

Excel Compiler User Guide

Contents

1	Overview	1
2	Installation	2
3	Basic Examples	3
3.1	Compiling a Monte Carlo Simulation	3
3.2	Classic Monte Carlo.....	5
3.3	Compiling a Deterministic Function.....	6
4	Financial Application of Monte Carlo Simulation: Pricing an Asian Call Option	8

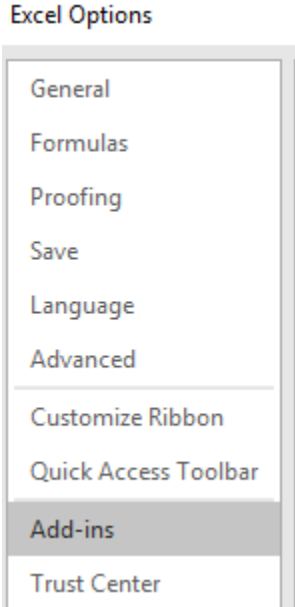
1 Overview

- Primary Uses:
 - Transcribing Excel functions into VBA code.
 - Running large Monte Carlo simulations quickly.
- Monte Carlo Simulation:
 - Repeated sampling of random variables used to understand a distribution of possible outcomes.
 - Each random sample has its associated outcome incorporated into a distribution. This distribution's statistical properties can be observed and studied.
 - The more random samples taken, the more accurate the estimates of the actual distribution's statistical parameters are.
- Example of a Monte Carlo Application:
 - Evaluating the effects of uncertainty in different market parameters on the value of a portfolio of financial instruments. Index and stock prices, along with interest rates and variance, can all be modelled using random processes simulated many times.
- Benefits of Using Excel Compiler:
 - Allows for a wide variety of Excel calculations, both random and deterministic, to be transcribed into VBA code.
 - Faster than Classic Monte Carlo method based on reiterated Excel spreadsheet calculation.
- Note:
 - If the compiled procedure's VBA code exceeds 64kb size, Excel will return an error statement saying 'Procedure too large'. In this case, the code needs to be manually broken down into two or more smaller procedures.
 - Whenever a random variable determines a compiled cell's value, Excel Compiler writes a VBA subroutine for the cell. Whenever a deterministic function determines a compiled cell's value, Excel Compiler writes a VBA function for the cell.

Excel Compiler User Guide

2 Installation


1. Select the File tab, then Options and Add-ins.



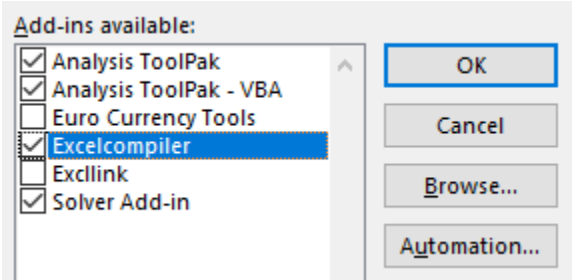
2. In the Manage section at the bottom, select Excel Add-ins and then click Go.



3. On the right-hand side of the window, select Browse, and open the Excel Compiler file (ExcelCompiler.xlam).

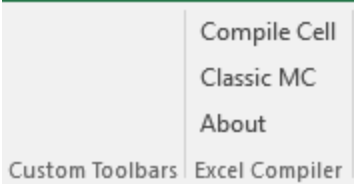
Name	Date modified	Type	Size
 ExcelCompiler.xlam	5/17/2018 9:54 AM	Microsoft Excel A...	44 KB

4. Excel Compiler will now be shown in the list of available Add-ins. Select the checkbox, and click OK.



5. Excel Compiler is now displayed in the Add-ins tab of Excel:

Excel Compiler User Guide



3 Basic Examples

3.1 Compiling a Monte Carlo Simulation

1. Open an Excel spreadsheet. In cells A1 through A10, input the formula = **NORMSINV(RAND())**. This creates a sample of 10 values from the standard normal distribution.

The screenshot shows an Excel spreadsheet with the formula bar containing `=NORMSINV(RAND())`. The spreadsheet has columns A through J and rows 1 through 10. Column A contains 10 values: -0.26934, -0.58443, 0.673984, 0.418396, 1.094208, 0.081465, -0.64705, 1.452795, 0.667268, and -1.02207.

	A	B	C	D	E	F	G	H	I	J
1	-0.26934									
2	-0.58443									
3	0.673984									
4	0.418396									
5	1.094208									
6	0.081465									
7	-0.64705									
8	1.452795									
9	0.667268									
10	-1.02207									

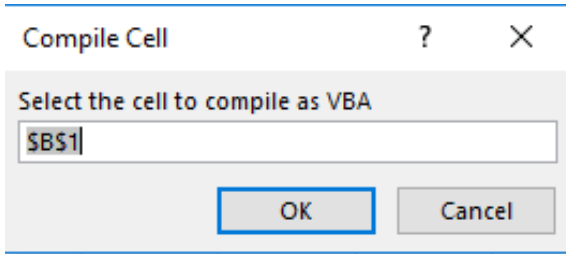
2. In cell B1, calculate the sum of cells A1 through A10.

The screenshot shows the same Excel spreadsheet as above, but now cell B1 contains the value -2.69356. The formula bar shows `=SUM(A1:A10)`. The values in column A are the same as in the previous screenshot.

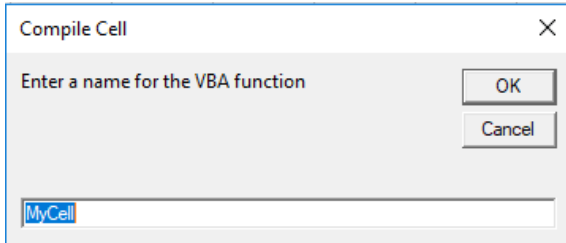
	A	B	C	D	E	F	G	H	I	J
1	1.783221	-2.69356								
2	-1.85719									
3	-0.66211									
4	-0.96568									
5	-0.51663									
6	-1.27314									
7	0.900763									
8	-0.42135									
9	0.371073									
10	-0.05253									

3. In the Add-Ins tab, select the “Compile Cell” button. Select cell B1 to compile as VBA.

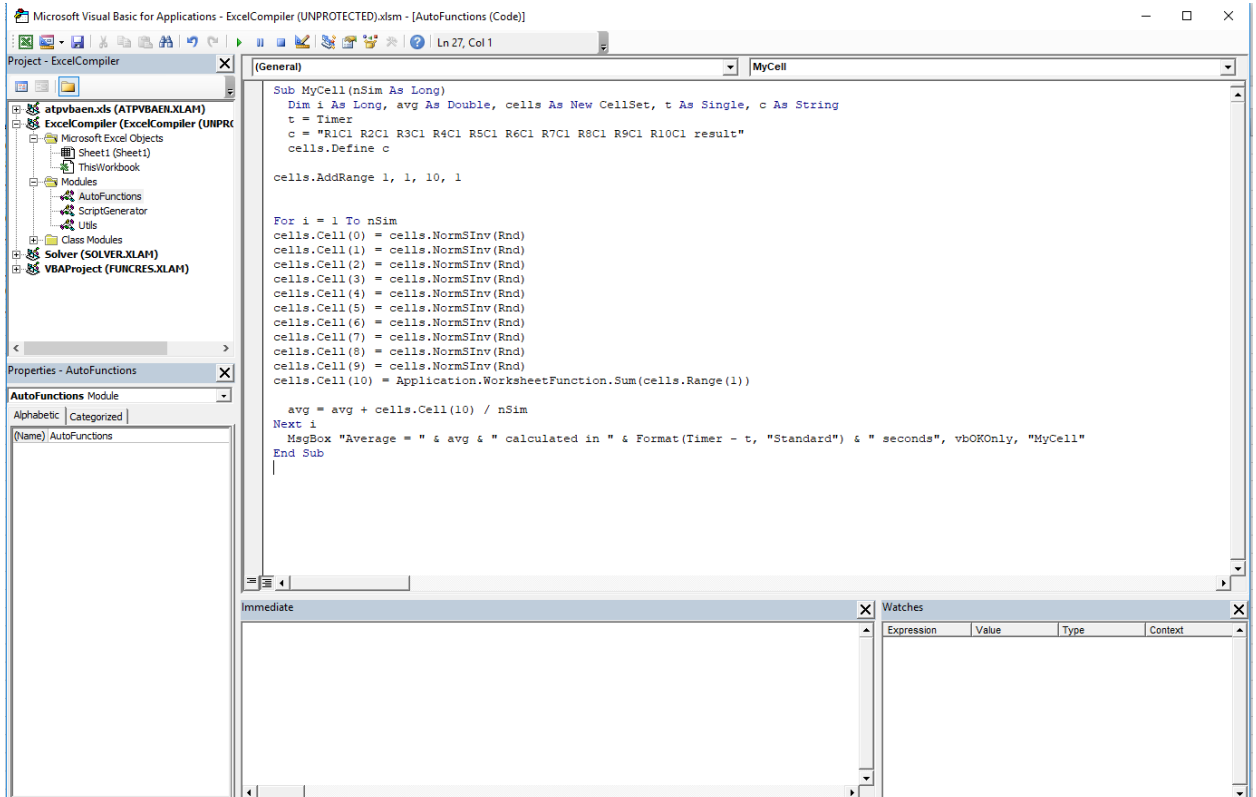
Excel Compiler User Guide



4. Choose any name for the VBA subroutine.



5. After the cell is compiled successfully, go to the Visual Basic editor in the Developer tab. In the 'AutoFunctions' VBA module of your spreadsheet, the cell is now written in VBA code as a procedure with the name previously specified.

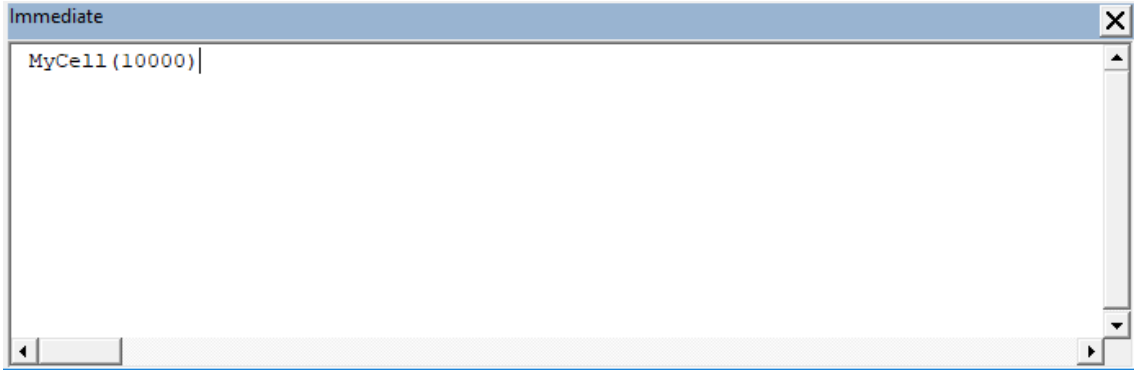


The above subroutine takes the number of simulations `nSim` as parameter. The code is split between a header section and a main section. In the header section each cell is defined by its row and column number (for example, cell A2 is defined as `R2C1`), then ranges are defined. In

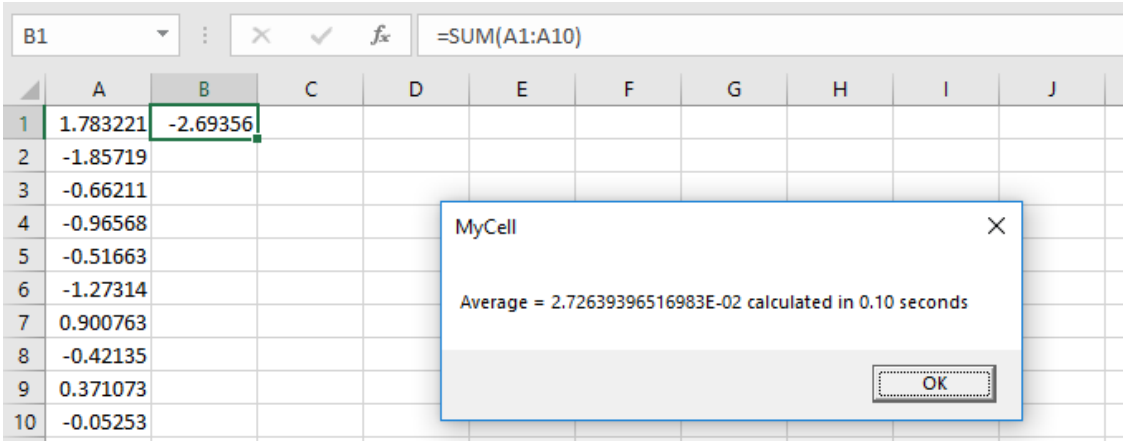
Excel Compiler User Guide

the main section, Monte Carlo simulations are executed within a loop. Note how the target cell (cells.Cell(10)) is calculated as the sum of the range (cells.Range(1)) corresponding to cells A1 through A10. The mean variable avg is then updated. On the last simulation, avg is equal to the actual average of all the compiled cell simulations.

6. To run the Monte Carlo simulations, select the Immediate window and type the name of the subroutine followed by a space and the number of simulations to be performed.



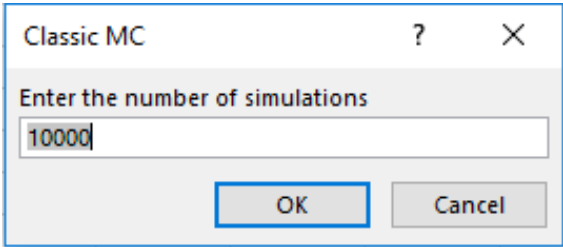
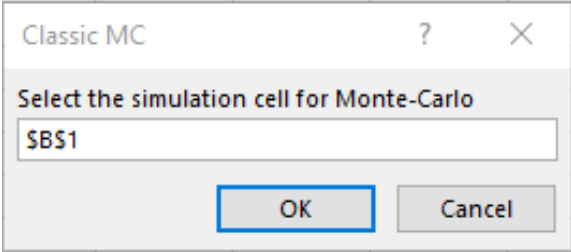
7. After you press *Enter*, the average of all of the simulations performed along with the time elapsed appears in the message box. In this example, given that the standard normal distribution is centered at zero, the average (shown below) is close to zero for a large number of simulations.



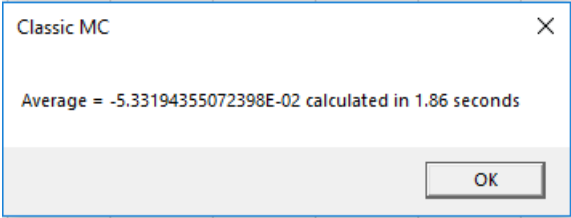
3.2 Classic Monte Carlo

1. In some cases Excel Compiler will fail to produce code. As long as your spreadsheet is working properly, you can still run Monte Carlo simulations using the "Classic MC" button:

Excel Compiler User Guide



- 2. In this example we can compare the time between the compiled and classic methods. On an Intel Xeon 3.1GHz processor, the compiled method is about 19 times faster. In complex simulations, the speed gain can be even larger.



3.3 Compiling a Deterministic Function

- 1. In cells A1 through A5, input any set of numbers.
- 2. Make cells B1 through B5 the square of their adjacent values in the A column.

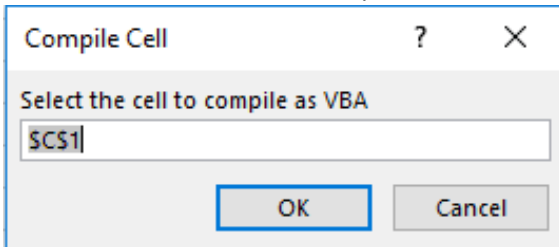
	A	B	C	D	E	F	G	H	I
1	2	4							
2	3	9							
3	4	16							
4	9	81							
5	4	16							

- 3. In cell C1, create a function that returns the natural logarithm of the sum of cells B1 through B5.

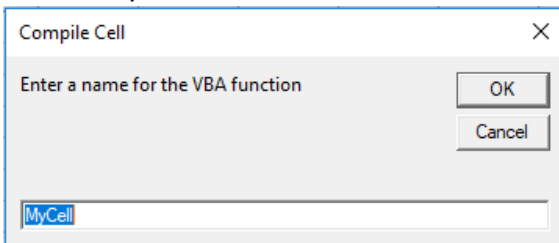
Excel Compiler User Guide

	A	B	C	D	E	F	G	H	I
1	2	4	4.836282						
2	3	9							
3	4	16							
4	9	81							
5	4	16							

4. In the Add-Ins tab, select Compile Cell. Select cell C1 to compile as VBA.



5. Choose any name for the VBA function.

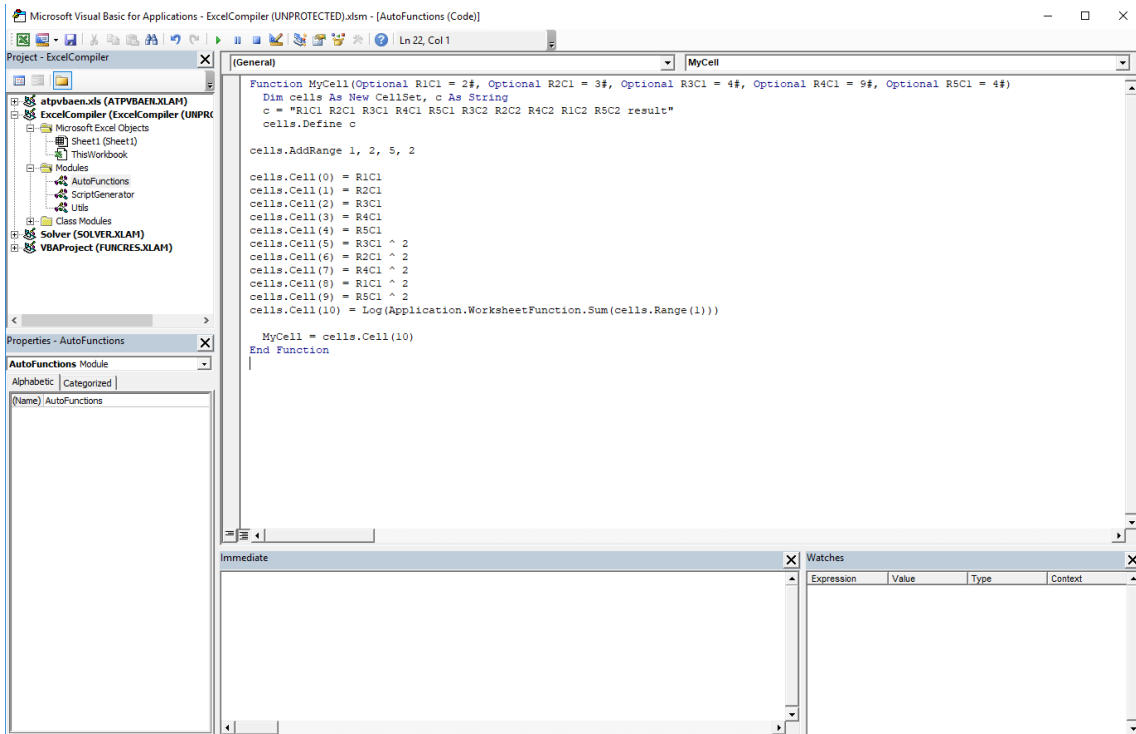


6. The newly created function can be invoked directly from the spreadsheet. Notice how the function automatically returns the natural logarithm of the sum of cells B1 through B5.

	A	B	C	D	E	F	G	H	I
1	2	4	4.836282		4.836282				
2	3	9							
3	4	16							
4	9	81							
5	4	16							
6									

7. As is shown in the function's VBA code, the default values for each input are the original values in cells A1 through A5. The first input (R1C1) corresponds to the value in A1, the second input corresponds to the value in A2, and so on. Within the Excel spreadsheet, the values in the A column determine the values in the B column, which in-turn determine the value in the C column. Thus, Excel Compiler creates a function with five inputs corresponding to each of the values in the A column, since these are the original given parameters. The same compilation process is used with all other deterministic functions.

Excel Compiler User Guide



The VBA code above follows a similar logic to the subroutine shown in the previous Monte Carlo example. However, because the function is deterministic, there is no need for a for loop specifying the number of simulations to be performed.

8. If a different set of input values is listed in parentheses, the function operates on these inputs as if they were the original values in the A column. Each of the values listed in parentheses is squared, added together, and the natural logarithm of this sum is returned.

	A	B	C	D	E	F	G	H	I
1	2	4	4.836282		5.164786				
2	3	9							
3	4	16							
4	9	81							
5	4	16							
6									

4 Financial Application of Monte Carlo Simulation: Pricing an Asian Call Option

- In this example, the payoff of a 5-year Asian call option with semi-annual fixings is simulated:

$$\max\left(\frac{1}{n} \sum_{i=1}^n S(t_i) - K, 0\right)$$

Excel Compiler User Guide

$S(t_i)$ = stock price at time t_i

n = total number of stock prices used to calculate the average

K = strike price

- From an initial price of \$100 on the valuation date, ten future stock prices are simulated in accordance with the standard Black-Scholes model
- Day-count convention: ACT/ACT (actual days/actual number of days in year)
- Continuous risk-free rate = 1.50% p.a.
- Volatility = 20%
- Initial stock price = \$100
- Strike price = \$100

1. Open the file named *OptionPayoffs.xlsx*. The parameters listed above are organized in this Excel spreadsheet such that the Asian call payoff can be calculated.

		=MAX(0,AVERAGE(D15:D24)-C7)				
	A	B	C	D	E	F
1	Valuation Date		1/1/2018			
2	Payoff Date		1/1/2023			
3	Day Count		ACT/ACT			
4	Contin. Risk-Free Rate		1.50%			
5	Volatility		20%			
6	Initial Stock Price		\$ 100.00			
7	Strike Price		\$ 100.00			
8						
9	Asian Call Payoff		\$ 39.58			
10	Average Asian Call Payoff					
11	Asian Call Price		\$ -			
12						
13	Date	ACT/ACT	StdNormal	Stock Price		
14	1/1/2018	0		\$ 100.00		
15	7/1/2018	0.495890411	0.6380919	\$ 109.13		
16	1/1/2019	1	0.7491572	\$ 121.08		
17	7/1/2019	1.495890411	-0.1593228	\$ 118.10		
18	1/1/2020	1.998175182	-0.3766485	\$ 111.68		
19	7/1/2020	2.496350365	0.5156054	\$ 119.81		
20	1/1/2021	3.000684463	-0.231509	\$ 115.64		
21	7/1/2021	3.496235455	0.6428034	\$ 126.28		
22	1/1/2022	4.000547645	2.1930258	\$ 172.00		
23	7/1/2022	4.496166484	0.590418	\$ 186.44		
24	1/1/2023	5.000456413	1.0424507	\$ 215.65		

In the stock price column, the initial value as of the valuation date (1/1/2018) is set equal to \$100. Future stock prices are simulated for each date using the following formula:

Excel Compiler User Guide

$$S_{t+\Delta t} = S_t \times e^{(r_f - \sigma^2/2) \times \Delta t + \sigma \times \sqrt{\Delta t} \times \varepsilon}$$

S_t = stock price at time t ;

Δt = time interval;

r_f = continuous risk free rate;

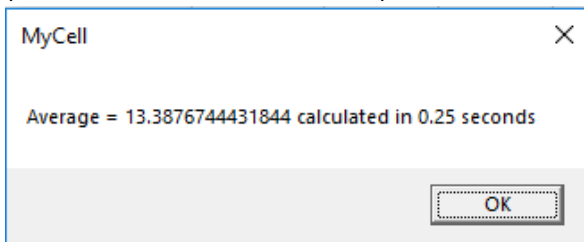
σ = volatility;

ε = random sample from the standard normal distribution

The Asian call payoff is then equal to the average stock price observed every 6 months in excess of the strike price. Recall that an option payoff is never less than zero.

After running Monte Carlo simulations, the Asian call price will be the present value of the mean payoff.

2. Compile the Asian call payoff in cell C9, then run the compiled subroutine for 10,000 simulations (see section 3.1 for instructions). Write down the result, click OK and enter it in cell C10.



Cell C11 now contains the price for the Asian call option calculated using a Monte Carlo simulation.

	A	B	C	D	E	F
1	Valuation Date		1/1/2018			
2	Payoff Date		1/1/2023			
3	Day Count		ACT/ACT			
4	Contin. Risk-Free Rate		1.50%			
5	Volatility		20%			
6	Initial Stock Price		\$ 100.00			
7	Strike Price		\$ 100.00			
8						
9	Asian Call Payoff		\$ 13.42			
10	Average Asian Call Payoff		\$ 13.39			
11	Asian Call Price		\$ 12.42			
12						
13	Date	ACT/ACT	StdNormal	Stock Price		
14	1/1/2018	0		\$ 100.00		
15	7/1/2018	0.495890411	-1.5827345	\$ 79.82		
16	1/1/2019	1	2.6850437	\$ 116.58		
17	7/1/2019	1.495890411	0.7475263	\$ 129.20		
18	1/1/2020	1.998175182	-0.3207786	\$ 123.14		
19	7/1/2020	2.496350365	-0.0917816	\$ 121.26		
20	1/1/2021	3.000684463	-0.2088822	\$ 117.42		
21	7/1/2021	3.496235455	-0.4266336	\$ 110.30		
22	1/1/2022	4.000547645	1.4340472	\$ 134.87		
23	7/1/2022	4.496166484	-1.4849995	\$ 109.16		
24	1/1/2023	5.000456413	-1.1485409	\$ 92.49		