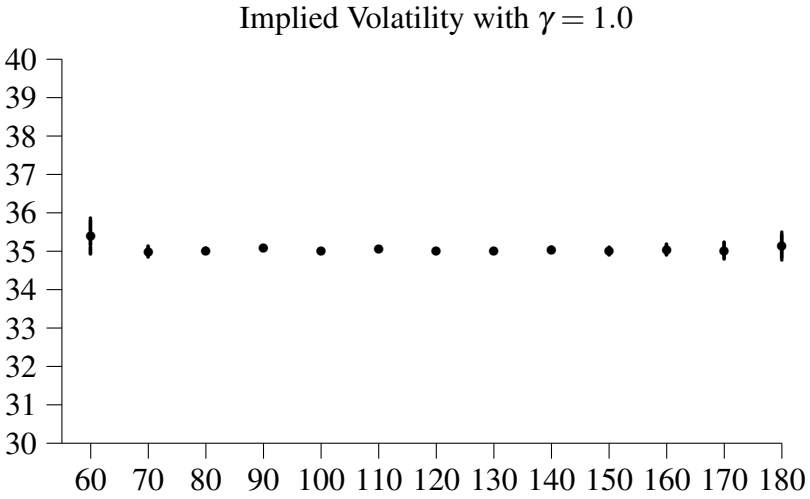
MTH 4600 - Project 7
Option Valuation with Stochastic Volatility

Martin Bachurin
Anton Gurshumov
John Hemminger

Baruch College

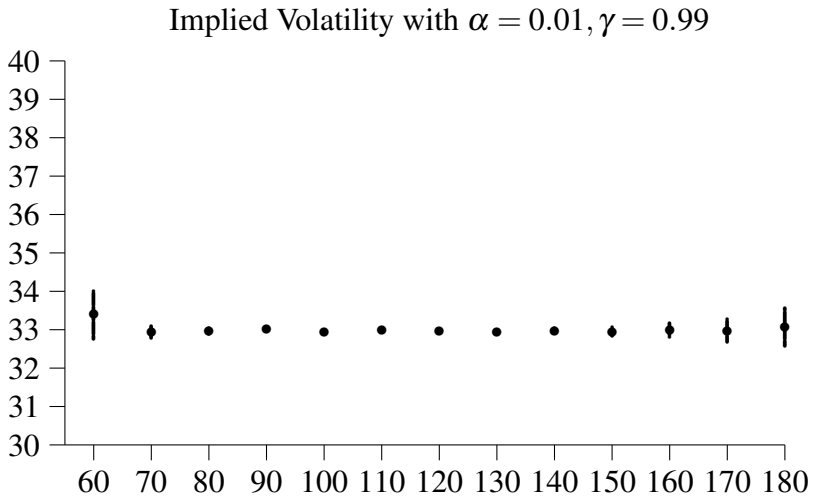
May 1, 2023

For question one, we used the GARCH(1,1) model to price the value of a call option. We simulated a million realizations of the half-year price path and discounted it back to time zero using strike 0. This resulted in the call option value being 100.00 +/- 0.01. This will be the framework of pricing used for the following problems, where we calculate the implied volatility of the derivative using various strike prices and parameters of calculating the daily volatility.

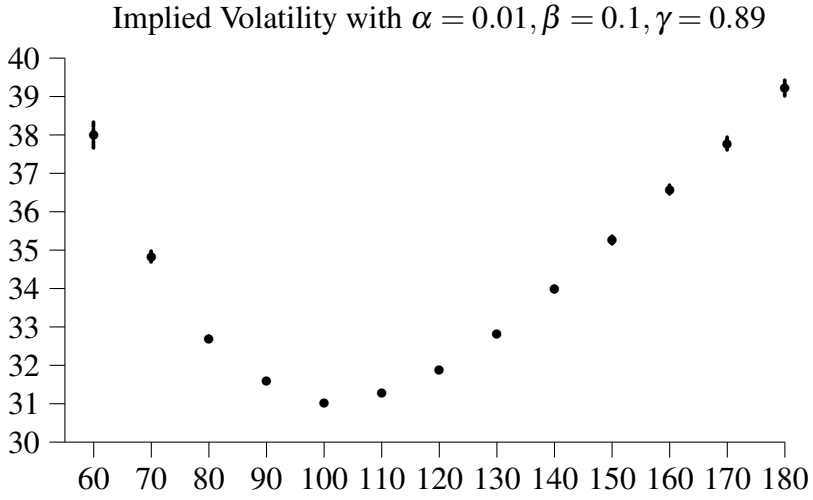


For this problem, we plotted the implied volatility against various strike prices and yielded the following results. The plot shows that the implied volatility is around 35%, with very little variation in terms of the strike price. The reasoning behind the implied volatility having little variation in terms of the strike price is caused by the Black-Scholes volatility framework, where the implied volatility is held as a constant. This is seen in the figure above. Additionally, the stock price evolution is based solely on historical data under these parameters. Hence the daily volatility remains the same as we move through time. Note that although the implied volatility does not change much, the confidence interval is more tightly grouped around strikes that are closer to being at the money. We believe this is caused by the error of calculation being magnified at more extreme values of the strike price because the price of call options as a function of the strike is convex. Therefore, the further we are from being at the money, the greater the rate of change is of the price of the call option. This causes the implied volatility calculation to be less accurate in these areas. For example, the price of a call being 0.13 cents +/- 0.01 carries more of an impact when measuring the implied volatility compared to the price of a call that is at the money. This effect carries on as we look at the remaining problems.

The figure below describes a process similar to the first for the second problem. However, we are now considering an anchored daily volatility into consideration. Visually this causes the implied volatility to be constant at around 33%, shifting the constant down from 35%. This makes sense because in each step of calculating the daily volatility under GARCH(1,1), σ_{LT}, is set to be 30%; Therefore, with each iteration, we take into account 1% of σ_{LT}, causing us to trend to 30% at a slow rate. The realization of the stock price path, the call option price, is still living in the Black-Scholes volatility framework and is not realistic to the market interpretation of the asset prices. Therefore the implied volatility appears to be constant over various strike prices.



For this problem, we can see that the daily implied volatility looks very different for various strike prices. This shows that we do not approach a constant implied volatility as before since our GARCH(1,1) model significantly depends on the most recent volatility data when pricing the call option. This is caused by the evolution of the underlying stock itself. The real-world feature this results in is a volatility model that looks more realistic and usable for various strike prices. The figure shown is commonly known as a volatility smile. For example, we can find a minimum implied volatility of 31%, which happens to be when the call price is at the money. This figure also implies that we can no longer use the Black-Scholes framework to model all option values price because volatility is no longer held constant.



Our group separated the work relatively uniformly over the interval of time required to complete the assignment. We met on a regular basis and worked through each problem at a time, always listening to criticism and different points of view. We all coded together, and if something was unclear to someone when writing the code, we always explained precisely what was happening line by line. Our group has a great interest in these projects and subject matter; therefore, we have had no issues with scheduling and collaborating as a team. We genuinely enjoy working on these projects together.

Appendix

Listing 1: Code:1

```
int main () {

   int i, j, N;
   double s_start, s, T, r, sigma, sigma2,  sigma2_start, mu, alpha, beta,
          gamma, dt, Z, R, U, sigma2_LT;
   double s_til,s_anti,s_bar,s_2bar,callPrice, s_final;
   double K = 0.0;
   double n =0.0, discount;
   double C, Ctil;
   double S0,C0, C0til,X,C0bar = 0.0 ,C02bar =0.0;
   double stdhatC,error, elapsed_time, t_star, epsilon;
   int counter, test , done;

   // Specify the alpha, beta, and gamma parameters.
   j = GetInteger ("Which problem (2, 3, or 4) are you working on?... ");
   if (j == 2) {
      alpha = 0.0; beta = 0.0; gamma = 1.0;
   }
   else if (j== 3) {
      alpha = 0.01; beta = 0.0; gamma = 0.99;
   }
   else {
      alpha = 0.01; beta = 0.10; gamma = 0.89;
   }
   // Time to expiration.

   T = 0.5;

   // Risk-free interest rate.
   r = 0.05;

   // Discount factor
   discount = exp(-r*T);

   // Number of days to expiration (252 days per year), and one day in years.
   N = 252 * T;
   dt = 1.0/252.0;

   // Convert "r" to a daily risk-free rate.
   r *= dt;

   // Current stock price.
   s_start = 100.0;

   // Current stock price variance.
```

```
sigma2_start = 0.35 * 0.35;

// Current daily stock price variance.
sigma2_start *= dt;

// Annual long-term variance.
sigma2_LT = 0.30 * 0.30;

// Daily long-term variance.
sigma2_LT *= dt;

// Seed the RNG.
MTUniform (1);

Time();
// Generate and display 5 illustrative paths.

//}
for (j = 1; j <= 1000000; j++) {

   // Initialize the stock price and the volatility to their starting (time 0)
   //    values.
   s = s_start;
   s_til = s_start;
   sigma2 = sigma2_start;

   // Now march forward in time day by day.
   for (i = 1; i <= N; i++) {

      // Compute the drift parameter that corresponds to the volatility for
      //    this period.
      mu = r - 0.5*sigma2;

      // Compute the stock price at day "i"...

      // First get a standard normal Z.
      U = MTUniform (0);
      Z = PsiInv (U);

      // Apply current volatility.
      sigma = sqrt(sigma2);
      R = sigma * Z;

      // Update the stock price.
      s *= exp (mu + R);

      s_til *= exp (mu - R);

      // Update the stochastic volatility according to the GARCH(1,1) model.
```

```
            sigma2 = alpha * sigma2_LT  +  beta * R*R +  gamma * sigma2;

            if(i==N){
                if(s - K > 0){C = s - K;}
                    else{C = 0;}
                if(s_til - K > 0){Ctil = s_til - K;}
                    else{Ctil = 0;}
            }
        }

        // Discounting the call price to time 0, and calculating the antithetic call price
        C0 = discount * C;
        C0til = discount * Ctil;
        X = (C0 + C0til) / 2.0;

        // Computing sample moments
        C0bar  = ((j-1)*C0bar + X) / j;
        C02bar  = ((j-1)*C02bar + X*X) / j;

        // Computing the error, and displaying results
        error = 1.96 * sqrt((C02bar-C0bar*C0bar)/1000000.0);
        printf ("Expected present value is %6.2f +/- %6.2f ", C0bar, error);
        printf ("with 95%% confidence.\n");

    }
    Exit();
}
```

Appendix

Listing 2: Code:2-4

```
int main () {

   int i, j, N;
   double s_start, s, T, r, sigma, sigma2,  sigma2_start, mu, alpha, beta,
          gamma, dt, Z, R, U, sigma2_LT;
   double s_til,Isigma_low,Isigma_high,Isigma;
   double K = 50.0;
   double n =0.0, discount;
   double C,Ctil;
   double S0,C0, C0til,X,C0bar = 0.0 ,C02bar =0.0;
   double stdhatC,error, elapsed_time, t_star, epsilon;
   int counter, test , done;

   // Specify the alpha, beta, and gamma parameters.
   j = GetInteger ("Which problem (2, 3, or 4) are you working on?... ");
   if (j == 2) {
      alpha = 0.0; beta = 0.0; gamma = 1.0;
   }
   else if (j== 3) {
      alpha = 0.01; beta = 0.0; gamma = 0.99;
   }
   else {
      alpha = 0.01; beta = 0.10; gamma = 0.89;
   }
   epsilon = 0.01;
   // Time to expiration.

   T = 0.5;

   // Risk-free interest rate.
   r = 0.05;

   // Discount factor
   discount = exp(-r*T);
   // Number of days to expiration (252 days per year), and one day in years.
   N = 252 * T;
   dt = 1.0/252.0;

   // Convert "r" to a daily risk-free rate.
   r *= dt;

   // Current stock price.
   s_start = 100.0;

   // Current stock price variance.
```

```
sigma2_start = 0.35 * 0.35;

// Current daily stock price variance.
sigma2_start *= dt;

// Annual long-term variance.
sigma2_LT = 0.30 * 0.30;

// Daily long-term variance.
sigma2_LT *= dt;

// Seed the RNG.
MTUniform (1);

Time();

// Increment the strike price for each simulation

for(int step = 1; step <= 13; ++step){
     K += 10.0;
     for (j = 1; j <= 1000000; j++) {

          // Initialize the stock price and the volatility to their starting (time 0)
          //    values.
          s = s_start;
          s_til = s_start;
          sigma2 = sigma2_start;

          // Now march forward in time day by day.
          for (i = 1; i <= N; i++) {

               // Compute the drift parameter that corresponds to the volatility for
               //    this period.
               mu = r - 0.5*sigma2;

               // Compute the stock price at day "i"...

               // First get a standard normal Z.
               U = MTUniform (0);
               Z = PsiInv (U);

               // Apply current volatility.
               sigma = sqrt(sigma2);
               R = sigma * Z;

               // Update the stock price.
               s *= exp (mu + R);

               s_til *= exp (mu - R);
```

```
                // Update the stochastic volatility according to the GARCH(1,1) model.
                sigma2 = alpha * sigma2_LT  +  beta * R*R +  gamma * sigma2;

                if(i==N){
                    if(s - K > 0){C = s - K;}
                    else{C = 0;}
                    if(s_til - K > 0){Ctil = s_til - K;}
                    else{Ctil = 0;}

                }

            }

  // Discounting the call price to time 0, and calculating the antithetic call price
            C0 = discount * C;
            C0til = discount * Ctil;
            X = (C0 + C0til) / 2.0;

            // Compute sample moments
            C0bar  = ((j-1)*C0bar + X) / j;
            C02bar  = ((j-1)*C02bar + X*X) / j;

        }
        // Compute error and display empirical call price at time 0.
        error = 1.96 * sqrt((C02bar-C0bar*C0bar)/1000000.0);

        std::cout<<" with K = "<<K<<"\n";
        printf ("Expected present value is %6.2f +/- %6.2f ", C0bar, error);
        printf ("with 95%% confidence.\n\n");
        std::cout<<" with K = "<<K<<"\n";

        // Compute the implied volitility with the current strike.

        Isigma = ImpliedVol (T,s_start, K, 0.05,C0bar);

  // Compute the confidence interval with the current strike, and display the results.

        Isigma_low = ImpliedVol (T, s_start, K, 0.05, C0bar - 0.01);
        Isigma_high = ImpliedVol (T, s_start, K, 0.05, C0bar + 0.01);
        std::cout<<"Implied volitility when K = "<<K<<" "<<Isigma * 100<<"\n\n";
        std::cout<<"CI when K = "<<K<<" "<<Isigma_low * 100<<" "<<Isigma_high * 100<<"\n";
    }
    Exit();
}
```