

## Introduction

An Excel financial model processes inputs using functions and formulae to produce output calculations. This article illustrates what happened when the 'user', in late February 2026, prompted GPT 5.3 (Codex-OpenAI, released 6 Feb 2026) and Sonnet 4.6 (Claude-Anthropic, released 17 Feb 2026) to explore how efficiently they can produce an Excel spreadsheet from scratch, via Python, that matches an existing (human prepared) model: the convertible bond pricing model in the article 'Security Valuation- Convertible Bond Pricing'.

Rather than prompt Codex and Claude to verify that the uploaded model was 'correct' (the calculations could be correct, but the methodology flawed), it was decided to:

- (1) Write a detailed prompt, giving input and methodology instructions only, without uploading the model or stating what the price should be (Instructions → Python → Excel).

The 'user' would then compare the result to the actual model, and give further prompts to clarify and correct errors in the model prepared by Codex / Claude after each prompt. Hopefully, the price would converge to 115.00. This tested how well the methodology was understood and how formulae were written.

- (2) Write a basic prompt, upload the model and state the price should be 115.00 (Appendix 3-Test 2: Excel → Python → Excel 'roundtrip').

This tested how good they were at analysing an existing model and preparing another version (hopefully with formulae improvements).

<sup>1</sup> These numbers are nice round numbers simply due to excel manipulation: the straight debt price of 100.00 was set by adjusting the redemption premium (given an assumed yield) and the equity component 15.00 by adjusting the existing share price.

## Methodology

The results are only as good as the prompting in the chats, particularly in Test 1 (building the excel model from instructions). If all errors from the first prompt are addressed in the second prompt, a perfect score should follow (unless new errors surface after prompt 2 or the LLM goes in a different direction of analysis, which certainly happened).

The LLM excel models produced after each prompt were reviewed in the following module order: conversion value, probability of conversion, blended discount rate, convertible price. A prompt followed when the first error was spotted, whereas what should have been done was to go through the whole model and check all errors, but . . . . no thanks.

Two runs were performed in each test:

- Test 1: Codex in Test 1 / Run 1 / Prompt 1 was the first attempt at giving instructions. Once Codex matched the 115.00 convertible price (Python code = Excel outputs result), the cumulative prompts from that Run 1 were included in Prompt 1 for Run 2. This was then used when testing Claude for Test 1 and the same procedure was used for Claude's Run 2.
- Test 2, both LLMs had the same prompt. The convertible pricing model was uploaded with a basic prompt saying reproduce it and make sure the result is 115.00. No guidance on methodology was given

New chats or threads were started with each run without access to past chats.

## Test 1

### *Codex / GPT 5.3*

The first prompt was sent to Codex (see below – Codex – Run 1 – Prompt 1), with detailed instructions about how to build the model, though no knowledge about how it would interpret it (the writer has zero coding knowledge).

An arbitrary 'call cap' was introduced by Codex, without any prompting (removed at Prompt 2). CB price = 116.102593. Treatment of the coupon was then clarified (Prompt 2), since, although the prompt stated "Bond coupons are added at each node to values (exception conversion value)," no coupon was added to the call price at time 5 node 3 (or in any of the convertible price formulae). Prompt 4 had to clarify that the coupon should be added at the time step (6) when the bond is redeemed. CB price = 117.259502 (user: "heading in the wrong direction!").

Despite the prompt stating how the continuing value should be calculated from next period values (convertible price, risk neutral probability, blended discount rate – after Goldman Sachs (1994)), the Tsiveriotis and Fernandes (1998) approach was used (separate equity and debt components discounted at risk free rate and credit adjusted rate, respectively). This was corrected in Prompt 5. CB price = 117.0000 (user: "I think I know the problem")- Prompt 6: "The coupon is not added at the root node" CB price = 115.0000. Bingo!

A second run was performed with the first prompt including the extra prompts in run1. Codex got 115.00000 after one prompt: 784 lines of Python code, 9 minutes.

## *Claude / Sonnet 4.6*

The same prompt used for the Codex Run 2 test was used (matched in one prompt). Unfortunately, Claude stumbled a bit here, although its first attempt at prompt 1 was 114.4662, so things looked promising. The issues were:

- Excel had hard coded numbers for output cells in the lattice, rather than embedded formulae (as prompt 1 stated). Claude: “You're right. Let me rebuild those with proper Excel formulas throughout.”
- The probability of conversion is 0.00 if the CB price equals the bond call or redemption price and 1.00 if it equals the conversion value. Claude: “You're right — the probability of conversion at interior nodes should reflect the actual decision at that node” No change to price.
- The risk neutral probability equation adjusted for the dividend, when the dividend was built into the cum-div / ex-div share price. The price increases to 115.5947

Back to the bond coupon. . . . At prompt 5, Claude got a bit worked up when the user suggested the coupon wasn't added to the continuing (hold) value. Claude was convinced the user was at fault, referring to a previous model version, when in fact it had not read prompt 1 correctly. Prompt 6 reiterated that the Goldman Sachs and not Tsiveriotis and Fernandes method should be used. Claude: “You're absolutely right. I misread the GS1994 methodology. . . . No separate equity/debt discounting — that's the T&F approach.” Misread?

As for Codex, a second run was performed with all prompts from Run 1 included, but this time Claude completely ignored the dividend. Prompt 2 instructed it to read the prompt again (the user took the opportunity to mention that hard coded numbers appeared in some lattice trees – not good!). CB price 114.988159 (so close!). Prompt 3 reminded that the conversion value is based on the ex-div share price and not cum-div (CB price no 114.591889) – of course this should have been spotted by the user after prompt 1. The last error (again, could have been spotted earlier). The 115.0000 price was only reached after the final prompt (4) which addressed another issued about the coupon! 877 lines of Python code, 26 minutes.

## **Test 2**

### *Codex / GPT 5.3*

Codex produced 250 lines of Python (vs. 784 in test 1) and the correct Excel model in 12 mins (see App. 3). The prompt instructed no 64 bit encoding (whatever that means), since, as the write now knows, the whole uploaded model could be converted lock, stock and barrel into Python code in gibberish code.

## *Claude / Sonnet 4.6*

Claude produced 581 lines of Python (vs. 877 in test 1) and the correct Excel model in 22 mins (see App. 3 for a note on prompting issues and Claude freezing).

### Conclusions

A number of observations can be made from this surprisingly entertaining exercise:

- Prompting from the user has to be accurate and complete: any subsequent prompt can take the LLM in the wrong direction and lead to bloated formulae (see App.2 end of Codex section, where the probability of conversion prompts got a bit messy – the user ended up saying ‘do it your own way’)
- Prompting needs to be read completely by the LLM: quite often simple instructions were ignored and were only followed after a prompt.
- Python output didn’t always agree to Excel output, even though that was a requirement
- Excel output cells did not always have embedded formulae, but hard coded values from Python (despite this being requirement in all prompts)
- Best practice Excel model design is not always followed:
  - The fill right rule: the first column should be filled and copied right: the final time step had different formulae to intermediate time steps, which hinders model flexibility if more periods are to be added.
  - Input scenario inflexibility: some of the input alternatives were ignored in the LLM excel models.
  - Formulae were very long and not broken down. No flags were used.
- Claude was more interesting to observe, as it gave more insight into its thinking. Its language was less formal and (dare one say) robotic, and more pleasing to ‘converse’ with.
- Claude formatted the Excel output much better (shading, borders etc)

## CHAT HISTORY

### TEST 1: INSTRUCTIONS → PYTHON → EXCEL

#### CODEX / GPT 5.3

Run 1 (matched after 6 prompts, 19mins): 843 lines of Python Code

Font colour scheme: Prompts, Codex chat, Claude chat, [Notes]

#### PROMPT 1

“(A) General task:

You must write standalone, human-readable Python code that:

- Prices a convertible bond using the binomial tree (lattice) method
- Generates an Excel spreadsheet with embedded formulae that can be changed (via assumptions sheet)
- Results in a convertible bond price rounded to the nearest 4- 6 decimal places
- Test the model

(B) Python constraints:

- Use only standard Python and common numerical libraries (numpy, pandas, openpyxl/xlsxwriter or similar for Excel I/O).
- Do not rely on any 64-bit or platform-specific compiled extensions; the code must run in a plain 64-bit Python environment but not depend on embedded binary code.

(C) Model assumptions:

(1) Bond terms

- Face value = \$100.000
- Redemption amount = \$129.318
- Maturity = 6 years
- Coupon = 2.000%
- Coupon payments per year = 1 starting at time step 1
- Time steps in model = 6 (valuation date at time 0)
- Yield = 6.000% p.a. continuous rate
- Time convention = actual / actual (365 days in year)
- Callable at time steps 4, 5 and 6 at the following premium to face value: 18.351814%, 23.670281% and 29.317624% respectively

(2) Conversion features

- Share price at valuation date = \$14.3745776350793
- Volatility = 40.0%
- Risk free rate = 5.000% p.a. continuous rate
- Conversion ratio = 5.00 shares
- Conversion at any node / time step = yes
- Dividend = 1.00% per time step (discrete rate)

Share prices are adjusted for the dividend just paid:

- (1) at  $t=1$ , 'cum-div' share price is  $t=0$  ex-dividend share price (the input ex-div price) adjusted by up and down factors which results in the  $t=1$  'cum-div' share price,
- (2) the dividend for  $t=1$  is the  $t=1$  cum-div price multiplied by the 1.00% dividend yield,
- (3) the 'ex-div' price is the cum-div price less the dividend,
- (4) the next period cum-div is the previous period ex-div price adjusted for  $u$  and  $d$  etc

(3) Binomial features

- Parameters based on CRR methodology
- Discount rate based on blended rate
- Goldman Sachs (1994) and not Tsiveriotis and Fernandes (1998)
- All rates are the same each period
- Discrete compounding in the lattice (from continuous rates)
- Bond coupons are added at each node to values (exception conversion value)
- The discount rates and all rates based on the input assumptions must be converted to discrete rates

(4) Excel model structure

- Only inputs section must have hard coded numbers
- Lattice must only have embedded formulae
- Time steps are on top row (no column gaps between steps)
- Diagonal layout with up movements on top row
- Separate lattices for: share price, conversion value, convertible price, debt and equity components, holder action, probability of conversion, blended rate

(D) Model verification and summary

- Test the model
- Prepare a full audit and verification report
- Prepare Python scrip for download
- Prepare standalone Excel spreadsheet for download

Codex 1

Implemented and generated all requested artifacts. Convertible bond price ( $t=0$ ): 116.102593 (6 d.p.)

What was included:

- CRR binomial convertible bond pricer with GS (1994)-style debt/equity decomposition (notTF).
- Dividend handling per your ex-div/cum-div sequence using dividend yield.
- Callable schedule at steps 4/5/6 and node-level conversion allowed at all steps.
- Excel with assumptions hard-coded only on ASSUMPTIONS; lattice blocks are embedded formulas on lattice

## PROMPT 2

Please remove the no call cap

[User: This wasn't asked for!]

Codex 2

I'll remove the artificial no-call sentinel from the generator and switch formulas to explicit non-call-period logic ( $t < 4$ ) so early nodes only compare convert vs hold.

I've regenerated the files successfully; next I'm doing a quick formula spot-check to confirm the no-call cap row is gone and pre-call steps use direct convert vs hold logic.

Root convertible value = 116.102593

## PROMPT 3

The coupon must be added to the call price. The convertible price will include a coupon unless it represents the conversion value, when no coupon is added

Codex 3

Call payoff now includes coupon:

Python lattice uses  $\text{call\_value} = \text{call\_price} + \text{coupon\_amount}$  at callable steps.

Conversion value remains coupon-free

New root price: 116.467248

## PROMPT 4

The coupon is added to the final node when called or redeemed

Codex 4

Final-node non-conversion payoff now includes coupon in both code and Excel:

Root price: 117.259502

## PROMPT 5

The blended rate must be calculated using the probability of conversion and this blended rate is then used to discount the next period price (Goldman Sachs 1994 and not Tsiveriotis and Fernandes (1998))

Codex 5

I changed the model so the blended rate is now used directly to discount continuation value. At each non-terminal node, I compute:

hold\_conversion\_prob = p \* prob\_up + (1-p) \* prob\_down  
hold\_blended\_rate = hold\_conversion\_prob \* r\_f + (1-hold\_conversion\_prob) \* y  
hold\_value = E[next CB value] / (1 + hold\_blended\_rate) + coupon  
Decision logic then applies Convert / Call / Hold.  
This is now aligned with Goldman Sachs (1994) blended-rate discounting (notTF).

Root CB price: 117.000000

### PROMPT 6

The coupon is not added at the root node

Codex 6

I verified the current root node already includes coupon mathematically, but I'll make it explicit in the recursion structure so the coupon is visibly added at the node decision (except conversion) rather than hidden inside the hold formula. This will keep behavior aligned with your rule and make root-node treatment unambiguous.

Root hold ex-coupon: 115.000000...

[User: Finally go there!]

Run 2 (matched after 1 prompt, 9 mins): 784 lines of Python code

### PROMPT 1

The Run 1 initial prompt 1 was amended as follows:

- Underlined sections were deleted
- The following was added either as a new instruction or clarification:

#### (1) Bond terms

"There is no call cap"

#### "(2) Coupon treatment

The coupon is added to the call amount when comparing payoffs

The coupon is added when the bond is priced as straight debt (for example at maturity when not converting)

The coupon is added to the continuing value

The coupon is not added to the conversion value

The coupon is not added at the root node"

#### (3) Binomial features

"Discount rate based on blended rate based on probability of conversion based on Goldman Sachs (1994)

Do not use Tsiveriotis and Fernandes (1998)“

(4) Excel model structure

“No hard coded values except in inputs or assumptions section“

#### Codex 1

I’m going to build a standalone Python script that computes the 6-step convertible bond lattice, writes an Excel workbook with editable assumptions and embedded formulas, then runs verification checks and produces an audit report. First I’ll inspect the workspace and then create the script, workbook output, and report files.

The workspace is essentially empty, so I’ll create a new script from scratch and include everything needed: model engine, Excel generator with formula cells, verification checks, and an audit report writer. I’ll quickly confirm the Python packages available before writing it.

I’ve confirmed xlswriter and openpyxl are available (while numpy/pandas are not), so I’m implementing the model with pure Python + Excel libraries only. Next I’m writing a single standalone script that prices the bond, creates the formula-driven workbook, runs checks, and emits a detailed audit report file.

The script is in place. I’m running it now to generate the Excel workbook and audit report, then I’ll validate the outputs and confirm the Python and spreadsheet values match.

I’m tightening the Excel model so it avoids hardcoded step numbers in formulas by referencing each column’s time-step header, and I’m moving the non-call sentinel into the assumptions table as a named input. Then I’ll rerun the script and recheck verification.

The model still passes all checks after that formula cleanup. I’m doing a final pass on the generated report and output artifacts, then I’ll give you the exact file paths and a concise summary of results.

Python root value: 115.000000001362

Excel cached root value: 115.000000001362

All implemented checks passed (probability bounds, component consistency, blended-rate bounds, formula presence, and Python-vs-Excel match).

[User: “10/10“]

## Codex Excel model (Run 2)

### Assumptions Sheet

	A	B	C
1	<b>Parameter</b>	<b>Value</b>	<b>Notes</b>
2	Face value	100.0000000000	USD
3	Redemption amount	129.3180000000	USD
4	Maturity (years)	6	Years
5	Coupon rate	2.000000%	Per annum
6	Coupon payments per year	1	Frequency
7	Model time steps	6	Lattice periods
8	Yield (continuous)	6.000000%	Continuous p.a.
9	Risk-free rate (continuous)	5.000000%	Continuous p.a.
10	Credit-adjusted rate (continuous)	6.000000%	Continuous p.a.
11	Volatility	40.000000%	Per annum
12	Spot share price (t=0 ex-div)	14.3745776351	USD
13	Conversion ratio (shares)	5.0000000000	Shares per bond
14	Dividend per step (discrete)	1.000000%	Per step
15	Call step 1	4	Time step index
16	Call premium 1	18.351814%	Premium to face
17	Call step 2	5	Time step index
18	Call premium 2	23.670281%	Premium to face
19	Call step 3	6	Time step index
20	Call premium 3	29.317624%	Premium to face
22			
23	<b>Derived parameters</b>		
24	Time step (years)	1.0000000000	Years per step
25	Coupon amount per step	2.0000000000	USD
26	Up factor (u)	1.4918246976	CRR up move
27	Down factor (d)	0.6703200460	CRR down move
28	Risk-free discrete rate	5.127110%	Per step
29	Credit discrete rate	6.183655%	Per step
30	Yield discrete rate	6.183655%	Per step
31	Risk-neutral probability (p)	0.4637235463	CRR probability
32			

### Summary sheet

	A	B
1	<b>Metric</b>	<b>Value</b>
2	Convertible bond value at t=0	115.0000000014
3	Conversion value at t=0	71.8728881754
4	Debt component at t=0	85.0827990004
5	Equity component at t=0	29.9172010010
6	Probability of conversion at t=0	0.2601495739
7	Blended discount rate at t=0 (discrete)	0.0590879492
8	Debt + Equity consistency check	115.0000000014
9	Difference vs. convertible value	0.0000000000
10		

Lattice sheet

	A	B	C	D	E	F	G	H
23	<b>Conversion Value Lattice</b>							
24	<b>Top row = all up moves</b>							
25		0	1	2	3	4	5	6
26		71.872888	106.149532	156.772929	231.538950	341.961369	505.044951	745.904146
27			47.696059	70.442618	104.037156	153.653148	226.931325	335.156337
28				31.651908	46.746908	69.040810	101.966817	150.595450
29					21.004740	31.022035	45.816644	67.666897
30						13.939099	20.586745	30.404697
31							9.250221	13.661711
32								6.138602
34	<b>Convertible Price Lattice</b>							
35	<b>Top row = all up moves</b>							
36		0	1	2	3	4	5	6
37		115.000000014	136.262731	170.207683	231.538950	341.961369	505.044951	745.904146
38			109.284823	117.483333	130.479194	153.653148	226.931325	335.156337
39				110.626012	115.343070	120.351814	125.670281	150.595450
40					115.343070	120.351813	125.670281	131.317624
41						120.351813	125.670281	131.317624
42							125.670281	131.317624
43								131.317624
45	<b>Probability of Conversion Lattice</b>							
46	<b>Top row = all up moves</b>							
47		0	1	2	3	4	5	6
48		26.0150%	44.5681%	71.2408%	100.0000%	100.0000%	100.0000%	100.0000%
49			9.9719%	21.5040%	46.3724%	100.0000%	100.0000%	100.0000%
50				-	-	-	-	100.0000%
51					-	-	-	-
52						-	-	-
53							-	-
56	<b>Blended Discount Rate Lattice</b>							
57	<b>Top row = all up moves</b>							
58		0	1	2	3	4	5	6
59		5.9088%	5.7128%	5.4310%	5.1271%	5.1271%	5.1271%	5.1271%
60			6.0783%	5.9565%	5.6937%	5.1271%	5.1271%	5.1271%
61				6.1837%	6.1837%	6.1837%	6.1837%	5.1271%
62					6.1837%	6.1837%	6.1837%	6.1837%
63						6.1837%	6.1837%	6.1837%
64							6.1837%	6.1837%
67	<b>Debt Component Lattice</b>							
68	<b>Top row = all up moves</b>							
69		0	1	2	3	4	5	6
70		85.082799	75.533048	48.950442	0.000000	0.000000	0.000000	0.000000
71			98.387061	92.219773	69.972919	0.000000	0.000000	0.000000
72				110.626012	115.343070	120.351814	125.670281	0.000000
73					115.343070	120.351813	125.670281	131.317624
74						120.351813	125.670281	131.317624
75							125.670281	131.317624
78	<b>Equity Component Lattice</b>							
79	<b>Top row = all up moves</b>							
80		0	1	2	3	4	5	6
81		29.917201	60.729683	121.257241	231.538950	341.961369	505.044951	745.904146
82			10.897761	25.263560	60.506274	153.653148	226.931325	335.156337
83				0.000000	0.000000	0.000000	0.000000	150.595450
84					0.000000	0.000000	0.000000	0.000000
85						0.000000	0.000000	0.000000
86							0.000000	0.000000
89	<b>Holder Action Lattice</b>							
90	<b>Top row = all up moves</b>							
91		0	1	2	3	4	5	6
92		Hold	Hold	Hold	Convert	Convert	Convert	Convert
93			Hold	Hold	Hold	Called -> Convert	Convert	Convert
94				Hold	Hold	Hold	Called -> Cash	Convert
95					Hold	Hold	Hold	Redeem/Called Cash
96						Hold	Hold	Redeem/Called Cash
97							Hold	Redeem/Called Cash

The formulae for the top rows (up moves) at time 5 and 6 are as follows:

	5	6	
Convertible Price	505.044951		=MAX(G25,MIN((CouponAmt+(RiskNeutralProb*H36+(1-RiskNeutralProb)*H37)/(1+((RiskNeutralProb*H47+(1-RiskNeutralProb)*H48)*RiskFreeDisc+(1-(RiskNeutralProb*H47+(1-RiskNeutralProb)*H48))*CreditDisc))),IF(G35=CallStep1,FaceValue*(1+CallPrem1)+(CouponAmt),IF(G35=CallStep2,FaceValue*(1+CallPrem2)+(CouponAmt),IF(G35=CallStep3,FaceValue*(1+CallPrem3)+(CouponAmt),LargeNumber))))))
		745.904146	=MAX(H25,MIN((RedemptionAmount+(CouponAmt)),IF(H35=CallStep1,FaceValue*(1+CallPrem1)+(CouponAmt),IF(H35=CallStep2,FaceValue*(1+CallPrem2)+(CouponAmt),IF(H35=CallStep3,FaceValue*(1+CallPrem3)+(CouponAmt),LargeNumber))))))
Probability of Conversion	100.0000%		=IF(MAX(G25,IF(G35=CallStep1,FaceValue*(1+CallPrem1)+(CouponAmt),IF(G35=CallStep2,FaceValue*(1+CallPrem2)+(CouponAmt),IF(G35=CallStep3,FaceValue*(1+CallPrem3)+(CouponAmt),LargeNumber))))<MAX(G25,(CouponAmt+(RiskNeutralProb*H36+(1-RiskNeutralProb)*H37)/(1+((RiskNeutralProb*H47+(1-RiskNeutralProb)*H48)*RiskFreeDisc+(1-(RiskNeutralProb*H47+(1-RiskNeutralProb)*H48))*CreditDisc))),IF(G25>=IF(G35=CallStep1,FaceValue*(1+CallPrem1)+(CouponAmt),IF(G35=CallStep2,FaceValue*(1+CallPrem2)+(CouponAmt),IF(G35=CallStep3,FaceValue*(1+CallPrem3)+(CouponAmt),LargeNumber))),1,0),IF(G25>=(CouponAmt+(RiskNeutralProb*H36+(1-RiskNeutralProb)*H37)/(1+((RiskNeutralProb*H47+(1-RiskNeutralProb)*H48)*RiskFreeDisc+(1-(RiskNeutralProb*H47+(1-RiskNeutralProb)*H48))*CreditDisc))),1,(RiskNeutralProb*H47+(1-RiskNeutralProb)*H48)))
		100.0000%	=IF(H25>=MIN((RedemptionAmount+(CouponAmt)),IF(H35=CallStep1,FaceValue*(1+CallPrem1)+(CouponAmt),IF(H35=CallStep2,FaceValue*(1+CallPrem2)+(CouponAmt),IF(H35=CallStep3,FaceValue*(1+CallPrem3)+(CouponAmt),LargeNumber))))),1,0)
Blended Discount Rate	5.1271%		=G47*RiskFreeDisc+(1-G47)*CreditDisc
		5.1271%	=H47*RiskFreeDisc+(1-H47)*CreditDisc
Debt Component	0.000000		=G36*(1-G47)
		0.000000	=H36*(1-H47)
Equity Component	505.044951		=G36*G47
		745.904146	=H36*H47

Holder Action	Convert	=IF(MAX(G25,IF(G35=CallStep1,FaceValue*(1+CallPrem1)+(CouponAmt),IF(G35=CallStep2,FaceValue*(1+CallPrem2)+(CouponAmt),IF(G35=CallStep3,FaceValue*(1+CallPrem3)+(CouponAmt),LargeNumber))))<MAX(G25,(CouponAmt+(RiskNeutralProb*H36+(1-RiskNeutralProb)*H37)/(1+((RiskNeutralProb*H47+(1-RiskNeutralProb)*H48)*RiskFreeDisc+(1-(RiskNeutralProb*H47+(1-RiskNeutralProb)*H48))*CreditDisc))),IF(G25>=IF(G35=CallStep1,FaceValue*(1+CallPrem1)+(CouponAmt),IF(G35=CallStep2,FaceValue*(1+CallPrem2)+(CouponAmt),IF(G35=CallStep3,FaceValue*(1+CallPrem3)+(CouponAmt),LargeNumber))),"Called -> Convert","Called -> Cash"),IF(G25>=(CouponAmt+(RiskNeutralProb*H36+(1-RiskNeutralProb)*H37)/(1+((RiskNeutralProb*H47+(1-RiskNeutralProb)*H48)*RiskFreeDisc+(1-(RiskNeutralProb*H47+(1-RiskNeutralProb)*H48))*CreditDisc))),"Convert","Hold"))
	Convert	=IF(H25>=MIN((RedemptionAmount+(CouponAmt)),IF(H35=CallStep1,FaceValue*(1+CallPrem1)+(CouponAmt),IF(H35=CallStep2,FaceValue*(1+CallPrem2)+(CouponAmt),IF(H35=CallStep3,FaceValue*(1+CallPrem3)+(CouponAmt),LargeNumber))),"Convert","Redeem/Called Cash")

In practice these formulae would be broken down into smaller parts with flags and other helper columns.

In another run for Codex, the wording for the probability of conversion prompt in Run 2 was different:

“Discount rate based on blended rate based on probability of conversion based on Goldman Sachs (1994) Probability of conversion is based on next period probabilities (using the equation), except it will equal 1.00 if the convertible price is the conversion value or 0.00 if the convertible price is the redemption or call value or continuing value. Do not use Tsiveriotis and Fernandes (1998)”

It was later expanded: “The probability of conversion should be: If CB price = conversion value → prob = 1.0. If CB price = straight debt value or continuing value → prob = 0.0. Otherwise → prob = p × P\_up + (1-p) × P\_down (expected future probability)”

This prompt wasn’t brilliant, as it missed a bit about the call having zero conversion probability. The formula that now resulted was the following (time 5 as an example, each cell):

```
=IF(ABS(IF(Assumptions!G28=1,MIN(MAX(ConversionValue!G2,((Assumptions!$B$23*ConvertibleValue!H2+(1-Assumptions!$B$23)*ConvertibleValue!H3)/(1+(Assumptions!$B$23*PCov!H2+(1-Assumptions!$B$23)*PCov!H3)*Assumptions!$B$18+(1-Assumptions!$B$23*PCov!H2+(1-Assumptions!$B$19)))+IF(G1>0,Assumptions!$B$17,0))),MAX(ConversionValue!G2,(Assumptions!G29+IF(G1>0,Assumptions!$B$17,0))),MAX(ConversionValue!G2,((Assumptions!$B$23*ConvertibleValue!H2+(1-Assumptions!$B$23)*ConvertibleValue!H3)/(1+(Assumptions!$B$23*PCov!H2+(1-Assumptions!$B$23)*PCov!H3)*Assumptions!$B$18+(1-Assumptions!$B$23*PCov!H2+(1-Assumptions!$B$23)*PCov!H3)*Assumptions!$B$19))+IF(G1>0,Assumptions!$B$17,0))))-ConversionValue!G2)<=Assumptions!$B$24,1,IF(AND(Assumptions!G28=1,ABS(IF(Assumptions!G28=1,MIN(MAX(ConversionValue!G2,((Assumptions!$B$23*ConvertibleValue!H2+(1-Assumptions!$B$23)*ConvertibleValue!H3)/(1+(Assumptions!$B$23*PCov!H2+(1-Assumptions!$B$23)*PCov!H3)*Assumptions!$B$18+(1-Assumptions!$B$23*PCov!H2+(1-Assumptions!$B$19)))+IF(G1>0,Assumptions!$B$17,0))),MAX(ConversionValue!G2,(Assumptions!G29+IF(G1>0,Assumptions!$B$17,0))),MAX(ConversionValue!G2,((Assumptions!$B$23*ConvertibleValue!H2+(1-
```



Run 1 (matched after 6 prompts): 729 lines of Python Code

PROMPT 1

The same prompt as for Codax Run 2 (matched first time) was used.

Claude 1

I'll build this step by step. Let me first read the relevant skill files, then implement the model. Now let me build the comprehensive Python model and Excel spreadsheet. The model runs perfectly. Now let me recalculate the Excel formulas and copy the Python script to outputs: Model Results \$114.4662

PROMPT 2

The spreadsheet has hard coded numbers in the lattice output sections. Lattice cells must have embedded formulae and not numbers.

Claude 2

You're right. Let me rebuild those with proper Excel formulas throughout. The challenge is that each lattice node depends on its two child nodes, so I need to write backward-induction formulas referencing adjacent cells within each sheet, plus cross-references to other sheets for conversion values and call prices.

262 formulas, zero errors. Let me do one final spot-check — verify the Excel-computed root node matches Python's \$114.4662:

PROMPT 3

Before the final node, the probability of conversion will be zero if the bond is called (convertible price = call price + coupon) or redeemed (convertible price is redemption amount + coupon) and 1.00 if the convertible price is the conversion value, otherwise the formulae as given in ProbConversion Interior is correct.

Claude 3

You're right — the probability of conversion at interior nodes should reflect the actual decision at that node, not just the discounted expectation from children. This means the probability formula needs to check the same call/convert/hold conditions as the other lattices. Now I understand the full picture. The fix is clear: interior nodes need the same decision tree as terminal nodes — returning 1 if converting, 0 if called/redeemed, and pc\_cont only when holding. I also need to fix the Python model to match, then regenerate everything.

CB price unchanged at \$114.4662 — the D+E split and blended rates will now differ slightly since they draw on the corrected probabilities

#### PROMPT 4

The dividend is taken into account in the share price so the risk neutral probability equation in Assumptions B26 must not include the dividend yield reduction

Claude 4

You're right. The dividend is already embedded in the share price tree via the  $(1-q)^t$  factor, so it must not also be removed from the risk-neutral probability. The correct CRR  $p$  under continuous risk-free rate (no dividend adjustment) is simply:  $p = (e^{(r-dt)} - d) / (u - d)$ . ... Price \$115.5947

#### PROMPT 5

The continuing value in the convertible bond pricing tree must include the coupon. For example E7 in ConvertiblePrice sheet does not include the coupon.

[User: This one really threw a spanner in the works]

Claude 5

The coupon is included in cont — the problem is that cont is only used as the floor for MAX vs CV, but when the bond is not called and not converting (i.e.  $cont > CV$ ), the CB value returned is cont itself which correctly includes the coupon. Let me look more carefully at the callable node formula and the equity/debt component formulas to find where the coupon is actually missing. Now I can see the bug clearly. In the equity component at a non-callable hold node (e.g. E7,  $t=3, j=2$ ), when the bond is held (not converting), it returns:

$$(p \cdot EQ_{up} + (1-p) \cdot EQ_{dn}) / (1+rf) \leftarrow \text{NO COUPON}$$

But the debt component correctly returns:

$$(p \cdot DEBT_{up} + (1-p) \cdot DEBT_{dn}) / (1+cr) + coupon \leftarrow \text{COUPON ADDED}$$

All values match. The issue must be that the user is seeing the previous version of the file (before the formula fix). But let me re-read the problem statement carefully — the user says "the continuing value must include the coupon" and cites E7 as an example that doesn't. In `fml_cb` for non-callable nodes, cont is formed as `ec+dc+ca`. But `ca_expr(t)` returns `CPN_` for  $t > 0$  and "0" for  $t = 0$ . Let me check: E7 is  $t=3, j=2$ .  $t=3 > 0$ , so `ca = CPN_`. That IS included.

[User: There then follows a very long period of (hilarious?) analysis and questioning]:

"All values match. The issue must be that the user is seeing the previous version of the file (before the formula fix). But let me re-read the problem statement carefully. ... But — I just noticed something. ... Wait, re-reading the user's statement more carefully. ... E7 in the user's spreadsheet might be a different cell if they opened it in Excel and the row/column layout differs. ..."

Wait — no.... Let me think about what the user is actually seeing. They say E7 doesn't include the coupon. The formula at E7 IS: =MAX((eq\_cont + debt\_cont + CPN\_), CV) — coupon IS there.... I think the user may genuinely be pointing to a version discrepancy... The most productive thing to do is to take the user's feedback at face value, carefully re-examine fml\_cb for non-callable interior nodes, and make sure the coupon is properly added...

Let me look at this from the user's perspective: if they look at E7 in ConvertiblePrice and expand the formula manually, they see ec + dc + cpn. But if they look at E7 in EquityComponent, they see ec (no coupon). And E7 in DebtComponent shows dc + cpn. Perhaps the confusion is that the user wants the ConvertiblePrice cont to be computed as EquityComponent + DebtComponent (which already includes coupon in the debt part), not ec + dc + cpn separately. This would mean the ConvertiblePrice formula should reference the own-sheet children (i.e., CB\_up and CB\_dn) rather than separately summing EQ and DEBT children.

Actually — this is exactly the structural issue. The ConvertiblePrice formula currently recomputes ec and dc from scratch from EQ/DEBT children, then adds coupon. But if instead it just referenced  $(p*CB_{up} + (1-p)*CB_{dn})/(1+blended\_rate) + coupon$ , it would be cleaner. However under GS1994 we discount equity at rf and debt at credit rate separately, so we can't use a single blended rate on the combined CB value for the continuation — we must split.

I'm now confident the user is referring to the previously delivered file, not the current one.... Now let me look at this from a completely fresh angle.... I'm now very confident the formulas are correct and the user is referring to the older file.

All correct. But I wonder if the user means something different — perhaps they want the ConvertiblePrice formula to NOT split equity/debt and just use the blended discounted CB children + coupon? [User: "Bingo"] Let me check whether that approach gives the same answer as GS1994:

Actually — I think I now understand the user's actual point. Looking at the structure more carefully:

The cont in the ConvertiblePrice formula is computed as:

$$ec + dc + coupon = (p*EQ_{up} + (1-p)*EQ_{dn})/(1+rf) + (p*DEBT_{up} + (1-p)*DEBT_{dn})/(1+cr) + coupon$$

All formulas are correct in the current file. The user must have been looking at the previous version. [User: "Oh no"]

After an exhaustive audit I can confirm the coupon is correctly included in every interior node of the ConvertiblePrice lattice.

[User: "Oh no it isn't"]

## PROMPT 6

The convertible price must be calculated using the blended discount rate and not valuing debt and equity separately. The first prompt clearly says: "Discount rate based on blended rate based on probability of conversion based on Goldman Sachs (1994) Do not use Tsiveriotis and Fernandes (1998)".

## Claude 6

You're absolutely right. I misread the GS1994 methodology. The correct approach is: Compute the blended discount rate at each node:  $r_{\text{blend}} = p_{\text{conv}} \times r_f + (1-p_{\text{conv}}) \times \text{credit}$ . Discount the total CB value (not split equity/debt) at this single blended rate. No separate equity/debt discounting — that's the T&F approach. This simplifies the tree significantly. The debt/equity component sheets should still show the split, but the CB price itself is computed via the blended rate on the combined continuation value. Now I have a complete picture. I need to rewrite the model fundamentally. The GS1994 approach is:

ProbConversion is computed first (same node-level logic as before).  $\text{BlendedRate} = p_{\text{conv}} \times r_f + (1-p_{\text{conv}}) \times \text{credit}$  at each node.  $\text{CB price} = (p \times \text{CB}_{\text{up}} + (1-p) \times \text{CB}_{\text{dn}}) / (1 + \text{blended\_rate}) + \text{coupon}$  — one combined discounting step

The equity/debt split is then derived from the CB price, not computed separately as inputs to CB

Model runs correctly — CB = \$115.00 using the GS1994 single blended-rate tree.

[User: !!!]

## Run 2 (matched after 4 prompts, 26 mins): 877 lines of Python Code

### PROMPT 1

The same prompt as for Claude Run 1 plus prompts from Run 1

### Claude 1

The model is correct. At  $t=6$ , the call price (131.317624) is slightly below the redemption+coupon (131.318), so the issuer calls at the lower amount. Now let me build the comprehensive Excel spreadsheet: Convertible Bond Price at Root: \$114.988159

### PROMPT 2

The dividend has been ignored. Read the prompt again under section (4) dividend treatment. Also, all lattices must have embedded formulae with no hard coded numbers, including the probability of conversion sheet and holder action sheet.

### Claude 2

Let me re-read the specification carefully and examine the current implementation. Now I understand all the issues clearly. Let me rebuild the full model with:

Explicit dividend recurrence in share price lattice (cum-div = prev ex-div  $\times$  u/d)

Formula-driven Prob\_Conversion (IF formulas referencing CB and CV)

Formula-driven Holder\_Action (IF formulas)

CB root price: \$114.988159 (Python = Excel)

### PROMPT 3

The conversion value = ex-div share price x conversion ratio, not cum-div. Change.

Claude 3

Let me fix this- conversion value should use ex-div price, not cum-div.

New CB root = \$114.591889.

### PROMPT 4

In the convertible price formula, the next period coupon is not discounted when calculating the continuing (hold) value, but the coupon for the current period is added in full.

Claude 4

Let me check the current continuing value formula carefully. Found it. Line 112:  $\text{cont\_val} = \text{disc} * \text{exp\_val} + (0.0 \text{ if } t == 0 \text{ else } \text{COUPON} * \text{disc})$  — coupon is being discounted ( $\text{COUPON} * \text{disc}$ ). The spec says the coupon for the current period is added in full (not discounted).

CB root is now \$115.000000. The coupon is now added in full (not discounted) to the continuing value at each non-root node.

### Claude Excel model (Run 2)

	A	B
3	<b>A. BOND TERMS</b>	
4	Face Value (\$)	100.000
5	Redemption Amount (\$)	129.318
6	Maturity (years)	6
7	Coupon Rate (% p.a.)	2.000%
8	Coupon Payments Per Year	1
9	Time Steps in Model	6
10	Yield (continuous % p.a.)	6.000%
11	Callable at Time Steps	4, 5 and 6
12	Call Premium at t=4 (% of face)	18.351814%
13	Call Premium at t=5 (% of face)	23.670281%
14	Call Premium at t=6 (% of face)	29.317624%
15		
16	<b>B. CONVERSION &amp; EQUITY PARAMETERS</b>	
17	Share Price at Valuation Date (\$)	14.3745776351
18	Volatility (% p.a.)	40.000%
19	Risk-Free Rate (continuous % p.a.)	5.000%
20	Credit-Adjusted Rate (cont % p.a.)	6.000%
21	Conversion Ratio (shares)	5.00
22	Dividend Yield (% per step, discrete)	1.000%
23		

	A	B
23		
24	<b>C. DERIVED PARAMETERS (auto-calculated)</b>	
25	Time Step Length dt (years)	1.000000
26	Annual Coupon (\$)	2.000
27	Up Factor $u = \exp(\sigma\sqrt{dt})$	1.491825
28	Down Factor $d = 1/u$	0.670320
29	Risk-Neutral Prob $p = (\exp(r_f dt) - d) / (u - d)$	0.463724
30	Risk-Neutral Prob $q = 1 - p$	0.536276
31	Risk-Free Rate (discrete per step)	5.127110%
32	Credit-Adj. Rate (discrete per step)	6.183655%
33	Yield (discrete per step)	6.183655%
34	Call Price t=4 (\$ excl. coupon)	118.351814
35	Call Price t=5 (\$ excl. coupon)	123.670281
36	Call Price t=6 (\$ excl. coupon)	129.317624
37		

Conversion Value Lattice = Conversion Ratio × Ex-Div Share Price (\$)							
Node	t=0	t=1	t=2	t=3	t=4	t=5	t=6
j=6							745.904146
j=5						505.044951	335.156337
j=4					341.961369	226.931325	150.595450
j=3				231.538950	153.653148	101.966817	67.666897
j=2			156.772929	104.037156	69.040810	45.816644	30.404697
j=1		106.149532	70.442618	46.746908	31.022035	20.586745	13.661711
j=0	71.872888	47.696059	31.651908	21.004740	13.939099	9.250221	6.138602

Assumptions | Share\_Price | **Conversion\_Value** | Blended\_Rate | Prob\_Conversion

Convertible Bond Price Lattice (\$)							
Node	t=0	t=1	t=2	t=3	t=4	t=5	t=6
j=6							745.904146
j=5						505.044951	335.156337
j=4					341.961369	226.931325	150.595450
j=3				231.538950	153.653148	125.670281	131.317624
j=2			170.207683	130.479194	120.351814	125.670281	131.317624
j=1		136.262731	117.483333	115.343070	120.351813	125.670281	131.317624
j=0	<b>115.000000</b>	109.284823	110.626012	115.343070	120.351813	125.670281	131.317624

Conversion\_Value | Blended\_Rate | Prob\_Conversion | **Convertible\_Price** | Holder\_Actio

Probability of Conversion Lattice (formula-driven)							
Node	t=0	t=1	t=2	t=3	t=4	t=5	t=6
j=6							100.0000%
j=5						100.0000%	100.0000%
j=4					100.0000%	100.0000%	100.0000%
j=3				100.0000%	100.0000%	0.0000%	0.0000%
j=2			71.2408%	46.3724%	0.0000%	0.0000%	0.0000%
j=1		44.5681%	21.5040%	0.0000%	0.0000%	0.0000%	0.0000%
j=0	26.0150%	9.9719%	0.0000%	0.0000%	0.0000%	0.0000%	0.0000%

Conversion\_Value | Blended\_Rate | **Prob\_Conversion** | Convertible\_Price | Holder\_Actio

Blended Discount Rate — Goldman Sachs (1994) = p_conv_next·r_f + (1-p_conv_next)·r_cr							
Node	t=0	t=1	t=2	t=3	t=4	t=5	t=6
j=6							n/a
j=5						5.1271%	n/a
j=4					5.1271%	5.1271%	n/a
j=3				5.1271%	5.6937%	5.6937%	n/a
j=2			5.4310%	5.6937%	6.1837%	6.1837%	n/a
j=1		5.7128%	5.9565%	6.1837%	6.1837%	6.1837%	n/a
j=0	5.9088%	6.0783%	6.1837%	6.1837%	6.1837%	6.1837%	n/a

Conversion\_Value | **Blended\_Rate** | Prob\_Conversion | Convertible\_Price | Holder\_Actio

	A	B	C	D	E	F	G	H	I	J
2	<b>DEBT COMPONENT = <math>CB \times (1 - Prob\_Conv)</math></b>									
3	<b>Node</b>	<b>t = 0</b>	<b>t = 1</b>	<b>t = 2</b>	<b>t = 3</b>	<b>t = 4</b>	<b>t = 5</b>	<b>t = 6</b>		
4	j = 6							0.000000		
5	j = 5						0.000000	0.000000		
6	j = 4					0.000000	0.000000	0.000000		
7	j = 3				0.000000	0.000000	125.670281	131.317624		
8	j = 2			48.950442	69.972919	120.351814	125.670281	131.317624		
9	j = 1		75.533048	92.219773	115.343070	120.351813	125.670281	131.317624		
10	j = 0	85.082799	98.387061	110.626012	115.343070	120.351813	125.670281	131.317624		

Conversion\_Value    Blended\_Rate    Prob\_Conversion    Convertible\_Price    Holder\_Action    **Debt\_Equity**

	A	B	C	D	E	F	G	H	I	J
14	<b>EQUITY COMPONENT = <math>CB \times Prob\_Conv</math></b>									
15	<b>Node</b>	<b>t = 0</b>	<b>t = 1</b>	<b>t = 2</b>	<b>t = 3</b>	<b>t = 4</b>	<b>t = 5</b>	<b>t = 6</b>		
16	j = 6							745.9041		
17	j = 5						505.0450	335.1563		
18	j = 4					341.9614	226.9313	150.5954		
19	j = 3				231.5389	153.6531	0.0000	0.0000		
20	j = 2			121.2572	60.5063	0.0000	0.0000	0.0000		
21	j = 1		60.7297	25.2636	0.0000	0.0000	0.0000	0.0000		
22	j = 0	29.9172	10.8978	0.0000	0.0000	0.0000	0.0000	0.0000		

Conversion\_Value    Blended\_Rate    Prob\_Conversion    Convertible\_Price    Holder\_Action    **Debt\_Equity**

	A	B	C	D	E	F	G	H	I	
1	<b>Holder / Issuer Action Lattice (formula-driven)</b>									
2	<b>Node</b>	<b>t = 0</b>	<b>t = 1</b>	<b>t = 2</b>	<b>t = 3</b>	<b>t = 4</b>	<b>t = 5</b>	<b>t = 6</b>		
3	j = 6							Convert		
4	j = 5						Convert (on call)	Convert		
5	j = 4					Convert (on call)	Convert (on call)	Convert		
6	j = 3				Convert	Convert (on call)	Call	Call		
7	j = 2			Hold	Hold	Call	Call	Call		
8	j = 1		Hold	Hold	Hold	Hold	Call	Call		
9	j = 0	Hold	Hold	Hold	Hold	Hold	Call	Call		

Conversion\_Value    Blended\_Rate    Prob\_Conversion    Convertible\_Price    **Holder\_Action**

The formulae for the top rows (up moves) at time 5 and 6 are as follows:

	5	6	
Convertible Price	505.044951		=IF(Conversion_Value!G4>=(1/(1+Blended_Rate!G4))*(Assumptions!\$B\$29*H3+Assumptions!\$B\$30*H4)+Assumptions!\$B\$26,Conversion_Value!G4,IF(Assumptions!\$B\$35+Assumptions!\$B\$26<(1/(1+Blended_Rate!G4))*(Assumptions!\$B\$29*H3+Assumptions!\$B\$30*H4)+Assumptions!\$B\$26,IF(Conversion_Value!G4>=Assumptions!\$B\$35+Assumptions!\$B\$26,Conversion_Value!G4,Assumptions!\$B\$35+Assumptions!\$B\$26),(1/(1+Blended_Rate!G4))*(Assumptions!\$B\$29*H3+Assumptions!\$B\$30*H4)+Assumptions!\$B\$26))
		745.904146	=IF(Conversion_Value!H3>=Assumptions!\$B\$36+Assumptions!\$B\$26,Conversion_Value!H3,IF(Assumptions!\$B\$36+Assumptions!\$B\$26<Assumptions!\$B\$5+Assumptions!\$B\$26,IF(Conversion_Value!H3>=Assumptions!\$B\$36+Assumptions!\$B\$26,Conversion_Value!H3,Assumptions!\$B\$36+Assumptions!\$B\$26),Assumptions!\$B\$5+Assumptions!\$B\$26))
Probability of Conversion	100.0000%		=IF(ROUND(Convertible_Price!G4,6)=ROUND(Conversion_Value!G4,6),1,IF(ROUND(Convertible_Price!G4,6)=ROUND(Assumptions!\$B\$35+Assumptions!\$B\$26,6),0,(Assumptions!\$B\$29*H3+Assumptions!\$B\$30*H4)))
		100.0000%	=IF(ROUND(Convertible_Price!H3,6)=ROUND(Conversion_Value!H3,6),1,0)
Blended Discount Rate	5.1271%		=(Assumptions!\$B\$29*Prob_Conversion!H3+Assumptions!\$B\$30*Prob_Conversion!H4)*Assumptions!\$B\$31+(1-(Assumptions!\$B\$29*Prob_Conversion!H3+Assumptions!\$B\$30*Prob_Conversion!H4))*Assumptions!\$B\$32
		-	
Debt Component	0.000000		=Convertible_Price!G4*(1-Prob_Conversion!G4)
		0.000000	=Convertible_Price!H3*(1-Prob_Conversion!H3)
Equity Component	505.044951		=Convertible_Price!G4*Prob_Conversion!G4
		745.904146	=Convertible_Price!H3*Prob_Conversion!H3
Holder Action	Convert		=IF(ROUND(Convertible_Price!G4,6)=ROUND(Conversion_Value!G4,6),IF(Assumptions!\$B\$35+Assumptions!\$B\$26<(1/(1+Blended_Rate!G4))*(Assumptions!\$B\$29*Convertible_Price!H3+Assumptions!\$B\$30*Convertible_Price!H4)+Assumptions!\$B\$26,"Convert (on call)","Convert"),IF(ROUND(Convertible_Price!G4,6)=ROUND(Assumptions!\$B\$35+Assumptions!\$B\$26,6),"Call","Hold"))
		Convert	=IF(ROUND(Convertible_Price!H3,6)=ROUND(Conversion_Value!H3,6),"Convert",IF(ROUND(Convertible_Price!H3,6)=ROUND(Assumptions!\$B\$36+Assumptions!\$B\$26,6),"Call","Redeem"))

PROMPT 1

“Write the python code for the attached file [note: the user’s model from Appendix 1] and produce an Excel spreadsheet that produces the same 115.00 price.

The python code should be based on the uploaded file and should allow a separate standalone excel model to be produced with embedded formulas and lattice structure (time periods on top row, wedge layout).

This should be standalone code without reference to the file and should allow a new excel stand alone file to be produced in whatever layout you prefer so long as it has embedded formulae and a lattice structure. It does not have to replicate the original. If I delete the original, the code should still produce a model that has a price of 115.00

The code must be transparent, stand alone, with no reference to the original file, so no 64 bit encoding.”

Codex 1

I’m going to inspect the attached workbook first so I can extract the model inputs and lattice logic, then I’ll generate standalone Python code that builds a new Excel file (with formulas) reproducing the 115.00 price. Next I’ll map the workbook structure (sheets, ranges, key formula cells including the 115.00 output) so the standalone generator mirrors the mechanics, not the file itself.

I’m pulling only the core input block (rates, up/down factors, probabilities, redemption/coupon settings) so we can build a clean standalone lattice model instead of recreating the full workbook complexity. I now have enough inputs from your file (vol, rates, conversion ratio, coupon, redemption) and I’m testing a clean 6-step binomial backward-induction spec to lock the root value to 115.00 before writing the generator script. I identified the key missing piece: the source model uses probability-of-conversion blending (not delta) for discounting. I’m implementing that recursion now because it should bring the standalone lattice to the exact 115.00 level. I found the exact driver for the 115.00 match: the first node has zero coupon accrual (like your source), while later nodes include coupon.

Implemented and generated both deliverables. Root price formula is in Inputs!B29 and is designed to evaluate to 115.00.

[User: “Wow”]

Codex Excel model (Run 1)

	A	B
1	<b>Standalone Convertible Lattice Inputs</b>	
2	Spot price S0	14.37457764
3	Equity volatility (sigma)	0.4
4	Dividend yield per step	0.01
5	Risk-free cc rate	0.05
6	Credit spread	0.01
7	Risky cc rate	0.06
8	Step size dt (years)	1
9	Conversion ratio	5
10	Coupon per period	2
11	Redemption amount (ex-final)	129.317624
13	Number of steps	6
14	Callable from step	4
15		
16	$u = \exp(\text{sigma} * \sqrt{\text{dt}})$	1.491824698
17	$d = 1/u$	0.670320046
18	u ex-div	1.476906451
19	d ex-div	0.663616846
20	Risk-free step rate	0.051271096
21	Risky step rate	0.061836547
22	Risk-neutral probability p	0.463723546
23		

	A	B	C	D	E	F	G	H	I
25	<b>Straight bond floor schedule B(t)</b>								
26			0	1	2	3	4	5	6
27	B(t)		100	106.1836547	110.6260121	115.3430695	120.3518135	125.6702809	131.317624

Inputs		Lattice								
	A	B	C	D	E	F	G	H	I	
1	<b>Convertible Bond Lattice (Formula Driven)</b>									
2	Section	No	0	1	2	3	4	5	6	
3	<b>Stock ex-div S(t,j)</b>									
4		0	14.374578	21.229906	31.354586	46.307790	68.392274	101.008990	149.180829	
5		1		9.539212	14.088524	20.807431	30.730630	45.386265	67.031267	
6		2			6.330382	9.349382	13.808162	20.393363	30.119090	
7		3				4.200948	6.204407	9.163329	13.533379	
8		4					2.787820	4.117349	6.080939	
9		5						1.850044	2.732342	
10		6							1.227720	
11	<b>Conversion value CV(t,j)</b>									
12		0	71.872888	106.149532	156.772929	231.538950	341.961369	505.044951	745.904146	
13		1		47.696059	70.442618	104.037156	153.653148	226.931325	335.156337	
14		2			31.651908	46.746908	69.040810	101.966817	150.595450	
15		3				21.004740	31.022035	45.816644	67.666897	
16		4					13.939099	20.586745	30.404697	
17		5						9.250221	13.661711	
18		6							6.138602	
19	<b>Prob. of conversion PC(t,j)</b>									
20		0	26.0150%	44.5681%	71.2408%	100.0000%	100.0000%	100.0000%	100.0000%	
21		1		9.9719%	21.5040%	46.3724%	100.0000%	100.0000%	100.0000%	
22		2			0.0000%	0.0000%	0.0000%	0.0000%	100.0000%	
23		3				0.0000%	0.0000%	0.0000%	0.0000%	
24		4					0.0000%	0.0000%	0.0000%	
25		5						0.0000%	0.0000%	
26		6							0.0000%	
27	<b>Convertible value V(t,j)</b>									
28		0	115.000000	136.262731	170.207683	231.538950	341.961369	505.044951	745.904146	
29		1		109.284823	117.483333	130.479194	153.653148	226.931325	335.156337	
30		2			110.626012	115.343070	120.351814	125.670281	150.595450	
31		3				115.343070	120.351814	125.670281	131.317624	
32		4					120.351814	125.670281	131.317624	
33		5						125.670281	131.317624	
34		6							131.317624	

The ex-div share price is calculated this time using ex-div up and down factors:  $u = 1.491825 \times (100\% - 1\% \text{ discrete dividend yield}) = 1.476906$ ,  $d = (1 / 1.491825) \times (100\% - 1\%) = 0.663617$

The formulae for the top rows (up moves) at time 5 and 6 are as follows:

	5	6	
Convertible Price	505.044951		=IF(H\$2>=Inputs!\$B\$14,Inputs!\$H\$27,0)>0,MAX(Inputs!\$H\$27,H12,MIN((Inputs!\$B\$22*I28+(1-Inputs!\$B\$22)*I29)/(1+((Inputs!\$B\$22*I20+(1-Inputs!\$B\$22)*I21))*Inputs!\$B\$20+(1-(Inputs!\$B\$22*I20+(1-Inputs!\$B\$22)*I21))*Inputs!\$B\$21))+IF(H\$2=0,Inputs!\$B\$12,Inputs!\$B\$10),IF(H\$2>=Inputs!\$B\$14,Inputs!\$H\$27,0)),MAX(Inputs!\$H\$27,H12,(Inputs!\$B\$22*I28+(1-Inputs!\$B\$22)*I29)/(1+((Inputs!\$B\$22*I20+(1-Inputs!\$B\$22)*I21))*Inputs!\$B\$20+(1-(Inputs!\$B\$22*I20+(1-Inputs!\$B\$22)*I21))*Inputs!\$B\$21))+IF(H\$2=0,Inputs!\$B\$12,Inputs!\$B\$10)))
		745.904146	=MAX(I12,Inputs!\$I\$27)
Probability of Conversion	100.0000%		=IF(ABS(H28-H12)<0.00000001,1,IF(ABS(H28-Inputs!\$H\$27)<0.00000001,0,(Inputs!\$B\$22*I20+(1-Inputs!\$B\$22)*I21)))
		100.0000%	=IF(I12>Inputs!\$I\$27,1,0)

The blended discount rate and holder action options are not shown separately but embedded in the calculations.

Run 1 (matched after 1 prompt split in two, 22 mins ): 581 lines of Python Code

PROMPT 1

Write the python code for the attached file that produces the same 115.00 price. The python code should do what the uploaded model does. It should be standalone code without reference to the uploaded excel file (written from scratch, no reference to the original file, so no 64 bit encoding).

Note: the single prompt approach – see Codex – meant Claude working for 15 mins, at which point it appeared to shut down (“Claude’s response could not be fully generated”). The prompt was split in two, which meant working for 11 mins for both prompt, without stalling.

Claude 1

Let me read the Excel file to understand its structure and calculations. Now I have a thorough understanding of the model. This is a Convertible Bond (CB) pricing model using a binomial tree with the Goldman Sachs (1994) method. Let me now read the formulas to understand the exact computation logic.

Now I have enough information to write the Python code. Let me build it based on my understanding of the model:

- It’s a CRR binomial tree for a callable convertible bond
- Goldman Sachs (1994) / Tsiveriotis-Fernandes (1998) blended discount rate approach
- The model uses M16=1 (blended rate using delta/probability of conversion)
- The M17=0 means use probability of conversion (not delta) for blending
- 6 periods, annual steps

Close but not 115.00. The blended rate approach needs refinement. Let me look more carefully at how the Excel model computes probabilities and blended rates at period 0. ... Period 0 does NOT add a coupon (the valuation date is the last coupon date), but periods 1-6 do. The coupon is added to the discounted continuation value.

Now let me also verify the terminal period calculation. At the terminal period, the coupon (2.0) is already included in the redemption\_amount flow. The redemption\_amount is 129.3176... and the cb\_price shows 131.3176... which means terminal cb = redemption\_amount + coupon = 131.3176. Now let me fix the code properly. Still off. Let me trace through the tree more carefully to match the Excel node by node. The logic is working correctly for intermediate nodes.

My issue must be in how I’m building the tree — the share price construction might differ. ... The tree is built from ex-div prices — each node’s cum-div = prior node’s ex-div × u (or d). ... This means each period’s ex-div price =  $S_0 * u^{up} * d^{dn} * (1 - div\_yield)^{period}$ . ... Now I have the correct share price formula.

The model produces exactly 115.00. ... Here’s a summary of the key modelling choices that were critical to match the Excel:

- Share price tree construction — the most important fix. Each node's cum-div price is the parent's ex-div price  $\times$  u or d (not the cum-div price). This means ex-div prices follow:  $S_{\text{ex}}(\text{period}, j) = S_0 \times u^{(\text{up})} \times d^{(\text{down})} \times (1 - \text{div\_yield})^{\text{period}}$ , where the dividend yield compounds with each passing period.
- No coupon at period 0 — the valuation date coincides with the last coupon payment date, so no coupon is added in the period 0 rollback.
- Goldman Sachs (1994) blended rate — discount rate =  $P_{\text{conv}} \times \text{rf\_eff} + (1 - P_{\text{conv}}) \times \text{risky\_eff}$ , where  $P_{\text{conv}}$  is the risk-neutral probability of eventual conversion propagated backward through the tree.
- Call logic — the issuer calls only when continuation  $>$  call\_price + coupon; the holder converts when conv\_value  $\geq$  call\_price + coupon. The call prices accrete each period at the risky rate net of coupon.

## PROMPT 2

The python code should allow a separate standalone excel spreadsheet to be produced with embedded formulas and lattice structure (time periods on top row, wedge layout). No hardcoded values must appear in the Excel lattice (embedded formulae only).

Claude 2

Let me first examine the Excel model structure carefully to understand the exact lattice layout before building anything. Now let me plan the layout carefully before writing any code. I need to think through the full structure. . . . Now recalculate formulas and check for errors: CB price at period 0 = 115.00 exactly. Let me verify some more nodes against the reference model:

INPUTS sheet — all 11 hardcoded assumptions (blue, editable) plus derived parameters computed entirely from formulas (black): rf/risky effective rates, u/d/p, coupon cash. Also contains the call schedule (accreting prices via recurrence  $CP_t = CP_{t-1} \times (1 + \text{risky\_eff}) - \text{coupon}$ ) and straight bond values for all 7 periods.

TREE sheet — full 6-period binomial lattice in wedge layout (periods 0→6 across columns, nodes stacking down). Each of the 28 nodes has 10 formula-driven rows:

Zero hardcoded values in the lattice — every cell formula references INPUTS. Changing any assumption (volatility, rates,  $S_0$ , callable period, etc.) causes the entire tree and CB price to recalculate automatically.

[User: "Bravo"]

# Claude Excel model (Run 1)

	A	B	C	D	E
1	<b>CALLABLE CONVERTIBLE BOND – PRICING MODEL</b>				
3	<b>BOND PARAMETERS</b>				
4	Face Value (FV)		100.00		Bond principal / par value
5	Coupon Rate (annual)		2.00%		Annual coupon rate
6	Coupon Frequency		1		1 = Annual   2 = Semi-annual
7	Maturity (years)		6		Bond term in years
8	Redemption Amount (RA)		129.3176		Principal repaid at maturity (accreting, excl. final coupon)
10	<b>RATE &amp; VOLATILITY PARAMETERS</b>				
11	Risk-Free Rate (rf, cont. comp.)		5.000%		Continuously compounded
12	Credit Risk Premium (crp, cont.)		1.000%		Continuously compounded
13	Equity Volatility ( $\sigma$ )		40.0%		Annualised
14	Discrete Dividend Yield ( $q$ )		1.0%		Per period (applied to ex-div price)
16	<b>EQUITY &amp; CONVERSION</b>				
17	Share Price at Valuation ( $S_0$ )		14.3746		Ex-div at valuation date
18	Conversion Ratio (CR)		5.00		Shares per bond = Face / Conversion Price
19	Callable From Period		4		First period the issuer can exercise the call option
21	<b>DERIVED PARAMETERS (formulas – do not edit)</b>				
22	Number of Steps (N = Maturity $\times$ Freq)		6		Binomial steps
23	Time Step $\Delta t = 1 / \text{Freq}$		1.0000		Years per step
24	Risky Rate (cont.) = rf + crp		6.000%		
25	rf effective per period $\exp(\text{rf} \cdot \Delta t) - 1$		5.12711%		
26	Risky eff. per period $\exp(\text{risky} \cdot \Delta t) - 1$		6.18365%		
27	Up factor $u = \exp(\sigma \cdot \sqrt{\Delta t})$		1.49182		
28	Down factor $d = 1/u$		0.67032		
29	Risk-neutral prob up p		0.46372		
30	Risk-neutral prob down $q = 1 - p$		0.53628		
31	Coupon Cash per Period		2.0000		
33	<b>CALL SCHEDULE (call price accretes each period at the risky rate, net of coupon)</b>				
34		Period	Callable?	Price (excl. coug)	Call Price + Coupon
35		1	No	104.1837	106.1837
36		2	No	108.6260	110.6260
37		3	No	113.3431	115.3431
38		4	Yes	118.3518	120.3518
39		5	Yes	123.6703	125.6703
40		6	Yes	129.3176	131.3176
42	<b>STRAIGHT BOND VALUE AT EACH PERIOD (PV of future cash flows discounted at risky rate)</b>				
43		Period	Periods Remaining	Straight Bond Value	
44		0	6	100.0000	
45		1	5	104.1837	
46		2	4	108.6260	
47		3	3	113.3431	
48		4	2	118.3518	
49		5	1	123.6703	
50		6	0	0.0000	

	A	B	C	D	E	F	G	H
1	<b>CALLABLE CONVERTIBLE BOND – BINOMIAL PRICING TREE</b>							
3	<b>Period →</b>	<b>0</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>
5	S (ex-div)	14.374578	21.229906	31.354586	46.307790	68.392274	101.008990	149.180829
6	Conversion Value	71.872888	106.149532	156.772929	231.538950	341.961369	505.044951	745.904146
7	Straight Bond	100.000000	104.183655	108.626012	113.343070	118.351814	123.670281	0.000000
8	P(Conversion)	26.0150%	44.5681%	71.2408%	100.0000%	100.0000%	100.0000%	100.0000%
9	Blended Rate	5.9088%	5.7128%	5.4310%	5.1271%	5.1271%	5.1271%	5.1271%
10	<b>CB Price ★</b>	<b>115.000000</b>	<b>136.262731</b>	<b>170.207683</b>	<b>231.538950</b>	<b>341.961369</b>	<b>505.044951</b>	<b>745.904146</b>
11	Continuation	115.000000	136.262731	170.207683	231.223560	340.541755	501.994502	
12	Call Price	-	104.183655	108.626012	113.343070	118.351814	123.670281	131.317624
13	Issuer Action	-	-	-	-	-	-	-
14	Holder Action	Hold	Hold	Hold	Convert	Convert	Convert	Convert
15								
16	S (ex-div)		9.539212	14.088524	20.807431	30.730630	45.386265	67.031267
17	Conversion Value		47.696059	70.442618	104.037156	153.653148	226.931325	335.156337
18	Straight Bond		104.183655	108.626012	113.343070	118.351814	123.670281	0.000000
19	P(Conversion)		9.9719%	21.5040%	46.3724%	100.0000%	100.0000%	100.0000%
20	Blended Rate		6.0783%	5.9565%	5.6937%	5.6937%	5.1271%	5.1271%
21	<b>CB Price ★</b>		<b>109.284823</b>	<b>117.483333</b>	<b>130.479194</b>	<b>153.653148</b>	<b>226.931325</b>	<b>335.156337</b>
22	Continuation		109.284823	117.483333	130.479194	165.327989	226.662011	
23	Call Price		104.183655	108.626012	113.343070	118.351814	123.670281	131.317624
24	Issuer Action		-	-	-	-	-	-
25	Holder Action		Hold	Hold	Hold	Convert	Convert	Convert
26								
27	S (ex-div)			6.330382	9.349382	13.808162	20.393363	30.119090
28	Conversion Value			31.651908	46.746908	69.040810	101.966817	150.595450
29	Straight Bond			108.626012	113.343070	118.351814	123.670281	0.000000
30	P(Conversion)			0.0000%	0.0000%	0.0000%	0.0000%	100.0000%
31	Blended Rate			6.1837%	6.1837%	6.1837%	5.6937%	5.1271%
32	<b>CB Price ★</b>			<b>110.626012</b>	<b>115.343070</b>	<b>120.351814</b>	<b>125.670281</b>	<b>150.595450</b>
33	Continuation			110.626012	115.343070	120.351814	134.701564	
34	Call Price			108.626012	113.343070	118.351814	123.670281	131.317624
35	Issuer Action			-	-	-	Call	-
36	Holder Action			Hold	Hold	Hold	Accept Call	Convert
37								
38	S (ex-div)				4.200948	6.204407	9.163329	13.533379
39	Conversion Value				21.004740	31.022035	45.816644	67.666897
40	Straight Bond				113.343070	118.351814	123.670281	0.000000
41	P(Conversion)				0.0000%	0.0000%	0.0000%	0.0000%
42	Blended Rate				6.1837%	6.1837%	6.1837%	6.1837%
43	<b>CB Price ★</b>				<b>115.343070</b>	<b>120.351814</b>	<b>125.670281</b>	<b>131.317624</b>
44	Continuation				115.343070	120.351814	125.670281	
45	Call Price				113.343070	118.351814	123.670281	131.317624
46	Issuer Action				-	-	-	-
47	Holder Action				Hold	Hold	Hold	Redeem
48								
49	S (ex-div)					2.787820	4.117349	6.080939
50	Conversion Value					13.939099	20.586745	30.404697
51	Straight Bond					118.351814	123.670281	0.000000
52	P(Conversion)					0.0000%	0.0000%	0.0000%
53	Blended Rate					6.1837%	6.1837%	6.1837%
54	<b>CB Price ★</b>					<b>120.351814</b>	<b>125.670281</b>	<b>131.317624</b>
55	Continuation					120.351814	125.670281	
56	Call Price					118.351814	123.670281	131.317624
57	Issuer Action					-	-	-
58	Holder Action					Hold	Hold	Redeem

	A	B	C	D	E	F	G	H
1	<b>CALLABLE CONVERTIBLE BOND – BINOMIAL PRICING TREE</b>							
3	Period →	0	1	2	3	4	5	6
59								
60	S (ex-div)						1.850044	2.732342
61	Conversion Value						9.250221	13.661711
62	Straight Bond						123.670281	0.000000
63	P(Conversion)						0.0000%	0.0000%
64	Blended Rate						6.1837%	6.1837%
65	<b>CB Price ★</b>						<b>125.670281</b>	<b>131.317624</b>
66	Continuation						125.670281	
67	Call Price						123.670281	131.317624
68	Issuer Action						-	-
69	Holder Action						Hold	Redeem
70								
71	S (ex-div)							1.227720
72	Conversion Value							6.138602
73	Straight Bond							0.000000
74	P(Conversion)							0.0000%
75	Blended Rate							6.1837%
76	<b>CB Price ★</b>							<b>131.317624</b>
77	Continuation							
78	Call Price							131.317624
79	Issuer Action							-
80	Holder Action							Redeem
81								
82								

	5	6	
Convertible Price	505.044951		=IF(INPUTS!C39="Yes",IF(G6>=(INPUTS!D39+INPUTS!C31),G6,IF(G11>(INPUTS!D39+INPUTS!C31),(INPUTS!D39+INPUTS!C31),MAX(G11,G6,G7))),MAX(G11,G6,G7))
		745.904146	=MAX(H6,(INPUTS!C8+INPUTS!C31))
Probability of Conversion	100.0000%		=IF(INPUTS!C39="Yes",IF(G6>=(INPUTS!D39+INPUTS!C31),1,IF(G11>(INPUTS!D39+INPUTS!C31),0,IF(G10=G6,1,IF(G10=G11,(INPUTS!C29*H8+INPUTS!C30*H19),0))),IF(G10=G6,1,IF(G10=G11,(INPUTS!C29*H8+INPUTS!C30*H19),0)))
		100.0000%	=IF(H6>=(INPUTS!C8+INPUTS!C31),1,0)
Blended Discount Rate	5.1271%		=(INPUTS!C29*H8+INPUTS!C30*H19)*INPUTS!C25+(1-(INPUTS!C29*H8+INPUTS!C30*H19))*INPUTS!C26
		5.1271%	=IF(H6>=(INPUTS!C8+INPUTS!C31),INPUTS!C25,INPUTS!C26)
Holder Action	Convert		=IF(INPUTS!C39="Yes",IF(G6>=(INPUTS!D39+INPUTS!C31),"Convert",IF(G11>(INPUTS!D39+INPUTS!C31),"Accept Call",IF(G10=G6,"Convert","Hold"))),IF(G10=G6,"Convert","Hold"))
		Convert	=IF(H6>=(INPUTS!C8+INPUTS!C31),"Convert","Redeem")