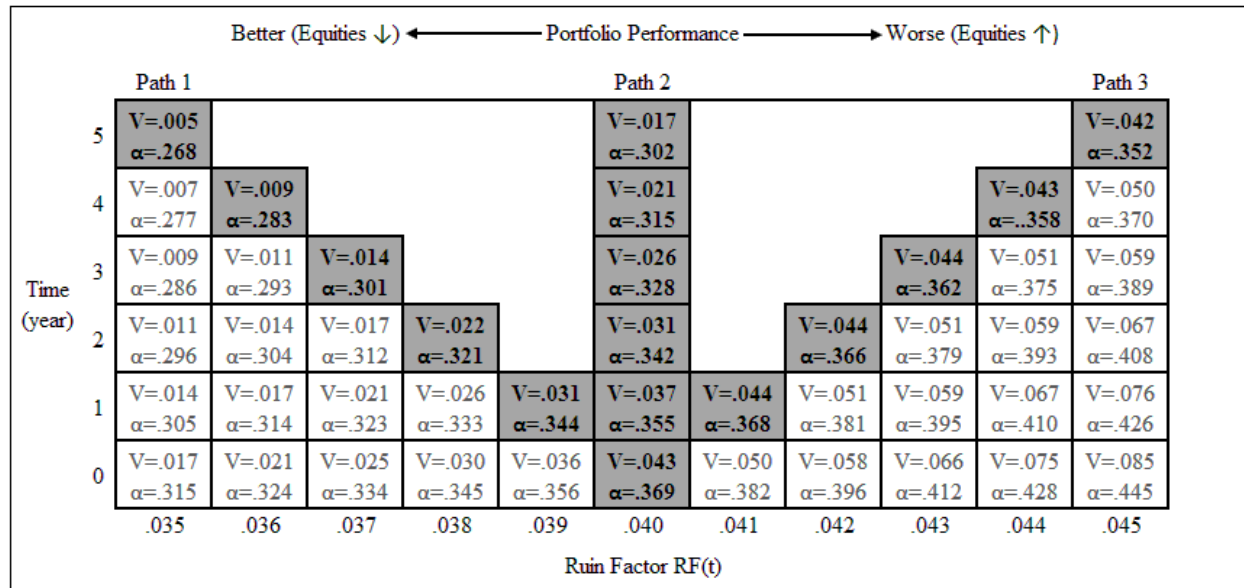


Figure 7 displays the results for $E_R=0.0\%$. The optimal $P(\text{Ruin})$ using a 4% initial withdrawal rate is 0.043 (success probability = 0.957) with starting asset allocation of 36.9% in equities. The same pattern of equities decreasing during favorable market returns, and increasing with unfavorable returns holds true. The 95.7% success rate applies to the retiree who begins retirement and agrees to always act optimally at future time points.

Note that the grids in Figures 6 and 7 apply to all retirees using any initial withdrawal rate. If two retirees begin time $t=0$ with different withdrawal rates and end up with equal ruin factors at time t then their asset allocations, returns, and probability of ruin will be identical for the remainder of retirement. This also means they will chart the exact same course through the grid after the point of intersection, and W_R plays no role in the analysis. The intuition behind this conclusion is that the retiree with higher W_R must experience favorable returns and chart a

leftward path prior to the point of intersection with a retiree using a lower W_R . While charting this leftward path the retiree accumulates enough wealth to support larger withdrawals. No assessment is made here regarding the likelihood of such an intersection occurring, however.

4.3 Numerical Accuracy of the Solution

An RF(t) discretization using $P_R=5,000$ was run for comparison with the solution above. When $P_R=5,000$, 13,751 buckets are created for each time t and 412,530 total cells exist in the Figure 5 grid. An expense ratio of $E_R=0.0\%$ was assumed making this discretization directly comparable with Figure 7. As shown in Table 2, the larger P_R changes $P(\text{Ruin})$ and α modestly, but from a practical sense the results using $P_R=1,000$ are more than adequate when $T_D=30$.

Table 2. Comparison of V and α for $P_R=1,000$ vs. $P_R=5,000$ with $E_R=0.0\%$

Time (t)	RF(t) (= W_R)	$V_d(0, RF(0))$ [$\alpha_d(0, RF(0))$]	
		$P_R=1,000^1$	$P_R=5,000$
0	0.036	0.02051 [0.324]	0.02035 [0.323]
0	0.038	0.03032 [0.345]	0.03014 [0.344]
0	0.040	0.04271 [0.369]	0.04251 [0.368]
0	0.042	0.05770 [0.396]	0.05750 [0.396]
0	0.044	0.07516 [0.428]	0.07495 [0.427]

¹ From Figure 7, row $t=0$.

4.4 Consequences of Suboptimal Strategies

The practical worth of the model in (22) and its implementation from Section 4.2 can be gauged by comparing it with suboptimal strategies. Fixed portfolios having $\alpha \in \{0.00, 0.25, 0.50, 0.75, 1.00\}$ were considered under the same market assumptions set forth in Section 4.1, with $T_D=30$, $E_R=0.0\%$, and $W_R=0.04$. With fixed α , simulation can be used to estimate $P(\text{Ruin})$ or the DP can be solved separately for each of the 5 asset allocations above with $|\alpha_{\{i\}}|=1$. The DP thus takes the minimum over one α at each cell in the grid and $V(0, RF(0)=0.04)$ represents a numerical approximation to the true $P(\text{Ruin})$ under that fixed α strategy. The results of both methods are shown in Table 3 and directly compare with $V(0, RF(0)=0.04)$ in Figure 7. Of the 5

considered, an $\alpha=0.50$ portfolio generated the lowest $P(\text{Ruin})$ having a success rate of 91.5%, compared with 95.7% under the optimal strategy. The consequence of using one of the suboptimal fixed α strategies considered here is thus to accept at least a doubling of $P(\text{Ruin})$.

Table 3. Comparable $P(\text{Ruin})$ using Fixed α Strategies

Fixed α	Simulation ¹	DP Solved for $\alpha_{\{t\}}=\{\alpha\}$ ²
0.00	0.3850	0.3852
0.25	0.1203	0.1209
0.50	0.0852	0.0854
0.75	0.1071	0.1071
1.00	0.1462	0.1461

¹ $N=2.5$ million per simulation

² $P_R=1,000$

4.5 Algorithm Scalability

Let $k=T_D+1$, $M=P_\alpha$, and $N=(RF_{\text{Max}}*P_R)+1$. The discrete DP algorithm in (27) scales at $O(k*M*N^2)$ since each cell in a row of the Figure 5 grid requires an expected value computation that accesses N cells at the next time point, which happens for M asset allocations over all N cells in each of k rows. Therefore, without pruning/parallel processing, the implementation from Section 4.3 has a 25X longer runtime than the implementation from Section 4.2 (k and M did not change, but N is 5X larger). Significant and scalable efficiencies are achievable, however.¹⁰

4.6 Incorporating Random Time of Final Withdrawal (T_D)¹¹

A valid criticism of the implementations in Sections 4.2 and 4.3 is that T_D was fixed. A random T_D can be incorporated by selecting a large value S_{Max} then solving the discrete DP for $T_D=S_{\text{Max}}$. The resulting grid has dimensions $S_{\text{Max}}+1$ by $RF_{\text{Max}}+1$ and contains the solution for all $T_D \leq S_{\text{Max}}$. For example, the rows for $t=S_{\text{Max}}-31$ thru S_{Max} reflect the solution for $T_D=30$, after renumbering the time points. Thus, the grid for $T_D=s$, $s=0, 1, \dots, S_{\text{Max}}$ will be taken as the last $s+1$ rows of the solution for $T_D=S_{\text{Max}}$. Let,

¹⁰ For example, consider the value of $RF(t)$ such that $V(t, RF(t))=1$ over time & the processing of adjacent grid cells.

¹¹ See newly added Section 4.6.1 below for an improved model that incorporates random T_D into the optimization.

$$V_s(t, RF(t)) = V(t, RF(t)) | T_D=s \quad (32a)$$

$$\alpha_s(t, RF(t)) = \alpha(t, RF(t)) | T_D=s \quad (32b)$$

be the known V and α from the solution for $T_D=s$ at time t and ruin factor $RF(t)$, $t=0, 1, \dots, s$. Note that $V(t, RF(t))$ and $\alpha(t, RF(t))$, as defined in (22) are now random variables since they depend on T_D , which is random. Interest is thus in $E[V(t, RF(t))]$ and $E[\alpha(t, RF(t))]$ over all T_D for each $(t, RF(t))$. Let $T_D | T_D \geq k$ be a conditional random variable representing the time of final withdrawal given that the retiree makes the required withdrawal at time $t=k$. There are $S_{Max}+1$ such random variables, one for each $k=0, 1, 2, \dots, S_{Max}$, and the PMFs are built as:¹²

$$P(T_D = k + i | T_D \geq k) = \begin{cases} \frac{P(T_D=k+i)}{\sum_{j=0}^{S_{Max}-k} P(T_D=k+j)} , \text{ for } i = 0, \dots, S_{Max} - k \\ 0 , \text{ O.W.} \end{cases} \quad (33)$$

Finally, a new random T_D grid with $S_{Max}+1$ rows and $RF_{Max}+1$ columns has each cell $(t, RF(t))$ populated with $E[V(t, RF(t))]$ and $E[\alpha(t, RF(t))]$, where using the *law of total expectation*:

$$E[V(t, RF(t))] = E_{T_D \geq t} [E[V(t, RF(t)) | T_D \geq t]] \quad (34a)$$

$$= \sum_{i=0}^{S_{Max}-t} E[V(t, RF(t)) | T_D = t + i] * P(T_D = t+i | T_D \geq t) \quad (34b)$$

$$= \sum_{i=0}^{S_{Max}-t} V_{t+i}(t, RF(t)) * P(T_D = t + i | T_D \geq t) \quad (34c)$$

and, similarly:

$$E[\alpha(t, RF(t))] = \sum_{i=0}^{S_{Max}-t} \alpha_{t+i}(t, RF(t)) * P(T_D = t + i | T_D \geq t). \quad (35)$$

4.6.1 Optimal Model for Random T_D

The technique to incorporate random T_D in Section 4.6 above uses expected values of V and α that were derived from fixed T_D solutions. There is no guarantee that $E[\alpha]$ will generate ruin probabilities of $E[V]$ when implemented, however. It turns out that other heuristics can be

¹² Data to construct these PMFs can be downloaded from life tables published at SSA.gov, for example.

formed using fixed T_D solutions that perform better in simulations. A reason why the technique from Section 4.6 can perform sub optimally is that for fixed $RF(t)$ the optimal *solution* α is not always a decreasing function of time. For fixed T_D the low-volatility portfolio plays an important role at the last decision point (time $t=T_D-1$). Decision-rules may be needed to select an optimal α in the code when probabilities are tied (e.g., $P(\text{Ruin})=0 \forall \alpha$ at consecutive small $RF(t)$ values). For fixed small $RF(t)$, *these decisions* can result in α decreasing below the low-volatility portfolio as t increases, then stepping up to it at $t=T_D-1$ when a single withdrawal remains. Computing $E[\alpha]$ across fixed T_D solutions when α is non-monotone over time, and attaching it to $E[V]$ produces a heuristic that can perform sub optimally.¹³ Here, S_{Max} is chosen such that $P(T_D > S_{\text{Max}}) = 0$.

To overcome this the user can construct their own heuristic from fixed T_D solutions or a random T_D can be directly incorporated into the model during optimization. In the latter case, select a large value $t=S_{\text{Max}}$ and build conditional discrete hazard probabilities $P(T_D=i \mid T_D \geq i)$ for $i=0, 1, 2, \dots, S_{\text{Max}}-1$. The induction process begins at $t=S_{\text{Max}}$ but now must account for the fact that any time t may represent T_D . The random T_D value function is defined as:

$V_R(t, RF(t)) = \text{Optimal (minimum) probability of ruin after time } t \text{ given } RF(t) > 0$

$\alpha_R(t, RF(t)) = \text{Value of } \alpha \text{ required to achieve } V_R(t, RF(t)).$

For random T_D , given $T_D \geq t$, $\text{Ruin}^C(t)$ occurs if the withdrawal is successfully made at time t and $\text{Ruin}^C(\leq S_{\text{Max}})$ occurs at time t if the withdrawal just made is the last (that is, $T_D=t$). It will be assumed that T_D and $\hat{r}_{(t,\alpha)}$ are independent R.V.s $\forall t=1, \dots, S_{\text{Max}}$. Induction steps for the random T_D model are detailed below with times $t=S_{\text{Max}}-2$ and $t=S_{\text{Max}}-3$ derived in Appendix G.

¹³ This section has changed to correct a misstatement in the previous version. It now explicitly refers to the behavior of the *solution* α , NOT the *theoretical optimal* α , over time and for fixed $RF(t)$. Whether or not the theoretical optimal α is monotone over time for fixed $RF(t)$ is a statement that would require formal proof which has not been supplied here. Note that tied probabilities can often be addressed by changing the calculation order of terms for $V(t, RF(t))$. See the attached code in Appendix H, namely the function `optimize()` for one way to address ties.

4.6.1.1 Induction at Time $t=S_{\text{Max}}$

Assume the retiree arrives at time $t=S_{\text{Max}}$ and makes their last withdrawal. $\text{RF}(S_{\text{Max}}) (> 0)$ need not be calculated since there are no more withdrawals and $\text{Ruin}^C(\leq S_{\text{Max}})$ has occurred. At $t=S_{\text{Max}}$, $P(\text{Ruin})=0$ and a B.C. for the value function is $V_R(S_{\text{Max}}, \text{RF}(S_{\text{Max}})) = 0, \forall \text{RF}(S_{\text{Max}}) > 0$.

4.6.1.2 Induction at Time $t=S_{\text{Max}} - 1$

Assume the retiree arrives at time $t=S_{\text{Max}}-1$, makes their withdrawal and has at most one remaining. $\text{RF}(S_{\text{Max}}-1) (> 0)$ is calculated based on the portfolio's return just observed, $\hat{r}_{(S_{\text{Max}}-1, \alpha)}$. The retiree seeks α to minimize $P(\text{Ruin}(S_{\text{Max}}))$, which can be expressed as:

$$V_R(S_{\text{Max}}-1, \text{RF}(S_{\text{Max}}-1)) = \text{Min}_{(0 \leq \alpha \leq 1)} \left\{ P(\text{Ruin}(S_{\text{Max}})) \right\} \quad (36a)$$

$$\rightarrow V_R(S_{\text{Max}}-1, \text{RF}(S_{\text{Max}}-1)) = \text{Min}_{(0 \leq \alpha \leq 1)} \left\{ 1 - P(\text{Ruin}^C(S_{\text{Max}})) \right\} \quad (36b)$$

$$\rightarrow V_R(S_{\text{Max}}-1, \text{RF}(S_{\text{Max}}-1)) =$$

$$\text{Min}_{(0 \leq \alpha \leq 1)} \left\{ 1 - \left[P(T_D = S_{\text{Max}}-1 \mid T_D \geq S_{\text{Max}}-1) * (1) \right. \right. \\ \left. \left. + \underbrace{P(T_D > S_{\text{Max}}-1 \mid T_D \geq S_{\text{Max}}-1) * P(\hat{r}_{(S_{\text{Max}}, \alpha)} > \text{RF}(S_{\text{Max}}-1))}_{\text{This term = } P(T_D = S_{\text{Max}} \mid T_D \geq S_{\text{Max}}-1)} \right] \right\} \quad (36c)$$

$$\rightarrow V_R(S_{\text{Max}}-1, \text{RF}(S_{\text{Max}}-1)) =$$

$$\text{Min}_{(0 \leq \alpha \leq 1)} \left\{ P(T_D > S_{\text{Max}}-1 \mid T_D \geq S_{\text{Max}}-1) * [1 - P(\hat{r}_{(S_{\text{Max}}, \alpha)} > \text{RF}(S_{\text{Max}}-1))] \right\} \quad (36d)$$

$$\rightarrow V_R(S_{\text{Max}}-1, \text{RF}(S_{\text{Max}}-1)) =$$

$$\text{Min}_{(0 \leq \alpha \leq 1)} \left\{ P(T_D > S_{\text{Max}}-1 \mid T_D \geq S_{\text{Max}}-1) * [1 - (1 - F_{\hat{r}_{(S_{\text{Max}}, \alpha)}}(\text{RF}(S_{\text{Max}}-1)))] \right\}, \quad (36e)$$

given the known ruin factor $\text{RF}(S_{\text{Max}}-1)$. Note that $V_R(S_{\text{Max}}-1, \text{RF}(S_{\text{Max}}-1))$ can alternatively be expressed as:

$$V_R(S_{Max}-1, RF(S_{Max}-1)) = \text{Min}_{(0 \leq \alpha \leq 1)} \left\{ \begin{array}{l} P(T_D > S_{Max}-1 \mid T_D \geq S_{Max}-1)^* \\ [1 - (1 - F_{\hat{f}(S_{Max}, \alpha)}(RF(S_{Max}-1))) * (1 - E_{\hat{f}(S_{Max}, \alpha)}^+[V_R(S_{Max}, RF(S_{Max}))])] \end{array} \right\} \quad (36f)$$

$RF(S_{Max}) = RF(S_{Max}-1)/(\hat{f}(S_{Max}, \alpha) - RF(S_{Max}-1))$

$P[\text{Ruin}^C(S_{Max}) \text{ given } T_D > S_{Max} - 1.]$

$\text{Expected prob. of no ruin after time } t=S_{Max}, \text{ given } \text{Ruin}^C(S_{Max}). \text{ (This expression} = 1.)$

with optimal $\tilde{\alpha} = \alpha_R(S_{Max}-1, RF(S_{Max}-1))$. (See Appendix G for induction at $t=S_{Max}-2$ & $t=S_{Max}-3$.)

4.6.1.3 Value Function at Time t

Assume the retiree arrives a time t, successfully makes their withdrawal and updates the ruin factor $RF(t) (> 0)$ based on the portfolio return just observed, $\hat{f}_{(t, \alpha)}$. The goal is to find α that minimizes the probability of ruin after time t, which for a given α is expressed as:

$$P(\text{Ruin}) = P(\text{Ruin}(t+1) \cup \text{Ruin}(t+2) \cup \dots \cup \text{Ruin}(S_{Max})) \quad (37a)$$

$$= 1 - P(\text{Ruin}^C(t+1) \cap \text{Ruin}^C(t+2) \cap \dots \cap \text{Ruin}^C(S_{Max})) \quad (37b)$$

$$= 1 - [P(T_D = t \mid T_D \geq t) * (1) + P(T_D > t \mid T_D \geq t) * P(\hat{f}_{(t+1, \alpha)} > RF(t) \cap \text{Ruin}^C(t+2) \cap \dots \cap \text{Ruin}^C(S_{Max}))] \quad (37c)$$

$$= 1 - [1 - P(T_D > t \mid T_D \geq t) + P(T_D > t \mid T_D \geq t) * \{P(\hat{f}_{(t+1, \alpha)} > RF(t)) * P(\text{Ruin}^C(t+2) \cap \dots \cap \text{Ruin}^C(S_{Max}) \mid \hat{f}_{(t+1, \alpha)} > RF(t))\}] \quad (37d)$$

$$= P(T_D > t \mid T_D \geq t) * [1 - \{P(\hat{f}_{(t+1, \alpha)} > RF(t)) * P(\text{Ruin}^C(t+2) \cap \dots \cap \text{Ruin}^C(S_{Max}) \mid \hat{f}_{(t+1, \alpha)} > RF(t))\}] \quad (37e)$$

$$= P(T_D > t \mid T_D \geq t) * [1 - \{(1 - F_{\hat{f}_{(t+1, \alpha)}}(RF(t))) * P(\text{Ruin}^C(t+2) \cap \dots \cap \text{Ruin}^C(S_{Max}) \mid \hat{f}_{(t+1, \alpha)} > RF(t))\}] \quad (37f)$$

The value function at (t, $RF(t)$) is the minimum of $P(\text{Ruin})$, which is the minimum of (37f) over $(0 \leq \alpha \leq 1)$. It can be shown (see Appendix G) that this is expressed as:

The choice of α at time t is a balancing act between minimizing $P(\text{Ruin})$ at time $t+1$, and after time $t+1$.

$$\begin{aligned}
 V_R(t, RF(t)) = & \left\{ \begin{array}{l} \text{Min}_{(0 \leq \alpha \leq 1)} \left\{ \begin{array}{l} P(T_D > t \mid T_D \geq t)^* \\ \{ 1 - (1 - F_{\hat{f}(t+1, \alpha)}(RF(t)))^* \\ (1 - E_{\hat{f}(t+1, \alpha)}^+ \left[V_R(t+1, \frac{RF(t)}{\hat{f}(t+1, \alpha) - RF(t)}) \right]) \} \end{array} \right\} \\ \text{For } 0 \leq t \leq S_{\text{Max}}-1, \\ RF(t) > 0, \\ \text{and B.C., } V_R(S_{\text{Max}}, RF(S_{\text{Max}})) = 0. \end{array} \right. \quad (38)
 \end{aligned}$$

Optimality is achieved at $\tilde{\alpha} = \alpha_R(t, RF(t))$ and the expectation is over the conditional R.V. $\hat{f}_{(t+1, \alpha)}^+$. The difference between $V_R(t, RF(t))$ in (38) and $V(t, RF(t))$ in (22) is that for random T_D $P(\text{Ruin})$ is multiplied by $P(T_D > t \mid T_D \geq t)$ at time t . Note too that $V_R(t, RF(t))$ is bounded above by $P(T_D > t \mid T_D \geq t)$ at each $(t, RF(t))$, which is intuitive since $P(\text{Ruin})$ at any future time point cannot exceed the probability of living to attempt another withdrawal. The DP in (38) can be discretized exactly as (22) was for fixed T_D in (27). This was done using life tables from SSA.gov for same-age male/female (M/F) couples who retire at 65, see below.

Table 4. Comparison of $P(\text{Ruin})$ for Optimal Random T_D vs. Fixed α Strategies

Demo ¹	W_R	Suboptimal	Optimal	% Decrease in $P(\text{Ruin})$
		$P(\text{Ruin})$ [Best Fixed α] ²	$P(\text{Ruin})$ [α at $t=0$] ³	
(M/F) Couple	4%	0.042 [0.45]	0.030 [0.359]	28.0%
	5%	0.135 [0.60]	0.102 [0.488]	24.7%
	6%	0.252 [0.80]	0.206 [0.682]	18.2%

¹ $S_{\text{Max}}=47$ for M/F Couple with retirement beginning at age 65 for both.

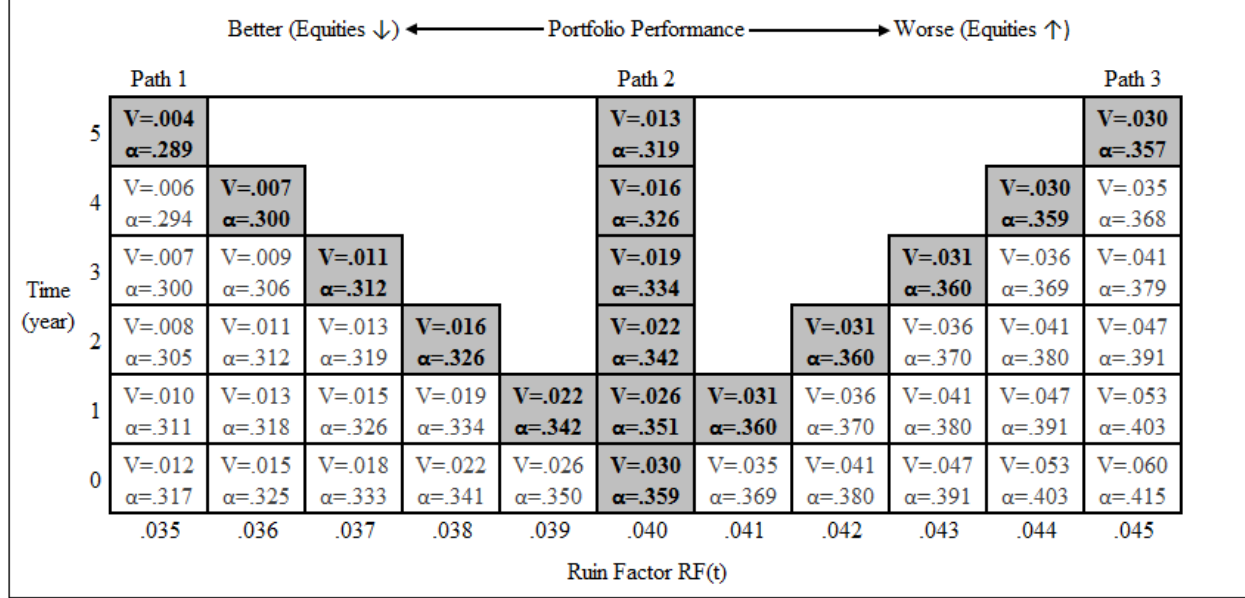
² $N=2.5$ million per simulation. Fixed α test set: $\{\alpha: \alpha=0.00 \text{ to } 1.00 \text{ by } 0.05\}$.

³ $P_R=5,000$. $P_a=1,000$ for $RF(t) \leq 0.5000$, $P_a=100$ for $RF(t) > 0.5000$. $E_R=0.0\%$.

As indicated in Table 4, $P(\text{Ruin})$ decreases and the differences between optimal and suboptimal solutions narrows when compared with the fixed $T_D=30$ model in Figure 7. The corresponding solution grid for M/F Couples is shown below in Figure 8 and reveals the familiar

pattern of equities decreasing as a percentage of the portfolio when returns are favorable, and increasing when returns are unfavorable.

Figure 8. Select Discrete Implementation Cells for M/F Couple Random T_D w/ $E_R=0.0\%$



4.6.2 Extension of Random T_D Model to Multiple Retirees

Define a multi-person unit (MPU) as a group that pools retirement funds and distributes systematic withdrawals amongst surviving members at each time point. The MPU makes withdrawals while it is alive and it is alive when at least one member is alive. Since systematic withdrawals are used ruin can occur. The general MPU consists of K females and L males. Let F_i and M_j represent the remaining lifetimes for female i and male j , respectively. At MPU retirement start F_i and M_j are independent discrete RVs with know PMF and:

$$T_D = \max\{F_1, \dots, F_i, \dots, F_K, M_1, \dots, M_j, \dots, M_L\} \quad (39)$$

Let $F_{TD}(\cdot)$ and $f_{TD}(\cdot)$ represent the CDF and PMF of T_D , respectively. The hazard probabilities $P(T_D=t \mid T_D \geq t)$ are derived as $f_{TD}(t) / [1 - F_{TD}(t-1)]$ for $t=0, 1, 2, \dots, S_{Max}$ with $F_{TD}(-1) = 0$. The PMF is constructed as $f_{TD}(t) = F_{TD}(t) - F_{TD}(t-1)$ and the CDF is built by recognizing that the maximum of a set is less than or equal to a given value *if-and-only-if* all members of the set are

less than or equal to that value. Namely,

$$T_D \leq t \leftrightarrow \max\{F_1, \dots, F_i, \dots, F_K, M_1, \dots, M_j, \dots, M_L\} \leq t \quad (40a)$$

$$\leftrightarrow (F_1 \leq t) \cap \dots \cap (F_K \leq t) \cap (M_1 \leq t) \cap \dots \cap (M_K \leq t) \quad (40b)$$

$$\rightarrow F_{TD}(t) = P(T_D \leq t) = P(F_1 \leq t) * \dots * P(F_K \leq t) * P(M_1 \leq t) * \dots * P(M_K \leq t). \quad (40c)$$

The individual RHS probabilities in (40c) are derived from published SSA life tables and the random T_D model is solved by treating the MPU as an individual with known hazard probabilities. The solution is an optimal decumulation strategy and the MPU can select W_R by first determining their desired success rate. Note that the same-age M/F couple analysis from the previous section is a special-case MPU having $K+L=2$ ($K=L=1$).



A risk with the random T_D models proposed here is that the hazard probabilities can become dated, rendering the solution obsolete. Suppose an MPU is formed and several members perish unexpectedly in year 1. This MPU will have remaining hazard probabilities that differ markedly from those used to construct the optimal strategy and the optimization should be rerun. The same applies to a couple where one member dies early in retirement. As $K+L$ increases the risk lessens, meaning that couples are most exposed. They are also the easiest to adjust. Simply transfer the living member to the random T_D model solved for single retirees of that gender starting at the appropriate time point. Random T_D MPU models should thus be monitored over time and updated when realized hazard probabilities differ from those used in the optimization.

4.7 Making Adjustments over Time

Adjustments to the optimal strategy can be made as follows. Suppose a retiree charting Path 1 in Figure 7 is dissatisfied with the decreasing α , and a retiree charting Path 3 is dissatisfied with the increasing $P(\text{Ruin})$. Both prefer a shift to Path 2 after making the time $t=5$ withdrawal. Table 5 provides the real rates of return that would track the ruin factor bucket

midpoints for each path. Both retirees began time $t=0$ with $W_R=4\%$ and an initial account balance of $\$A$. In time $t=0$ dollars the retirees withdraw $(4\%)*(\$A)/\text{year}$. The Path 1 retiree can shift to Path 2 by increasing the real withdrawal amount to $(4%)*(1.143)*(\$A)$ for $t \geq 6$. The Path 3 retiree can shift by lowering the real withdrawal amount to $(4%)*(0.889)*(\$A)$ for $t \geq 6$. The new rates are thus $W_R=(4%)*(1.143)=4.57\%$ and $W_R=(4%)*(0.889)=3.56\%$, respectively, if based on $\$A$ at time $t=0$. At time $t=6$ the retirees begin their new strategy and withdraw $(4.57%)*\prod_{i=1}^6 (1+I_i)*(\$A)$ and $(3.56%)*\prod_{i=1}^6 (1+I_i)*(\$A)$, respectively. Essentially, they halt their original plans and begin anew with time $t=5$ as the new time $t=0$, and horizon of $T_D=25$ years. The starting balances are $(1.143)*(\$A)*\prod_{i=1}^5 (1+I_i)$ and $(0.889)*(\$A)*\prod_{i=1}^5 (1+I_i)$, respectively (see Table 5). By shifting to Path 2, both use $W_R=4\%=RF(5)$, but it is now based on their time $t=5$ balances.¹⁴ The same process is followed when provisioning for emergencies.

Table 5. Real Returns that Generate Paths 1 and 3 from Figure 7*

Time (t)	Path 1			Path 3		
	Real Return $r_{(t, \alpha)}$	Real Account Balance	RF(t)	Real Return $r_{(t, \alpha)}$	Real Account Balance	RF(t)
1	6.56%		.039	1.56%		.041
2	6.53%		.038	1.71%		.042
3	6.50%		.037	1.87%		.043
4	6.48%		.036	2.03%		.044
5	6.46%		.035	2.18%		.045
		$(1.143)*(\$A)$			$(0.889)*(\$A)$	

* Returns are rounded.

4.8 Model Extensions/Enhancements

All implementations in Section 4 assume stock/bond returns originate as *iid* samples from their underlying distributions (see Appendix F). The *independence* assumption is a necessary condition for model development, but the *identical distribution* assumption is not. Means and variances for stock/bond returns can be defined as functions of time which would allow users to apply their own forecasts if these differ from historical averages.

¹⁴ Note that this result serves as justification for the conclusion reached at the end of Section 4.2, namely that charting a leftward course through the grid builds wealth sufficient to support larger withdrawals.

The expense ratio E_R has been defined as a single value that accounts for all expenses paid by the retiree to their financial firm. If stock/bond funds are used and the retiree pays no other expenses then E_R can be defined more precisely as $E_R = E_{R(\alpha)} = \alpha * E_{R(s)} + (1-\alpha) * E_{R(b)}$, where $E_{R(s)}$ and $E_{R(b)}$ are the exact expenses paid on stock/bond funds, respectively. Defining the inflation/expense-adjusted return at time t as $\hat{r}_{(t,\alpha)} = (1+r_{(t,\alpha)})(1-E_{R(t,\alpha)})$ allows the model to incorporate different expenses for different asset classes at each time t . Minimizing over α at each t now adjusts the joint density to account for the exact expenses incurred using that α .

The models developed here use only two asset classes, stocks and bonds. Nothing prevents additional asset classes from being incorporated except that the CDF of linear combinations of (possibly correlated) returns at each time t is needed. Depending on the levels of precision selected, clever coding may be required to keep the runtime reasonable since the minimization is now over more than just α .

5.0 Summary/Conclusion

The model proposed in (22) yields an optimal retirement decumulation strategy. Once distributional assumptions regarding asset class returns are made, however, the model becomes an estimate of the optimal strategy. As formulated, the model is intractable under common distributional assumptions and therefore must be discretized. The discretized solution is thus a numerical approximation to the estimate of an optimal strategy. Since the user controls the discretization's precision, the approximation can be driven to any desired degree of accuracy. This leaves distributional assumptions as the determining factor of how close the user's solution is to true optimality, and different users are sure to make different distributional assumptions.

The optimal decumulation strategy set forth here is based on an objective of minimizing the probability of ruin, not maximizing terminal wealth. Retirees seeking to maximize bequest

wealth may use this approach by increasing W_R to gain more risk exposure, then investing excess funds aggressively outside of their retirement portfolio. However, it may be more prudent for such a retiree to employ a model built on a wealth maximization objective, for example that proposed by Fan, Murray, and Pittman (2013). There is evidence to suggest, however, that retirees often bias towards loss prevention, not wealth maximization. A 2012 study by ING Retirement Research Institute found that 80% of target-date fund users prefer a portfolio that protects against losses, and 66% of non target-date fund users agreed.

A usage scenario for an advisor who seeks to employ the model proposed here is as follows. The advisor codes or has the DP in (27) coded based on the market return assumptions they perceive most appropriate for the coming retirement horizon using either a fixed or random T_D , and desired level of precision.¹⁵ The result is a grid of the form presented in Figure 5. The advisor then customizes this grid for each retiree by shading various regions the retiree indicates they are comfortable and uncomfortable entering. A plan is then set forth at time $t=0$ which indicates precisely when and what type of adjustments will be made if the retiree encroaches on a region they have indicated is intolerable. The retiree thus takes comfort in the knowledge that a strategy exists, and they remain fully informed of what actions will be taken, and when they will be taken, to modify that strategy based on their portfolio's performance over time.

Supporting Appendices



¹⁵ See Appendix H for a full C++ implementation.

Appendix A: Derivation of the Ruin Factor RF(t)

The condition for Ruin(1) was developed in Section 3.3. Given Ruin^C(1), the event Ruin(2) can be formulated similarly. Namely, assuming Ruin^C(1) has occurred it follows that,

$$\begin{aligned} \text{Ruin}(2) \mid \text{Ruin}^C(1) &\leftrightarrow [\$A*(1+R_{(1,a)})*(1-E_R) - \$A*(W_R)*(1+I_1)]*(1+R_{(2,a)})*(1-E_R) \\ &\leq \$A*(W_R)*(1+I_1)*(1+I_2) \end{aligned} \quad (\text{A.1a})$$

$$\leftrightarrow [(1+r_{(1,a)})*(1-E_R) - W_R]*(1+r_{(2,a)})*(1-E_R) \leq W_R \quad (\text{A.1b})$$

$$\leftrightarrow (1+r_{(2,a)})*(1-E_R) \leq W_R / [(1+r_{(1,a)})*(1-E_R) - W_R] \quad (\text{A.1c})$$

$$\leftrightarrow \hat{r}_{(2,a)} \leq W_R / [\hat{r}_{(1,a)} - W_R] \quad (\text{A.1d})$$

where the term divided $[(1+r_{(1,a)})*(1-E_R) - W_R] > 0$ since it is precisely the condition that must be satisfied to avoid ruin at time $t=1$, see (2c).

Thus, at time $t=2$, $P(\text{Ruin}(2) \mid \text{Ruin}^C(1)) = F_{\hat{r}_{(2,a)}}(W_R / [\hat{r}_{(1,a)} - W_R])$. Extending the analysis, suppose $(\text{Ruin}^C(1) \cap \text{Ruin}^C(2))$ has occurred, the condition required to experience ruin at $t=3$ is formulated as:

$$\begin{aligned} \text{Ruin}(3) \mid \text{Ruin}^C(1) \cap \text{Ruin}^C(2) &\leftrightarrow \{[\$A*(1+R_{(1,a)})*(1-E_R) - \$A*(W_R)*(1+I_1)]*(1+R_{(2,a)})*(1-E_R) - \\ &\quad \$A*(W_R)*(1+I_1)*(1+I_2)\}*(1+R_{(3,a)})*(1-E_R) \\ &\leq \$A*(W_R)*(1+I_1)*(1+I_2)*(1+I_3) \end{aligned} \quad (\text{A.2a})$$

$$\begin{aligned} &\leftrightarrow \{[(1+r_{(1,a)})*(1-E_R) - W_R]*(1+r_{(2,a)})*(1-E_R) - W_R\}*(1+r_{(3,a)})*(1-E_R) \\ &\leq W_R \end{aligned} \quad (\text{A.2b})$$

$$\begin{aligned} &\leftrightarrow (1+r_{(3,a)})*(1-E_R) \\ &\leq W_R / \{[(1+r_{(1,a)})*(1-E_R) - W_R]*(1+r_{(2,a)})*(1-E_R) - W_R\} \end{aligned} \quad (\text{A.2c})$$

$$\leftrightarrow \hat{r}_{(3,a)} \leq W_R / \{[\hat{r}_{(1,a)} - W_R]*(\hat{r}_{(2,a)} - W_R)\} \quad (\text{A.2d})$$

Where the term just divided, $\{(1+r_{(1,a)})*(1-E_R) - W_R\}*(1+r_{(2,a)})*(1-E_R) - W_R\} > 0$ since it is precisely the condition needed to avoid ruin at time $t=2$, see (A.1b). Thus:

$$\text{Ruin}(3) \mid \text{Ruin}^C(1) \cap \text{Ruin}^C(2) \leftrightarrow \hat{r}_{(3,a)} \leq \left(\frac{\{W_R / [\hat{r}_{(1,a)} - W_R]\}}{\{(\hat{r}_{(2,a)} - W_R / [\hat{r}_{(1,a)} - W_R])\}} \right) \quad (\text{A.3})$$

Therefore, at time $t=3$, $\text{Ruin}(3)$ has conditional probability of occurrence:

$$P(\text{Ruin}(3) \mid \text{Ruin}^C(1) \cap \text{Ruin}^C(2)) = F_{\hat{r}_{(3,a)}} \left(\frac{\{W_R / [\hat{r}_{(1,a)} - W_R]\}}{\{(\hat{r}_{(2,a)} - W_R / [\hat{r}_{(1,a)} - W_R])\}} \right) \quad (\text{A.4})$$

The following pattern reveals itself. At times $t=0$, $t=1$, $t=2$, ruin occurs as, see (2d), (A.1d), and (A.2d):

$$\text{Ruin}(1) \leftrightarrow \hat{r}_{(1,a)} \leq W_R \quad (\text{A.5a})$$

$$\text{Ruin}(2) \mid \text{Ruin}^C(1) \leftrightarrow \hat{r}_{(2,a)} \leq W_R / [\hat{r}_{(1,a)} - W_R] \quad (\text{A.5b})$$

$$\begin{aligned} \text{Ruin}(3) \mid \text{Ruin}^C(1) \cap \text{Ruin}^C(2) \\ \leftrightarrow \hat{r}_{(3,a)} \leq \left(\frac{\{W_R / [\hat{r}_{(1,a)} - W_R]\}}{\{(\hat{r}_{(2,a)} - W_R / [\hat{r}_{(1,a)} - W_R])\}} \right) \end{aligned} \quad (\text{A.5c})$$

The *ruin factor* at time t is defined as the quantity on the RHS of each condition above and denoted $\text{RF}(t)$, that is:

$$\text{RF}(0) = W_R \quad (\text{A.6a})$$

$$\text{RF}(1) = W_R / [\hat{r}_{(1,a)} - W_R] \quad (\text{A.6b})$$

$$\text{RF}(2) = \left(\frac{\{W_R / [\hat{r}_{(1,a)} - W_R]\}}{\{(\hat{r}_{(2,a)} - W_R / [\hat{r}_{(1,a)} - W_R])\}} \right) \quad (\text{A.6c})$$

and expressing these quantities recursively:

$$\text{RF}(0) = W_R \quad (\text{A.7a})$$

$$\text{RF}(1) = \text{RF}(0) / [\hat{r}_{(1,a)} - \text{RF}(0)] \quad (\text{A.7b})$$

$$\text{RF}(2) = \text{RF}(1) / [\hat{r}_{(2,a)} - \text{RF}(1)] \quad (\text{A.7c})$$

Replacing the notation in (A.5a), (A.5b), and (A.5c) by the terms defined in (A.7a), (A.7b), and (A.7c) yields:

$$\text{Ruin}(1) \leftrightarrow \hat{r}_{(1,\alpha)} \leq \text{RF}(0) \quad (\text{A.8a})$$

$$\text{Ruin}(2) \mid \text{Ruin}^C(1) \leftrightarrow \hat{r}_{(2,\alpha)} \leq \text{RF}(1) \quad (\text{A.8b})$$

$$\text{Ruin}(3) \mid \text{Ruin}^C(1) \cap (\text{Ruin}^C(2)) \leftrightarrow \hat{r}_{(3,\alpha)} \leq \text{RF}(2), \quad (\text{A.8c})$$

and in general, $\text{Ruin}(t)$ occurs $\leftrightarrow \hat{r}_{(t,\alpha)} \leq \text{RF}(t-1)$, which will be shown next for any time t .

For the general case of time t , assume that $(\text{Ruin}^C(1) \cap \text{Ruin}^C(2) \cap \dots \cap \text{Ruin}^C(t-1))$ has occurred. $\text{Ruin}(t)$ occurs if there are insufficient funds to make the withdrawal at time t , namely:

$$\text{Ruin}(t) \mid \text{Ruin}^C(1) \cap \dots \cap \text{Ruin}^C(t-1) \leftrightarrow$$

$$\begin{aligned} & \$A^*(1-E_R)^t * \prod_{j=1}^t (1+I_j)(1+r_{(j,\alpha)}) - \prod_{j=1}^t (1+I_j) * \sum_{k=2}^t \left\{ \$A^*(W_R)^*(1-E_R)^{t-k+1} \prod_{j=k}^t (1+r_{(j,\alpha)}) \right\} \\ & \leq \$A^*(W_R)^* \prod_{j=1}^t (1+I_j) \end{aligned} \quad (\text{A.9})$$

Where the term on the left is the accumulation of (account values - withdrawals) compounded for another time point, repeated until time t . Each term in (A.9) contains $\$A$ along with the accumulated compounding of all inflation rates $\prod_{j=1}^t (1+I_j)$, therefore these terms cancel throughout. Note that inflation thus plays no direct role in determining ruin for the retiree, except for the degree to which it impacts $r_{(t,\alpha)}$, recalling that $r_{(t,\alpha)} \mid I_t, R_{(t,\alpha)}$ is defined conditionally.

After cancelling:

$$\text{Ruin}(t) \mid \text{Ruin}^C(1) \cap \dots \cap \text{Ruin}^C(t-1) \leftrightarrow$$

$$(1-E_R)^t * \prod_{j=1}^t (1+r_{(j,\alpha)}) - \sum_{k=2}^t \left\{ (W_R)^*(1-E_R)^{t-(k-1)} \prod_{j=k}^t (1+r_{(j,\alpha)}) \right\} \leq W_R \quad (\text{A.10})$$

The term $(1+r_{(t,\alpha)})$ is also common on the LHS, and is factored along with $(1-E_R)$, leaving:

$$\text{Ruin}(t) \mid \text{Ruin}^C(1) \cap \dots \cap \text{Ruin}^C(t-1) \leftrightarrow$$

$$(1+r_{(t,\alpha)})^*(1-E_R)^* \left[\underbrace{\left((1-E_R)^{t-1} * \prod_{j=1}^{t-1} (1+r_{(j,\alpha)}) - \sum_{k=2}^{t-1} \left\{ (W_R)^*(1-E_R)^{(t-1)-(k-1)} \prod_{j=k}^{t-1} (1+r_{(j,\alpha)}) \right\} \right)}_{\leq W_R} - W_R \right] \quad (\text{A.11})$$

Considering the inner bracketed term shown above, both $(1+ r_{(t-1,a)})$ and $(1-E_R)$ are common throughout and are thus factored, resulting in:

$$\text{Ruin}(t) \mid \text{Ruin}^C(1) \cap \dots \cap \text{Ruin}^C(t-1) \leftrightarrow$$

$$\begin{aligned} & (1+ r_{(t,a)})^*(1-E_R)^* \\ & \left(\left((1+r_{(t-1,a)})^*(1-E_R)^* \left\{ \underbrace{(1-E_R)^{t-2} * \prod_{j=1}^{t-2} (1+r_{(j,a)}) - \sum_{k=2}^{t-2} \{ (W_R)^*(1-E_R)^{(t-2)-(k-1)} \prod_{j=k}^{t-2} (1+r_{(j,a)}) \}}_{- W_R} \right\} - W_R \right) \right) \\ & \leq W_R \end{aligned} \tag{A.12}$$

Continuing, the inner term above contains both $(1+ r_{(t-2,a)})$ and $(1-E_R)$ and these terms are thus factored, leaving:

$$\text{Ruin}(t) \mid \text{Ruin}^C(1) \cap \dots \cap \text{Ruin}^C(t-1) \leftrightarrow$$

$$\begin{aligned} & (1+ r_{(t,a)})^*(1-E_R)^* \\ & \left(\left(\left((1+ r_{(t-1,a)})^*(1-E_R)^* \right. \right. \right. \\ & \quad \left((1+r_{(t-2,a)})^*(1-E_R)^* \left\{ \underbrace{(1-E_R)^{t-3} * \prod_{j=1}^{t-3} (1+r_{(j,a)}) - \sum_{k=2}^{t-3} \{ W_R^*(1-E_R)^{(t-3)-(k-1)} \prod_{j=k}^{t-3} (1+r_{(j,a)}) \}}_{- W_R} \right\} \right) \\ & \quad \left. - W_R \right) \\ & \quad \left. - W_R \right) \\ & \leq (W_R) \end{aligned} \tag{A.13}$$

Replacing $(1+ r_{(t,a)})^*(1-E_R)$ with $\hat{r}_{(t,a)}$ for all t and repeating the above for the remainder of the expression leaves ruin at time t , given no ruin prior to time t , defined as:

$$\text{Ruin}(t) \mid \text{Ruin}^C(1) \cap \dots \cap \text{Ruin}^C(t-1) \leftrightarrow$$

$$\hat{f}_{(t,\alpha)} * \left[\hat{f}_{(t-1,\alpha)} * \left[\hat{f}_{(t-2,\alpha)} * \dots * \underbrace{\left[\hat{f}_{(2,\alpha)} * \left[\hat{f}_{(1,\alpha)} - W_R \right] - W_R \right]}_{[1]} - W_R \right] - W_R \right] \leq W_R \quad (\text{A.14})$$

$\underbrace{\hspace{10em}}_{[2]}$
 \dots
 $\underbrace{\hspace{10em}}_{[t-2]}$
 $\underbrace{\hspace{10em}}_{[t-1]}$

Where the bracketed terms must be > 0 as they define precisely the condition that must be met for ruin to be avoided at that time, which is $< t$. That is, the term [1] can only exceed zero when $\text{Ruin}^C(1)$ occurs. Dividing through both sides by all but $\hat{f}_{(t,\alpha)}$ leaves:

$$\text{Ruin}(t) \mid \text{Ruin}^C(1) \cap \dots \cap \text{Ruin}^C(t-1) \leftrightarrow$$

$$\hat{f}_{(t,\alpha)} \leq W_R / \hat{f}_{(t-1,\alpha)} * \left[\hat{f}_{(t-2,\alpha)} * \dots * \underbrace{\left[\hat{f}_{(2,\alpha)} * \underbrace{\left[\hat{f}_{(1,\alpha)} - W_R \right]}_{[1]} - W_R \right]}_{[2]} - W_R \right] \quad (\text{A.15})$$

The RHS denominator is now unfolded by first factoring [1] from [2] leaving the new [2], say [2*], as:

$$[2^*] = \left[\hat{f}_{(1,\alpha)} - W_R \right] * \left[\hat{f}_{(2,\alpha)} - W_R / (\hat{f}_{(1,\alpha)} - W_R) \right] \quad (\text{A.16})$$

The term [2*] is then factored out in this manner, and the process is repeated. Note that $W_R / [2^*] = \text{RF}(1) / [\hat{f}_{(2,\alpha)} - \text{RF}(1)] = \text{RF}(2)$. The next term [3] becomes [3*], where:

$$[3^*] = \left[\hat{f}_{(1,\alpha)} - W_R \right] * \left[\hat{f}_{(2,\alpha)} - W_R / (\hat{f}_{(1,\alpha)} - W_R) \right] * \left[\hat{f}_{(3,\alpha)} - W_R / \left[\left[\hat{f}_{(1,\alpha)} - W_R \right] * \left[\hat{f}_{(2,\alpha)} - W_R / (\hat{f}_{(1,\alpha)} - W_R) \right] \right] \right] \quad (\text{A.17})$$

Note that $W_R / [3^*] = RF(2) / \hat{r}_{(3,\alpha)} - RF(2) = RF(3)$. Continuing to factor the denominator in this manner leaves the last term $[t-1^*]$, where $W_R / [t-1^*] = RF(t-2) / \hat{r}_{(t-1,\alpha)} - RF(t-2) = RF(t-1)$.

Finally,

$$\text{Ruin}(t) \mid \text{Ruin}^C(1) \cap \dots \cap \text{Ruin}^C(t-1)$$

$$\leftrightarrow \hat{r}_{(t,\alpha)} \leq RF(t-2) / [\hat{r}_{(t-1,\alpha)} - RF(t-2)] \quad (\text{A.18})$$

$$\leftrightarrow \hat{r}_{(t,\alpha)} \leq RF(t-1). \quad (\text{A.19})$$

(Continued on next page ...)

Appendix B: Full Details of Retirees 1 and 2

Table B.1. The Full Experience of Retiree 1 from Section 3.4

Retiree 1										
Year	Ruin Factor RF(t)	Start Year Total Balance	Total Return (1 + Rt)	Inflation (1 + It)	Inflation-Adjusted Return (1 + rt)	Expense Ratio (1 - ER)	Inflation-Adjusted EoY Withdrawal Rate	End Year Total Balance	End Year Real Balance	Terminal Ruin Factor
0	0.040	\$100,000	1.044	1.0250	1.019	0.995	0.0410	\$99,815	\$97,380	
1	0.041	\$99,815	1.190	1.0245	1.161	0.995	0.0420	\$113,961	\$108,524	
2	0.037	\$113,961	1.155	1.0239	1.128	0.995	0.0430	\$126,637	\$117,775	
3	0.034	\$126,637	1.040	1.0237	1.016	0.995	0.0440	\$126,699	\$115,106	
4	0.035	\$126,699	0.932	1.0237	0.910	0.995	0.0451	\$112,923	\$100,218	
5	0.040	\$112,923	1.051	1.0239	1.026	0.995	0.0462	\$113,421	\$98,307	
6	0.041	\$113,421	0.824	1.0229	0.806	0.995	0.0472	\$88,322	\$74,839	
7	0.053	\$88,322	1.102	1.0235	1.077	0.995	0.0483	\$92,049	\$76,209	
8	0.052	\$92,049	1.059	1.0232	1.035	0.995	0.0494	\$92,042	\$74,476	
9	0.054	\$92,042	0.882	1.0234	0.862	0.995	0.0506	\$75,727	\$59,876	
10	0.067	\$75,727	0.853	1.0228	0.834	0.995	0.0517	\$59,121	\$45,706	
11	0.088	\$59,121	0.819	1.0222	0.801	0.995	0.0529	\$42,901	\$32,446	
12	0.123	\$42,901	1.021	1.0222	0.999	0.995	0.0541	\$38,195	\$28,259	
13	0.142	\$38,195	1.004	1.0223	0.982	0.995	0.0553	\$32,632	\$23,618	
14	0.169	\$32,632	0.859	1.0225	0.840	0.995	0.0565	\$22,236	\$15,739	
15	0.254	\$22,236	1.220	1.0226	1.193	0.995	0.0578	\$21,206	\$14,680	
16	0.272	\$21,206	1.038	1.0223	1.015	0.995	0.0591	\$15,995	\$10,830	
17	0.369	\$15,995	1.249	1.0225	1.222	0.995	0.0604	\$13,843	\$9,167	
18	0.436	\$13,843	1.045	1.0225	1.022	0.995	0.0618	\$8,214	\$5,319	
19	0.752	\$8,214	1.180	1.0217	1.155	0.995	0.0631	\$3,333	\$2,113	
20	1.893	\$3,333	0.835	1.0216	0.818	0.995	0.0645	Ruin(20)	Ruin(20)	-1.753

(Continued on next page ...)

Table B.2. The Full Experience of Retiree 2 from Section 3.4

Retiree 2

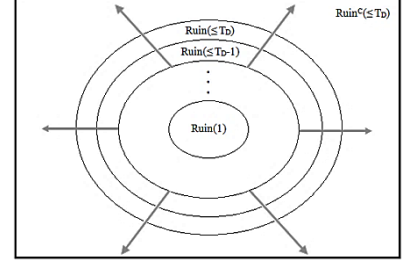
Year	Ruin Factor RF(t)	Start Year Total Balance	Total Return (1 + Rt)	Inflation (1 + It)	Inflation-Adjusted Return (1 + rt)	Expense Ratio (1 - ER)	Inflation-Adjusted EoY Withdrawal Rate	End Year Total Balance	End Year Real Balance	Terminal Ruin Factor
0	0.040	\$100,000	1.048	1.0250	1.022	1.000	0.0410	\$100,657	\$98,202	
1	0.041	\$100,657	0.912	1.0261	0.889	0.995	0.0421	\$87,165	\$82,874	
2	0.048	\$87,165	1.098	1.0255	1.071	0.995	0.0431	\$90,912	\$84,289	
3	0.047	\$90,912	1.179	1.0248	1.151	0.995	0.0442	\$102,243	\$92,501	
4	0.043	\$102,243	1.008	1.0246	0.984	0.995	0.0453	\$98,024	\$86,555	
5	0.046	\$98,024	1.088	1.0241	1.062	0.995	0.0464	\$101,483	\$87,503	
6	0.046	\$101,483	0.739	1.0244	0.722	0.995	0.0475	\$69,892	\$58,827	
7	0.068	\$69,892	1.180	1.0243	1.152	0.995	0.0487	\$77,202	\$63,441	
8	0.063	\$77,202	0.994	1.0245	0.971	0.995	0.0499	\$71,397	\$57,268	
9	0.070	\$71,397	1.006	1.0237	0.983	0.995	0.0511	\$66,396	\$52,024	
10	0.077	\$66,396	1.391	1.0238	1.359	0.995	0.0523	\$86,668	\$66,329	
11	0.060	\$86,668	1.208	1.0230	1.181	0.995	0.0535	\$98,851	\$73,955	
12	0.054	\$98,851	1.163	1.0229	1.137	0.995	0.0547	\$108,882	\$79,633	
13	0.050	\$108,882	1.062	1.0229	1.039	0.995	0.0559	\$109,506	\$78,297	
14	0.051	\$109,506	1.016	1.0226	0.994	0.995	0.0572	\$104,988	\$73,403	
15	0.054	\$104,988	1.149	1.0231	1.123	0.995	0.0585	\$114,157	\$78,012	
16	0.051	\$114,157	1.091	1.0241	1.065	0.995	0.0599	\$117,877	\$78,656	
17	0.051	\$117,877	1.215	1.0237	1.187	0.995	0.0614	\$136,367	\$88,884	
18	0.045	\$136,367	0.936	1.0236	0.914	0.995	0.0628	\$120,661	\$76,833	
19	0.052	\$120,661	1.011	1.0239	0.988	0.995	0.0643	\$114,985	\$71,509	
20	0.056	\$114,985	1.057	1.0236	1.033	0.995	0.0658	\$114,367	\$69,486	
21	0.058	\$114,367	1.078	1.0239	1.053	0.995	0.0674	\$115,902	\$68,779	
22	0.058	\$115,902	1.055	1.0238	1.030	0.995	0.0690	\$114,732	\$66,503	
23	0.060	\$114,732	1.214	1.0242	1.185	0.995	0.0707	\$131,522	\$74,436	
24	0.054	\$131,522	1.093	1.0242	1.067	0.995	0.0724	\$135,753	\$75,016	
25	0.053	\$135,753	0.932	1.0238	0.910	0.995	0.0741	\$118,437	\$63,923	
26	0.063	\$118,437	1.013	1.0233	0.990	0.995	0.0758	\$111,753	\$58,943	
27	0.068	\$111,753	1.302	1.0237	1.272	0.995	0.0776	\$137,023	\$70,595	
28	0.057	\$137,023	1.214	1.0250	1.184	0.995	0.0796	\$157,544	\$79,187	
29	0.051	\$157,544	1.215	1.0240	1.187	0.995	0.0815	\$182,351	\$89,504	
30	0.045	\$182,351	1.138	1.0232	1.112	0.995	0.0834	\$198,143	\$95,049	0.042

(Continued on next page ...)

Appendix C: Induction at Times T_D-2 and T_D-3 for Fixed T_D

C.1 Induction at Time $t=T_D-2$

Assume the retiree arrives at time $t=T_D-2$, makes their 3rd last withdrawal and has two remaining. The ruin factor $RF(T_D-2) (> 0)$ is calculated based on the portfolio's return just observed, $\hat{r}_{(T_D-2,\alpha)}$. The retiree now faces the restricted sample



space $S=\{\text{Ruin}(T_D-1), \text{Ruin}(T_D), \text{Ruin}^C(\leq T_D)\}$, as shown, and seeks to make the optimal asset allocation decision to minimize $P(\text{Ruin}(T_D-1) \cup \text{Ruin}(T_D))$, which is the probability of ruin at any future time point. Using (10), $P(\text{Ruin})$ is now defined as:

$$P(\text{Ruin}(T_D-1) \cup \text{Ruin}(T_D)) = 1 - P(\text{Ruin}^C(T_D-1) \cap \text{Ruin}^C(T_D)) \quad (\text{C.1a})$$

$$= 1 - P(\text{Ruin}^C(T_D-1)) * \underbrace{P(\text{Ruin}^C(T_D) \mid \text{Ruin}^C(T_D-1))}. \quad (\text{C.1b})$$

Note that the optimal value for this probability was derived at time $t=T_D-1$ for all $RF(t) > 0$.

The value function is thus:

$$V(T_D-2, RF(T_D-2)) = \text{Min}_{(0 \leq \alpha \leq 1)} \left\{ 1 - P(\text{Ruin}^C(T_D-1)) * P(\text{Ruin}^C(T_D) \mid \text{Ruin}^C(T_D-1)) \right\} \quad (\text{C.2a})$$

$$\rightarrow V(T_D-2, RF(T_D-2)) = \text{Min}_{(0 \leq \alpha \leq 1)} \left\{ 1 - P(\hat{r}_{(T_D-1,\alpha)} > RF(T_D-2)) * \left\{ \begin{array}{l} P(\hat{r}_{(T_D,\tilde{\alpha})} > RF(T_D-1) \mid \hat{r}_{(T_D-1,\alpha)} > RF(T_D-2)) \end{array} \right\} \right\} \quad (\text{C.2b})$$

$$\rightarrow V(T_D-2, RF(T_D-2)) = \text{Min}_{(0 \leq \alpha \leq 1)} \left\{ 1 - P(\hat{r}_{(T_D-1,\alpha)} > RF(T_D-2)) * \left\{ \begin{array}{l} \left[\frac{P(\hat{r}_{(T_D,\tilde{\alpha})} > RF(T_D-1) \cap \hat{r}_{(T_D-1,\alpha)} > RF(T_D-2))}{P(\hat{r}_{(T_D-1,\alpha)} > RF(T_D-2))} \right] \end{array} \right\} \right\} \quad (\text{C.2c})$$

Note that since $RF(T_D-1)$ is a function of $\hat{r}_{(T_D-1,\alpha)}$, these integrals cannot be split despite the independence of $\hat{r}_{(T_D,\tilde{\alpha})}$ and $\hat{r}_{(T_D-1,\alpha)}$.

$$\rightarrow V(T_D-2, RF(T_D-2)) = \text{Min}_{(0 \leq \alpha \leq 1)} \left\{ 1 - (1 - F\hat{r}_{(T_D-1,\alpha)}(RF(T_D-2))) * \left\{ \begin{array}{l} \left[\frac{\int_{RF(T_D-2)}^{\infty} \int_{RF(T_D-1)}^{\infty} f(\hat{r}_{(T_D,\tilde{\alpha})}, \hat{r}_{(T_D-1,\alpha)}) d(\hat{r}_{(T_D,\tilde{\alpha})}) d(\hat{r}_{(T_D-1,\alpha)})}{\int_{RF(T_D-2)}^{\infty} f(\hat{r}_{(T_D-1,\alpha)}) d(\hat{r}_{(T_D-1,\alpha)})} \right] \end{array} \right\} \right\} \quad (\text{C.2d})$$

and, invoking the assumption that portfolio returns are independent between time points,

$$\begin{aligned} \rightarrow V(T_D-2, RF(T_D-2)) = & \left\{ \begin{array}{l} \text{This term is } [1 - V(T_D-1, RF(T_D-1))] \text{ found earlier. An} \\ \text{optimal policy must be followed at each stage of the DP.} \\ 1 - (1 - F_{\hat{r}(T_D-1, \alpha)}(RF(T_D-2)))^* \\ \left[\frac{\int_{RF(T_D-2)}^{\infty} f(\hat{r}(T_D-1, \alpha)) \left[\int_{RF(T_D-1)}^{\infty} f(\hat{r}(T_D, \tilde{\alpha})) d(\hat{r}(T_D, \tilde{\alpha})) \right] d(\hat{r}(T_D-1, \alpha))}{\int_{RF(T_D-2)}^{\infty} f(\hat{r}(T_D-1, \alpha)) d(\hat{r}(T_D-1, \alpha))} \right] \end{array} \right\} \quad (C.2e) \end{aligned}$$

$$\begin{aligned} \rightarrow V(T_D-2, RF(T_D-2)) = & \left\{ \begin{array}{l} \text{By definition, this integral is the expected} \\ \text{value of } [1 - V(\cdot)] \text{ over the R.V. } \hat{r}(T_D-1, \alpha)^+. \\ 1 - (1 - F_{\hat{r}(T_D-1, \alpha)}(RF(T_D-2)))^* \\ \left[\frac{\int_{RF(T_D-2)}^{\infty} f(\hat{r}(T_D-1, \alpha)) [1 - V(T_D-1, RF(T_D-1))] d(\hat{r}(T_D-1, \alpha))}{\int_{RF(T_D-2)}^{\infty} f(\hat{r}(T_D-1, \alpha)) d(\hat{r}(T_D-1, \alpha))} \right] \end{array} \right\} \quad (C.2f) \end{aligned}$$

Now, since $RF(T_D-1)$ is a function of $\hat{r}(T_D-1, \alpha)$, namely,

$$RF(T_D-1) = \frac{RF(T_D-2)}{\hat{r}(T_D-1, \alpha) - RF(T_D-2)} \quad (C.3)$$

the value function in (C.2f) can be written as¹⁶,

$$\begin{aligned} V(T_D-2, RF(T_D-2)) = & \left\{ \begin{array}{l} 1 - (1 - F_{\hat{r}(T_D-1, \alpha)}(RF(T_D-2)))^* \\ (1 - E_{\hat{r}(T_D-1, \alpha)}^+ \left[V(T_D-1, \frac{RF(T_D-2)}{\hat{r}(T_D-1, \alpha) - RF(T_D-2)}) \right]) \end{array} \right\} \quad (C.4) \end{aligned}$$

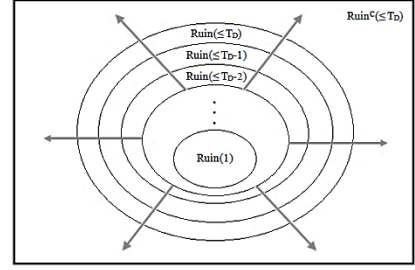
with optimality achieved at $\alpha = \alpha(T_D-2, RF(T_D-2))$ and the expectation over the conditional R.V.

$\hat{r}(T_D-1, \alpha)^+ = (\hat{r}(T_D-1, \alpha) \mid \hat{r}(T_D-1, \alpha) > RF(T_D-2))$ where $\{\hat{r}(T_D-1, \alpha) > RF(T_D-2)\} \equiv \{RF(T_D-1) > 0\}$.

¹⁶ An intuitive explanation of the need to take expectations is that $V(T_D-1, X)$, was already found for all positive ruin factors X , treating X as constant. At the current induction step, it is discovered that X is random with known PDF under our control via α . In the optimization over α , $E_X[V(T_D-1, X)]$ is then evaluated across the various PDFs of X .

C.2 Induction at Time $t=T_D - 3$

Assume the retiree arrives at time $t=T_D-3$, makes their 4th-last withdrawal and has 3 left.¹⁷ The ruin factor $RF(T_D-3)$ (> 0) is calculated based on the portfolio return just observed, $\hat{r}_{(T_D-3,\alpha)}$. The retiree is facing the restricted sample space



$S = \{Ruin(T_D-2), Ruin(T_D-1), Ruin(T_D), Ruin^C(\leq T_D)\}$ (shown at right) and seeks to make the optimal asset allocation decision to minimize $P(Ruin(T_D-2) \cup Ruin(T_D-1) \cup Ruin(T_D))$, which is the probability of ruin at any future time point. Under the restricted sample space $P(Ruin)$ is now defined as:

$$P(Ruin(T_D-2) \cup Ruin(T_D-1) \cup Ruin(T_D)) = 1 - P(Ruin^C(T_D-2) \cap Ruin^C(T_D-1) \cap Ruin^C(T_D)) \quad (C.5a)$$

$$= 1 - P(Ruin^C(T_D-2)) * \underbrace{P(Ruin^C(T_D-1) \cap Ruin^C(T_D) | Ruin^C(T_D-2))}_{\text{restricted sample space}} \quad (C.5b)$$

Note that the optimal value for this probability was derived at time $t=T_D-2$ for all $RF(t) > 0$.

The value function is thus:

$$V(T_D-3, RF(T_D-3)) = \text{Min}_{(0 \leq \alpha \leq 1)} \left\{ \begin{array}{l} 1 - P(Ruin^C(T_D-2)) * \\ P(Ruin^C(T_D-1) \cap Ruin^C(T_D) | Ruin^C(T_D-2)) \end{array} \right\} \quad (C.6a)$$

$$\rightarrow V(T_D-3, RF(T_D-3)) =$$

$$\text{Min}_{(0 \leq \alpha \leq 1)} \left\{ \begin{array}{l} 1 - P(\hat{r}_{(T_D-2,\alpha)} > RF(T_D-3)) * \\ P(\hat{r}_{(T_D-1,\tilde{\alpha})} > RF(T_D-2) \cap \hat{r}_{(T_D,\tilde{\alpha})} > RF(T_D-1) | \hat{r}_{(T_D-2,\alpha)} > RF(T_D-3)) \end{array} \right\} \quad (C.6b)$$

Dynamic programming requires an optimal policy be followed at each future stage, and these $\tilde{\alpha}$ reflect those optimal values.

¹⁷ Induction at time $t=T_D-3$ is nearly identical to induction at time $t=T_D-2$, and the process is generalized for time t in Section 3.7.3.

$$\rightarrow V(T_{D-3}, RF(T_{D-3})) = \text{Min}_{(0 \leq \alpha \leq 1)} \left\{ 1 - P(\hat{r}_{(T_{D-2}, \alpha)} > RF(T_{D-3}))^* \right. \\ \left. \left[\frac{P(\hat{r}_{(T_{D-1}, \tilde{\alpha})} > RF(T_{D-2}) \cap \hat{r}_{(T_D, \tilde{\alpha})} > RF(T_{D-1}) \cap \hat{r}_{(T_{D-2}, \alpha)} > RF(T_{D-3}))}{P(\hat{r}_{(T_{D-2}, \alpha)} > RF(T_{D-3}))} \right] \right\} \quad (C.6c)$$

Applying the definition of a conditional probability.

$$\rightarrow V(T_{D-3}, RF(T_{D-3})) = \text{Min}_{(0 \leq \alpha \leq 1)} \left\{ 1 - (1 - F_{\hat{r}_{(T_{D-2}, \alpha)}}(RF(T_{D-3})))^* \right. \\ \left. \frac{\int_{RF(T_{D-3})}^{\infty} \int_{RF(T_{D-2})}^{\infty} \int_{RF(T_{D-1})}^{\infty} f(\hat{r}_{(T_D, \tilde{\alpha})}, \hat{r}_{(T_{D-1}, \tilde{\alpha})}, \hat{r}_{(T_{D-2}, \alpha)}) d(\hat{r}_{(T_D, \tilde{\alpha})}) d(\hat{r}_{(T_{D-1}, \tilde{\alpha})}) d(\hat{r}_{(T_{D-2}, \alpha)})}{\int_{RF(T_{D-3})}^{\infty} f(\hat{r}_{(T_{D-2}, \alpha)}) d(\hat{r}_{(T_{D-2}, \alpha)})} \right\} \quad (C.6d)$$

The multivariate density of the next 3 real returns integrated over the condition of no ruin.

$$\rightarrow V(T_{D-3}, RF(T_{D-3})) = \text{Min}_{(0 \leq \alpha \leq 1)} \left\{ 1 - (1 - F_{\hat{r}_{(T_{D-2}, \alpha)}}(RF(T_{D-3})))^* \right. \\ \left. \frac{\int_{RF(T_{D-3})}^{\infty} f(\hat{r}_{(T_{D-2}, \alpha)}) \int_{RF(T_{D-2})}^{\infty} \int_{RF(T_{D-1})}^{\infty} f(\hat{r}_{(T_D, \tilde{\alpha})}, \hat{r}_{(T_{D-1}, \tilde{\alpha})}) d(\hat{r}_{(T_D, \tilde{\alpha})}) d(\hat{r}_{(T_{D-1}, \tilde{\alpha})}) d(\hat{r}_{(T_{D-2}, \alpha)})}{\int_{RF(T_{D-3})}^{\infty} f(\hat{r}_{(T_{D-2}, \alpha)}) d(\hat{r}_{(T_{D-2}, \alpha)})} \right\} \quad (C.6e)$$

By the assumption of independence.

$$\rightarrow V(T_{D-3}, RF(T_{D-3})) = \text{Min}_{(0 \leq \alpha \leq 1)} \left\{ 1 - (1 - F_{\hat{r}_{(T_{D-2}, \alpha)}}(RF(T_{D-3})))^* \right. \\ \left. \left[\frac{\int_{RF(T_{D-3})}^{\infty} f(\hat{r}_{(T_{D-2}, \alpha)}) [1 - V(T_{D-2}, RF(T_{D-2}))] d(\hat{r}_{(T_{D-2}, \alpha)})}{\int_{RF(T_{D-3})}^{\infty} f(\hat{r}_{(T_{D-2}, \alpha)}) d(\hat{r}_{(T_{D-2}, \alpha)})} \right] \right\} \quad (C.6f)$$

Note that the retiree has some control over the next ruin factor via choice of α at this time point.

Since $RF(T_{D-2})$ is a function of $\hat{r}_{(T_{D-2}, \alpha)}$, namely,

$$RF(T_{D-2}) = \frac{RF(T_{D-3})}{\hat{r}_{(T_{D-2}, \alpha)} - RF(T_{D-3})} \quad (C.6g)$$

the expression above is by definition the expected value of $[1 - V(T_{D-2}, RF(T_{D-2}))]$ over the random variable $\hat{r}_{(T_{D-2}, \alpha)}^+$, and the value function can be written as:

$$\begin{aligned}
& V(T_D-3, RF(T_D-3)) = \\
& \text{Min}_{(0 \leq \alpha \leq 1)} \left\{ \begin{array}{l} 1 - (1 - F_{\hat{f}_{(T_D-2, \alpha)}}(RF(T_D-3)))^* \\ (1 - E_{\hat{f}_{(T_D-2, \alpha)}}^+ \left[V(T_D - 2, \frac{RF(T_D-3)}{\hat{f}_{(T_D-2, \alpha)} - RF(T_D-3)}) \right]) \end{array} \right\} \quad (C.7)
\end{aligned}$$

$\boxed{RF(T_D-2)}$
 \downarrow
 \bullet

Optimality is achieved at $\alpha = \alpha(T_D-3, RF(T_D-3))$ and the expectation is over the conditional R.V.

$\hat{f}_{(T_D-2, \alpha)}^+ = (\hat{f}_{(T_D-2, \alpha)} | \hat{f}_{(T_D-2, \alpha)} > RF(T_D-3))$ where $\{\hat{f}_{(T_D-2, \alpha)} > RF(T_D-3)\} \equiv \{RF(T_D-2) > 0\}$.

(Continued on next page ...)

Appendix D: Derivation of Ruin Factor Bucket Probabilities

P(RF(t+1) in Bucket #1)

$$= P(0 < RF(t+1) \leq (1.5)*(1/P_R)) \quad (D.1a)$$

$$= P(0 < \frac{RF(t)}{\hat{r}_{(t+1,\alpha)} - RF(t)} \leq (1.5)*(1/P_R)) \quad (D.1b)$$

$$= P(\frac{1}{0} > \frac{\hat{r}_{(t+1,\alpha)} - RF(t)}{RF(t)} \geq \frac{1}{(1.5)*(1/P_R)}) \quad (D.1c)$$

$$= P(\infty > \hat{r}_{(t+1,\alpha)} \geq RF(t) * (1 + \frac{1}{(1.5)*(1/P_R)})) \quad (D.1d)$$

$$= P(RF(t) * (1 + \frac{1}{(1.5)*(1/P_R)}) \leq \hat{r}_{(t+1,\alpha)} < \infty) \quad (D.1e)$$

$$= \boxed{1 - F_{\hat{r}_{(t+1,\alpha)}}(RF(t) * (1 + \frac{1}{(1.5)*(1/P_R)}))} \quad (D.1f)$$

P(RF(t+1) in Bucket #i)
for $(2 \leq i \leq (P_R)*RF_{Max})$

$$= P(i/P_R - 1/(2P_R) < RF(t+1)) \leq i/P_R + 1/(2P_R)) \quad (D.2a)$$

$$= P((1/P_R)*(i - 1/2) < \frac{RF(t)}{\hat{r}_{(t+1,\alpha)} - RF(t)} \leq (1/P_R)*(i + 1/2)) \quad (D.2b)$$

$$= P(RF(t)*(1 + \frac{1}{(\frac{1}{P_R})*(i-\frac{1}{2})}) > \hat{r}_{(t+1,\alpha)} \geq RF(t)*(1 + \frac{1}{(\frac{1}{P_R})*(i+\frac{1}{2})})) \quad (D.2c)$$

$$= P(RF(t)*(1 + \frac{1}{(\frac{1}{P_R})*(i+\frac{1}{2})}) \leq \hat{r}_{(t+1,\alpha)} < RF(t)*(1 + \frac{1}{(\frac{1}{P_R})*(i-\frac{1}{2})})) \quad (D.2d)$$

$$= \boxed{F_{\hat{r}_{(t+1,\alpha)}}(RF(t)*(1 + \frac{1}{(\frac{1}{P_R})*(i-\frac{1}{2})})) - F_{\hat{r}_{(t+1,\alpha)}}(RF(t)*(1 + \frac{1}{(\frac{1}{P_R})*(i+\frac{1}{2})}))} \quad (D.2e)$$

P(RF(t+1) in
Bucket #(P_R)*RF_{Max}+1)

$$= P(RF_{Max} + 1/(2P_R) < RF(t+1) \leq \infty) \quad (D.3a)$$

$$= P(RF_{Max} + 1/(2P_R) < \frac{RF(t)}{\hat{r}_{(t+1,\alpha)} - RF(t)} \leq \infty) \quad (D.3b)$$

$$= P\left(\frac{1}{RF_{Max} + 1/(2P_R)} > \frac{\hat{r}_{(t+1,\alpha)} - RF(t)}{RF(t)} \geq \frac{1}{\infty}\right) \quad (D.3c)$$

$$= P(RF(t) * (1 + \frac{1}{RF_{Max} + 1/(2P_R)}) > \hat{r}_{(t+1,\alpha)} \geq RF(t)) \quad (D.3d)$$

$$= P(RF(t) \leq \hat{r}_{(t+1,\alpha)} < RF(t) * (1 + \frac{1}{RF_{Max} + 1/(2P_R)})) \quad (D.3e)$$

$$= \boxed{F_{\hat{r}_{(t+1,\alpha)}}(RF(t) * (1 + \frac{1}{RF_{Max} + 1/(2P_R)})) - F_{\hat{r}_{(t+1,\alpha)}}(RF(t))} \quad (D.3f)$$

Lastly, in (26) these probabilities are conditional on $RF(t+1) > 0 \leftrightarrow \hat{r}_{(t+1,\alpha)} > RF(t)$, therefore each must be divided by $P(\hat{r}_{(t+1,\alpha)} > RF(t)) = 1 - F_{\hat{r}_{(t+1,\alpha)}}(RF(t))$ for their sum to equal 1. What has been derived is the conditional probability that the next ruin factor is in any of the pre-defined buckets, given that no ruin occurs at the next time point.

(Continued on next page ...)

Appendix E: Limiting Behavior of $V_d(t, RF(t))$ as # Buckets Increases

Note that each probability in the conditional PMF of (26) is a difference of CDF values at the bucket limits with the discrete ruin factor centering each bucket.¹⁸ Consider all discrete ruin factors, $x \in [1/(2P_R), RF_{\text{Max}}+1/(2P_R)] = [a, b]$, which is a closed and bounded region. Each CDF difference takes the form,

$$F(x+\delta/2) - F(x-\delta/2) \quad (\text{E.1})$$

where the x 's are equidistant and δ is the distance between them. As $(\# \text{ of } x) \rightarrow \infty$, $\delta \rightarrow 0$. In the summation, these probabilities are multiplied by another function of the midpoints, say $g(x)$ (representing the value function at time $t+1$). The summation in $V_d(\cdot)$ thus has the form,

$$\sum_{x \in [a,b]} [F(x + \delta/2) - F(x - \delta/2)] * g(x) \quad (\text{E.2})$$

Note that the difference of CDFs, $F(x)$, represents an area under the PDF, $f(x)$, having width δ and can be approximated by a rectangle of height $f(x)$, where x is the midpoint. That is,

$$F(x + \delta/2) - F(x - \delta/2) \approx \delta * f(x) \quad (\text{E.3})$$

$$\rightarrow \sum_{x \in [a,b]} [F(x + \frac{\delta}{2}) - F(x - \frac{\delta}{2})] * g(x) \approx \sum_{x \in [a,b]} \delta * f(x) * g(x), \quad (\text{E.4})$$

where the last term represents a Riemann sum. Taking the limit as the # of midpoints (or buckets) approaches infinity yields,

$$\lim_{(\# \text{ of } x) \rightarrow \infty} \sum_{x \in [a,b]} \left[F\left(x + \frac{\delta}{2}\right) - F\left(x - \frac{\delta}{2}\right) \right] * g(x) = \lim_{(\# \text{ of } x) \rightarrow \infty} \sum_{x \in [a,b]} \delta * f(x) * g(x) \quad (\text{E.5a})$$

By definition of a definite integral.

$\bullet = \int_a^b f(x) * g(x) dx \quad (\text{E.5b})$

$$= E_x[g(x)], \quad (\text{E.5c})$$

provided there is no area to the right of b under the density function $f(x)$. The point is to select

RF_{Max} large enough to ensure this is the case for all α .

¹⁸ The first probability included the segment $(0, 1/(2P_R))$ to make the approximation more accurate, but as $P_R \rightarrow \infty$ this is not necessary, and the first probability could also be expressed as the difference of CDF values.

Appendix F: Derivation of Historical Stock and Bond Distributions

The hypothesis that real total stock and bond returns originate from a normal distribution cannot be rejected (Anderson-Darling p-values 0.707 and 0.243 respectively). Other distributions provide an acceptable fit, such as the lognormal distribution, however only using a non-zero location parameter in the case of real stock returns. The unconditional underlying probability distributions for real total returns at year t will thus be taken as, $N(\mu, \sigma)$:

$$\text{S\&P 500 Returns} = r_{(s,t)} \sim N(0.0825, 0.2007) \quad (\text{F.1a})$$

$$\text{10 year T-Bond Returns} = r_{(b,t)} \sim N(0.0214, 0.0834) \quad (\text{F.1b})$$

In addition, a small positive correlation of $\rho = 0.04387$ was measured between real stock and bond returns at the same time point, which will be carried through the analysis. As noted, these are unconditional distributions and would roughly trace out the shape of corresponding histograms plotted using these returns. If the returns from either investment exhibit serial correlation they cannot be treated as random samples from their unconditional distributions. It is well known, for example, that the inflation rate exhibits strong serial correlation.

To determine the nature of serial correlation within real stock and bond returns, it is first assumed that the average returns for each is stable, or that the processes are stationary. The overall mean for each process is estimated by the sample average, which is subtracted from the returns leaving them as deviations from their respective means. A stationary process that is autoregressive of order p , denoted $AR(p)$, is one that explicitly models the current observation as a linear function of the past p observations, under the assumption that they are correlated. The general $AR(p)$ model is defined as (Box, et al.),

$$Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \dots + \phi_p Y_{t-p} + a_t \quad (\text{F.2})$$

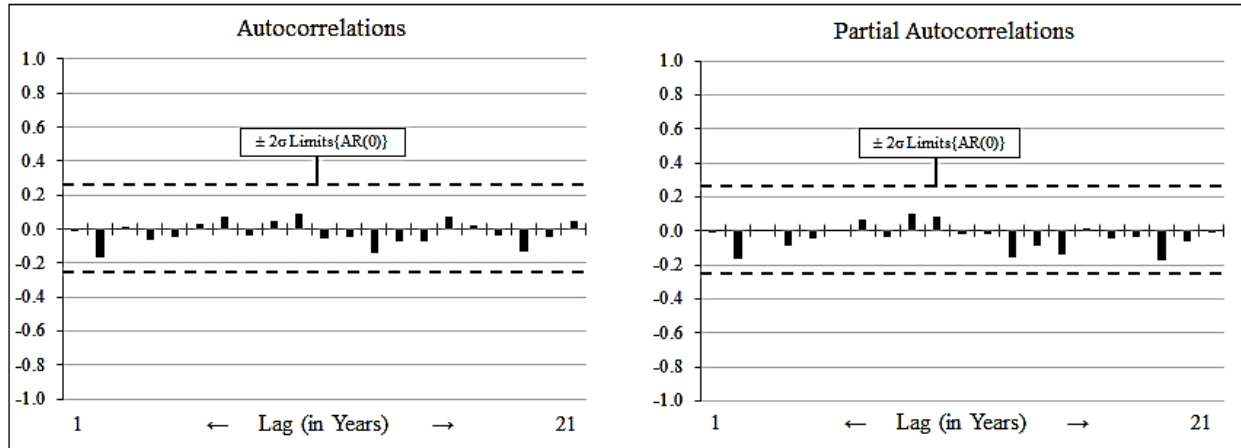
where Y_t reflects the centered observation at time t and $a_t \sim N(0, \sigma_a^2)$ are *iid* error terms.

Note that the Y 's are assumed correlated despite the fact that the error terms are *iid*. This is because, for instance, Y_t and Y_{t+1} are both functions of the same random error term a_t . To determine the autoregressive order of a stationary process the autocorrelations and partial autocorrelations are examined. The autocorrelations at lag k , are estimated by the sample correlations of all observations k time points apart and the partial autocorrelations at lag k reflect the correlation of observations k time points apart, after accounting for the correlation at lags $< k$. If an $AR(k)$ model is fit to the data, then the estimate of ϕ_k is also an estimate of the k -th partial autocorrelation. It is appropriate to examine both of these statistics up to lag $N/4$, where N reflects the number of data points collected ($N=86$, Box, et al.). Two important findings are that both the autocorrelations and partial autocorrelations at lags $> p$ are approximately $N(0, \sqrt{1/N})$ in an $AR(p)$ process for large N (Box, et al.). Since approximately 95% of the data in a normal distribution are within 2 standard deviations of the mean, this result can be used to assess the lag at which serial correlation in a stationary process ends (Box, et al.).

The autocorrelation estimates for inflation-adjusted stock returns are shown below on the left side of Figure E.1. None of the 21 autocorrelations exceed the $2\sqrt{1/N}$ ($= 0.21567$) threshold, represented by the dashed line. This indicates that an $AR(0)$ model is appropriate. The partial autocorrelations were estimated by fitting successive $AR(k)$ models, for $k=1, 2, \dots, 21$, where the ϕ -estimates for each model were taken as solutions to the Yule-Walker equations (Box, et al.). The partial autocorrelation estimates are shown on the right side of Figure E.1, and confirm the finding that real stock returns do not exhibit serial correlation. Since the autocorrelation estimates shown (not partial) and their large sample distribution apply to any linear stationary process, including a moving average $MA(q)$, or mixed autoregressive-moving average $ARMA(p, q)$ process, it is concluded that these returns originate as completely random

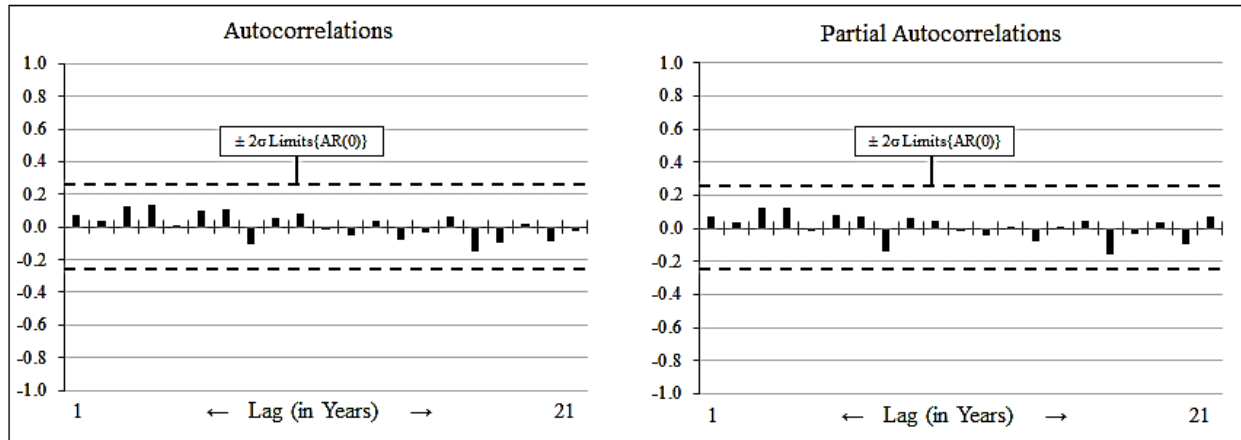
samples. Note that the autocorrelations tail off after p in an $AR(p)$ process, but drop off abruptly after q in an $MA(q)$ process (Box, et al.).

Figure E.1. Analysis of Serial Correlation in Real Stock Returns



A similar analysis was performed on real bond returns and the results are shown in Figure E.2 below. None of the 21 autocorrelations or partial autocorrelations exceed the $2\sqrt{1/N}$ ($= 0.21567$) tolerance limits indicating that an $AR(0)$ process is appropriate for these returns. Again, since the autocorrelation estimates and their large sample distribution apply to any linear stationary process, it is concluded that these returns also originate as completely random samples. Note that these findings support the theory that stock and bond markets are efficient (at least in the weak sense), meaning that prices adjust for any predictive capacity inherent in historical patterns, thereby instantly removing it. Lastly, it will be assumed that real stock and bond returns jointly follow a bivariate normal distribution.

Figure E.2. Analysis of Serial Correlation in Real Bond Returns



The stationarity assumption above is typically tested after fitting an AR(p) model to the data. The fitted model defines a characteristic polynomial and the stationarity hypothesis is accepted if all roots of this polynomial have magnitude > 1 . (Complex roots would appear as conjugate pairs, $a \pm bi$, with magnitude $\sqrt{a^2 + b^2}$.) Since an AR(0) process was found appropriate for both returns, the stationarity test was not performed. Note that MA(q) models are stationary by construct, and ARMA(p, q) models are stationary when their autoregressive component meets the condition just stated (Box, et al.). Standard regression models were also fit and indicate that the average real return for stocks and bonds does not change as a function of time. This supports the finding that real stock and bond returns originate completely at random, from their underlying identical distributions.

Due to the nature of compounding, investors generally do not experience the arithmetic mean return of a security over time. The geometric mean is a more accurate reflection of the compounded average performance. Based on the 86 years of historical data used here, the geometric means of inflation-adjusted stock and bond returns are 0.063 and 0.018, respectively. Eighty-six years of data were simulated (1 million times each) using the distributions for stocks

and bonds given in (F.1a) and (F.1b) and the average geometric mean values were 0.063 and 0.018, respectively, matching their historical counterparts.¹⁹

At each stage/state of the discrete DP (see Figure 5) the minimum probability of ruin is found over all asset allocations in the set $\alpha_{\{\cdot\}}$. Let $R_{(s,t)}$ and $R_{(b,t)}$ denote the actual total returns (not inflation-adjusted) at year t , for stocks and bonds respectively. A portfolio with stock and bond proportions α , and $(1-\alpha)$ thus has a total return represented by the R.V.,

$$R_{(t,\alpha)} = \alpha * R_{(s,t)} + (1-\alpha) * R_{(b,t)}, \quad (\text{F.3a})$$

with compounded return,

$$1 + R_{(t,\alpha)} = 1 + [\alpha * R_{(s,t)} + (1-\alpha) * R_{(b,t)}] \quad (\text{F.3b})$$

After adjusting for inflation, the real return for this portfolio at year t is given by the R.V.:

$$1 + r_{(t,\alpha)} | I_t, R_{(t,\alpha)} = [1 + R_{(t,\alpha)}] / (1 + I_t) = \{1 + [\alpha * R_{(s,t)} + (1-\alpha) * R_{(b,t)}]\} / (1 + I_t) \quad (\text{F.4a})$$

$$\rightarrow 1 + r_{(t,\alpha)} | I_t, R_{(t,\alpha)} = \{\alpha * [1 + R_{(s,t)}] + (1-\alpha) * [1 + R_{(b,t)}]\} / (1 + I_t) \quad (\text{F.4b})$$

$$\rightarrow 1 + r_{(t,\alpha)} | I_t, R_{(t,\alpha)} = \alpha * [1 + R_{(s,t)}] / (1 + I_t) + (1-\alpha) * [1 + R_{(b,t)}] / (1 + I_t) \quad (\text{F.4c})$$

$$\rightarrow 1 + r_{(t,\alpha)} | I_t, R_{(t,\alpha)} = \alpha * (1 + r_{(s,t)}) + (1-\alpha) * (1 + r_{(b,t)}) \quad (\text{F.4d})$$

$$\rightarrow 1 + r_{(t,\alpha)} | I_t, R_{(t,\alpha)} = \alpha * N(1.0825, 0.2007) + (1-\alpha) * N(1.0214, 0.0834) \quad (\text{F.4e})$$

$$\rightarrow 1 + r_{(t,\alpha)} | I_t, R_{(t,\alpha)} \sim N(\alpha * (1.0825) + (1-\alpha) * (1.0214), \quad (\text{F.4f})$$

$$\sqrt{\alpha^2 * (0.2007)^2 + (1 - \alpha)^2 * (0.0834)^2 + 2 * \alpha * (1 - \alpha) * (.00073)}),$$

where in general,

$$\text{Var}(\alpha X_1 + (1 - \alpha) X_2) = \alpha^2 \text{Var}(X_1) + (1-\alpha)^2 \text{Var}(X_2) + 2\alpha(1 - \alpha) \text{Cov}(X_1, X_2) \quad (\text{F.4g})$$

and,

$$\text{Cov}(X_1, X_2) = \rho * \sqrt{\text{Var}(X_1) \text{Var}(X_2)}, \quad (\text{F.4h})$$

¹⁹ Both negative and positive outliers were removed in the same relative proportions to avoid square roots of negative numbers. (About 1 outlier per 70,000 simulated histories was removed.)

holds for any two random variables X_1 and X_2 . Finally, assuming an expense ratio of $E_R = 0.5\%$ ($1 - E_R = 0.995$) is paid to the financial institution yearly, the inflation/expense-adjusted return at year t is given by the R.V.,

$$\hat{r}_{(t,\alpha)} = (1+r_{(t,\alpha)})(1-E_R) \sim N((0.995)*[\alpha*(1.0825) + (1-\alpha)*(1.0214)], \quad (F.5)$$

$$(0.995) * \sqrt{\alpha^2 * (0.2007)^2 + (1 - \alpha)^2 * (0.0834)^2 + 2 * \alpha * (1 - \alpha) * (.00073)}).$$

(Continued on next page ...)

Appendix G: Induction at Times $S_{\text{Max}-2}$ and $S_{\text{Max}-3}$ for Random T_D

G.1 Induction at Time $t=S_{\text{Max}} - 2$

Assume the retiree arrives at time $t=S_{\text{Max}-2}$, makes their withdrawal and has at most two remaining. $\text{RF}(S_{\text{Max}-2}) (> 0)$ is calculated based on the portfolio return just observed, $\hat{r}_{(S_{\text{Max}-2}, \alpha)}$. The retiree seeks α to minimize $P(\text{Ruin}) = P(\text{Ruin}(S_{\text{Max}-1}) \cup \text{Ruin}(S_{\text{Max}}))$, which for a given α can be expressed as: (Note: $t=S_{\text{Max}}$, $S_{\text{Max}-1}$ derived in Sections 4.6.1.1 & 4.6.1.2, respectively.)

$$P(\text{Ruin})$$

$$= P(\text{Ruin}(S_{\text{Max}-1}) \cup \text{Ruin}(S_{\text{Max}})) \quad (\text{G.1a})$$

$$= 1 - P(\text{Ruin}^C(S_{\text{Max}-1}) \cap \text{Ruin}^C(S_{\text{Max}})) \quad (\text{G.1b})$$

$$= 1 - \left\{ \begin{aligned} &P(T_D=S_{\text{Max}-2} \mid T_D \geq S_{\text{Max}-2}) * (1) \\ &+ P(T_D > S_{\text{Max}-2} \mid T_D \geq S_{\text{Max}-2}) * P(\hat{r}_{(S_{\text{Max}-1}, \alpha)} > \text{RF}(S_{\text{Max}-2}) \cap \text{Ruin}^C(S_{\text{Max}})) \end{aligned} \right\} \quad (\text{G.1c})$$

$$= P(T_D > S_{\text{Max}-2} \mid T_D \geq S_{\text{Max}-2}) *$$

$$[1 - P(\hat{r}_{(S_{\text{Max}-1}, \alpha)} > \text{RF}(S_{\text{Max}-2})) * P(\text{Ruin}^C(S_{\text{Max}}) \mid \hat{r}_{(S_{\text{Max}-1}, \alpha)} > \text{RF}(S_{\text{Max}-2}))] \quad (\text{G.1d})$$

$$P(\text{Ruin}^C(S_{\text{Max}}) \mid \hat{r}_{(S_{\text{Max}-1}, \alpha)} > \text{RF}(S_{\text{Max}-2}))$$

$$= \left[\frac{1}{P(\hat{r}_{(S_{\text{Max}-1}, \alpha)} > \text{RF}(S_{\text{Max}-2}))} \right] * P(\hat{r}_{(S_{\text{Max}-1}, \alpha)} > \text{RF}(S_{\text{Max}-2}) \cap \text{Ruin}^C(S_{\text{Max}})) \quad (\text{G.1e})$$

$$P(\hat{r}_{(S_{\text{Max}-1}, \alpha)} > \text{RF}(S_{\text{Max}-2}) \cap \text{Ruin}^C(S_{\text{Max}}))$$

These 2 events are independent since $\hat{r}_{(S_{\text{Max}-1}, \alpha)}$ and T_D are independent R.V.s.

$$= P[(\hat{r}_{(S_{\text{Max}-1}, \alpha)} > \text{RF}(S_{\text{Max}-2}) \cap (T_D=S_{\text{Max}-1} \mid T_D > S_{\text{Max}-2}))$$

These 2 events are mutually exclusive.

\cup

$$(\hat{r}_{(S_{\text{Max}-1}, \alpha)} > \text{RF}(S_{\text{Max}-2}) \cap (T_D=S_{\text{Max}} \mid T_D > S_{\text{Max}-2}) \cap \hat{r}_{(S_{\text{Max}}, \tilde{\alpha})} > \text{RF}(S_{\text{Max}-1})) \quad (\text{G.1f})$$

These 2 events are NOT independent despite the fact $\hat{r}_{(S_{\text{Max}-1}, \alpha)}$ & $\hat{r}_{(S_{\text{Max}}, \tilde{\alpha})}$ are ind. R.V.s since $\text{RF}(S_{\text{Max}-1})$ is a function of $\hat{r}_{(S_{\text{Max}-1}, \alpha)}$.

The event of making the withdrawal at $S_{\text{Max}-1}$ and also avoiding ruin at S_{Max} can happen 2 ways: Make the withdrawal at $S_{\text{Max}-1}$ and experience death before S_{Max} , or make the withdrawal at $S_{\text{Max}-1}$, then again at S_{Max} . Note that all this unfolds given $T_D > S_{\text{Max}-2} \equiv T_D \geq S_{\text{Max}-1}$, see (G.1c).

As noted, the intersection of events over T_D and $(\hat{r}_{(S_{Max}-1,\alpha)}, \hat{r}_{(S_{Max},\tilde{\alpha})})$ are independent since random market returns are independent of the retiree's random time of final withdrawal. Also, the union of events requiring $T_D=S_{Max}-1$ and $T_D=S_{Max}$ are mutually exclusive since both cannot occur simultaneously. Finally, the events $\hat{r}_{(S_{Max}-1,\alpha)} > RF(S_{Max}-2)$ and $\hat{r}_{(S_{Max},\tilde{\alpha})} > RF(S_{Max}-1)$ are not independent since $RF(S_{Max}-1) = RF(S_{Max}-2) / [\hat{r}_{(S_{Max}-1,\alpha)} - RF(S_{Max}-2)]$. The probability statement in (G.1f) can thus be written as:

$$P(\hat{r}_{(S_{Max}-1,\alpha)} > RF(S_{Max}-2) \cap \text{Ruin}^C(S_{Max}))$$

$$= P(\hat{r}_{(S_{Max}-1,\alpha)} > RF(S_{Max}-2)) * P(T_D=S_{Max}-1 \mid T_D \geq S_{Max}-1) \quad (G.1g)$$

$$+ P(\hat{r}_{(S_{Max}-1,\alpha)} > RF(S_{Max}-2) \cap \hat{r}_{(S_{Max},\tilde{\alpha})} > RF(S_{Max}-1)) * P(T_D=S_{Max} \mid T_D \geq S_{Max}-1)$$

$$= P(T_D=S_{Max}-1 \mid T_D \geq S_{Max}-1) * \quad (G.1h)$$

$$\int_{RF(S_{Max}-2)}^{\infty} f(\hat{r}_{(S_{Max}-1,\alpha)}) d(\hat{r}_{(S_{Max}-1,\alpha)})$$

$$+ P(T_D=S_{Max} \mid T_D \geq S_{Max}-1) *$$

$$\int_{RF(S_{Max}-2)}^{\infty} \int_{RF(S_{Max}-1)}^{\infty} f(\hat{r}_{(S_{Max}-1,\alpha)}, \hat{r}_{(S_{Max},\tilde{\alpha})}) d(\hat{r}_{(S_{Max},\tilde{\alpha})}) d(\hat{r}_{(S_{Max}-1,\alpha)})$$

$$= \int_{RF(S_{Max}-2)}^{\infty} f(\hat{r}_{(S_{Max}-1,\alpha)}) * \left[\right] d(\hat{r}_{(S_{Max}-1,\alpha)}) \quad (G.1i)$$

$$P(T_D=S_{Max}-1 \mid T_D \geq S_{Max}-1) + P(T_D=S_{Max} \mid T_D \geq S_{Max}-1) * \int_{RF(S_{Max}-1)}^{\infty} f(\hat{r}_{(S_{Max},\tilde{\alpha})}) d(\hat{r}_{(S_{Max},\tilde{\alpha})})$$

If the optimal $\alpha=\tilde{\alpha}$ is used at $t=S_{Max}-1$ then this term is precisely $\text{Min}_{\alpha}[P(\text{Ruin}^C(S_{Max}))]$ given the retiree arrives at $t=S_{Max}-1$ and successfully makes the withdrawal, which by definition is $1 - V_R(S_{Max}-1, RF(S_{Max}-1))$, see (36c). It will be shown below that $\alpha=\tilde{\alpha}$ must be used at $t=S_{Max}-1$ to minimize $V_R(S_{Max}-2, RF(S_{Max}-2))$ at $t=S_{Max}-1$.

Substituting the term from (G.1i) back into (G.1e) reveals it is nothing more than the expectation of $[1 - V_R(S_{Max}-1, RF(S_{Max}-1))]$ over the conditional R.V. $\hat{r}_{(S_{Max}-1,\alpha)}^+$. Thus, if using $\alpha=\tilde{\alpha}$ at $t=S_{Max}-1$, the original probability from (G.1a) for a given α at $t=S_{Max}-2$ is:

P(Ruin)

$$= P(T_D > S_{\text{Max}-2} \mid T_D \geq S_{\text{Max}-2})^*$$

$$[1 - P(\hat{f}_{(S_{\text{Max}-1}, \alpha)} > \text{RF}(S_{\text{Max}-2})) * (1 - E\hat{f}_{(S_{\text{Max}-1}, \alpha)}^+ [V_R(S_{\text{Max}} - 1, \frac{\text{RF}(S_{\text{Max}-2})}{\hat{f}_{(S_{\text{Max}-1}, \alpha)} - \text{RF}(S_{\text{Max}-2})})])] \quad (\text{G.1j})$$

For P(Ruin) to be minimized over all α at $t=S_{\text{Max}-2}$, the function in $[\cdot]$ must take its minimum value at each $\text{RF}(S_{\text{Max}-1})$, which occurs precisely at $V_R(S_{\text{Max}-1}, \text{RF}(S_{\text{Max}-1}))$.

Finally, the value function at $t=S_{\text{Max}-2}$ is defined as:

$$V_R(S_{\text{Max}-2}, \text{RF}(S_{\text{Max}-2})) =$$

$$\text{Min}_{(0 \leq \alpha \leq 1)} \left\{ P(T_D > S_{\text{Max}-2} \mid T_D \geq S_{\text{Max}-2})^* \right. \\ \left. \left\{ 1 - (1 - F\hat{f}_{(S_{\text{Max}-1}, \alpha)}(\text{RF}(S_{\text{Max}-2}))) * (1 - E\hat{f}_{(S_{\text{Max}-1}, \alpha)}^+[V_R(S_{\text{Max}-1}, \text{RF}(S_{\text{Max}-1}))]) \right\} \right\} \quad (\text{G.1k})$$

G.2 Induction at Time $t=S_{\text{Max}} - 3$

Assume the retiree arrives at time $t=S_{\text{Max}-3}$, makes their withdrawal (at most three remain), and updates $\text{RF}(S_{\text{Max}-3}) (> 0)$ based on the return just observed, $\hat{f}_{(S_{\text{Max}-3}, \alpha)}$. The retiree seeks α to minimize $P(\text{Ruin}) = P(\text{Ruin}(S_{\text{Max}-2}) \cup P(\text{Ruin}(S_{\text{Max}-1}) \cup \text{Ruin}(S_{\text{Max}})))$, which for a given α can be expressed as:

P(Ruin)

$$= P(\text{Ruin}(S_{\text{Max}-2}) \cup \text{Ruin}(S_{\text{Max}-1}) \cup \text{Ruin}(S_{\text{Max}})) \quad (\text{G.2a})$$

$$= 1 - P(\text{Ruin}^C(S_{\text{Max}-2}) \cap \text{Ruin}^C(S_{\text{Max}-1}) \cap \text{Ruin}^C(S_{\text{Max}})) \quad (\text{G.2b})$$

$$= 1 - \left\{ \begin{aligned} &P(T_D = S_{\text{Max}-3} \mid T_D \geq S_{\text{Max}-3})^*(1) \\ &+ P(T_D > S_{\text{Max}-3} \mid T_D \geq S_{\text{Max}-3})^* \\ &P(\hat{f}_{(S_{\text{Max}-2}, \alpha)} > \text{RF}(S_{\text{Max}-3}) \cap \text{Ruin}^C(S_{\text{Max}-1}) \cap \text{Ruin}^C(S_{\text{Max}})) \end{aligned} \right\} \quad (\text{G.2c})$$

$$= P(T_D > S_{\text{Max}-3} \mid T_D \geq S_{\text{Max}-3})^* \quad (\text{G.2d})$$

$$[1 - P(\hat{r}_{(S_{\text{Max}-2}, \alpha)} > \text{RF}(S_{\text{Max}-3})) * P(\text{Ruin}^C(S_{\text{Max}-1}) \cap \text{Ruin}^C(S_{\text{Max}}) \mid \hat{r}_{(S_{\text{Max}-2}, \alpha)} > \text{RF}(S_{\text{Max}-3}))]$$

$$P(\text{Ruin}^C(S_{\text{Max}-1}) \cap \text{Ruin}^C(S_{\text{Max}}) \mid \hat{r}_{(S_{\text{Max}-2}, \alpha)} > \text{RF}(S_{\text{Max}-3}))$$

$$= \left[\frac{1}{P(\hat{r}_{(S_{\text{Max}-2}, \alpha)} > \text{RF}(S_{\text{Max}-3}))} \right] * P(\hat{r}_{(S_{\text{Max}-2}, \alpha)} > \text{RF}(S_{\text{Max}-3}) \cap \text{Ruin}^C(S_{\text{Max}-1}) \cap \text{Ruin}^C(S_{\text{Max}})) \quad (\text{G.2e})$$

$$P(\hat{r}_{(S_{\text{Max}-2}, \alpha)} > \text{RF}(S_{\text{Max}-3}) \cap \text{Ruin}^C(S_{\text{Max}-1}) \cap \text{Ruin}^C(S_{\text{Max}}))$$

$$= P[(\hat{r}_{(S_{\text{Max}-2}, \alpha)} > \text{RF}(S_{\text{Max}-3}) \cap (T_D = S_{\text{Max}-2} \mid T_D \geq S_{\text{Max}-2}))$$

U

$$(\hat{r}_{(S_{\text{Max}-2}, \alpha)} > \text{RF}(S_{\text{Max}-3}) \cap (T_D = S_{\text{Max}-1} \mid T_D \geq S_{\text{Max}-2}) \cap \hat{r}_{(S_{\text{Max}-1}, \tilde{\alpha})} > \text{RF}(S_{\text{Max}-2}))$$

U

$$(\hat{r}_{(S_{\text{Max}-2}, \alpha)} > \text{RF}(S_{\text{Max}-3}) \cap (T_D = S_{\text{Max}} \mid T_D \geq S_{\text{Max}-2}) \cap \hat{r}_{(S_{\text{Max}-1}, \tilde{\alpha})} > \text{RF}(S_{\text{Max}-2}) \cap \hat{r}_{(S_{\text{Max}}, \tilde{\alpha})} > \text{RF}(S_{\text{Max}-1}))$$

Given $T_D > S_{\text{Max}-3}$, there are 3 mutually exclusive ways this intersection of events can occur:

Invoking the independence of T_D and $\hat{r}(\cdot)$, (G.2f) can be written as:

$$= P(T_D = S_{\text{Max}-2} \mid T_D \geq S_{\text{Max}-2}) * P(\hat{r}_{(S_{\text{Max}-2}, \alpha)} > \text{RF}(S_{\text{Max}-3}))$$

$$+ P(T_D = S_{\text{Max}-1} \mid T_D \geq S_{\text{Max}-2}) * P(\hat{r}_{(S_{\text{Max}-2}, \alpha)} > \text{RF}(S_{\text{Max}-3}) \cap \hat{r}_{(S_{\text{Max}-1}, \tilde{\alpha})} > \text{RF}(S_{\text{Max}-2})) \quad (\text{G.2g})$$

$$+ P(T_D = S_{\text{Max}} \mid T_D \geq S_{\text{Max}-2}) * P(\hat{r}_{(S_{\text{Max}-2}, \alpha)} > \text{RF}(S_{\text{Max}-3}) \cap \hat{r}_{(S_{\text{Max}-1}, \tilde{\alpha})} > \text{RF}(S_{\text{Max}-2}) \cap \hat{r}_{(S_{\text{Max}}, \tilde{\alpha})} > \text{RF}(S_{\text{Max}-1}))$$

$$= P(T_D = S_{\text{Max}-2} \mid T_D \geq S_{\text{Max}-2})^*$$

$$\int_{\text{RF}(S_{\text{Max}-3})}^{\infty} f(\hat{r}_{(S_{\text{Max}-2}, \alpha)}) d(\hat{r}_{(S_{\text{Max}-2}, \alpha)})$$

$$+ P(T_D = S_{\text{Max}-1} \mid T_D \geq S_{\text{Max}-2})^*$$

Independent R.V.s

$$\int_{\text{RF}(S_{\text{Max}-3})}^{\infty} \int_{\text{RF}(S_{\text{Max}-2})}^{\infty} f(\hat{r}_{(S_{\text{Max}-2}, \alpha)}, \hat{r}_{(S_{\text{Max}-1}, \tilde{\alpha})}) d(\hat{r}_{(S_{\text{Max}-1}, \tilde{\alpha})}) d(\hat{r}_{(S_{\text{Max}-2}, \alpha)}) \quad (\text{G.2h})$$

$$+ P(T_D = S_{\text{Max}} \mid T_D \geq S_{\text{Max}-2})^*$$

Independent R.V.s

$$\int_{\text{RF}(S_{\text{Max}-3})}^{\infty} \int_{\text{RF}(S_{\text{Max}-2})}^{\infty} \int_{\text{RF}(S_{\text{Max}-1})}^{\infty} f(\hat{r}_{(S_{\text{Max}}, \tilde{\alpha})}, \hat{r}_{(S_{\text{Max}-1}, \tilde{\alpha})}, \hat{r}_{(S_{\text{Max}-2}, \alpha)}) d(\hat{r}_{(S_{\text{Max}}, \tilde{\alpha})}) d(\hat{r}_{(S_{\text{Max}-1}, \tilde{\alpha})}) d(\hat{r}_{(S_{\text{Max}-2}, \alpha)})$$

Since the sum of integrals with respect to $\hat{r}_{(S_{\text{Max}-2}, \alpha)}$ is the integral of the sum, (G.2h) becomes:

$$= \int_{RF(S_{Max-3})}^{\infty} f(\hat{r}_{(S_{Max-2}, \alpha)}) * \left[\right] d(\hat{r}_{(S_{Max-2}, \alpha)}) \quad (G.2i)$$

$$\begin{aligned} & \left[P(T_D = S_{Max-2} \mid T_D \geq S_{Max-2}) \right. \\ & + P(T_D = S_{Max-1} \mid T_D \geq S_{Max-2}) * \int_{RF(S_{Max-2})}^{\infty} f(\hat{r}_{(S_{Max-1}, \tilde{\alpha})}) d(\hat{r}_{(S_{Max-1}, \tilde{\alpha})}) \\ & + P(T_D = S_{Max} \mid T_D \geq S_{Max-2}) * \\ & \left. \int_{RF(S_{Max-2})}^{\infty} \int_{RF(S_{Max-1})}^{\infty} f(\hat{r}_{(S_{Max}, \tilde{\alpha})}, \hat{r}_{(S_{Max-1}, \tilde{\alpha})}) d(\hat{r}_{(S_{Max}, \tilde{\alpha})}) d(\hat{r}_{(S_{Max-1}, \tilde{\alpha})}) \right] \end{aligned}$$

This entire term is recognized as $\text{Min}_{\alpha}[1 - P(\text{Ruin})]$, (see G.1c) given that the retiree arrives at $t=S_{Max-2}$ and successfully makes the withdrawal, which is by definition $1-V_R(S_{Max-2}, RF(S_{Max-2}))$. Following an optimal policy at future time points is required to achieve a minimum for $V_R(S_{Max-3}, RF(S_{Max-3}))$ at $t= S_{Max-3}$.

Substituting (G.2i) back into the RHS of (G.2e) reveals it is the expected value of $1-V_R(S_{Max-2}, RF(S_{Max-2}))$ with respect to the R.V. $\hat{r}_{(S_{Max-2}, \alpha)}^+$. This leaves (G.2d) as:

$P(\text{Ruin})$

$$\begin{aligned} &= P(T_D > S_{Max-3} \mid T_D \geq S_{Max-3}) * \\ & \left[1 - P(\hat{r}_{(S_{Max-2}, \alpha)} > RF(S_{Max-3})) * (1 - E\hat{r}_{(S_{Max-2}, \alpha)}^+ \left[V_R(S_{Max-2}, \frac{RF(S_{Max-3})}{\hat{r}_{(S_{Max-2}, \alpha)} - RF(S_{Max-3})}) \right]) \right] \quad (G.2j) \end{aligned}$$

The value function $V_R(S_{Max-3}, RF(S_{Max-3}))$ is the minimum of (G.2j) over $0 \leq \alpha \leq 1$, namely:

$$V_R(S_{Max-3}, RF(S_{Max-3})) =$$

$$\text{Min}_{(0 \leq \alpha \leq 1)} \left\{ P(T_D > S_{Max-3} \mid T_D \geq S_{Max-3}) * \left\{ 1 - (1 - F\hat{r}_{(S_{Max-2}, \alpha)}(RF(S_{Max-3}))) * (1 - E\hat{r}_{(S_{Max-2}, \alpha)}^+ [V_R(S_{Max-2}, RF(S_{Max-2}))]) \right\} \right\} \quad (G.2k)$$

Appendix H: Full C++ Implementation

The implementation consists of 5 functions and one header that are compiled into a 64-bit console application (executable). The application is launched by either double-clicking the executable or by invoking it from the command line or via a batch file. A single parameter is optionally accepted which is the number of tasks the user wants to process concurrently as it runs. If no value is supplied the code determines the maximum number of concurrent processing units on the machine running it and uses this value. The maximum number of concurrent processing units is generally the number of CPU cores. If an Intel® processor with Hyper-Threading® (HT) is used the capacity of each core doubles. For example, the runtime statistics given next are from a Dell® Precision® workstation with dual 10-core 2.5GHz Intel® Xeon® processors. Since all Xeon® processors use HT this PC can run a maximum of $2 \times 10 \times 2 = 40$ jobs concurrently. The application is multi-threaded and the task of processing each time point is split as follows. First the code approximates the RF(t) bucket at which heavy algorithm pruning begins. It then splits the remaining buckets into equal-sized collections and launches jobs to process all buckets simultaneously in separate threads. Once all threads for a given time point finish the results are aggregated and processing for the next time point begins.

When launched the application queries the user for the location of a directory that contains a control file and optionally a file of age probabilities (required for random T_D). The control file defaults to filename “control.txt” and the age probabilities file to “ageprobs.txt”. Both filenames can be changed in the header if desired. Examples of 6 control files (including runtime statistics) and the age probability file (derived from SSA.gov) are shown next. This application uses the C++ boost[©] library which will need to be installed (including binaries) on the machine creating the executable and it was optimized for speed when compiled using Microsoft® Visual Studio® (a project level setting).

Input File: control.txt

Contents: (A 3-line text file of space-delimited values as shown below.)

StockMean StockVar BondMean BondVar StockBondCov RF_{Max} E_R PrunePwr
 P_R P_α
NumRand Details

Example 1: (Fixed $T_D=50$, $P_R=1,000$, $P_\alpha=100$, $RF_{Max}=2.75$, $E_R=0.0\%$, Runtime=00:00:43)

0.082509 0.0402696529 0.021409 0.0069605649 0.0007344180 2.75 0.000 4.00
1000 100
0 50

Example 2: (Fixed $T_D=50$, $P_R=5,000$, $P_\alpha=1,000$, $RF_{Max}=2.75$, $E_R=0.5\%$, Runtime=00:28:02)

0.082509 0.0402696529 0.021409 0.0069605649 0.0007344180 2.75 0.005 4.00
5000 1000
0 50

Example 3: (Fixed $T_D=50$, $P_R=10,000$, $P_\alpha=1,000$, $RF_{Max}=2.75$, $E_R=1.0\%$, Runtime=01:42:36)

0.082509 0.0402696529 0.021409 0.0069605649 0.0007344180 2.75 0.010 4.00
10000 1000
0 50

Example 4: (Random $T_D/N=2$, $S_{Max}=46$, $P_R=1,000$, $P_\alpha=100$, $RF_{Max}=2.75$, $E_R=0.0\%$, Runtime=00:01:17)

0.082509 0.0402696529 0.021409 0.0069605649 0.0007344180 2.75 0.000 4.00
1000 100
2 M 65 F 67

Example 5: (Random $T_D/N=9$, $S_{Max}=50$, $P_R=5,000$, $P_\alpha=1,000$, $RF_{Max}=2.75$, $E_R=0.5\%$, Runtime=02:22:08)

0.082509 0.0402696529 0.021409 0.0069605649 0.0007344180 2.75 0.005 4.00
5000 1000
9 M 62 F 63 M 66 F 67 F 70 F 72 M 74 M 75 F 76

Example 6: (Random $T_D/N=1$, $S_{Max}=48$, $P_R=10,000$, $P_\alpha=1,000$, $RF_{Max}=2.75$, $E_R=1.0\%$, Runtime=11:01:24)

0.082509 0.0402696529 0.021409 0.0069605649 0.0007344180 2.75 0.010 4.00
10000 1000
1 F 65

Note: Runtime units are HH:MM:SS. The means/variances match the historical values that were derived in Appendix F. A 50-year fixed T_D solution contains all solutions for fixed T_D under 50 years. The parameter PrunePwr=4.0 indicates that heavy algorithm pruning begins when $P(\text{Ruin})$ equals the largest possible $P(\text{Ruin})$ for that time point to within 4 decimal places. Once ruin becomes a near certainty $\alpha=1$ and it is not necessary to evaluate all α -values. A higher PrunePwr value has a longer runtime and the user can confirm an equivalent solution by adjusting this value. The # years processed for random T_D is the larger of (111-min male age) and (113-min female age) where 111 and 113 are the maximum possible male/female ages, respectively, as derived from SSA.gov life-tables. (See the file "ageprobs.txt" below.) When T_D is random the last possible withdrawal is at $t=S_{Max}$ but the last decision point is at $t=S_{Max}-1$.

Input File: ageprobs.txt

```
50 0.005346265174752670 0.003283385442263650
51 0.005810688210135220 0.003544801480660440
52 0.006264310709811210 0.003795760877521360
53 0.006717933209487190 0.004015350349774660
54 0.007171555709163170 0.004234939822027960
55 0.007657579815958870 0.004475442577353010
56 0.008176005529874280 0.004747315257285670
57 0.008683630708083120 0.005071471144897680
58 0.009180455350585390 0.005458366881724930
59 0.009677279993087660 0.005897545826231530
60 0.010228107314122800 0.006399464619953360
61 0.010822136777984200 0.006943209979818680
62 0.011513371063204700 0.007518325264291610
63 0.012269408562664700 0.008103897190300420
64 0.013111850347777200 0.008741752323988580
65 0.014029895882835800 0.009452803948427840
66 0.015034345703546900 0.010247508705154100
67 0.016038795524258000 0.011104953311095500
68 0.017064846416382300 0.012035594407788100
69 0.018112498379919600 0.013049888636767600
70 0.019278956236229300 0.014168749281105900
71 0.020531818378191600 0.015392176340802900
72 0.021827882662980100 0.016667886608179200
73 0.023134747483475200 0.017943596875555500
74 0.024463213375383400 0.019282046992147100
75 0.025899684624357400 0.020787803373312600
76 0.027422560158983900 0.022429496094444400
77 0.028891433015077500 0.024050275532504500
78 0.030273901585518600 0.025629228404421100
79 0.031591566941720300 0.027208181276337700
80 0.032887631226508800 0.028891700563613000
81 0.034140493368471100 0.030721612832390500
82 0.035166544260595300 0.032561981742703900
83 0.035922581760055300 0.034339610803802000
84 0.036365403724024700 0.036033586732613200
85 0.036516611223916700 0.037539343113778700
86 0.036300600509785300 0.038804596739619200
87 0.035663368903097600 0.039693411270168200
88 0.034572514796733900 0.040132590214674900
89 0.032963234976454800 0.040028023799316100
90 0.030857130513673500 0.039285602250269300
91 0.028265001944096400 0.037894868925998300
92 0.025305655160496000 0.035845367184967500
93 0.022076294984231200 0.033189380234856200
94 0.018717328379487600 0.030021017849487100
95 0.015347561239037500 0.026371649953467900
96 0.012129001598479300 0.022481779302123700
97 0.009277660171944530 0.018591908650779500
98 0.006825938566552900 0.014890257547081000
99 0.004881842139370110 0.011585958821745600
100 0.003391368211863310 0.008804492173203810
101 0.002300514105499630 0.006545857601455560
102 0.001522875534626520 0.004726401974213920
103 0.000972048213591394 0.003335668649943010
104 0.000604829999567979 0.002269091213284120
105 0.000356417678316845 0.001495299739629630
106 0.000216010714131421 0.000951554379764307
107 0.000108005357065710 0.000575115284472933
108 0.000064803214239426 0.000345069170683760
109 0.000032401607119713 0.000188219547645687
110 0.000010800535706571 0.000094109773822844
111 0.000010800535706571 0.000052283207679358
112 0.000000000000000000 0.000020913283071743
113 0.000000000000000000 0.000010456641535872
```

Note: Columns in this file represent: age, male death probability, and female death probability for a 50-year old. They are computed from life-tables published at SSA.gov and each column sums to 1. For example, a male aged 50 has a 0.005346265174752670 probability of death before age 51. As seen, the maximum possible male age is 111 and the maximum possible female age is 113. There are no blank lines in this file and each field is delimited by a single space.

Code File: minpruin.cpp

```
/*
/ Copyright (c) 2014 Chris Rook, All Rights Reserved.
/
/ Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
/
/ 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
/ 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the
/ documentation and/or other materials provided with the distribution.
/ 3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this
/ software without specific prior written permission.
/
/ THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
/ TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR
/ CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
/ PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
/ LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
/ EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. (Source: http://opensource.org/licenses/BSD-3-Clause.)
/
/ Filename: minpruin.cpp (Defines the entry point for the console application.)
/
/ Function: main()
/
/ Summary:
/
/ Compiling the code will generate a single executable file that can be invoked with a double click or via a batch file with one parameter. When invoked
/ the directory location where input files reside and output files are written to is queried, stored as a string vbl and passed to other functions.
/ The control file (control.txt) is then read. If TD is specified as random the function derhrates() is invoked which reads the file of male/female
/ death probabilities (ageprobs.txt) and derives the hazard rates that apply to the group listed in the last line of the control file. The hazard rates
/ are stored locally and written to the file specified by the constant hrfil for reference. When TD is fixed, hazard rates are not needed or derived.
/ When invoked, the executable may have either zero or one argument(s). (See below.) The arrangement specified by the control file determines the
/ total number of years to process. Iteration always begins at the terminal year (i.e., total # of years minus 1) and ends at year 0. If TD is fixed
/ and 50 years are specified in the control file then processing starts at year=49 and ends at year=0. (This results in 50 alpha decisions made by the
/ retiree, the first at year=0 (i.e., t=0) and the last at year=49 (i.e., t=49). The total # of years to process for random TD depends on the specific
/ ages specified for the group. For example, if random TD for a 65 year old male and female couple is specified in the control file then a total of 48
/ years are processed since the maximum age allowed for a male is 111 and for a female is 113 (see the file ageprobs.txt). Processing 48 years of data
/ implies that 48 alpha decisions are made, here the first is at year=0 (both M/F at 65) and the last at year=47 (F is age 112). (See the paper for a
/ warning about dated hazard risk. Rerun the optimization if the hazard functions differ notably over time from those written to hrates.txt.) Using a
/ different ageprobs.txt file can change total # of years that need to be processed which, in general is: Max((Max male age - Min male age), (Max female
/ age - Min female age)). Here maximums are allowable and minimums are actual as specified in the control file. When processing each year (except the
/ terminal year) the function getprprobs() is invoked and reads the probabilities derived for the previous year. These are then passed to the function
/ optimize() which derives the probabilities and alphas for the current year and writes the results to a file for reference. Once complete, the function
/ combfiles() is invoked which combines all separate files into a single file to be read in as the prior probabilities during processing of the next year/
/ time point. When at year=0, combfiles() also concatenates all data from all prior years into both a vertical text file and 2 horizontal csv files, where
/ the csv files can be read into a spreadsheet. This is helpful when simulating the optimal solution to verify the result and only the csv file of optimal
/ alphas is needed. (Note: The terms "year" and "time point" are used interchangeably throughout this application and mean the same thing.)
/
/ Parameters:
/
/ The executable takes zero or one argument(s) when invoked. If an argument is specified it reflects the maximum number of jobs to process concurrently
/ by the application as it executes. If no argument is specified this value defaults to the number of independent processing units that exist on the
/ machine running the application. (This application was developed and tested on the Windows® operating system.)
/
/ Input Files:
/
/ 1.) Parameter control file of specific form whose name is defined by the string constant paramfile (defined in the header and located in the directory
/ specified by the user).
/ 2.) Age probability file of specific form whose name is set as the string constant ageprfile (defined in the header file and located in the directory
/ specified by the user. This file is read and processed by the function derhrates()).
/
```

```

/ Output Files:
/
/      1.) The derived hazard rates for the random arrangement specified in the control file. The hazard rate file is written by function derhrates() using
/          filename specified by string constant hrfile defined in the header to the directory specified by the user.
/      2.) FinalResults_V.txt (Vertical text file of all results from all years processed, written by combfiles() after year=0 finishes.)
/      3.) FinalAlphaResults_H.csv (Horizontal csv file of optimal alphas, written by combfiles() after year=0 finishes. Use this file to simulate the result.)
/      4.) FinalProbResults_H.csv (Horizontal csv file of optimal probabilities, written by combfiles() after year=0 finishes.)
/
/      Note: A number of temporary data files are written to the directory specified by the user during the processing of each year and across years. These
/            are removed by the application as it runs. They are created to avoid carrying large amounts of data inside arrays that are not needed. This was
/            a design decision made to speed up runtime. When processing a given year/time point, only the prior year's data is needed.
/
/ Return Value:
/
/      This function returns the integer value of 0 (success) or an error code (failure).
/-----*/

#include "stdafx.h"

int main(int argc, char *argv[])
{
    int precisions[2], numrand, numyears, p=0, rc, pllproc=0, * ages;
    long buckets[2], * * bktarys, bktsprun;
    double params[8];
    long double * hr, * prV;
    char * genders;
    string rootdir, trbin;
    vector<long> unqbkts;
    boost::thread * t;

    // Get the directory location where setup files are stored, strip any leading/trailing blanks.
    cout << "Enter the directory where the setup files reside (eg, c:\\mprsetup\\): " << endl;
    cin >> rootdir; cin.get();
    boost::algorithm::trim(rootdir);
    cout << endl;

    // Read in parameter control file. Load first line into array params[] and 2nd line into array precisions[].
    // (3rd line is processed conditionally based on the arrangement specified by the user i.e., fixed vs random TD.)
    ifstream getparams(rootdir+paramfile);
    if (getparams.is_open())
    {
        getparams >> params[0] >> params[1] >> params[2] >> params[3] >> params[4] >> params[5] >> params[6] >> params[7];
        getparams >> precisions[0] >> precisions[1];
        getparams >> numrand;
        if (numrand > 0)
        {
            genders = new char[numrand];
            ages = new int[numrand];
            if (numrand == 1)
                getparams >> genders[0] >> ages[0]; // Gender and age of single person with random TD.
            else // Random TD for group. Load arrays with genders and ages.
            {
                string people;
                std::getline(getparams, people);
                stringstream ss(people);
                for (int i=1; i <= numrand && ss.good(); ++i)
                {
                    ss >> genders[i-1] >> ages[i-1];
                    genders[i-1] = toupper(genders[i-1]);

                    // Data check for invalid gender. (Invalid ages are checked in derhrates().)
                    if (genders[i-1] != 'M' && genders[i-1] != 'F')

```

```

        {
            cout << "ERROR: Invalid gender in file " + rootdir + paramfile + " for person #" << i << ": " << genders[i-1] << endl;
            cout << "EXITING...main()..." << endl; cin.get();
            exit (EXIT_FAILURE);
        }
    }

    // Build the hazard rates if random TD is specified and no file exists in the directory. (Don't leave an old one sitting there.)
    ifstream gethrates(rootdir+hrfile);
    numyears = derhrates(rootdir, numrand, genders, ages, gethrates.is_open());
    hr = new long double[numyears+1];
    delete [] genders; genders = nullptr;
    delete [] ages; ages = nullptr;

    // File of hazard rates has now been built and written for reference, load it to the hr[] array.
    if (!gethrates.is_open())
        gethrates.open(rootdir+hrfile);

    // Read in survival probabilities that have been written to the file hrates.txt for this arrangement.
    // (Doing it this way is useful when needing to run a single year for debugging purposes.)
    if (gethrates.is_open())
    {
        while (!gethrates.eof() && p < numyears+1)
        {
            gethrates >> hr[p] >> trbin;
            p=p+1;
        }
        gethrates.close();
    }
    else
    {
        cout << "ERROR: Could not read file: " << rootdir + hrfile << endl;
        cout << "EXITING...main()..." << endl; cin.get();
        exit (EXIT_FAILURE);
    }
}
else
{
    // For fixed TD load hr[] with zeros (hrates.txt is not created).
    getparams >> numyears; // No SMax to account for here, so don't add an element.
    hr = new long double[numyears];
    for (int j=0; j<numyears; ++j)
        hr[j]=0.0;
}
getparams.close();
}
else
{
    cout << "ERROR: Could not read file: " << rootdir + paramfile << endl;
    cout << "EXITING...main()..." << endl; cin.get();
    exit (EXIT_FAILURE);
}

// Parse arguments to main(). Pllproc is optional, default is optimal # of threads on the machine running the application.
if (argc == 2)
    pllproc = stoi(argv[1]);
else if (argc != 1)
{
    cout << "ERROR: Parameter misspecification. Incorrect # of parameters to the executable (expecting zero or one...)." << endl;
    cout << "EXITING...main()..." << endl; cin.get();
    exit (EXIT_FAILURE);
}

```

```

}

// When pllproc==0, replace with the # of independent processing units on the computer running the application.
// Note that main() runs in its own thread but that is not accounted for here as it sits idle.
if (pllproc == 0)
    pllproc = boost::thread::hardware_concurrency();
if (pllproc == 1)
    pllproc = pllproc++; // If just one processing unit use 2 threads.

// Iterate over all years, launching separate threads to process optimal size collections of buckets concurrently.
for (int yr=numyears-1; yr>=0; yr--)
{
    cout << endl << "Processing for year " << yr << " has begun ..." << endl;
    // Process all buckets for terminal year in a single call to optimize().
    if (yr == (numyears-1))
    {
        buckets[0]=1;
        buckets[1]=(long) (params[5]*precisions[0] + 0.5);
        cout << "--> Processing buckets " << buckets[0] << " through " << buckets[1] << " ..." << endl;
        rc = optimize(rootdir, yr, numyears, params, precisions, buckets, hr, nullptr, vector<long>());
    }
    // Process buckets for all other years optimally based on both current/prior year characteristics.
    else
    {
        // Retrieve the prior year's probabilities which have been written to a file and load into an array.
        prV = new long double[(long) (params[5]*precisions[0] + 0.5)];
        unqbks=getprprobs(rootdir, yr, params[5], precisions[0], prV, hr[yr+1]);

        // Function call to determine pruning/parallel processing parameters for the current year.
        buckets[0]=0;
        buckets[1]=0;
        rc = optimize(rootdir, yr, numyears, params, precisions, buckets, hr, prV, unqbks);

        // Derive the optimal # of buckets to process per run.
        bktsprun = (long) (rc / (pllproc-1));
        cout << "--> # Buckets processed per thread (excluding last one): " << bktsprun << endl;

        // Build thread array of optimal size then launch calls to optimize() concurrently.
        bktarys = new long * [pllproc];
        t = new boost::thread[pllproc];
        for (int i=1; i<=pllproc; ++i)
        {
            // Define buckets then process them in concurrent threads.
            bktarys[i-1] = new long[2];
            bktarys[i-1][0]=bktsprun*(i-1) + 1;
            bktarys[i-1][1]=(i < pllproc)*bktsprun*(i) + (i==pllproc)*((long) (params[5]*precisions[0] + 0.5));
            cout << "--> Begin concurrent processing of buckets " << bktarys[i-1][0] << " through " << bktarys[i-1][1] << " ..." << endl;
            t[i-1] = boost::thread(optimize, rootdir, yr, numyears, boost::cref(params), boost::cref(precisions), boost::cref(bktarys[i-1]),
                                boost::cref(hr), boost::cref(prV), boost::cref(unqbks));
        }

        // Wait for all threads to finish, then proceed.
        for (int i=0; i<pllproc; ++i)
            t[i].join();

        // Free dynamic memory allocations.
        for (int i=0; i<pllproc; ++i)
        {
            delete [] bktarys[i]; bktarys[i] = nullptr;
        }
        delete [] bktarys; bktarys = nullptr;
        delete [] t; t = nullptr;
    }
}

```

```

        delete [] prV; prV = nullptr;

        // Clear contents of unqbkts vector and reset capacity. (This container is reused within the loop.)
        unqbkts.clear();
        unqbkts.shrink_to_fit();

        // Combine all files for each year processed, and if at year=0 concatenate data files from all years.
        combfiles(rootdir, yr, pllproc, bktsprun, params[5], precisions[0], numyears);
    }
    cout << "Processing for year " << yr << " has finished." << endl;
}

// Free memory of hazard rate array and exit.
delete [] hr; hr = nullptr;
return 0;
}

```

Code File: derhrates.cpp

```

/*
/ Copyright (c) 2014 Chris Rook, All Rights Reserved.
/
/ Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
/
/ 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
/ 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the
/ documentation and/or other materials provided with the distribution.
/ 3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this
/ software without specific prior written permission.
/
/ THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
/ TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR
/ CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
/ PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
/ LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
/ EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. (Source: http://opensource.org/licenses/BSD-3-Clause.)
/
/ Filename: derhrates.cpp
/
/ Function: derhrates()
/
/ Summary:
/
/ Derive the discrete hazard rates for the random TD arrangement specified by the last line of the control file. This function is not called for
/ fixed TD. The derived hazard rates are written to a file whose name is specified by the string constant hrfile (defined in the header file), and
/ the total # of years that need to be processed are determined by this function and returned by the call. All random TD models define a single
/ set of discrete hazard rates.
/
/ Parameters:
/
/ 1.) Root directory where input files reside and output files are written.
/ 2.) # of persons specified by this arrangement in the control file.
/ 3.) Array of genders for the random persons involved in this arrangement.
/ 4.) Array of ages for the random persons involved in this arrangement.
/ 5.) True/false indicator of whether or not the hazard rates file already exists. If it exists, this function only returns the total number of
/ years that need to be processed by this arrangement and the hazard rates are not re-derived. (Do not leave an old hazard rates file in the
/ root directory. If one is there it will be used but an alert message is given to the user. Kill the job with task manager if necessary.)
/
/ Input Files:
/
/ 1.) Age probability file of specific form whose name/location is set as the string constant ageprfile which is defined in the header file and

```



```

/          resides in the directory specified by the user.
/
/ Output Files:
/
/      1.) The derived hazard rates for the random arrangement specified in the control file using filename specified by hrfile which is defined in the
/          header file and resides in the directory specified by the user.
/
/ Return Value:
/
/      This function returns the # of years that need to be processed for this arrangement, representing the maximum # of alpha decisions to make.
/-----*/

#include "stdafx.h"

int derhrates(const string root, const int npersons, const char * gndrs, const int * ags, const bool hrexst)
{
    int age, prvage, remyrs, strtage=0, minmage=999, maxmage=0, maxamage, minfage=999, maxfage=0, maxafage;
    long double prob1, prob2, * * perspmf, * * perscdf, sumprobs, mchksum=0, fchksum=0, * tdcdf, * hrates;
    vector<long double> mprobs, fprobs;

    // Read age probabilities for M/F into corresponding arrays. These are TD probabilities starting at a given age and will need to be adjusted for each person
    // based on their age at retirement start. Also, use the file to derive the maximum possible M/F ages, which are ages that have non-zero probabilities.
    ifstream getprobs(root+ageprfile);
    if (getprobs.is_open())
    {
        while (!getprobs.eof())
        {
            getprobs >> age >> prob1 >> prob2;
            if (strtage == 0)
            {
                strtage = age;
                prvage = age-1;
            }
            if (age == prvage+1)
            {
                if (prob1 >= 0.0)
                    mprobs.push_back(prob1);
                if (prob2 >= 0.0)
                    fprobs.push_back(prob2);
                if (prob1 > 0.0)
                    maxamage=age;
                if (prob2 > 0.0)
                    maxafage=age;
                prvage = age;
            }
        }
        getprobs.close();
    }
    else
    {
        cout << "ERROR: Could not read file: " << root + ageprfile << endl;
        cout << "EXITING...derhrates()..." << endl; cin.get();
        exit (EXIT_FAILURE);
    }

    // Confirm that the probabilities read sum to 1.00 for each gender (allow for floating pt precision diffs).
    for (int i=(int) mprobs.size()-1; i>= 0; --i)
        mchksum=mchksum+mprobs[i];
    for (int i=(int) fprobs.size()-1; i>= 0; --i)
        fchksum=fchksum+fprobs[i];
    if (min(mchksum,fchksum) < (1.00 - pow(0.1,15)) || max(mchksum,fchksum) > (1.00 + pow(0.1,15)))
    {

```

```

        cout.setf(ios_base::fixed, ios_base::floatfield); cout.precision(25);
        cout << "ERROR: Probabilities in " << root + ageprfile << " do not sum to 1 for one or both genders. See below..." << endl;
        cout << "--> Male probability sum = " << mchksum << endl;
        cout << "--> Female probability sum = " << fchksum << endl;
        cout << "EXITING...derhrates()..." << endl; cin.get();
        exit (EXIT_FAILURE);
    }

    // Find the minimum male/female ages specified in the control file at retirement start.
    for (int i=0; i<npersons; ++i)
    {
        if (gndrs[i] == 'M' && ags[i] < minmage)
            minmage = ags[i];
        else if (gndrs[i] == 'F' && ags[i] < minfage)
            minfage = ags[i];
    }

    // Find the maximum male/female ages specified in the control file at retirement start.
    for (int i=0; i<npersons; ++i)
    {
        if (gndrs[i] == 'M' && ags[i] > maxmage)
            maxmage = ags[i];
        else if (gndrs[i] == 'F' && ags[i] > maxfage)
            maxfage = ags[i];
    }

    // Check that no age specified on the last line of the control file is less than the start age in the probabilities file.
    if (minmage < strtage || minfage < strtage)
    {
        cout << "ERROR: An age in " << root + paramfile << " is less than the minimum age in " << root + ageprfile << "." << endl;
        cout << "--> Smallest male age (" << paramfile << ") = " << minmage << endl;
        cout << "--> Smallest female age (" << paramfile << ") = " << minfage << endl;
        cout << "--> Smallest allowable age (" << ageprfile << ") = " << strtage << endl;
        cout << "EXITING...derhrates()..." << endl; cin.get();
        exit (EXIT_FAILURE);
    }

    // Check that no male age specified on the last line of the control file is greater than the last (>0 prob) male age in the age probs file.
    if (maxmage > maxamage)
    {
        cout << "ERROR: A male age in " << root + paramfile << " is greater than the maximum allowed in " << root + ageprfile << "." << endl;
        cout << "--> Largest male age (" << paramfile << ") = " << maxmage << endl;
        cout << "--> Largest allowable male age (" << ageprfile << ") = " << maxamage << endl;
        cout << "EXITING...derhrates()..." << endl; cin.get();
        exit (EXIT_FAILURE);
    }

    // Check that no female age specified on the last line of the control file is greater than the last (>0 prob) female age in the age probs file.
    if (maxfage > maxafage)
    {
        cout << "ERROR: A female age in " << root + paramfile << " is greater than the maximum allowed in " << root + ageprfile << "." << endl;
        cout << "--> Largest female age (" << paramfile << ") = " << maxfage << endl;
        cout << "--> Largest allowable female age (" << ageprfile << ") = " << maxafage << endl;
        cout << "EXITING...derhrates()..." << endl; cin.get();
        exit (EXIT_FAILURE);
    }

    // Derive total # years to process for this arrangement (plus 1). (Includes SMax, but no alpha decision is made at SMax.)
    remyrs = max(maxamage-minmage+1, maxafage-minfage+1);

    // If the hazard rates file exists use it. Otherwise derive it and write to file (using the details on the last line of the control file).
    if (hrexst)

```

```

else
{
    cout << endl << "ALERT: Hazard rates file: " << root + hrfile << " already exists and will be used. (Last line of the control file is ignored.)" << endl;

    // Construct the person-specific probability arrays (pmfs and cdfs), shifting each to begin at t=0 which represents the start of retirement.
    // Build array of pointers to individual TD probability arrays (start at time t=0 for each person). Then build inner arrays that will
    // apply to each individual person retiring, containing remyrs elements. (Do this for both the individual person pmfs and cdfs.)
    perspmf = new long double * [npersons];
    perscdf = new long double * [npersons];
    tdcdf = new long double [remyrs];
    hrates = new long double [remyrs];
    for (int i=0; i<npersons; ++i)
    {
        // Check that age values in parameter file are valid. (Invalid gender values are checked in main().)
        if (ags[i] < strtage || gndrs[i] == 'M' && ags[i] > maxamage || gndrs[i] == 'F' && ags[i] > maxafage)
        {
            cout << "ERROR: Invalid age in file control.txt for person #" << i << " of gender=" << gndrs[i] << ": " << ags[i] << endl;
            cout << "EXITING...derhrates()..." << endl; cin.get();
            exit (EXIT_FAILURE);
        }

        // Create the individual person PMF and CDF arrays.
        perspmf[i] = new long double [remyrs];
        perscdf[i] = new long double [remyrs];

        // Populate the individual arrays for each person. These are probabilities of death that
        // will sum to one and a value exists for each of the remyrs processed by this arrangement.
        sumprobs = 0.0;
        for (int j=((int) mprobs.size()-1); j>=(ags[i]-strtage); j--)
        {
            if (gndrs[i] == 'M')
                sumprobs = sumprobs + mprobs[j];
            else if (gndrs[i] == 'F')
                sumprobs = sumprobs + fprobs[j];
        }

        // Load the individual person PMF array with the applicable conditional PMF value. (Initialize with zeros.)
        for (int j=0; j<remyrs; ++j)
            perspmf[i][j] = 0.0;
        for (int j=0; (gndrs[i] == 'M' && j<(maxamage-(ags[i]-1))) || (gndrs[i] == 'F' && j<(maxafage-(ags[i]-1))); ++j)
        {
            if (gndrs[i] == 'M')
                perspmf[i][j] = mprobs[ags[i]-strtage+j] / sumprobs;
            else if (gndrs[i] == 'F')
                perspmf[i][j] = fprobs[ags[i]-strtage+j] / sumprobs;
        }

        // Check that sum of probabilities equals 1 (within floating point precision). If not put out alert.
        sumprobs = 0.0;
        for (int j=0; j<remyrs; ++j)
            sumprobs = sumprobs + perspmf[i][j];
        if (sumprobs < (1.00 - pow(0.1,15)) || sumprobs > (1.00 + pow(0.1,15)))
        {
            cout.setf(ios_base::fixed, ios_base::floatfield); cout.precision(50);
            cout << "Alert: Sum of probabilities for Person " << i+1 << " is not 1.00." << endl << "Sum is = " << sumprobs << endl;
        }

        // Load the individual person cdf array with the applicable cumulative probability.
        // And load the TD cdf array with the applicable cumulative probability.
        for (int j=0; j<remyrs; ++j)
        {
            perscdf[i][j] = 0;
            for (int k=0; k<=j; ++k)

```

```

        perscdf[i][j] = perscdf[i][j] + perspmf[i][k];
    if (i==0)
        tdcdf[j] = perscdf[0][j];
    else
        tdcdf[j] = tdcdf[j]*perscdf[i][j];
    }
}

// Build the hazard rates and write to file.
hrates[0] = tdcdf[0];
for (int j=1; j<remyrs; ++j)
    hrates[j] = min((tdcdf[j] - tdcdf[j-1]) / (1.0 - tdcdf[j-1]), 1.00);

// Write hazard rate probabilities to file for reference, this file will be read back in if processing years in separate calls.
ofstream hrout(root+hrfile);
hrout.setf(ios_base::fixed, ios_base::floatfield);
for (int j=0; j<remyrs; ++j)
{
    hrout.precision(50); hrout << hrates[j] << " "; hrout.precision(0); hrout << "(t=" << j << ")" << endl;
}
hrout.close();

// Free dynamic memory allocations.
for (int i=0; i<npersons; ++i)
{
    delete [] perspmf[i]; perspmf[i] = nullptr;
    delete [] perscdf[i]; perscdf[i] = nullptr;
}
delete [] perspmf; perspmf = nullptr;
delete [] perscdf; perscdf = nullptr;
delete [] tdcdf; tdcdf = nullptr;
delete [] hrates; hrates = nullptr;
}

// Return # years to process for this arrangement, which is 1 less than the # written to the hr file (since SMax is written).
return (remyrs-1);
}

```

Code File: getprprobs.cpp

```

/*
/ Copyright (c) 2014 Chris Rook, All Rights Reserved.
/
/ Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
/
/ 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
/ 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the
/ documentation and/or other materials provided with the distribution.
/ 3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this
/ software without specific prior written permission.
/
/ THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
/ TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR
/ CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
/ PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
/ LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
/ EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. (Source: http://opensource.org/licenses/BSD-3-Clause.)
/
/ Filename: getprprobs.cpp
/
/ Function: getprprobs()

```

```

/
/ Summary:
/
/      This function reads the prior time point's probability file and populates an array with these values. It also returns a vector containing only the
/      bucket #'s from the prior time point that have unique probabilities assigned to them. When deriving the expected probability of ruin for any time
/      point after the next time point it is not necessary to process every bucket if sequential buckets hold the same probability. Therefore the optimize()
/      function only processes those buckets with unique probabilities. Note that the probability of falling into a new larger bucket that was formed from
/      multiple sequential buckets is higher.
/
/ Parameters:
/
/      1.) Root directory where input files reside and output files are written.
/      2.) Current year/time point being processed. (These start at 0 and end at one less than the total # being processed by the given arrangement specified
/          in the control file.)
/      3.) Maximum ruin factor value specified in the control file (i.e., RfMax).
/      4.) Precision value for the ruin factor discretization as specified in the control file (i.e., PR).
/      5.) An empty array of appropriate size (i.e., total # of buckets) long doubles that will be populated by this function.
/      6.) Value of the prior time point's hazard rate (0.00 if fixed TD). This is to determine when the highest possible probability has been reached and
/          all buckets from this time point onward can be treated as one collection.
/
/ Input Files:
/
/      1.) File of probabilities from the prior time point that has already been derived. This file has the form "Year_X_All_Buckets.txt", where X refers to
/          actual time point. Time points start at 0 and end at 1 less than the total # being processed by the given arrangement. Each time point reflects
/          a decision that needs to be made. Time t=0 reflects the first asset allocation decision at the start of retirement and time T=SMax-1 reflects the
/          last asset allocation decision made the year before SMax when TD is random. When TD is fixed, TD-1 is the last time point requiring an asset
/          allocation decision. Note that the prior year/time point probabilities being read are from the current value + 1.
/
/ Output Files:
/
/      None.
/
/ Return Value:
/
/      This function returns a vector of bucket #'s with unique probabilities (from the prior time point which has already been processed). (It also populates
/      the array Vp whose location is passed to this function.)
/-----*/

#include "stdafx.h"

vector<long> getprprobs(const string root, const int curyear, const double rfmax, const int pprec, long double * Vp, const long double prhrate)
{
    int inyr, prMax=0;
    long * bktPrn, inb;
    double inrf, inalpha;
    long double inprob, prevprob;
    const long nbuckets = (long) (rfmax*pprec + 0.5);
    vector<long> PrB;
    string ifname;
    ifstream fin;
    ifname = root + "Year_" + to_string((long long) (curyear+1)) + "_All_Buckets.txt";

    // Array to indicate (0/1) where unique probabilities from prior time point reside.
    bktPrn = new long [nbuckets];

    // Open and read the prior time point's probability file and load the values into an array.
    fin.open(ifname);
    if (fin.is_open())
    {
        while (!fin.eof())
        {
            fin >> inyr >> inrf >> inprob >> inalpha;

```

```

        if (inyr == (curyear + 1))
        {
            inb = (long) (inrf*pprec + 0.5);
            Vp[inb-1]=inprob;
        }
    }
    fin.close();
}
else
{
    cout << "ERROR: Could not read prior year probability file: " << ifname << endl;
    cout << "EXITING...getprprobs()..." << endl; cin.get();
    exit (EXIT_FAILURE);
}

// Check the data just read in for any issues and load pruning array.
for (long b=1; b <= nbuckets; ++b)
{
    // Populate the pruning array with 1's and 0's, where 1 indicates unique probability (last bucket in the sequence).
    if (b > 1 && b < nbuckets && Vp[b-1] != Vp[b] && prMax == 0) // Always process first and last buckets individually.
    {
        if (Vp[b] >= (1.00 - prhrate)) // The next bucket holds the maximum possible probability, stop scanning.
        {
            prMax = 1;
            bktPrn[b-1] = 1; // This bucket will account for a unique probability.
        }
        else if (b > 1 && b < nbuckets)
        {
            bktPrn[b-1] = 0; // This bucket will not account for a unique probability.
        }
        else
        {
            bktPrn[b-1] = 1; // First and last buckets will always be processed.
        }
    }

    if (b==1)
        prevprob=0;

    // Check for and report issues found in the prior time point's probability file.
    if ( (Vp[b-1] < 0) || (Vp[b-1] > (1.00 - prhrate) + (2.0*pow(0.1,16))) || (Vp[b-1] < (prevprob - pow(0.1,15))) )
    {
        cout.setf(ios_base::fixed, ios_base::floatfield);
        cout.precision(50);
        cout << "There is an issue with the previous years data (see below):" << endl;
        cout << "Vp[" << (b-2) << "] = " << Vp[b-2] << endl;
        cout << "Vp[" << (b-1) << "] = " << Vp[b-1] << endl;
        cout << "EXITING...getprprobs()..." << endl; cin.get();
        exit(EXIT_FAILURE);
    }
    prevprob = Vp[b-1];
}

// Build an array with values that are the bucket #'s from the prior time point that have a
// unique probability and thus will be processed during the expected value computation.
for (long b=1; b<=nbuckets; ++b)
    if (bktPrn[b-1] == 1)
        PrB.push_back(b);

// Free memory of dynamically sized arrays.
delete [] bktPrn; bktPrn = nullptr;

// For informational purposes.
cout << "--> # Unique bucket probs at prior time point reported by getprprobs(): " << PrB.size() << endl;

// Return the vector PrB.
return PrB;
}

```

Code File: optimize.cpp

```
/*
/ Copyright (c) 2014 Chris Rook, All Rights Reserved.
/
/ Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
/
/ 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
/ 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the
/ documentation and/or other materials provided with the distribution.
/ 3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this
/ software without specific prior written permission.
/
/ THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
/ TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR
/ CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
/ PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
/ LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
/ EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. (Source: http://opensource.org/licenses/BSD-3-Clause.)
/
/ Filename: optimize.cpp
/
/ Function: optimize()
/
/ Summary:
/
/ This function derives the optimal alpha and probability values for a single year/time point and bucket range specified by the call. When the first
/ and last buckets are set to 0 this function makes a single pass over the given year/time point and approximates the bucket number where heavy algorithm
/ pruning begins. Heavy pruning sets alpha=pa which is a decision rule that takes effect when the optimal probability of ruin exceeds the first k
/ decimal places of the maximum possible probability of ruin for this year/arrangement, where k is the last parameter specified on the first line of the
/ control file. Thus a value of 4.00 indicates that pruning begins when the probability of ruin exceeds the first 4 decimal places of the highest possible
/ probability of ruin for this year/arrangement. Note that this value is stored locally in the variable prnpwr and only has an impact when TD is random.
/ The initial pass approximates this value by only considering alpha=pa. Optimal bucket sizes are then formed based off of this approximation and then
/ the actual calls to this function are made where the point of heavy pruning is determined exactly. The approximation is only used to estimate optimal
/ bucket collection sizes for threaded processing. For all other cases the specific buckets listed are processed and written to a file with the bucket
/ boundaries contained in the file name. The function combfiles() is invoked by main() after all buckets for a year/time point have been processed and
/ written to a file and it aggregates all individual files then deletes them leaving a single file containing all relevant optimal values for each year/
/ time point. Once the final year/time point has been processed (i.e., t=0) the individual files for each year/time point are aggregated into final
/ (horizontal & vertical) files for the specific arrangement being modeled and the year-specific files are deleted.
/
/ Parameters:
/
/ 1.) Root directory where input files reside and output files are written.
/ 2.) Year value to process. (This is the time point value which always starts at zero and ends at # time points to process for this arrangement less 1.)
/ 3.) Total # of years/time points processed by the arrangement specified in the control file.
/ 4.) Array containing all values from the first line of the control file.
/ 5.) Array containing the precision values from the 2nd line of the control file.
/ 6.) Array containing the bucket limits for this call.
/ 7.) Array containing the hazard rates derived for this arrangement.
/ 8.) Array of optimal probabilities derived for the prior year/time point (i.e., current year/time point + 1)
/ 9.) Vector of unique bucket endpoints for prior year where collections of buckets with equal probabilities are treated as a single bucket in the
/ expected value calculation.
/
/ Input Files:
/
/ None. (The file of prior probabilities is read by the function getprprobs() and passed to this function via the parameter Vp.)
/
/ Output Files:
/
/ 1.) When processing the terminal time point the file "Year_X_All_Buckets.txt" is written to directory root. (Where X = nyrs-1 is the time point.)
/ 2.) When processing all other time points the file "Year_Y_Buckets_S_thru_E.txt" is written to directory root. (Where Y is the time point, S is the
```

```

/          start bucket #, and E is the end bucket #.  Once processing for this year has ended these files are aggregated by combfiles() into a single file
/          named "Year_Y_All_Buckets.txt".)
/
/ Return Value:
/
/          The approximate bucket # where heavy algorithm pruning begins when invoking this function with start and end buckets equal to 0.  Otherwise return 0.
/-----*/

#include "stdafx.h"

int optimize(const string root, const int y, const int nyrs, const double prms[8], const int prc[2], const long bkts[2], const long double * sprobs,
            const long double * Vp, const vector<long> & PrB)
{
    const int pr=prc[0], pa=prc[1];
    const double smn=prms[0], svr=prms[1], bmn=prms[2], bvr=prms[3], cv=prms[4], rfmax=prms[5], er=prms[6], prnpwr=prms[7];
    const long nbuckets=(long) (rfmax*pr + 0.5), nugbkts=(long) PrB.size();
    int prunept = 0, ties, stalpa=0;
    long stbkt=bkts[0], endbkt=bkts[1];
    double rf, alpha, OPT_alpha, mean, std, * Ac;
    long double OPT_pruin = 1.0, cdfval, eprob, * Vc,  rhs_cdf, lhs_cdf, pruin, tiethresh, pruneptprob;
    boost::math::normal normdist;
    ofstream fout;

    // Create the arrays to hold current year optimal alphas and probabilities.
    Ac = new double [nbuckets];
    Vc = new long double [nbuckets];

    // Before processing the current year, run through all buckets using only alpha=100% to find the pruning point.
    if (stbkt == 0 && endbkt == 0)
    {
        stalpa = pa;
        stbkt = 1;
        endbkt = nbuckets;
    }

    // Process year specified.
    if (y == (nyrs-1)) // Last decision point (either TD-1 or SMax-1).
    {
        if (bkts[0] == 0 && bkts[1] == 0) // This should not happen, return with error.
        {
            cout << "ERROR: Attempt to find pruning point at terminal year." << endl;
            cout << "EXITING...optimize()..." << endl; cin.get();
            exit (EXIT_FAILURE);
        }
        for (long b = stbkt; b <= endbkt; ++b) // Get min PRuin at the last time point.
        {
            rf = (double) b / pr;
            for (int a=0; a <= pa; ++a)
            {
                alpha = (double) a / pa;
                mean = (1.00-er)*(1.00 + alpha*smn + (1.00-alpha)*bmn);
                std = (1.00-er)*sqrt(pow(alpha,2)*svr + pow(1.00-alpha,2)*bvr + 2.00*(alpha)*(1.00-alpha)*(cv));
                normdist = boost::math::normal(mean,std);
                cdfval = cdf(normdist, rf);
                if ((a == 0) || (cdfval < 0.5 && cdfval < OPT_pruin) || (cdfval >= 0.5 && cdfval <= OPT_pruin))
                {
                    OPT_pruin = cdfval;
                    OPT_alpha = alpha;
                }
            }
            Ac[b-1] = OPT_alpha;
            Vc[b-1] = (1.00 - sprobs[y]) * OPT_pruin;
        }
    }
}

```



```

}

// Write results to file. These will be loaded to an array when processing the next time point (i.e., the one before this one).
string outfile = root + "Year_" + to_string((long long) y) + "_All_Buckets.txt";
fout.open(outfile);
for (long b = stbkt; b <= endbkt; ++b)
{
    fout.setf(ios_base::fixed, ios_base::floatfield);
    fout << y << " ";
    fout.precision(10); fout << (double) b / pr << " ";
    fout.precision(50); fout << Vc[b-1] << " ";
    fout.precision(10); fout << Ac[b-1] << endl;
}
fout.close();
}
else // Process all other years.
{
    // One-half of the maximum possible PRuin is the tie threshold, in general.
    tiethresh = (0.5)*((1.00 - sprobs[y]));

    // Prune point #2 begins at set number of decimals after max probability for this arrangement.
    pruneprob = floor(pow(10.0,prnpwr)*(1.00 - sprobs[y])) / pow(10.0,prnpwr);

    if (bkts[0] == 0 && bkts[1] == 0)
    {
        cout.setf(ios_base::fixed, ios_base::floatfield);
        cout.precision(10); cout << "--> Value of pruneprob reported by optimize() is: " << pruneprob << endl;
    }
    for (long b = stbkt; b <= endbkt; ++b) // PRuin at any future time point.
    {
        ties = 0; // Initialize for le/gt comparisons.
        rf = (double) b / pr; // Derive ruin factor.
        OPT_alpha = 99.00; // Initialize to unrealistic value.
        OPT_pruin = 99.00; // Initialize to unrealistic value.
        for (int a=stalpa; a <= pa; ++a)
        {
            if ((prunepnt == 0 && (a == stalpa || OPT_pruin > 0.00)) || (prunepnt == 1 && a == pa))
            {
                alpha = (double) a / pa;
                mean = (1.00-er)*(1.00 + alpha*smn + (1.00-alpha)*bmn);
                std = (1.00-er)*sqrt(pow(alpha,2)*svr + pow(1.00-alpha,2)*bvr + 2.00*(alpha)*(1.00-alpha)*(cv));
                normdist = boost::math::normal(mean,std);
                cdfval = cdf(normdist, rf);
                if (cdfval == 1.00)
                    eprob = 1.00 - sprobs[y+1];
                else
                {
                    //-----//
                    rhs_cdf = 1.00;
                    lhs_cdf = cdf(normdist, rf*(1 + (pr/1.5)));
                    eprob = (rhs_cdf - lhs_cdf)*Vp[0]; // First bucket, unique processing.
                    rhs_cdf = lhs_cdf;
                    for (long pb=2; pb<=nuqbkets; ++pb) // All others but last bucket, standard processing
                    { // for unique probs only.
                        lhs_cdf = cdf(normdist, rf*(1.0 + pr/(PrB[pb-1] + 0.5)));
                        eprob = eprob + (rhs_cdf - lhs_cdf)*Vp[PrB[pb-1]-1];
                        rhs_cdf = lhs_cdf;
                    }
                    eprob = eprob + (rhs_cdf - cdfval)*(1.00 - sprobs[y+1]); // Last bucket, unique processing.
                    eprob = eprob / (1.00 - cdfval); // Make the probability conditional.
                    //-----//
                }
            }
        }
    }
}

```

```

        // Deal with numerical instability near zero.
        if (ties == 0)
        {
            pruin = (1.00 - sprobs[y])*(cdfval + eprob - (cdfval*eprob));
            if (pruin > tiethresh)
                ties = 1;
        }

        // Deal with numerical instability near one.
        if (ties == 1)
            pruin = (1.00) - (sprobs[y] + (1.00-cdfval)*(1.00-eprob) - sprobs[y]*(1.00-cdfval)*(1.00-eprob));

        // Update optimal values.
        if ((a == 0) || (ties == 0 && pruin < OPT_pruin) || (ties == 1 && pruin <= OPT_pruin))
        {
            OPT_pruin = pruin;
            OPT_alpha = alpha;
        }
    }

    // Load optimal values into corresponding arrays.
    Ac[b-1] = OPT_alpha;
    Vc[b-1] = OPT_pruin;

    // Set pruning for this timepoints processing, allow for floating point precision diffs.
    if (prunepnt == 0 && Vc[b-1] >= (pruneprob - (pow(0.1,16) + pow(0.1,17))))
    {
        prunepnt = 1;

        // Initial run finds the (approximate) bucket where pruning starts then uses this information to split the remaining buckets by
        // sizes that will run in shortest time. For this run, end the call as soon as the bucket number has been found.
        if (bkts[0] == 0 && bkts[1] == 0)
        {
            cout << "--> Pruning RC returned by optimize() is: " << b << endl;
            return b;
        }
    }
}

// Write current year to file with bucket boundaries contained in the name.
string outfile = root + "Year_" + to_string((long long) y) + "_Buckets_" + to_string((long long) stbkt) + "_thru_" + to_string((long long) endbkt)
+ ".txt";

fout.open(outfile);
for (long b = stbkt; b <= endbkt; ++b)
{
    fout.setf(ios_base::fixed, ios_base::floatfield);
    fout << y << " ";
    fout.precision(10); fout << (double) b / pr << " ";
    fout.precision(50); fout << Vc[b-1] << " ";
    fout.precision(10); fout << Ac[b-1] << endl;
}
fout.close();
}

// Free dynamic memory allocations and exit.
delete [] Ac; Ac = nullptr;
delete [] Vc; Vc = nullptr;
return 0;
}

```

Code File: combfiles.cpp

```
/*
/ Copyright (c) 2014 Chris Rook, All Rights Reserved.
/
/ Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
/
/ 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
/ 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the
/ documentation and/or other materials provided with the distribution.
/ 3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this
/ software without specific prior written permission.
/
/ THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
/ TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR
/ CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
/ PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
/ LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
/ EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. (Source: http://opensource.org/licenses/BSD-3-Clause.)
/
/ Filename: combfiles.cpp
/
/ Function: combfiles()
/
/ Summary:
/
/ Function to aggregate the individual bucket-specific output files written by optimize() into single files for each time point. These files are
/ subsequently read by getprprobs() when processing the next time point. At the last time point (i.e., year=0) all files for the individual time
/ points are aggregated into 3 files then deleted. The first is a (space-delimited) vertical file containing all data for all time points. This
/ file is named FinalResults_V.txt. The remaining 2 files are comma-separated horizontal files for import into a spreadsheet, and these files are
/ named FinalAlphaResults_H.csv and FinalProbResults_H.csv.
/
/ Parameters:
/
/ 1.) Root directory where input files reside and output files are written.
/ 2.) Current year/time point number being processed. (Years start at 0 and end at one less than the total # of years being processed by the given
/ arrangement specified in the control file.)
/ 3.) Number of concurrent processes used to derive the optimal values for each time point.
/ 4.) Number of buckets processed per thread within each time point. (Except the last one. This is specific to each time point.)
/ 5.) Maximum ruin factor value specified in the control file (i.e., RFMax).
/ 6.) Precision value for the ruin factor discretization as specified in the control file (i.e., PR).
/ 7.) Total number of years processed by the arrangement specified in the control file.
/
/ Input Files:
/
/ 1.) For a given time point, input files of the form "Year_X_Buckets_S_thru_E.txt" are read from the root directory. (Where X is the time point,
/ S is the start bucket #, and E is the end bucket #. This function aggregates these individual files into a single file named: Year_X_All_Buckets.txt.
/ 2.) Year_X_All_buckets.txt (Written by this function for each time point and all such files are read by this function and aggregated at the last time
/ point, which is year=0.)
/
/ Output Files:
/
/ 1.) Year_X_All_Buckets.txt, where X reflects the year/time point being processed.
/ 2.) FinalResults_V.txt (Vertical text file of all results from all years processed.)
/ 3.) FinalAlphaResults_H.csv (Horizontal csv file of just optimal alphas. Use this file to simulate and confirm the result.)
/ 4.) FinalProbResults_H.csv (Horizontal csv file of optimal probabilities.)
/
/ Note: The last 3 output files above are only written at the last time point when all processing ends (i.e., year=0). The first file is written at every
/ time point.
/
/ Return Value:
```

```

/
/      None.
/-----*/

#include "stdafx.h"

void combfiles(const string root, const int curyear, const int pllruns, const long bcktsprun, const double rfmax, const int pprec, const int nyrs)
{
    int numfiles=0, fnlyear;
    const long nbkts=(long) (rfmax*pprec + 0.5);
    long fnlbkt;
    double fnlrf, fnlalpha;
    long double fnlprob;
    string fndfname, fndfile, concatyr, concatall;
    ifstream chk4file, normin;
    ofstream fnloutp, fnlouta;

    numfiles = 0;
    for (int j=1; j<=pllruns; ++j)
    {
        if (j <= (pllruns-1))
            fndfname = "Year_" + to_string((long long) curyear) + "_Buckets_" + to_string((long long) bcktsprun*(j-1) + 1) + "_thru_" +
                to_string((long long) bcktsprun*(j)) + ".txt";
        else
            fndfname = "Year_" + to_string((long long) (curyear)) + "_Buckets_" + to_string((long long) bcktsprun*(j-1) + 1) + "_thru_" +
                to_string((long long) nbkts) + ".txt";
        fndfile = root + fndfname;

        // Construct string for concatenation.
        concatyr=concatyr + " " + fndfile;

        // Attempt to open file and update file counter if successful.
        chk4file.open(fndfile);
        if (chk4file.is_open())
            numfiles = numfiles + 1;
        chk4file.close();
    }
    if (numfiles == pllruns)
    {
        // Concatenate files into 1 and remove individual files.
        concatyr="type" + concatyr + " > " + root + "Year_" + to_string((long long) curyear) + "_All_Buckets.txt 2>nul & del " + root + "Year_" +
            to_string((long long) (curyear)) + "_Buckets*.txt";
        system(concatyr.c_str());
    }
    else
    {
        cout << "ERROR: Attempt to combine files after year=" << curyear << " has failed." << endl;
        cout << "EXITING...combfiles()..." << endl; cin.get();
        exit (EXIT_FAILURE);
    }

    // If at last year then build the normalized and transposed final data files and delete all intermediate data files.
    if (curyear == 0)
    {
        // Build normalized data file.
        concatall="type" + root + "Year_" + to_string((long long) curyear) + "_All_Buckets.txt";
        for (int j=curyear+1; j<nyrs; ++j)
            concatall = concatall + " " + root + "Year_" + to_string((long long) j) + "_All_Buckets.txt";
        concatall=concatall + " > " + root + "FinalResults_V.txt 2>nul & del " + root + "Year_*.txt";
        system(concatall.c_str());

        // Build transposed probability/alpha data files.
    }
}

```

```

long double * * fnlprobs;                                // Pointer to long double pointer for probabilities.
double * * fnlalphas;                                    // Pointer to double pointer for alphas.
fnlprobs = new long double * [nyrs];                     // Use long double pointer to hold array of long double pointers.
fnlalphas = new double * [nyrs];                         // Use double pointer to hold array of double pointers.
for (int j=0; j<nyrs; ++j)                               // (A pointer to any type can hold an array of that type.)
{
    fnlprobs[j] = new long double [nbkts];               // Each array value now points to array of long doubles (2D array).
    fnlalphas[j] = new double [nbkts];                  // Each array value now points to array of doubles (2D array).
}

// Open normalized file just created, transpose it and write to csv file.
normin.open(root+"FinalResults_V.txt");
if (normin.is_open())
{
    while (!normin.eof())
    {
        normin >> fnlyear >> fnlrf >> fnlprob >> fnlalalpha;
        fnlbkt = (long) (fnlrf*pprec + 0.5);
        if (fnlbkt >= 1 && fnlbkt <= (long) (rfmax*pprec + 0.5))
        {
            fnlprobs[fnlyear][fnlbkt-1] = fnlprob;
            fnlalphas[fnlyear][fnlbkt-1] = fnlalalpha;
        }
    }
    normin.close();
}
else
{
    cout << "ERROR: Could not read file: " << root+"FinalResults_V.txt" << endl;
    cout << "EXITING...combfiles()..." << endl; cin.get();
    exit (EXIT_FAILURE);
}

// Transpose and write to file.
fnloutp.open(root+"FinalProbResults_H.csv");  fnloutp.setf(ios_base::fixed, ios_base::floatfield);  fnloutp << "RF";
fnlouta.open(root+"FinalAlphaResults_H.csv");  fnlouta.setf(ios_base::fixed, ios_base::floatfield);  fnlouta << "RF";

for (long i=0; i<=nbkts; ++i)                // The first line (i=0) of the csv files holds column headers.
{
    if (i > 0)
    {
        float _rf_; _rf_ = (float) i/pprec;
        fnloutp.precision(10); fnloutp << _rf_;
        fnlouta.precision(10); fnlouta << _rf_;
    }
    for (int j=0; j<nyrs; ++j)
    {
        if (i == 0)
        {
            fnloutp << ", Time (t=" << j << ")";  fnlouta << ", Time (t=" << j << ")";
        }
        else
        {
            fnloutp.precision(50); fnloutp << "," << fnlprobs[j][i-1];
            fnlouta << "," << fnlalphas[j][i-1];
        }
    }
    fnloutp << endl;  fnlouta << endl;
}
fnloutp.close();
fnlouta.close();

```

```

        // Free dynamic memory allocations.
        for (int j=0; j<(nyrs); ++j)
        {
            delete [] fnlprobs[j];  fnlprobs[j] = nullptr;
            delete [] fnlalphas[j];  fnlalphas[j] = nullptr;
        }
        delete [] fnlprobs;  fnlprobs = nullptr;
        delete [] fnlalphas;  fnlalphas = nullptr;
    }
}

```

Code File: stdafx.h

```

/*
/ Copyright (c) 2014 Chris Rook, All Rights Reserved.
/
/ Filename:  stdafx.h  (header file to include)
/ -----*/

#pragma once

// Header files

#include "targetver.h"          // Define based on user's target operating system version.  (See compiler for reference.)
#include <stdio.h>
#include <tchar.h>
#include <iostream>
#include <fstream>
#include <string>
#include <math.h>
#include <stdlib.h>
#include <boost/math/distributions.hpp>
#include <boost/algorithm/string.hpp>
#include <boost/thread/thread.hpp>
#include <windows.h>
#include <dos.h>
#include <vector>

using namespace std;

// Define constants.

const string paramfile="control.txt";          // Name of parameter control input file.
const string ageprfile="ageprobs.txt";         // Name of age probability input file.
const string hrfile="hrates.txt";              // Name of hazard rate probability output file.

// Define Function prototypes.

int optimize(const string root, const int strtpr, const int nyrs, const double prms[8], const int prc[2], const long bkts[2], const long double * sprobs,
             const long double * Vp, const vector<long> & PrB);
int derhrates(const string root, const int numpersons, const char * ppl, const int * ags, const bool hrexst);
void combfiles(const string root, const int curyear, const int pllruns, const long bcktsprun, const double rfmax, const int pprec, const int nyears);
vector<long> getprprobs(const string root, const int curyear, const double rfmax, const int pprec, long double * Vp, const long double prhrate);

```

Output File: hrates.txt

```
0.000000000000000003119716336171020000000000000000000 (t=0)
0.00000000000000000219225366981865290000000000000000000 (t=1)
0.00000000000000000113691519087302570000000000000000000 (t=2)
0.00000000000000000020003863625408732000000000000000000 (t=3)
0.00000000000000000019508109036756375000000000000000000 (t=4)
0.00000000000000000013122232017542715000000000000000000 (t=5)
0.00000000000000000068276004654995606000000000000000000 (t=6)
0.00000000000000000029370323648036120000000000000000000 (t=7)
0.00000000000000000010922010790464607000000000000000000 (t=8)
0.00000000000000000003614532741498721900000000000000000 (t=9)
0.000010840167532523471000000000000000000000000000000 (t=10)
0.000029845843540258257000000000000000000000000000000 (t=11)
0.000076209096960356155000000000000000000000000000000 (t=12)
0.000181886473346565420000000000000000000000000000000 (t=13)
0.000407608210176165860000000000000000000000000000000 (t=14)
0.000859042642106637900000000000000000000000000000000 (t=15)
0.001704890145610573400000000000000000000000000000000 (t=16)
0.003193594391314505700000000000000000000000000000000 (t=17)
0.005658974674907143800000000000000000000000000000000 (t=18)
0.009499324966886255100000000000000000000000000000000 (t=19)
0.015118189091587747000000000000000000000000000000000 (t=20)
0.022875081638296994000000000000000000000000000000000 (t=21)
0.033015922824457408000000000000000000000000000000000 (t=22)
0.045637651785438903000000000000000000000000000000000 (t=23)
0.060691746109167360000000000000000000000000000000000 (t=24)
0.077963887369063345000000000000000000000000000000000 (t=25)
0.097137396255244896000000000000000000000000000000000 (t=26)
0.117840663186852730000000000000000000000000000000000 (t=27)
0.139575950610399850000000000000000000000000000000000 (t=28)
0.161900102252575400000000000000000000000000000000000 (t=29)
0.184467351117651360000000000000000000000000000000000 (t=30)
0.207135581355305380000000000000000000000000000000000 (t=31)
0.229171968292873700000000000000000000000000000000000 (t=32)
0.250574407204215200000000000000000000000000000000000 (t=33)
0.270977195463241840000000000000000000000000000000000 (t=34)
0.290085531901921880000000000000000000000000000000000 (t=35)
0.308009615818462420000000000000000000000000000000000 (t=36)
0.325077620015427620000000000000000000000000000000000 (t=37)
0.342464754108966370000000000000000000000000000000000 (t=38)
0.360104446807174790000000000000000000000000000000000 (t=39)
0.379553934696363280000000000000000000000000000000000 (t=40)
0.399307081845850230000000000000000000000000000000000 (t=41)
0.420237804626677040000000000000000000000000000000000 (t=42)
0.441048411168636250000000000000000000000000000000000 (t=43)
0.466088322927087180000000000000000000000000000000000 (t=44)
0.495210226193286640000000000000000000000000000000000 (t=45)
0.532495020963441370000000000000000000000000000000000 (t=46)
0.546267277606566020000000000000000000000000000000000 (t=47)
0.598662873465986230000000000000000000000000000000000 (t=48)
0.754149916748012240000000000000000000000000000000000 (t=49)
0.999999999702304800000000000000000000000000000000000 (t=50)
```

Note: “hrates.txt” is a file derived by the function derhrates() and is written for random T_D arrangements. It contains the discrete hazard rates for the person/MPU defined on the last line of the control file. (See Section 4.6.2 for reference.) The probabilities shown above are from the 5th “control.txt” example (MPU of size $N=9$). All random arrangements have a single set of discrete hazard rates which is the only random-specific input required by the random T_D model. Because an MPU is defined by a single hazard rate file the MPU size does not impact runtime. In the example above, $S_{Max}=50$ and a maximum of 50 decisions need to be made, the first at $t=0$ and the last at $t=S_{Max}-1=49$. No decision is made at time $t=S_{Max}$ since $P(T_D > S_{Max}) = 0$.

Output File: FinalResults_V.txt

```
-----
0 0.0016000000 0.0000000000000000000019272096920837062000000000000000 0.1980000000
0 0.0017000000 0.000000000000000000005785455286366354000000000000000 0.1990000000
0 0.0018000000 0.0000000000000000000016213529004089414000000000000000 0.1990000000
0 0.0019000000 0.0000000000000000000042735423336906475000000000000000 0.2000000000
.
.
.
10 0.0108000000 0.0000000022286488318214245000000000000000000000000000 0.2140000000
10 0.0109000000 0.0000000025603507963221465000000000000000000000000000 0.2140000000
10 0.0110000000 0.0000000029365856846809467000000000000000000000000000 0.2150000000
10 0.0111000000 0.0000000033626746638999980000000000000000000000000000 0.2150000000
.
.
.
20 0.0296000000 0.0000107948199183127730000000000000000000000000000000 0.2360000000
20 0.0297000000 0.0000112517285274006340000000000000000000000000000000 0.2360000000
20 0.0298000000 0.0000117253459734435400000000000000000000000000000000 0.2370000000
20 0.0299000000 0.0000122161610152366390000000000000000000000000000000 0.2370000000
.
.
.
30 0.0934000000 0.0077559399110792319000000000000000000000000000000000 0.3640000000
30 0.0935000000 0.0078073369296789057000000000000000000000000000000000 0.3640000000
30 0.0936000000 0.0078589480851500408000000000000000000000000000000000 0.3650000000
30 0.0937000000 0.0079107717918250314000000000000000000000000000000000 0.3650000000
.
.
.
40 0.0001000000 0.0000000000000000000000000000000000000000000085482880000 0.1440000000
40 0.0002000000 0.0000000000000000000000000000000000000000000088469110000 0.1440000000
40 0.0003000000 0.0000000000000000000000000000000000000000000091611610000 0.1440000000
40 0.0004000000 0.0000000000000000000000000000000000000000000094839560000 0.1440000000
.
.
.
45 0.4978000000 0.0887638500846292530000000000000000000000000000000000 0.4170000000
45 0.4979000000 0.0888303869479259430000000000000000000000000000000000 0.4180000000
45 0.4980000000 0.0888969095018619090000000000000000000000000000000000 0.4200000000
45 0.4981000000 0.0889634242548212610000000000000000000000000000000000 0.4210000000
.
.
.
46 0.5223000000 0.0477533568952275690000000000000000000000000000000000 0.8510000000
46 0.5224000000 0.0478043157876719760000000000000000000000000000000000 0.8530000000
46 0.5225000000 0.0478552141727582870000000000000000000000000000000000 0.8550000000
46 0.5226000000 0.0479060463999401160000000000000000000000000000000000 0.8560000000
.
.
.
47 0.8091000000 0.0010488399156190109000000000000000000000000000000000 0.1740000000
47 0.8092000000 0.0010529378732509127000000000000000000000000000000000 0.1740000000
47 0.8093000000 0.0010570492840191212000000000000000000000000000000000 0.1750000000
47 0.8094000000 0.0010611749159532891000000000000000000000000000000000 0.1750000000
```

Note: Select rows from the file “FinalResults_V.txt” are displayed. This is a vertical file written by the application containing all model output. The values shown are from the 6th “control.txt” example. Column definitions are: time point $[t]$, ruin factor $RF(t)$, minimum probability of ruin $V_R(t, RF(t))$, and optimal asset allocation $\alpha_R(t, RF(t))$. In addition, there are 2 horizontal files which together contain the same information as the single vertical file and these are named: “FinalAlphaResults_H.csv” & “FinalProbResults_H.csv”. To simulate and confirm the result use the file “FinalAlphaResults_H.csv”. A method for doing this is discussed in the next section.

Additional Notes:

- The source code provided has not been independently validated. The user should conduct full code validation before using it within a production environment.²⁰
- To simulate a specific solution load the file “FinalAlphaResults_H.csv” into Microsoft® Excel® and use a simulator such as Oracle® Crystal Ball®. A vlookup table is used to retrieve the optimal alpha at each time point.²¹ (Recall that each RF(t) centers its bucket.)
- The basic approach for coding the DP suggested in Figure 5 was altered to avoid unacceptable runtimes. Most coding effort was spent modifying the implementation to reduce runtimes which also has the undesirable effect of complicating the code.
- This implementation has been tested on the Windows 7 Professional and Windows 7 Home Edition operating systems. Some file management operations in the code are Windows-specific and would need to be updated for use on other operating systems.
- The code provided will produce a similar but slightly better solution than was presented in Figures 6, 7, and 8. Runtimes are now such that there is no longer a need to partition the RF(t) dimension when discretizing it as was done in those solutions.
- Increasing the RF(t) discretization precision P_R results in a more accurate numerical approximation whereas increasing the alpha precision P_α lowers the probability of ruin $V(t, RF(t))$.

²⁰ The author would greatly appreciate being informed of code errors/bugs as well as techniques to reduce runtimes.

²¹ A template for simulating the optimal solution will be supplied upon request.

References

- Anton, H. (1988). *Calculus with Analytic Geometry* 3rd Edition. *John Wiley & Sons*. New York, NY.
- Bengen, W. P. (1994). Determining withdrawal rates using historical data. *Journal of Financial Planning*, 7(4), 171.
- Bengen, W. P. (2006). Baking a withdrawal plan 'layer cake' for your retirement clients. *Journal of Financial Planning*, 19(8), 44-46,48-51.
- Blanchett, David M, (2007). Dynamic allocation strategies for distribution portfolios: Determining the optimal distribution glide-path. *Journal of Financial Planning*, 20(12), 68-70,72-81.
- Box, G.E.P., Jenkins, G.M., Reinsel, G.C. (1994). *Time Series Analysis Forecasting and Control* 3rd Edition. *Prentice Hall*. Englewood Cliffs, NJ.
- Bradley, Stephen P., Arnoldo C. Hax, and Thomas L. Magnanti. *Applied Mathematical Programming*. Reading, MA: Addison-Wesley Publishing Company, 1977.
- Brigham, E.F., Ehrhardt, M.C. (2008). *Financial Management Theory and Practice* 12th Edition. *South-Western CENGAGE Learning*. Mason, OH.
- Charlson, J., Herbst, M., Kailin, L., Lutton, L., Rekenenthaler, J. (2009) Target-Date Series Research Paper: Industry Survey. *Morningstar, Inc.*
- Cohen, J., Gardner, G., Fan, Y.A. (2010) The date debate: Should target-date fund glide-paths be managed "to" or "through" retirement? *Russell Research. Russell Investments*.
- Cooley, Philip L., Carl M. Hubbard, and Daniel T. Walz. (1998). Retirement Savings: Choosing a Withdrawal Rate That Is Sustainable. *American Association of Individual Investors Journal*, 10, 3: 16–21
- Cooley, P. L., Hubbard, C. M., & Walz, D. T. (2011). Portfolio success rates: Where to draw the line. *Journal of Financial Planning*, 24(4), 48-50,52-54,56-60.
- Fan, Y.A., Murray, S., and Pittman, S. Optimizing Retirement Income: An Adaptive Approach Based on Assets and Liabilities. *The Journal of Retirement*, Vol. 1, No. 1 (Summer 2013), pp. 124-135.
- Guyton, J. T. (2004). Decision rules and portfolio management for retirees: Is the 'safe' initial withdrawal rate too safe? *Journal of Financial Planning*, 17(10), 54-54+.
- ING Retirement Research Institute. (2012) Participant Preferences in Target-Date Funds: Examining Perceptions and Expectations. (Link: <http://ing.us/rri/ing-studies/target-date-funds>)

Markowitz, H. (1952), PORTFOLIO SELECTION. *The Journal of Finance*, 7: 77–91. doi: 10.1111/j.1540-6261.1952.tb01525.x.

Milevsky, M. A., & Robinson, C. (2005). A sustainable spending rate without simulation. *Financial Analysts Journal*, 61(6), 89-100.

Pye, G. B. (2000). Sustainable investment withdrawals. *Journal of Portfolio Management*, 26(4), 73-83.

Stout, R. G. (2008). Stochastic optimization of retirement portfolio asset allocations and withdrawals. *Financial Services Review*, 17(1), 1-15.

Pfau, Wade D, Kitces, Michael E, (2014). Reducing retirement risk with a rising equity glide-path. *Journal of Financial Planning*, 27(1), 38-45.