# Executive Summary

The Report presents the valuation of Contingent Income Issuer Callable Yield Notes of the two-year term with a face value of $1000 linked to the Common Stock of Oracle Corporation. The notes provide an annual coupon of 11.65% per annum (paid quarterly) and have an embedded barrier where it does not provide any coupon if the underlying is less than 0.65 times its value on the pricing date and gives the issuer the right to call the note at all but first quarterly observation date.
Using the CRR method of binomial pricing we have estimated the value of the product to be $971.50 using data of implied volatility, dividend yield, and risk-free rate collected via Bloomberg terminal.
We also conducted sensitivity analysis and found that product value does not converge smoothly due to the embedded barrier as we increase the number of steps in the binomial tree but converge to a point where the relative position of the barrier is 0.3 from 1st largest and 1st smallest stock price at T1.
Volatility analysis showed that using implied volatility with a strike of 'At the Money' overvalues the product by underestimating the probability of the stock ending below the coupon barrier.

# 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 **2/24/2025 (Pricing Date)** to **2/24/2027 (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 |
   |---|---|
   | 2/24/27 | 0.921911 |

   Here in this case we do not need to calculate the weightage average discount rate as we already have the discount rate available for the valuation date with base as the pricing date.

   T1 - T0 = 730 days

   r(T0,T1) = -(365/(T1-T0)) * ln(discount rate)

$$r\,(T0, T1) \;=\; -\;(365/730)\;*\;ln(0.921911)$$
$$=\; 4.0653\%$$

2) For dividend yield between T0 ( Pricing Date) and T2 (Valuation Date), we used the Bloomberg command Oracle OPDF and collected the dividend yield = 1.109% for the period

3) For Implied Volatilities (IVs), we used Moneyness, and maturity matrix based on the underlying Oracle Corporation, American Style. We used Bloomberg command Oracle OVM and extracted IVs with Moneyness of 0.65 S0, 0.75 S0, 0.85 S0, S0, 1.05 S0, 1.10 S0, 1.20 S0 for all Time period between the pricing date and various observation dates for coupons.

# Valuation of the Product

**See Annexure 2**

| | |
|---|---|
| Product | Contingent Issuer Callable Product |
| Underlying | One Share of Oracle Corporation |
| Term | Almost 2 Years if not called Earlier |
| Coupon Payment | Dependent on the performance of underlying |
| Contingent coupon rate | 11.65% per Annum payable quarterly (2.9125% per quarter) |
| Pricing Date (T0) | 2/24/2025 |
| Valuation Date (T1) | 2/24/2027 |
| Face Value | $1000 |
| S0 (Value of underlying on Pricing date) | 169.96 |
| Coupon Barrier | 0.65 * S0 = 110.47 |
| Threshold value | 0.65 * S0 = 110.47 |
| Contingent Coupon Payment | If on observation dates underlying is greater than or equal to the Coupon barrier, the Contingent Coupon Payment of $29.125 per $1,000.00 in principal amount of Note will be made |
| Early Redemption | Issuer can redeem the note early on quarterly observation dates (except 1st observation |

| | |
|---|---|
| | date), If redeemed early, no further coupon will be paid post redemption |
| Early Redemption Value | $1000 ( Face value) + Any coupon if payable on the observation date based on value of the underlying being above or below the Coupon barrier |
| 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<br><br>If the Ending value is below Threshold value than $1000 + ($1000 * Underlying Srock return) + Any Applicable coupon based on Coupon barrier |
| Underlying Stock Return | (Ending Value - S0) / S0 |
| Observation Dates, Time period between observation dates and T0 | 05/27/2025, 92 Days<br>08/25/2025, 182 Days<br>11/24/2025, 273 days<br>02/24/2026, 365 days<br>05/26/2026, 456 days<br>08/24/2026, 546 days<br>11/24/2026, 638 days<br>02/24/2027 (Valuation Date), 730 days |
| Callable Dates | Same as Observation dates except for the 1st observation date on which Note is not callable |
| Implied Volatility (IV) used for the valuation of the Product ($\sigma$) | 35.46% (IV of Buy American Call at S0 for Time period of 730 days between T0 and T1) |
| Risk Free Rate r(T0,T1) | 4.0653% |
| Delta (Dividend Yield) ($\delta$) | 1.109% |
| Agent Commission | $18.50 |

- We chose the Buy American Call Volatility of Oracle Corporation stock which was similar to the Buy American Put for a given time period and strike price and were greater than Sell American Call volatility which were similar to the Sell American Put.

To provide a more realistic chance of stock going below the coupon barrier for a given choice of Strike price and time period chosen for IV, we chose the one that was comparatively high i.e. Buy American Call.

- For valuing the product we will be using the CRR method of binomial valuation, and since the product contains a barrier for coupon payment we will deploy a larger N to increase the accuracy

- As it is a CRR method of binomial valuation we use the following factors to calculate the Binomial Stock tree of underlying using the Up(u) and down(d) factors alongside number of steps (N = 10045) over time period (T) of 2 years, and also calculated Risk - Neutral probability (p) for valuation of the product

  dt = T/N = 2/10045
  u = $e^{\sigma * \sqrt{dt}}$
  d = 1/u
  p = $(e^{(r-\delta) * dt} - d) / (u - d)$

- We then created two dimensional array stock tree St[i,j] with S[0,0] = S0 and S[i,0] = S[i-1,0] * d and S[i, j] = S[i-1,j-1] * u for j (1,N+1)

- We converted observation dates into discrete steps within the binomial tree model, and any fractional steps resulting from non-integer time divisions are stored in correction factors (delimd and delid) to improve precision.

  Created 2 sets of observation date arrays one in which Note is callable and one in which Note is not callable

  Coupon payment if payable will be paid on dates which fall in the discrete time step we calculated and then stored as integer form by flooring it.
  Since in this method, we will be adding coupons earlier than anticipated we discounted that coupon for the time difference between the coupon added in the tree and the original time period of the coupon.

  For Ex) with N = 10045 and for the 1st observation date with T=92 days from pricing

  dt = 2/10045 = 0.0001991
  Time period of Observation date = 92/730 = 0.252054795
  Time Step for observation date = 0.252054795 / 0.0001991 = 1265.954
  Flooring the time step to include the coupon in the tree with integer value = 1265
  Time period for discounting the coupon = (1265.954 - 1265) * dt = 0.000188 (calling this variable as delid / delimd)

- We also created the function lambda $= (S_k - K) / (S_k - S_{k-1})$ where K is the barrier stock price, Sk is 1st stock price above K, and S(k-1) is stock 1st stock price below K at maturity
  We will later use this function to do a sensitivity analysis of our product valuation

- For valuation, we created a 2-dimensional array V([N+1, N+1])

  For the valuation of the product at maturity for every St we used the following methodology:

  If St[N,j] >= Coupon Barrier (K1) (Which is same as threshold barrier)
  Payoff = $1000 + ($1000 * 11.65%/4) = $1029.125

  Otherwise (Cases when the stock price is below the threshold and thus also below the coupon barrier)
  Payoff = $1000 (Final Stock value / S0)

  and thus calculated the payoff at maturity and then used backward induction via stock binomial tree to calculate Note value for different time steps and j using the following conditions

  For all St[i,j] points in binomial tree

  $V[i,j] = e^{-r*dt} * [V[i+1, j+1] * p + V[i+1, j] * (1-p)]$ with the following conditions when the time step is in observation date for coupons

  **If time step i is in observation date when Note is callable then**

  cv = $e^{-r*dt} * [V[i+1, j+1] * p + V[i+1, j] * (1-p)]$

  call = $e^{(-delid[i] * r)} * FV$ where FV is the face value of note = $1000, we are discounting this also like coupon as this cash flow is being also calculated like coupon earlier

  then V[i,j] = minimum[cv,call]
  and in this if S[i,j] > Coupon barrier then

  V[i,j] = minimum [cv,call] + $e^{(-delid[i] * r)} * (FV * coupon\ rate/periodicity)$

  **If the time step is in the observation date when the Note is not callable then**

  if S[i,j] >= Coupon barrier then

$$V[i,j] = e^{-r*dt} * [V[i+1,j+1] * p + V[i+1,j] * (1-p)] + e^{(-delimd[i]*r)} * (FV * coupon/per)$$

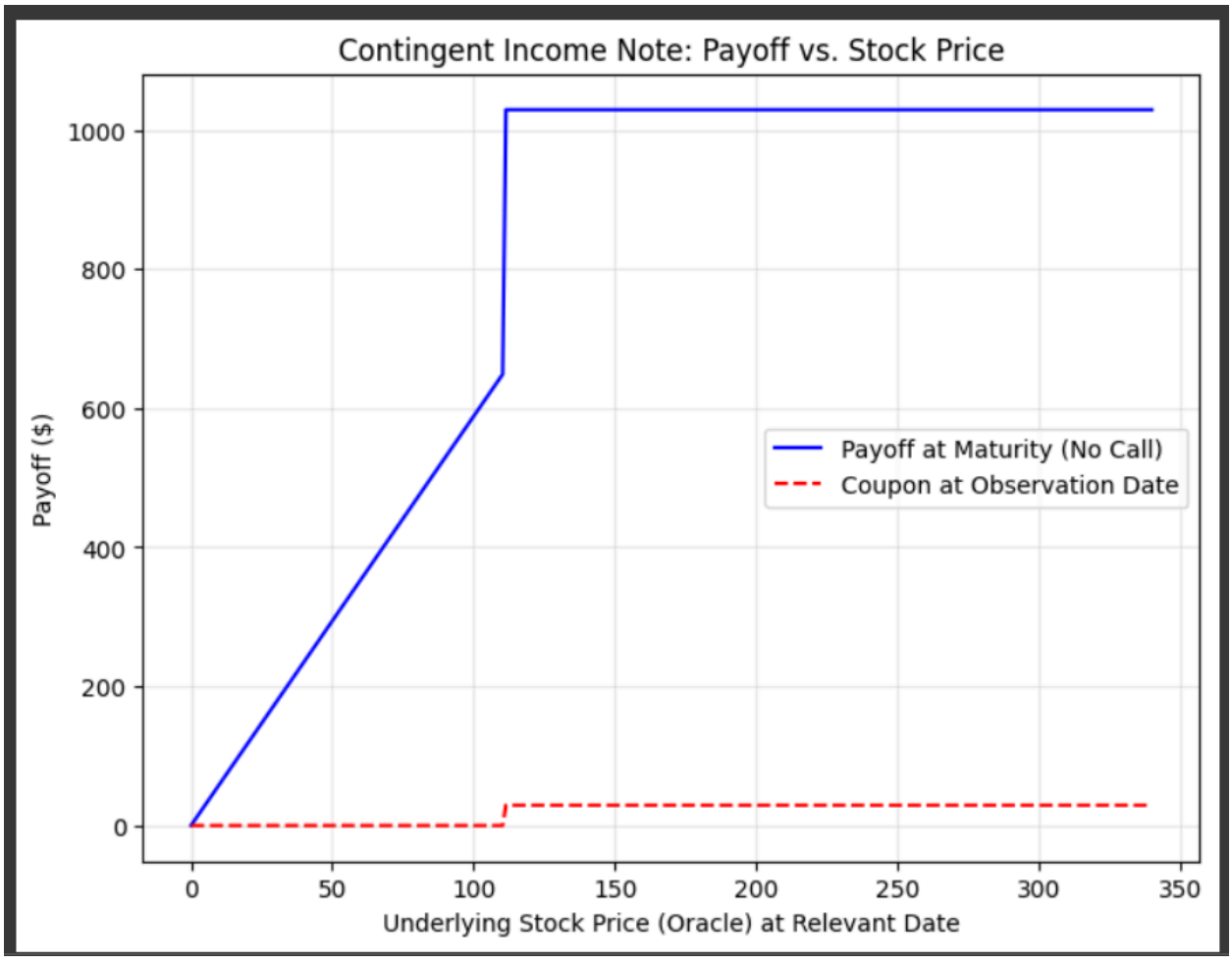
and thus doing backward induction in the binomial tree we calculate V[0,0] which is the value of the product at T0.

Using this methodology and all the data points provided above for N=10045 we get the value of note **V[0,0] = $971.50**

**Value including agent commission** = $971.50 + $18.50 = **$990**

```
CallYiNo(169.96,110.47,0.040653,0.01109,0.3546,2,10045,1000,0.1165,4)

{'num_steps': 10045,
 'Value': 971.5133334049818,
 'LambdaK1': 0.5525324831336504}
```

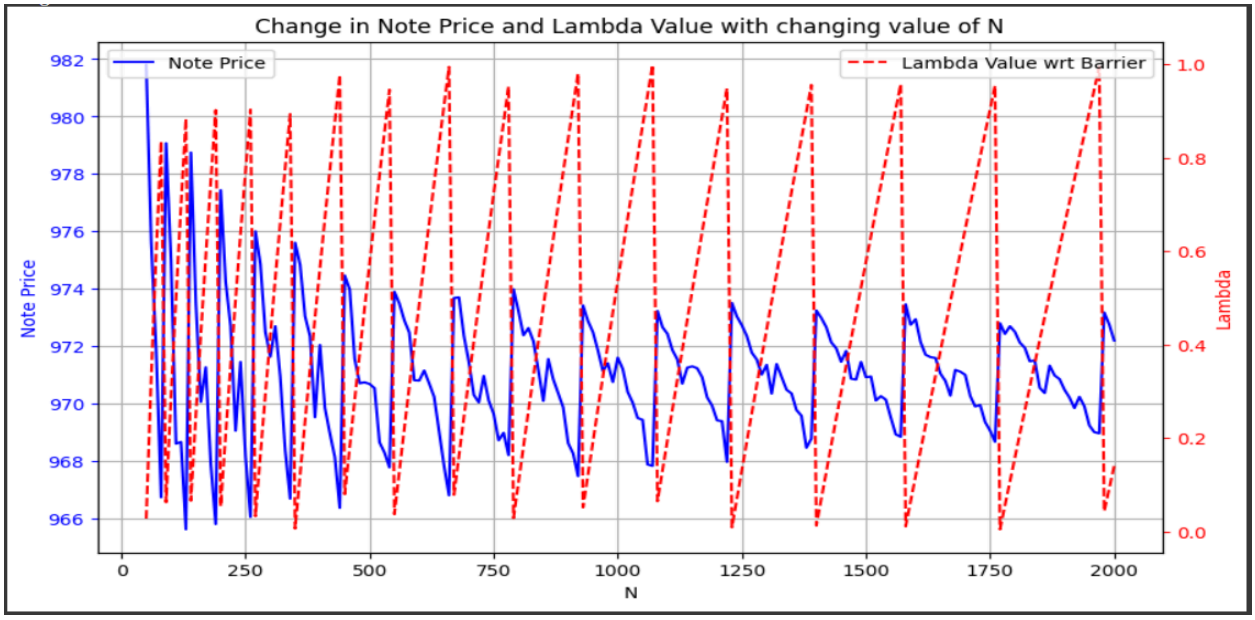


# Sensitivity Analysis

**See Annexure 3**

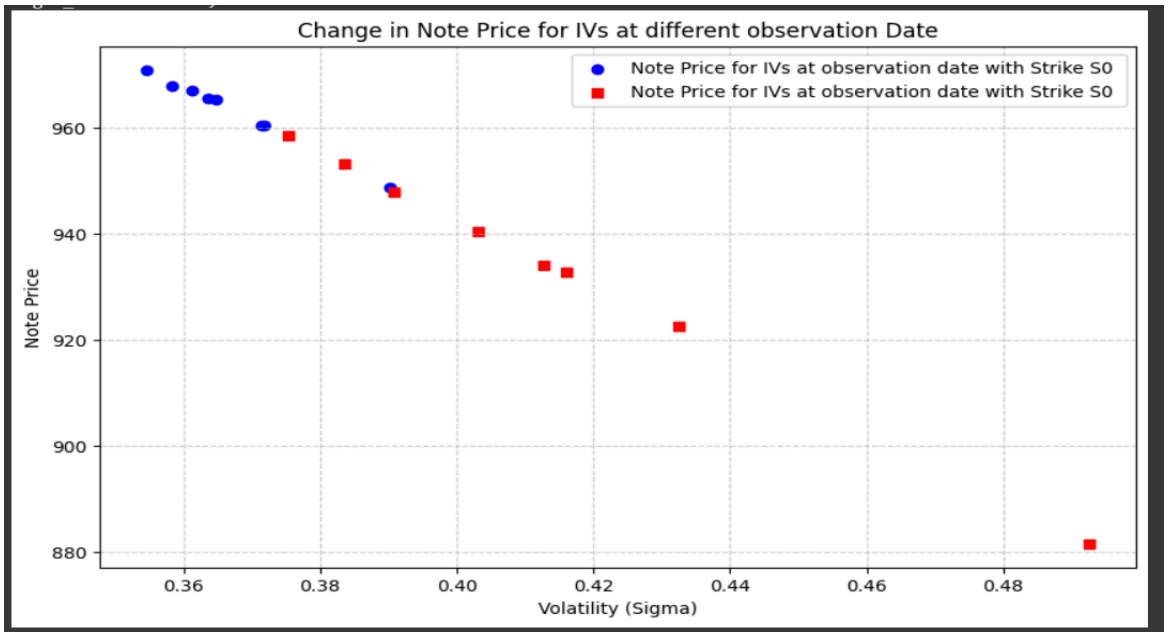
1) We have used r(T0,T1) for discounting all the coupons which are received during the life of product and since they are with different time periods, they will have different risk free rates and have considered observation date as payment date of coupon, both of which can cause over/under-valuation 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 and since time between coupon included in tree and its observation date is fairly near for large N, it does not cause significant error, as this is a barrier product thus most of the noise and sensitivity issue is coming from the barrier embedded in product and we are pretty certain that this assumption will not cause much noise and thus sensitivity in the value of product.

2) One of the major sources of error in the products with the embedded barrier is the barrier point of the underlying, to figure out how the product value varies as we vary N, we plotted the lambda calculated in the main section which is the relative distance of barrier from 1st largest and 1st smallest stock price at T1.
We varied N from 50 to 2000, while keeping all other variables constant and observed that the value of the product converges as we increased the N with the value converging to 971 when the value of lambda is 0.3.
From this, we observe that the value of the product does not converge continuously as we increase the N because of the embedded barrier in the product and this causes an error in the calculation of the product, thus creating a sensitivity issue while valuing the product



3) 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
though while using the CRR method we can only use 1 volatility to make sure the stock price tree recombines.
We did a sensitivity analysis for different volatilities observed at moneyness of 0.65 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 $948.77 to $971 with the highest value observed when the time to maturity is equal to product length (730 Days) and the lowest for the 1st observation date for coupon (92 days)
For moneyness of 0.65 S0, we observed even more variation since it is deep in the money strike with product valuation varying from $881.6 to $958.6 with the highest value observed when the time to maturity is equal to product length (730 Days) and lowest for the 1st observation date for coupon (92 days)
Even on removing 2 extreme outliers of lower-end valuation varies from $971 to $932.75.
Here we observe, If we use IV of S0 instead of 0.65 S0 (which is our barrier) we are overvaluing the product since it underweights the probability of stock ending below barrier value resulting in no coupon payment.
Also, volatilities for the respective strikes are lowest when the time to expiry is 730 days, and using that to value product paying out coupons at other observation dates overvalues the probability of stock being above barrier value
This can result in investors overpaying for the product as it presents a higher probability of coupon being paid than it should



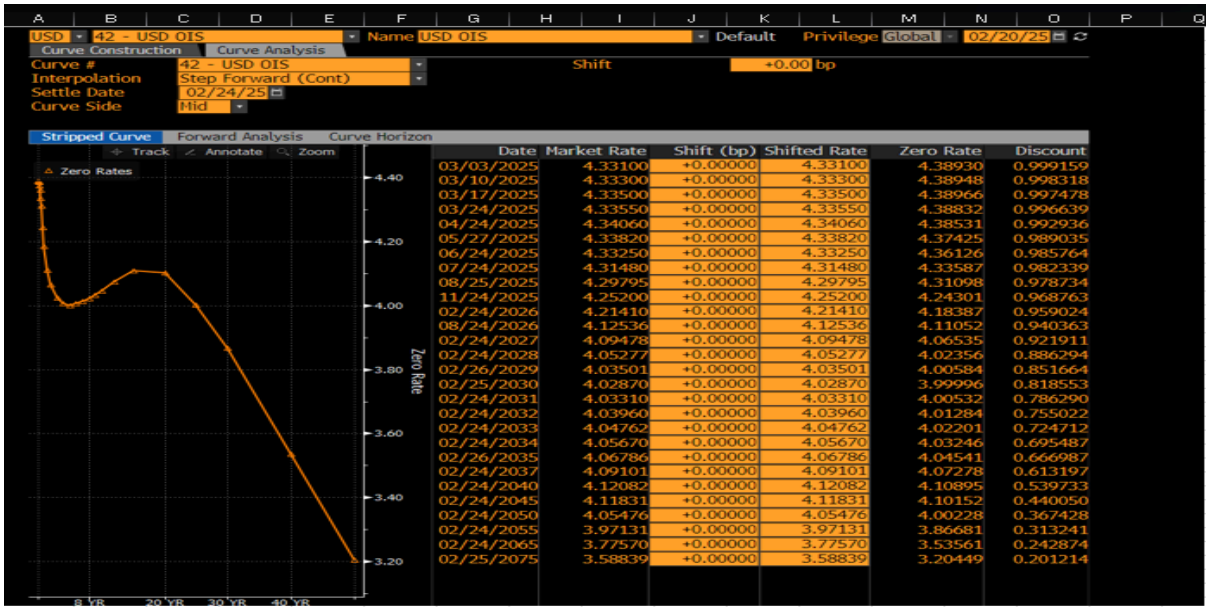
4) Though our underlying is a single stock, we used continuous dividend yield instead of discrete dividends to ensure that the stock price tree recombines, and since it is a forward-looking dividend yield from Bloomberg, with quarterly observation dates, it is a fairly reasonable assumption, and will not lead to observable error as discrete dividends would also have been observed at different dates than observation dates quarterly, thus cousin almost same effect on valuation

5) All these different values can lead to mispricing and overpaying for the product and lead to loss for the investor

6) Our valuation of $971.50 of the product is less than the term sheet valuation of the product of $972.90, undervaluing the product by 0.14%

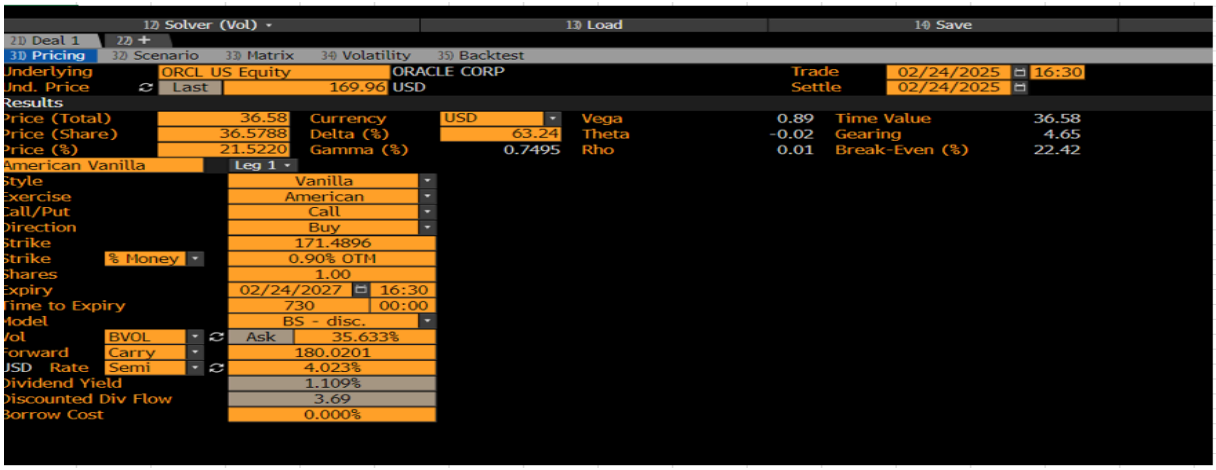
## Annexure 1

### 1) Risk-Free rate




| Date | Market Rate | Shift (bp) | Shifted Rate | Zero Rate | Discount |
|---|---|---|---|---|---|
| 03/03/2025 | 4.33100 | +0.00000 | 4.33100 | 4.38930 | 0.999159 |
| 03/10/2025 | 4.33300 | +0.00000 | 4.33300 | 4.38948 | 0.998318 |
| 03/17/2025 | 4.33500 | +0.00000 | 4.33500 | 4.38966 | 0.997478 |
| 03/24/2025 | 4.33550 | +0.00000 | 4.33550 | 4.38832 | 0.996639 |
| 04/24/2025 | 4.34060 | +0.00000 | 4.34060 | 4.38531 | 0.992936 |
| 05/27/2025 | 4.33820 | +0.00000 | 4.33820 | 4.37425 | 0.989035 |
| 06/24/2025 | 4.33250 | +0.00000 | 4.33250 | 4.36126 | 0.985764 |
| 07/24/2025 | 4.31480 | +0.00000 | 4.31480 | 4.33587 | 0.982339 |
| 08/25/2025 | 4.29795 | +0.00000 | 4.29795 | 4.31098 | 0.978734 |
| 11/24/2025 | 4.25200 | +0.00000 | 4.25200 | 4.24301 | 0.968763 |
| 02/24/2026 | 4.21410 | +0.00000 | 4.21410 | 4.18387 | 0.959024 |
| 08/24/2026 | 4.12536 | +0.00000 | 4.12536 | 4.11052 | 0.940363 |
| 02/24/2027 | 4.09478 | +0.00000 | 4.09478 | 4.06535 | 0.921911 |
| 02/24/2028 | 4.05277 | +0.00000 | 4.05277 | 4.02356 | 0.886294 |
| 02/26/2029 | 4.03501 | +0.00000 | 4.03501 | 4.00584 | 0.851664 |
| 02/25/2030 | 4.02870 | +0.00000 | 4.02870 | 3.99996 | 0.818553 |
| 02/24/2031 | 4.03310 | +0.00000 | 4.03310 | 4.00532 | 0.786290 |
| 02/24/2032 | 4.03960 | +0.00000 | 4.03960 | 4.01284 | 0.755022 |
| 02/24/2033 | 4.04762 | +0.00000 | 4.04762 | 4.02201 | 0.724712 |
| 02/24/2034 | 4.05670 | +0.00000 | 4.05670 | 4.03246 | 0.695487 |
| 02/26/2035 | 4.06786 | +0.00000 | 4.06786 | 4.04541 | 0.666987 |
| 02/24/2037 | 4.09101 | +0.00000 | 4.09101 | 4.07278 | 0.613197 |
| 02/24/2040 | 4.12082 | +0.00000 | 4.12082 | 4.10895 | 0.539733 |
| 02/24/2045 | 4.11831 | +0.00000 | 4.11831 | 4.10152 | 0.440050 |
| 02/24/2050 | 4.05476 | +0.00000 | 4.05476 | 4.00228 | 0.367428 |
| 02/24/2055 | 3.97131 | +0.00000 | 3.97131 | 3.86681 | 0.313241 |
| 02/24/2065 | 3.77570 | +0.00000 | 3.77570 | 3.53561 | 0.242874 |
| 02/25/2075 | 3.58839 | +0.00000 | 3.58839 | 3.20449 | 0.201214 |

### 2) Dividend Yield




## 3) Implied Volatility

31) Pricing 32) Scenario 33) Matrix 34) Volatility 35) Backtest
Underlying ORCL US Equity ORACLE... Exercise American Trade 02/24/2025 16:30 98) Save Settings
Und. Price 169.96USD Direction Buy Call 99) Export to Excel 97) Reset Settings

| Y-Axis | X-Axis | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Moneynes | Maturity | 730D | 638D | 546D | 456D | 365D | 273D | 182D | 92D |
| 65 | Volatility | 37.54% | 38.36% | 39.08% | 40.32% | 41.27% | 41.60% | 43.54% | 49.45% |
| 75 | Volatility | 37.25% | 36.78% | 37.31% | 37.65% | 37.87% | 39.55% | 40.02% | 43.26% |
| 85 | Volatility | 36.05% | 36.23% | 36.25% | 36.37% | 36.70% | 38.05% | 38.14% | 40.50% |
| 100 | Volatility | 35.46% | 35.83% | 36.13% | 36.47% | 36.37% | 37.15% | 37.18% | 39.02% |
| 105 | Volatility | 36.82% | 35.70% | 35.57% | 35.77% | 36.14% | 36.96% | 36.67% | 38.58% |
| 110 | Volatility | 35.22% | 35.32% | 35.32% | 35.87% | 36.83% | 36.73% | 36.24% | 38.12% |
| 120 | Volatility | 34.88% | 35.26% | 35.17% | 35.55% | 36.20% | 36.24% | 35.63% | 37.83% |

31) Pricing 32) Scenario 33) Matrix 34) Volatility 35) Backtest
Underlying ORCL US Equity ORACLE... Exercise American Trade 02/24/2025 16:30 98) Save Settings
Und. Price 169.96USD Direction Sell Call 99) Export to Excel 97) Reset Settings

| Y-Axis | X-Axis | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Moneynes | Maturity | 730D | 638D | 546D | 456D | 365D | 273D | 182D | 92D |
| 65 | Volatility | 37.07% | 36.53% | 37.68% | 37.90% | 37.35% | 41.14% | 41.67% | 44.15% |
| 75 | Volatility | 35.21% | 35.11% | 35.75% | 36.64% | 37.19% | 39.14% | 39.40% | 42.87% |
| 85 | Volatility | 35.25% | 34.15% | 34.88% | 35.82% | 36.31% | 37.59% | 37.79% | 39.94% |
| 100 | Volatility | 34.89% | 34.03% | 34.15% | 34.73% | 35.67% | 36.70% | 36.48% | 38.47% |
| 105 | Volatility | 33.26% | 34.15% | 34.62% | 35.27% | 35.76% | 36.54% | 36.25% | 38.11% |
| 110 | Volatility | 34.60% | 34.52% | 34.76% | 35.03% | 34.94% | 36.38% | 35.90% | 37.77% |
| 120 | Volatility | 34.34% | 34.48% | 34.61% | 34.92% | 35.19% | 35.89% | 35.25% | 37.39% |

## Annexure 2:

```
import numpy as np
import math
import matplotlib.pyplot as plt
from scipy.stats import norm
```

```
[9] def CallYiNo (S0,K1,r,delta,sigma,T,N,FV,coupon,per):      #k1 lower bound

      md = 1
      bd = np.zeros([md])
      bd = [92/365]


      nd = 6
      cd = np.zeros([nd])
      cd = [182/365,273/365,365/365,456/365,546/365,638/365]

      St = np.zeros([N+1,N+1])
      V = np.zeros([N+1,N+1])
      dt = T/N
      u = np.exp(sigma*np.sqrt(dt))
      d = 1/u
      p = (np.exp((r-delta)*dt)-d)/(u-d)


      imd = [i/dt for i in bd]
      imd1 = [math.floor(i) for i in imd]
      delimd = {}
      for i in range(md):
```

```
        delimd[imd1[i]] = dt * (imd[i] - imd1[i])

      id = [i/dt for i in cd]
      id1 = [math.floor(i) for i in id]
      delid = {}
      for i in range(nd):
        delid[id1[i]] = dt * (id[i] - id1[i])

      def Stock(u,d,dt):
        St[0,0] = S0
        for i in range (1,N+1):
              St[i, 0] = St[i-1, 0]*d
              for j in range(1, i+1):
                  St[i, j] = St[i-1, j-1]*u

      Stock(u,d,dt)

      def lambda_func1(K1,St):
        Fi_Pr = St[-1,:]
        Fi_Pr = np.sort(Fi_Pr)

        idx = np.searchsorted(Fi_Pr,K1)

        Sk = Fi_Pr[idx]
        Sk_1 = Fi_Pr[idx-1]

        lamb1 = (Sk - K1)/(Sk - Sk_1)

        return(lamb1)
```

```
    lambd1 = lambda_func1(K1,St)


    i = N
    for j in range (0,N+1):
      if St[i,j]>=K1:
        V[i,j] = FV + (FV * coupon / per)
      else:
        V[i,j] = FV * (St[i,j] /S0)


    for i in range (N-1,-1,-1):
      for j in range (0,i+1):
        V[i,j] = np.exp(-r*dt)*(V[i+1,j+1]*p + V[i+1,j]*(1-p))
        if i in id1:

            cv = np.exp(-r*dt)*( V[i+1,j+1]*p + V[i+1,j]*(1-p) )
            call = np.exp(-delid[i]*r) * FV
            V[i,j] = np.minimum(cv,call)
            if (St[i,j]>=K1):
              V[i,j] = V[i,j] + np.exp(-delid[i]*r) *  (FV * coupon / per)

        if i in imd1:

           if St[i,j] >= K1:
            V[i,j] = np.exp(-r*dt)*( V[i+1,j+1]*p + V[i+1,j]*(1-p) ) + np.exp(-delimd[i]*r) *  (FV * coupon / per)

    output = {'num_steps': N, 'Value': V[0,0], 'LambdaK1':lambd1 }

    return output
```

```
[24] S0 = 169.96
     barrier_percent = 0.65
     K1 = barrier_percent * S0
     FV = 1000
     annual_coupon_rate = 0.1165
     quarterly_rate = annual_coupon_rate / 4
     coupon_amt = FV * quarterly_rate
     def payoff_at_maturity(S):

         payoff = np.where(S >= K1,
                           FV + coupon_amt,
                           FV * (S / S0))
         return payoff
     def payoff_coupon_observation(S):
         payoff_coupon = np.where(S >= K1,
                                  coupon_amt,
                                  0.0)
         return payoff_coupon
     S_range = np.linspace(0, 2 * S0, 300)
     final_payoffs = payoff_at_maturity(S_range)
     coupon_payoffs = payoff_coupon_observation(S_range)
     plt.figure(figsize=(8, 6))
     plt.plot(S_range, final_payoffs, label='Payoff at Maturity (No Call)', color='blue')
     plt.plot(S_range, coupon_payoffs, label='Coupon at Observation Date', color='red', linestyle='--')
     plt.title("Contingent Income Note: Payoff vs. Stock Price")
     plt.xlabel("Underlying Stock Price (Oracle) at Relevant Date")
     plt.ylabel("Payoff ($)")
     plt.legend()
     plt.grid(True, alpha=0.3)
     plt.show()
```

## Annexure 3:

```
N_Val = np.arange(50,2001,10)
Note_val = []
Lambda_Val =[]

for N in N_Val:
  note = CallYiNo(169.69,110.47,0.040653,0.01109,0.3546,2,N,1000,0.1165,4)['Value']
  Note_val.append(note)
  lamd = CallYiNo(169.69,110.47,0.040653,0.01109,0.3546,2,N,1000,0.1165,4)['LambdaK1']
  Lambda_Val.append(lamd)

for i in range(len(N_Val)):
  print(f"N = {N_Val[i]}, Note Price = {Note_val[i]:.8f}, Lambda = {Lambda_Val[i]:.8f}")

plt.figure(figsize=(10,6))
fig, ax1 = plt.subplots(figsize=(10,6))
ax1.plot(N_Val, Note_val, 'b-', label='Note Price')
ax1.set_xlabel('N')
ax1.set_ylabel('Note Price', color='b')
ax1.tick_params('y', colors='b')

ax2 = ax1.twinx()
ax2.plot(N_Val, Lambda_Val, 'r--', label='Lambda Value wrt Barrier')
ax2.set_ylabel('Lambda', color='r')
ax2.tick_params('y', colors='r')
plt.title(" Change in Note Price and Lambda Value with changing value of N ")
ax1.grid(True)
ax1.legend(loc='upper left')
ax2.legend(loc='upper right')
plt.show()
```

```
[23] V_Sigma_S0 = [0.3902,0.3718,0.3715,0.3637,0.3647,0.3613,0.3583,0.3546]
     V_Sigma_65S0 = [0.4925,0.4325,0.4160,0.4127,0.4032,0.3908,0.3836,0.3754]

     Note_ValS0 = []
     Note_Val65S0 = []

     for sigma in V_Sigma_S0:
       note = CallYiNo(169.69,110.47,0.040653,0.01109,sigma,2,10045,1000,0.1165,4)['Value']
       Note_ValS0.append(note)

     for sigma in V_Sigma_65S0:
       note = CallYiNo(169.69,110.47,0.040653,0.01109,sigma,2,10045,1000,0.1165,4)['Value']
       Note_Val65S0.append(note)

     for i in range(len(V_Sigma_S0)):
       print(f"Sigma_S0 = {V_Sigma_S0[i]}, Note Price = {Note_ValS0[i]:.8f}")

     for i in range(len(V_Sigma_65S0)):
       print(f"Sigma_65S0 = {V_Sigma_65S0[i]}, Note Price = {Note_Val65S0[i]:.8f}")

     plt.figure(figsize=(10,6))
     plt.scatter(V_Sigma_S0, Note_ValS0, color='b', label='Note Price for IVs at observation date with Strike S0 ', marker='o')
     plt.scatter(V_Sigma_65S0, Note_Val65S0, color='r', label='Note Price for IVs at observation date with Strike S0', 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()
```