Approximating a life table by linear combinations of exponential distributions and valuing life-contingent options

Zhenhao Zhou

Department of Statistics and Actuarial Science
The University of Iowa
Iowa City, Iowa 52242

A key idea used in Gerber, Shiu and Yang (2012, 2013) is that the distribution of T_x, the time-until-death random variable, can be approximated by linear combinations of exponential distributions. In the accompanying folder, I have provided MATLAB functions for determining such approximations when T_x is prescribed by a life table. I also give MATLAB functions for valuing life-contingent options using the approximations. Here, the stock price process is modeled as the exponential of a Brownian motion plus an independent compound Poisson process. More specifically, the compound Poisson process is the sum of two independent compound Poisson processes, one for upward jumps and one for downward jumps. The jump distributions are combinations of exponential distributions.

First we make a choice for n, the number of exponential distributions. Then we seek the parameters $\alpha_1, \cdots, \alpha_n, \lambda_1, \cdots, \lambda_n,$, which minimize the weighted sum of squares,

$$\sum_{k\geq 1} w_k \left[{}_kp_x - \sum_{j=1}^{n} \alpha_j e^{-\lambda_j k} \right]^2 ,$$

subject to

$$\sum_{j=1}^{n} \alpha_j = 1$$

and $\lambda_1 > 0, \cdots, \lambda_n > 0$. We start with an initial guess of a set of $\lambda_1, \cdots, \lambda_n$. Then we use linear regression to solve for $\alpha_1, \cdots, \alpha_n$. With these α's, we use the "trust region algorithm" to come up with the next set of $\lambda_1, \cdots, \lambda_n$. Then we again use linear regression to solve for $\alpha_1, \cdots, \alpha_n$. And so on.

For life-contingent options, we assume that the stock price is modeled as

$$S(t) = S(0)e^{X(t)}, \quad t \geq 0,$$

where $\{X(t)\}$ is a Brownian motion (with drift and diffusion parameters μ and σ) extended by independent jumps in both directions. The downward jumps form an independent compound Poisson process; the frequency of these jumps is υ. Similarly, the upward jumps form another independent compound Poisson process with Poisson parameter ω. The pdf of each downward jump is

$$\sum_{j=1}^{m} A_j v_j e^{-v_j x}, \quad x > 0,$$

with $\sum_{j=1}^{m} A_j = 1$ and $0 < v_1 < v_2 < \ldots v_m$, and the pdf of each upward jump is

$$\sum_{k=1}^{n} B_k w_k e^{-w_k x}, \quad x > 0,$$

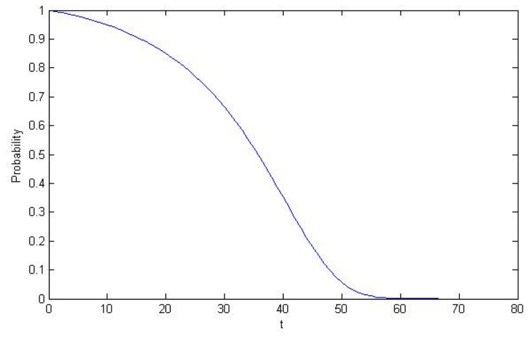
with $\sum_{k=1}^{n} B_k = 1$ and $0 < w_1 < w_2 < \ldots w_n$.

Under the assumption above, we use the formulas given in Gerber, Shiu and Yang (2013) to price life-contingent options numerically. All computations are performed with MATLAB, and the description of the functions are provided in the Appendix.

Numerical example

As an example, we use the data from `http://www.ssa.gov/oact/STATS/table4c6.html` to illustrate how the MATLAB functions work. Using the function y = empirical_distribution(t, data, 45), we can calculate the empirical probability $P(T_{45} > t)$.

Figure 1: Empirical distribution of $P(T_{45} > t)$



Because

$$\mathrm{E}[e^{-r\tau} S(\tau)] = S(0)\frac{\lambda}{\lambda + r - \Psi(1)}$$

where τ is an exponential random variable with parameter λ and

$$\Psi(z) = \mu z + \frac{1}{2}\sigma^2 z^2 - \nu \sum_{i=1}^{m} A_i \frac{z}{v_i - z} + \omega \sum_{i=1}^{m} B_i \frac{z}{w_i + z},$$

we need $\lambda > \Psi(1) - r$ to make the expectation above exist. For example, we choose the parameters as follows: $\mu = 0.01$; $r = 0.01$; $\sigma = 0.1$; $B = [0.2,\ 0.5,\ 0.3]$; $A = [0.2,\ 0.5,\ 0.3]$; $w = [\frac{1}{0.03},\ \frac{1}{0.02},\ \frac{1}{0.01}]$; $v = [\frac{1}{0.03},\ \frac{1}{0.02},\ \frac{1}{0.01}]$; $\omega = 20$; $\nu = 20$; and we have $\Psi(1) = 0.0314$. We can start with the initial guess of $\lambda_0 = [0.05,\ 0.1,\ 0.2,\ 0.5]$, using the function [lambda,linear_coeff] = fit_nonlinear(λ_0, t, data, 45, 0.0314); we obtain that the coefficients of the fitting distribution are

$$\alpha = [0.201 \times 10^7,\ 4.5134 \times 10^7,\ -4.6673 \times 10^7,\ -0.0471 \times 10^7]$$

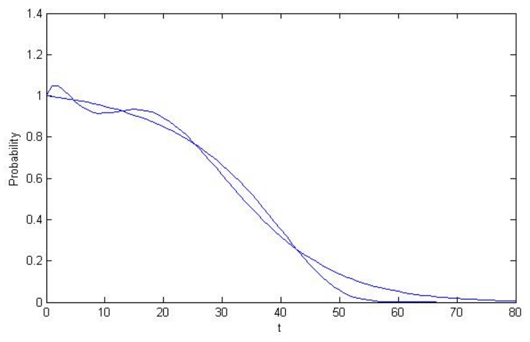
and the exponential parameters are

$$\lambda = [0.1526,\ 0.1507,\ 0.1508,\ 0.1546]$$

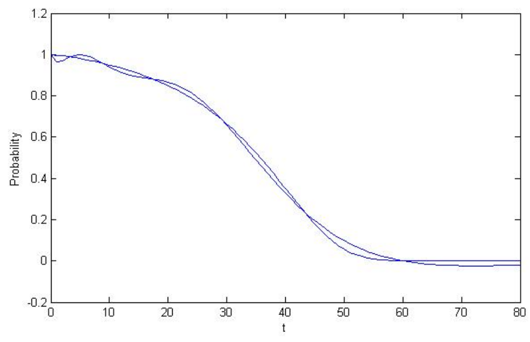
In Figure 2, we show the fitting results using linear combinations of 4, 6, 8 and 10 terms of exponential distributions. We can see the fitting result is good when we use 8 terms.

Figure 2: Fitting the empirical distribution of $P(T_{45} > t)$

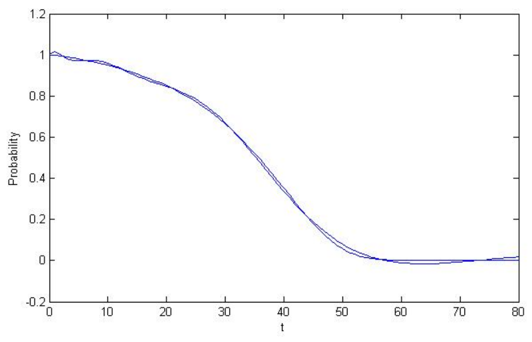
(a) 4 terms



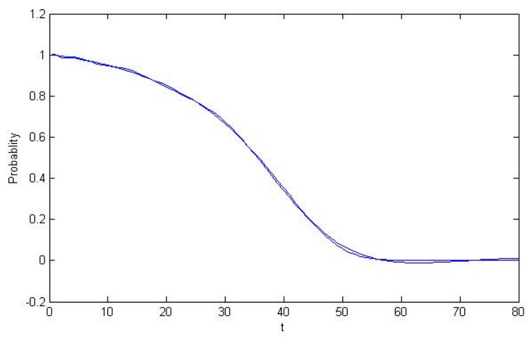
(b) 6 terms



(c) 8 terms



(d) 10 terms



In order to illustrate how the fitting results can be applied to option pricing, we price a call and a put option. We let the initial stock price $S(0) = 200$ and strike price $K = 205$. Using 4 terms of exponential distributions, we calculate that an approximate value of the call option is 215.7352, and for the put 20.3582. Using 6 terms of exponential distributions, we obtain 205.0381 as an approximate value of the call option, and 19.8565 for the put. Using 8 terms of exponential distributions, we obtain 218.1054 as an approximate value of the call option, and 20.0780 for the put. Using 10 terms of exponential distributions, we obtain 212.0643 as an approximate value of the call option, and 20.1211 for the put. Test Codes are provided in the package.

Appendix

Syntax:

y = empirical_distribution(t, data, x)

Description:

y = empirical_distribution(t, data, x) returns an array of empirical survival distribution using the data in the vector data. Specifically, it calculates the quantity ${}_tp_x$, which is the probability that (x) survives to age x+t.

Input argument:

data: Column vector which represents the number of surviving people at each age. We can find this data from life table.

x: a number which represents the age of a person.

t: a number which represents how many more years (x) will survive.

Syntax:

[lambda,linear_coeff] = fit_nonlinear(lambda_0, t, data, x, lb)

Description:

[lambda,linear_coeff] = fit_nonlinear(lambda_0, t, data, x, lb) returns linear coefficient $\frac{\alpha_i}{\lambda_i}$ and exponential coefficient λ_i of the fitting problem.

Input argument:

lambda_0 : a vector which represents the initial guess of λ .

data: column vector which represents the number of surviving people at each age. We can find this data from life table.

x: a number which represents the age of a person.

t: a number which represents how many more years (x) will survive.

lb: a number which represents the lower bound of lambda, which makes the expectation of stock price exist.

Syntax:

s = life_stock_expect(mu, sigma, up_weight, down_weight, up_exp, down_exp, up_freq, down_freq, lambda, S_initial, linear_coeff, interest_rate) ;

Description: s = life_stock_expect() calculates the value of $\mathrm{E}[e^{-r\mathrm{T}_x} S(\mathrm{T}_x)]$.

[put, call] = life_vanilla(mu, sigma, up_weight, down_weight, up_exp, down_exp, up_freq, down_freq, lambda, K, S_initial, linear_coeff, interest_rate) ;

Description: [put, call] = life_vanilla() calculates the price of call option $\mathrm{E}[e^{-r\mathrm{T}_x}\,[S(\mathrm{T}_x) - K]_+]$ and put option $\mathrm{E}[e^{-r\mathrm{T}_x}\,[K - S(\mathrm{T}_x)]_+]$.

[put, call]= life_fixed_lookback(mu, sigma, up_weight, down_weight, up_exp, down_exp, up_freq, down_freq, lambda, K, H, S_initial, linear_coeff, interest_rate) ;

Description: [put, call] = life_fixed_lookback() calculates the price of fixed-strike lookback call option $\mathrm{E}[e^{-r\mathrm{T}_x}\left[\max(H, \max\limits_{0\le t\le \mathrm{T}_x} S(t)) - K\right]_+]$ and fixed-strike lookback put option

$\mathrm{E}[e^{-r\mathrm{T}_x}\left[K - \min(H, \min\limits_{0\le t\le \mathrm{T}_x} S(t))\right]_+]$.

[name, price] = life_float_lookback(mu, sigma, up_weight, down_weight, up_exp, down_exp, up_freq, down_freq, lambda, H, S_initial, linear_coeff, interest_rate) ;

Description: [name, price] = life_float_lookback() calculates the the price of float-strike lookback call option $\mathrm{E}[e^{-r\mathrm{T}_x}\left[S(\mathrm{T}_x) - \min(H, \min\limits_{0\le t\le \mathrm{T}_x} S(t))\right]_+]$ with the condition that S_initial is larger than H and float-strike lookback put option $\mathrm{E}[e^{-r\mathrm{T}_x}\left[\max(H, \max\limits_{0\le t\le \mathrm{T}_x} S(t)) - S(\mathrm{T}_x)\right]_+]$ with the condition that S_initial is smaller than H.

[name, price] = life_fractional_float_lookback(mu, sigma, up_weight, down_weight, up_exp, down_exp, up_freq,down_freq, lambda, gamma, S_initial, linear_coeff, interest_rate) ;

Description: [name, price] = life_fractional_float_lookback() calculates the the price of fractional float-strike lookback call option $\mathrm{E}[e^{-r\mathrm{T}_x}\left[S(\mathrm{T}_x) - \gamma \min\limits_{0\leq t\leq \mathrm{T}_x} S(t)\right]_+]$ with the condition that S_initial is larger than H and fractional float-strike lookback put option $\mathrm{E}[e^{-r\mathrm{T}_x}\left[\gamma \max\limits_{0\leq t\leq \mathrm{T}_x} S(t) - S(\mathrm{T}_x)\right]_+]$ with the condition that S_initial is smaller than H.

[put, call] = life_knock_in(mu, sigma, up_weight, down_weight, up_exp, down_exp, up_freq, down_freq, lambda, K, L, S_initial, linear_coeff, interest_rate) ;

Description: If the condition that S_initial is larger than L holds, [put, call]= life_knock_in() calculates the the price of up-and-in call option $\mathrm{E}[e^{-r\mathrm{T}_x} I(\max\limits_{0\leq t\leq \mathrm{T}_x} S(t) \geq L)\,[S(\mathrm{T}_x) - K]_+]$ and up-and-in put option $\mathrm{E}[e^{-r\mathrm{T}_x} I(\max\limits_{0\leq t\leq \mathrm{T}_x} S(t) \geq L)\,[K - S(\mathrm{T}_x)]_+]$

[put, call] = life_knock_out(mu, sigma, up_weight, down_weight, up_exp, down_exp, up_freq, down_freq, lambda, K, L, S_initial, linear_coeff, interest_rate) ;

Description: If the condition that S_initial is smaller than L holds, [put, call]= life_knock_out() calculates the the price of up-and-out call option $\mathrm{E}[e^{-r\mathrm{T}_x} I(\max\limits_{0\leq t\leq \mathrm{T}_x} S(t) \leq L)\,[S(\mathrm{T}_x) - K]_+]$ and up-and-out put option $\mathrm{E}[e^{-r\mathrm{T}_x} I(\max\limits_{0\leq t\leq \mathrm{T}_x} S(t) \leq L)\,[K - S(\mathrm{T}_x)]_+]$

Input argument:

mu: a number which represents the drift of asset return;

sigma: a number which represents the volatility of asset return;

up_weight: a row vector which represents the weight of different exponential distributions in each up-jump of asset return;

down_weight: a row vector which represents the weight of different exponential distributions in each down-jump of asset return;

up_exp: a row vector which represents the parameters of different exponential distributions in each up-jump of asset return;

down_exp: a row vector which represents the parameters of different exponential distributions in each down-jump of asset return;

up_freq: a number which represents the frequency of Poisson process of upwards jumps;

down_freq: a number which represents the frequency of Poisson process of downwards jumps;

lambda: a row vector which represents the exponential coefficients of the fitted distribution;

linear_coeff: a row vector which represents the linear coefficients of the fitted distribution;

K: a number which represents the strike price of call or put options;

interest_rate: a number which represents the risk free rate;

L: a number which represents the barrier of a barrier option;

H: a number which represents the maximum or minimum level of the stock's historical price of lookback options;

Gamma: a number which represents the parameter in floating-strike lookback;