# Executive Summary

The project values Autocallable Contingent Income Barrier Notes with Memory Coupon of a two-year term at $970.81 with a face value of $1000 based on the performance of the common stock of Lockheed Martin (LMT). The note provides a quarterly coupon of 1.65% per quarter (6.60% per annum), given that the LMT stock closes above 70% of its value on the pricing date of November 25, 2024. We also did a sensitivity analysis for different possible values of volatilities under given circumstances and concluded that the product is overvalued when using implied volatilities of maturity for valuation

# Data Collection and Sources

**See Annexure 1**
The valuation is based on the following data sources:

1) Discount rates using Bloomberg command SWDF and then USD OIS to calculate OIS for the time period between **11/25/2024 (Pricing Date)** to **11/25/2026 (Valuation date)** of the product and then used the weightage average of the discount rates between the available future discount rate dates to calculate the risk-free rate between pricing date and the valuation date.
   All discount rate collected are for base T0 = Pricing Date,
   T1 = Valuation Date

| Date | Discount rate(d) | Days (T1-Discount Date) |
|---|---|---|
| 05/27/2026 | 0.940738 | 182 |
| 11/27/2026 | 0.923229 | 2 |

   d(T0,T1) = $0.940738 * 2/184 + 0.923229 * 182/184$
   $= 0.923419$

   T1-T0 = 730 Days

   r(T0,T1) = $(-365/(T0 - T1)) * ln(d(To, T1))$
   $= (-365/730) * ln(0.923419)$
   $= 3.984\%$

2) For dividend yield between T0 ( Pricing Date) and T2 (Valuation Date), we used the Bloomberg command Oracle OPDF and collected the continuous dividend yield = 2.647% for the period for common stock of LMT.
   We are using the continuous divided yield for our valuation because we will be using Monte Carlo simulations to price our product, and will use only stock prices on coupon

observation date for valuation, thus negating the effect of using discreet dividend or continuous ones to a larger extent as both will have mostly similar effect.

3) For Implied Volatilities (IVs), we used Moneyness and maturity matrix based on the underlying LMT Corporation, American Style. We used the Bloomberg command LMT OVM and extracted IVs with Moneyness of 0.70 S0, 0.75 S0, 0.80 S0, 0.85 S0, 0.90 S0, 0.95 S0,  S0, and 1.05 S0 for all Time period between the pricing date and various observation dates for coupons.
   For our valuation we used the IVs of S0 and 0.7 S0 of the valuation dates and used the average pricing found using two IVs to find the price of product.
   We used 0.7 S0 as it is the coupon barrier, and S0 as it is the notes' callable point.

   We used different sets of volatilities as valuing the product using one specific IV(k,t) is difficult because of different conditions of payout for different values of K (strike price wrt S0), and since IV is function of K,T we decided to use the two most important inflation points for note valuation of S0, 0.7S0 IV at valuation date.

4) We used 1 million simulations of Monte Carlo to find the price of note, we use such a high number of N as Monte carlo has high error count therefore it requires large N.

# Valuation of the Product

**See Annexure 2**

| | |
|---|---|
| Product | Autocallable Contingent Income Barrier Notes with Memory Coupon |
| Underlying | One Share of Lockheed Martin Corporation |
| Term | 2 Years if not called earlier |
| Coupon Payment | Dependent on the performance of the underlying |
| Contingent coupon rate | 6.60% per Annum payable quarterly (1.65% per quarter) |
| Pricing Date (T0) | 11/25/2024 |
| Valuation Date (T1) | 11/25/2026 |
| Face Value | $1000 |
| S0 (Value of underlying on Pricing date) | $521.89 |
| Coupon Barrier | 0.70 * S0 = $365.323 |

| | |
|---|---|
| Threshold value | 0.70 * S0 = $365.323 |
| Contingent Coupon Payment | If on observation dates underlying is greater than or equal to the Coupon barrier, the Contingent Coupon Payment of $16.50 per $1,000.00 in principal amount of Note will be made + and any previously unpaid Contingent Coupons, if applicable. |
| Auto Callability | The product will be automatically recalled early if the value of the underlying at the observation date is greater than or equal to its value on the pricing date.<br>Autocallable feature is valid for all observation dates except the 1st observation date |
| Early Callable Value | $1000 ( Face value) + applicable coupon payable on the observation date and any previously unpaid Contingent Coupons, if applicable. |
| Redemption Amount at maturity | If the Note is not called till Maturity, the redemption amount per $1000 will be<br><br>If the Ending Value on the observation date is greater than or equal to the Threshold value, then $1000 + applicable coupon based on the Coupon barrier + and any previously unpaid Contingent Coupons, if applicable.<br><br>If the Ending value is below Threshold value than $1000 + ($1000 * Underlying Srock return) + Any Applicable coupon based on Coupon barrier |
| Underlying Stock Return | (Ending Value - S0) / S0 |
| Observation Dates, Time period between observation dates, and T0 | 02/25/2025, 92 Days<br>05/27/2025, 183 Days<br>08/25/2025, 273 days<br>11/25/2025, 365 days<br>02/25/2026, 457 days<br>05/26/2026, 547 days<br>08/25/2026, 638 days<br>11/25/2026 (Valuation Date), 730 days |
| Callable Dates | Same as Observation dates except for the 1st observation date on which the Note is not callable |

| | |
|---|---|
| Implied Volatility (IV) used for the valuation of the Product ($\sigma$) | We used IV of 21.907% (moneyness 100) and IV of 26.773% (moneyness 70), and used the average of values from the two IV to value the note |
| Risk Free Rate r(T0,T1) | 3.984% |
| Delta (Dividend Yield) ($\delta$) | 2.647% |
| Agent Commission | $18.50 |

- For the valuation of the product, we will be using Monte Carlo simulations with a total of 1 million simulations to value the product

- To start off our Monte Carlo simulation, we 1st created an array of observation dates and then divided it by 365 to convert it into time steps, and created another array which stores the discount rate for various time periods wrt observation dates to discount any coupon back to T0 for its valuation.
  ex 1st coupon is after 92 days therefor ime step is 92/365

- In next steps, we created an empty array payoffs [ ] to store value od product for all1 million simulations.

- Then we started our valuation code by starting a loop of N=1 million to value the product 1 million times

- In that loop, we 1st created an array of stock prices with S[0] = S0 (initial value) and space for stock values at the observation date
  For calculating the stock price on the observation date, we 1st calculated the time period between the two observation dates using the time step array (ons_times) calculated earlier, and then calculated the stock price using

  $$S[j] = S[j-1] * e^{(r - delta + 0.5 * sigma^2) * dt + sigma * \sqrt{dt} * phi}$$

  where S [ j ] is stock on the jth observation dates starting from j=1, and S[0] is the stock price on the pricing date
  r is the risk-free rate = 3.984%, delta is the continuous dividend yield = 2.647%, sigma is the implied volatility, and phi is a random number from a normal distribution used in shock term of the stock price valuation.
  Here, the 1st term is called the drift term, while the other term is called the shock term in stock simulation as per the Monte Carlo

- Also created two variables: PV = 0 to store the value of the product, missed_coupons = 0 to capture the number of missed coupons on any observation dates, as the note has a callback option and gives all the coupons not given on previous observation dates on further observation dates if the condition is satisfied.

Then we created the following conditions for coupon valuation

- If We are in 1st observation date for coupon, since product is not callable, i.e. j=1 in S [ j] where j is the stock price on jth observation date starting from j=1
  We created the condition if
  S [1] >= K, where K is coupon barrier then
  PV = PV + FV * coupon_rate * df [ j-1]
  and if S[1] <K then missed coupon = missed coupon +1

  where FV is face value of not, coupon rate is quarterly coupon rate, Discount factor is discount factor array we created earlier
  here df is discount factor where we used [ j-1] as discount period rate starts from 1st observation date and its stored in array as discount factor[0]

- For remaining of observation dates except for final one which is valuation date we used following measure

  if S [ j ] >= CB, where CB is the call barrier, then this condition plays if the underlying is above the call barrier
  TC = FV * coupon_rate * (1+missed coupons), TC = total Coupon
  PV = PV + ((TC + FV) * df [j-1])
  called = True
  break to new simulation

  elif S[ j ] >= K
  TC = FV * coupon_rate * (1+missed coupons)
  PV = PV + TC * df[j-1]
  missed coupons = 0
  else missed coupons = missed coupon +1

- If not called till valuation date then

  Sf = S[-1] i.e. final stock price on valuation date
  if Sf >= K then,  here coupon barrier is same as threshold
  TC =  FV * coupon_rate * (1+missed coupons), TC = total Coupon
  PV = PV +( ( FV + TC) * df [-1] )

  else, i.e., stock is below threshold, then

PV = PV + ( FV * (Sf/S0) * df [-1]

then just append that PV of note to our created list payoffs

Run the same simulation 1 million times and take the mean of values in the payoff distribution in the end to find the PV of note for given variables.

- We do this for 2 IVs, IV of moneyness 100 = 22.041% and IV of moneyness 70 = 26.773% while keeping other variables same and then take the average of two to find value of the note.

Using this methodology value of note for IV of Moneyness 100 = $982.4508
Using this methodology value of note for IV of Moneyness 70 = $959.1655

Value of Note = (982.4508 + 959.1655)/2 = **$970.8081**

- Value of Note including Commission = 970.8081 + 18.5 = **$989.31**

```
IV = [.22041, 0.26773]       #Expiry day Vols ranging for Moneyness of 100,70
My = [100,70]

Value = []

for sigma in IV:
  val = C_Conti1(521.89, 0.03984, 0.02647,sigma, 365.323,521.89,1000,0.0165,1000000,42)
  Value.append(val)

for i in range(len(Value)):
  print(f" Moneyness = {My[i]}, Coupon Value = {Value[i]:.4f}")

average = np.mean(Value)

print (f"Average Coupon Value = {average:.4f}")
```

```
 Moneyness = 100, Coupon Value = 982.4508
 Moneyness = 70, Coupon Value = 959.1655
Average Coupon Value = 970.8081
```

## Sensitivity Analysis

**See Annexure 3**

- We have used r(T0,T1) for discounting all the coupons which are received during the life of product and since they are with different time periods, they will have different risk free rates and have considered observation date as payment date of coupon, both of which can cause over/under-valuation of product, though its a fairly cautious as risk free rate do not affect the value of option by large amount when compared to other variables such as implied volatility, coupon and call barrier, dividend yield.
  And we are doing Monte Carlo simulations; therefore, large number of simulations we

can mostly negate this error

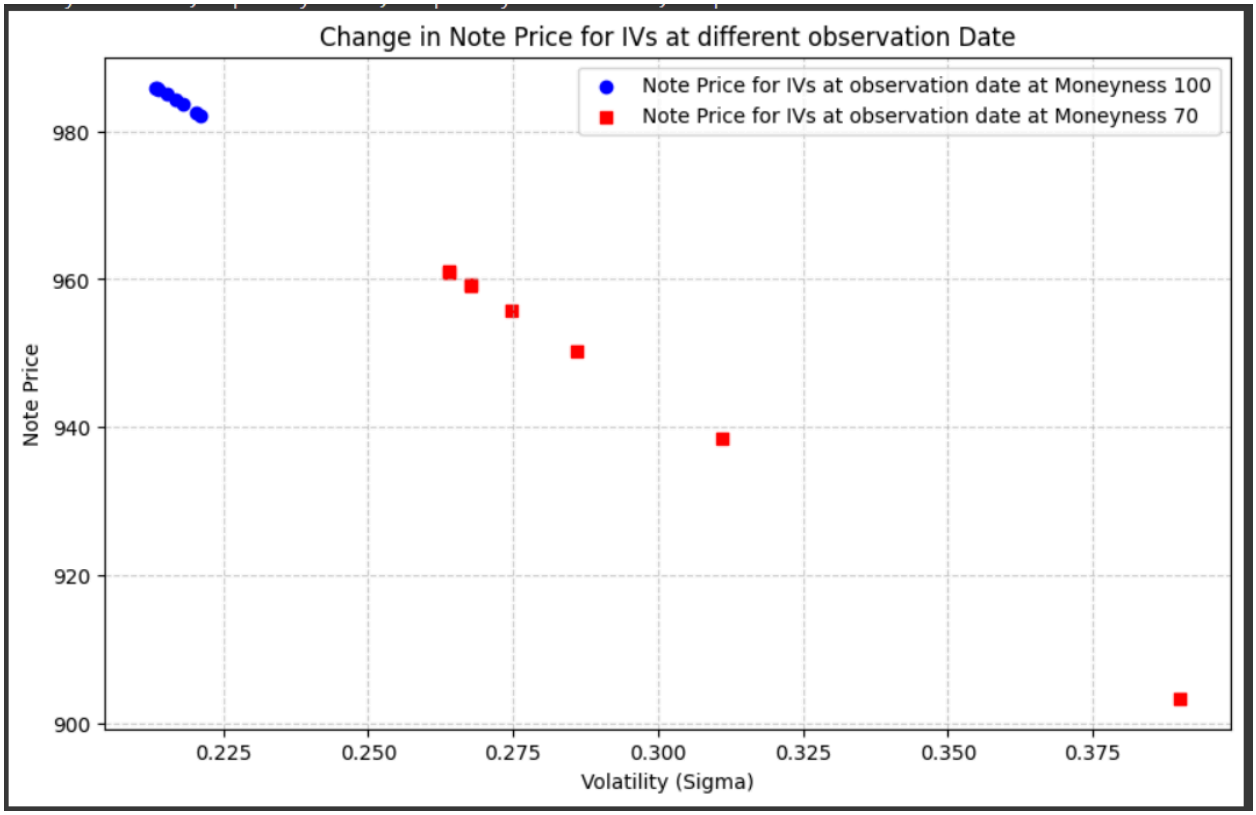
- As from the data collected for the implied volatilities at different moneyness and expiry, especially wrt the observation date, there are a lot of volatility choices which we can make while valuing the product
  While doing our valuations, we used 2 IVs (moneyness 100 and 70) of the valuation date (T1) and took the average between the two values to find the note's value
  We did a sensitivity analysis for different volatilities observed at moneyness of 0.70 S0 (Barrier value) and S0 for all the observation date time periods while keeping all other variables constant and plotting the volatility against observed product valuation.

  We observed when valuing for moneyness at S0 our product valuation varied from $985.9043 to $982.1597 with the lowest value observed when the IV of coupon day 638 and the Highest for the 1st observation date for coupon (92 days)
  For moneyness 70 we observed even more variation, since its deep in the money strike with values varying from 960.9727 for coupon day 547 IV to 903.32 for coupon day 92 IV
  Even on removing the extreme outlier lower end values vary from 935.55 to 960.97

  Here we observe, If we use only volatility of moneyness 100 we are overvaluing the product since our barries is 07 S0 i.e. moneyness 70 while if we use volatility of moneyness 70, we undervalue product as our payout is same even if underlying is at 0.8S0 or 0.75 S0 or 0.7 S0

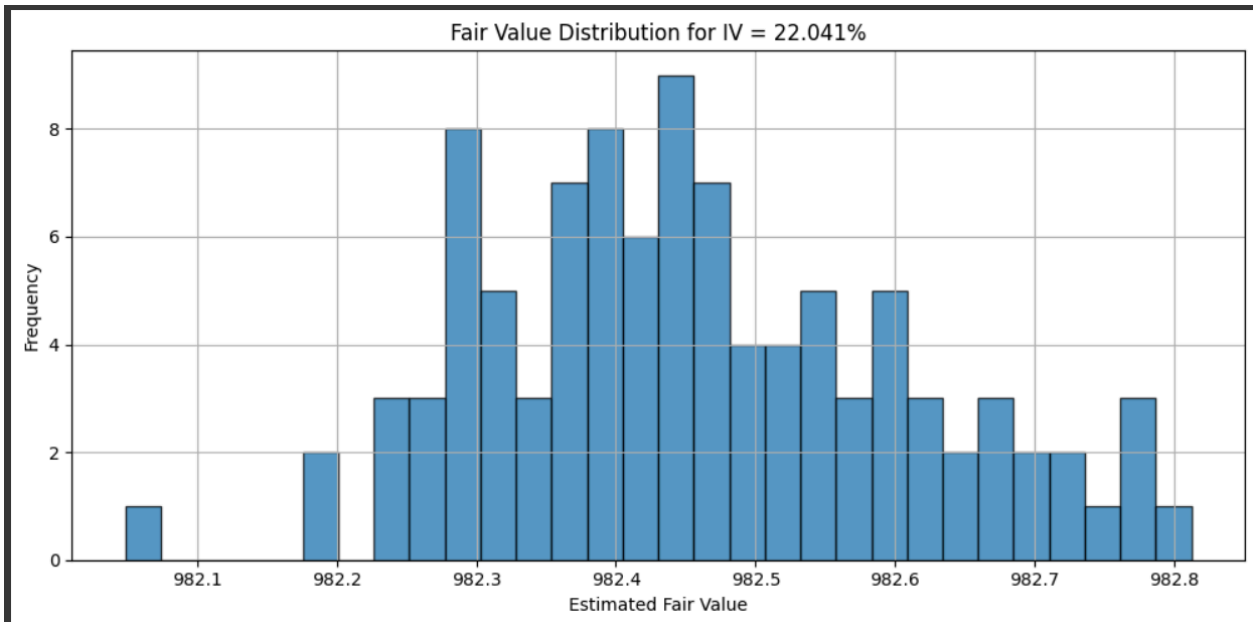
  Also, even using volatility of valuation date (day 730 from pricing) we are overvaluing the product even when we are taking average of value based on two IVs as there is higher probability of stock going below 0.7S0 per IVs of moneyness 70 of earlier coupon days and even if we receive those coupon later on due to memory coupon, we are overvaluing that coupon present value and thus overvaluing the product in general
  This can result in investors overpaying for the product as it presents a higher probability of coupon being paid than it should

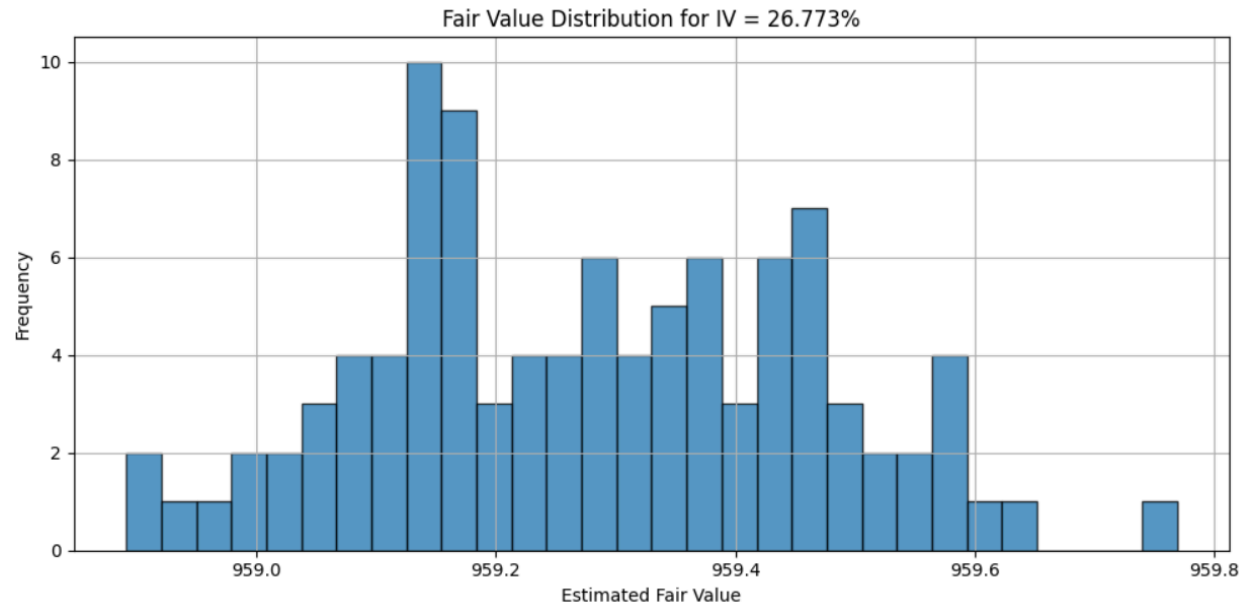
```
Moneyness = 100 ,Coupon Day = 92, Coupon Day IV = 0.21329, Coupon Value = 985.9043
Moneyness = 100 ,Coupon Day = 183, Coupon Day IV = 0.21684, Coupon Value = 984.1906
Moneyness = 100 ,Coupon Day = 273, Coupon Day IV = 0.21522, Coupon Value = 984.9662
Moneyness = 100 ,Coupon Day = 365, Coupon Day IV = 0.21349, Coupon Value = 985.8102
Moneyness = 100 ,Coupon Day = 457, Coupon Day IV = 0.21379, Coupon Value = 985.6538
Moneyness = 100 ,Coupon Day = 547, Coupon Day IV = 0.21797, Coupon Value = 983.6217
Moneyness = 100 ,Coupon Day = 638, Coupon Day IV = 0.22101, Coupon Value = 982.1597
Moneyness = 100 ,Coupon Day = 730, Coupon Day IV = 0.22041, Coupon Value = 982.4508
Moneyness = 70 ,Coupon Day = 92, Coupon Day IV = 0.38991, Coupon Value = 903.3292
Moneyness = 70 ,Coupon Day = 183, Coupon Day IV = 0.31096, Coupon Value = 938.5589
Moneyness = 70 ,Coupon Day = 273, Coupon Day IV = 0.28606, Coupon Value = 950.2858
Moneyness = 70 ,Coupon Day = 365, Coupon Day IV = 0.27472, Coupon Value = 955.7613
Moneyness = 70 ,Coupon Day = 457, Coupon Day IV = 0.26763, Coupon Value = 959.2162
Moneyness = 70 ,Coupon Day = 547, Coupon Day IV = 0.26402, Coupon Value = 960.9727
Moneyness = 70 ,Coupon Day = 638, Coupon Day IV = 0.26396, Coupon Value = 961.0016
Moneyness = 70 ,Coupon Day = 730, Coupon Day IV = 0.26773, Coupon Value = 959.1655
```




- We also ran our code 100 times in a loop for 1 million Monte carlo simulations for our our two IVs which we used in valuation i.e. 22.041% at moneyness 100 and 26.773% at moneyness 70 plotted the graph for same.
  We observe it looks like a normal distribution ( not perfectly as 1 million simulations were run 100 times only, due to constraints), but it gives us a fair idea of product value along with its Standard Deviation

```
The Mean value of Product with IV of 22.041% at moneyness 100 is: 982.4554
The Standard Deviation of Product with IV of 22.041% at moneyness 100 is: 0.1813
The Mean value of Product with IV of 26.773% at moneyness 70 is: 959.2804
The Standard Deviation of Product with IV of 26.773% at moneyness 70 is: 0.1813
The Mean value of Product with IV of 22.041% and 26.773% at moneyness 100 and 70 is: 970.8679
```




- Though our underlying is a single stock, we used continuous dividend yield instead of discrete dividends, as we are using Monte Carlo for our valuations and only valuing stock on observation days, it will not cause relatively that high of error if we use compared to using discrete dividends, and will not lead to high observable error as discrete dividends would also have been observed at different dates than observation dates quarterly, thus cousin almost same effect on valuation

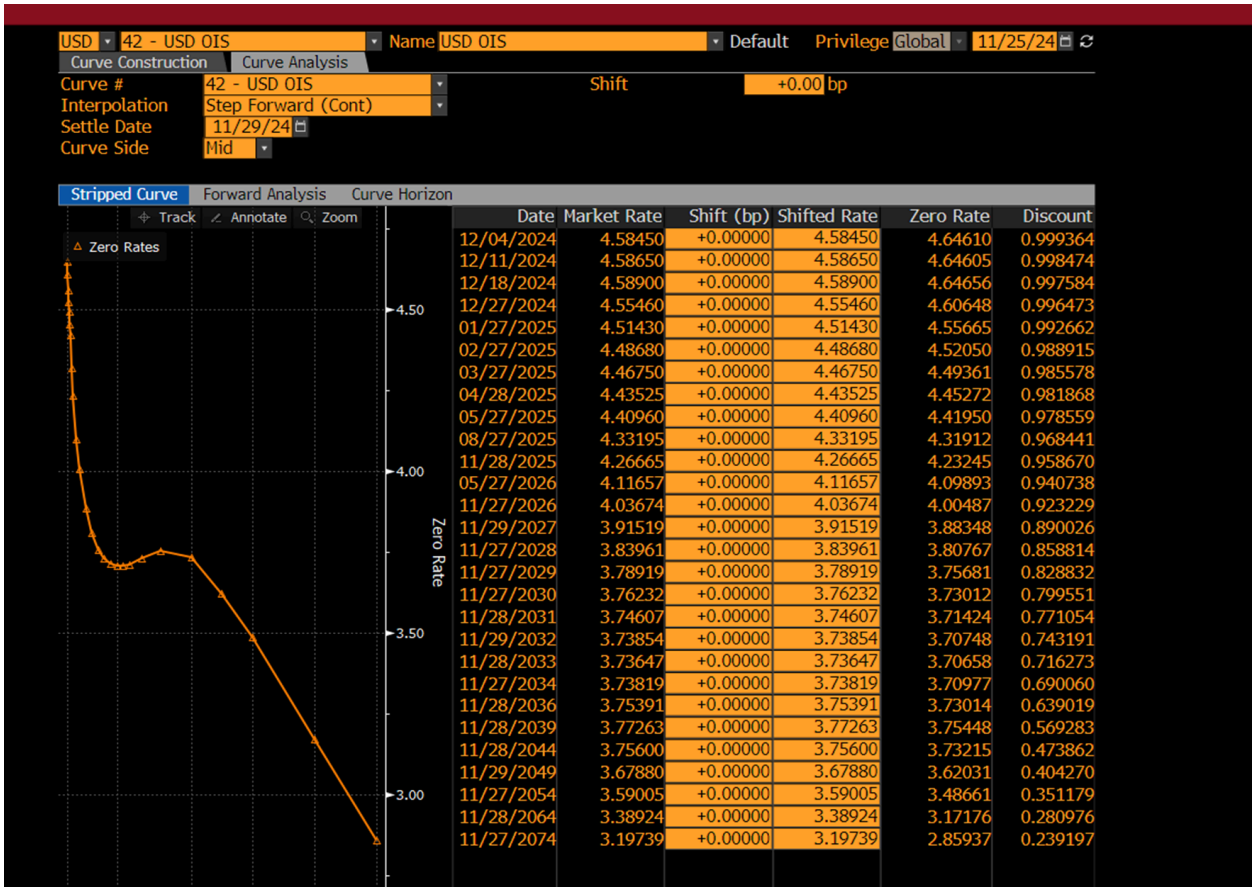
- Monte Carlo produces high error & therefore we have to select as large of simulations as possible. In our case, we chose 1 million simulations, given the constraints of our tech, the number can be enhanced and increased with better tech

- All these different assumptions of different variable values can lead to mispricing and overpaying for the product and lead to loss for the investor

- Our valuation of the product at $970.81 is less than the term sheet valuation of the product of  $972.65, undervaluing the product by 0.18%

**Annexure 1**

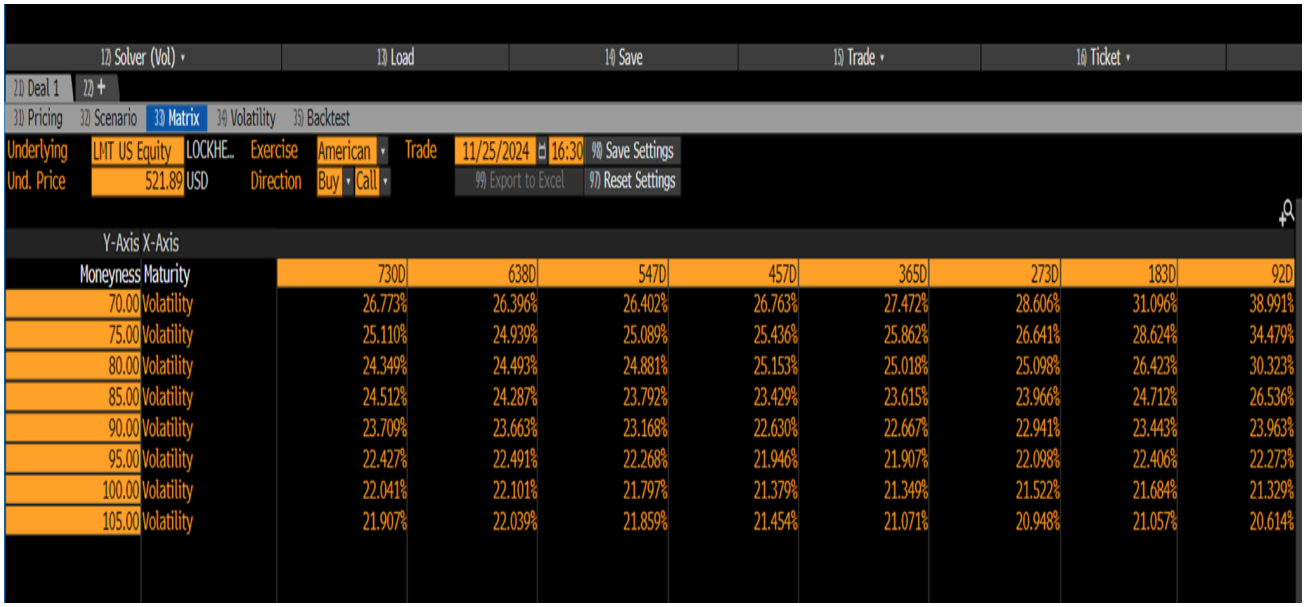
**1) Risk-Free rate**




| Date | Market Rate | Shift (bp) | Shifted Rate | Zero Rate | Discount |
|---|---|---|---|---|---|
| 12/04/2024 | 4.58450 | +0.00000 | 4.58450 | 4.64610 | 0.999364 |
| 12/11/2024 | 4.58650 | +0.00000 | 4.58650 | 4.64605 | 0.998474 |
| 12/18/2024 | 4.58900 | +0.00000 | 4.58900 | 4.64656 | 0.997584 |
| 12/27/2024 | 4.55460 | +0.00000 | 4.55460 | 4.60648 | 0.996473 |
| 01/27/2025 | 4.51430 | +0.00000 | 4.51430 | 4.55665 | 0.992662 |
| 02/27/2025 | 4.48680 | +0.00000 | 4.48680 | 4.52050 | 0.988915 |
| 03/27/2025 | 4.46750 | +0.00000 | 4.46750 | 4.49361 | 0.985578 |
| 04/28/2025 | 4.43525 | +0.00000 | 4.43525 | 4.45272 | 0.981868 |
| 05/27/2025 | 4.40960 | +0.00000 | 4.40960 | 4.41950 | 0.978559 |
| 08/27/2025 | 4.33195 | +0.00000 | 4.33195 | 4.31912 | 0.968441 |
| 11/28/2025 | 4.26665 | +0.00000 | 4.26665 | 4.23245 | 0.958670 |
| 05/27/2026 | 4.11657 | +0.00000 | 4.11657 | 4.09893 | 0.940738 |
| 11/27/2026 | 4.03674 | +0.00000 | 4.03674 | 4.00487 | 0.923229 |
| 11/29/2027 | 3.91519 | +0.00000 | 3.91519 | 3.88348 | 0.890026 |
| 11/27/2028 | 3.83961 | +0.00000 | 3.83961 | 3.80767 | 0.858814 |
| 11/27/2029 | 3.78919 | +0.00000 | 3.78919 | 3.75681 | 0.828832 |
| 11/27/2030 | 3.76232 | +0.00000 | 3.76232 | 3.73012 | 0.799551 |
| 11/28/2031 | 3.74607 | +0.00000 | 3.74607 | 3.71424 | 0.771054 |
| 11/29/2032 | 3.73854 | +0.00000 | 3.73854 | 3.70748 | 0.743191 |
| 11/28/2033 | 3.73647 | +0.00000 | 3.73647 | 3.70658 | 0.716273 |
| 11/27/2034 | 3.73819 | +0.00000 | 3.73819 | 3.70977 | 0.690060 |
| 11/28/2036 | 3.75391 | +0.00000 | 3.75391 | 3.73014 | 0.639019 |
| 11/28/2039 | 3.77263 | +0.00000 | 3.77263 | 3.75448 | 0.569283 |
| 11/28/2044 | 3.75600 | +0.00000 | 3.75600 | 3.73215 | 0.473862 |
| 11/29/2049 | 3.67880 | +0.00000 | 3.67880 | 3.62031 | 0.404270 |
| 11/27/2054 | 3.59005 | +0.00000 | 3.59005 | 3.48661 | 0.351179 |
| 11/28/2064 | 3.38924 | +0.00000 | 3.38924 | 3.17176 | 0.280976 |
| 11/27/2074 | 3.19739 | +0.00000 | 3.19739 | 2.85937 | 0.239197 |

2) **Dividend Yield**


3) **Implied Volatility**




Annexure2)

```
[1] import numpy as np
    import matplotlib.pyplot as plt
```

```
def C_Conti1( S0, r, delta, sigma, K, CB, FV, coupon_rate, N, seed=None):

    obs_days = np.array([92, 183, 273, 365, 457, 547, 638, 730])
    obs_times = obs_days / 365.0
    steps = len(obs_times)
    df = np.exp(-r * obs_times)


    if seed is not None:
        np.random.seed(seed)

    payoffs = []

    for _ in range(N):
        S = np.zeros(steps + 1)
        S[0] = S0
        missed_coupons = 0
        PV = 0
        called = False

        for j in range(1, steps + 1):
            dt_j = obs_times[j - 1] - obs_times[j - 2] if j > 1 else obs_times[0]
            phi = np.random.normal()
            S[j] = S[j - 1] * np.exp((r - delta - 0.5 * sigma**2) * dt_j + sigma * np.sqrt(dt_j) * phi)
```

```
#S0 initial Price, r ~ risk free, Delta ~diviyield, sigma ~ Impl Vol, K~Coupon barrier
#CB ~ Call barrier, FV~ Face value, coupon_rate~ Quaterlu coupon rate, N~ No of simulations

#Observation Time for different Coupon days wrt Pricing Day

#Discount Factor for different observation days

#List for Saving different Payoffs of all simulations

#Simulating Stock prices on observation days using Monte Carlo
```

```
[2]             if j == 1:
                    if S[j] >= K:
                        PV += FV * coupon_rate * df[j - 1]
                    else:
                        missed_coupons += 1
                elif S[j] >= CB:
                    TC = FV * coupon_rate * (1 + missed_coupons)
                    PV += TC * df[j - 1]
                    PV += FV * df[j - 1]
                    called = True
                    break
                elif S[j] >= K:
                    TC = FV * coupon_rate * (1 + missed_coupons)
                    PV += TC * df[j - 1]
                    missed_coupons = 0
                else:
                    missed_coupons += 1
```

```
#Condition for 1st observation day  Coupon

#Total number of missed Coupons
#Condition for Observation day bw 2nd and 2nd last
#TC = Total coupon
#PV = Present Value of Coupon on the observation day in question
```

```
            if not called:
                Sf = S[-1]
                if Sf >= K:
                    TC = FV * coupon_rate * (1 + missed_coupons)
                    PV += TC * df[-1]
                    PV += FV * df[-1]
                else:
                    PV += (FV * (Sf / S0)) * df[-1]

            payoffs.append(PV)

        return np.mean(payoffs)
```

```
#Condition for last observation day  which is also Valuation day

#Mean of all Notes value Calculated using Monte carlo Simulations
```

Annexure 3)

```
IV100 = [0.21329,0.21684, 0.21522, 0.21349, 0.21379, 0.21797, 0.22101, 0.22041 ]          #Vols for Moneyness 100 with Pricing and different Coupon dates
IV70 = [0.38991, 0.31096, 0.28606, 0.27472, 0.26763, 0.26402, 0.26396, 0.26773]          #Vols for Moneyness 70 with Pricing and different Coupon dates
Days = [92, 183, 273, 365, 457, 547, 638, 730]
Value100 = []
Value70 = []

for sigma in IV100:
  val100 = C_Conti1(521.89, 0.03984, 0.02647, sigma, 365.323, 521.89, 1000, 0.0165,1000000,42)
  Value100.append(val100)

for sigma in IV70:
  val70 = C_Conti1(521.89, 0.03984, 0.02647, sigma, 365.323, 521.89, 1000, 0.0165,1000000,42)
  Value70.append(val70)

for i in range(len(Value100)):
  print(f" Moneyness = {100} ,Coupon Day = {Days[i]}, Coupon Day IV = {IV100[i]}, Coupon Value = {Value100[i]:.4f}")

for i in range(len(Value70)):
  print(f" Moneyness = {70} ,Coupon Day = {Days[i]}, Coupon Day IV = {IV70[i]}, Coupon Value = {Value70[i]:.4f}")

plt.figure(figsize=(10, 6))
plt.scatter(IV100, Value100, color = 'b', label='Note Price for IVs at observation date at Moneyness 100', marker='o')
plt.scatter(IV70, Value70, color = 'r', label='Note Price for IVs at observation date at Moneyness 70', marker='s')
plt.xlabel('Volatility (Sigma)')
plt.ylabel('Note Price')
plt.title('Change in Note Price for IVs at different observation Date')
```

```
plt.grid(True, linestyle='--', alpha=0.6)
plt.legend()
plt.show()
```

```
distribution100 = []
distribution70 =[]

for seed in range(100):
  val100 = C_Conti1(521.89,0.03984, 0.02647,0.22041, 365.323,521.89,1000,0.0165,1000000,seed = seed)
  distribution100.append(val100)
  val70 = C_Conti1(521.89,0.03984, 0.02647,0.26773, 365.323,521.89,1000,0.0165,1000000,seed = seed)
  distribution70.append(val70)

mean_val100 = np.mean(distribution100)
std_val70 = np.std(distribution100)
mean_val70 = np.mean(distribution70)
std_val70 = np.std(distribution70)
Product_Value = np.mean([mean_val100, mean_val70])

# Plot
plt.figure(figsize=(10, 5))
plt.hist(distribution100, bins=30, edgecolor='k', alpha=0.75)
plt.title("Fair Value Distribution for IV = 22.041%")
plt.xlabel("Estimated Fair Value")
plt.ylabel("Frequency")
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
plt.figure(figsize=(10, 5))
plt.hist(distribution70, bins=30, edgecolor='k', alpha=0.75)
plt.title("Fair Value Distribution for IV = 26.773%")
plt.xlabel("Estimated Fair Value")
plt.ylabel("Frequency")
plt.grid(True)
plt.tight_layout()
plt.show()

print(f"The Mean value of Product with IV of 22.041% at moneyness 100 is: {mean_val100:.4f}")
print(f"The Standard Deviation of Product with IV of 22.041% at moneyness 100 is: {std_val70:.4f}")
print(f"The Mean value of Product with IV of 26.773% at moneyness 70 is: {mean_val70:.4f}")
print(f"The Standard Deviation of Product with IV of 26.773% at moneyness 70 is: {std_val70:.4f}")
print(f"The Mean value of Product with IV of 22.041% and 26.773% at moneyness 100 and 70 is: {Product_Value:.4f}")
```