MATH3090 Assignment 1

March 18, 2024

1 MATH3090 Assignment 1

Student: Hugo Burton (s4698512) Date Due: Tuesday March 19 @ 1pm

[1]:
```
import math
from typing import Dict
from colorama import Fore, Style
import numpy as np
from IPython.display import Markdown, display

import bond
import interest
import newtons
import display as dsp
```

1.1 Question 1 (6 marks)

1.1.1 Part A (3 marks)

Suppose a company issues a zero coupon bond with face value $10,000 and which matures in 20 years. Calculate the price given:

[2]:
```
face_value = 10_000
years_to_maturity = 20
```

i. An 8% discount compound annual yield, compounded annually.

[3]:
```
# Question i
interest_rate = 0.08
compounding_frequency_yr = 1

price_i = bond.price_zero_coupon_bond_discrete(
    face_value, years_to_maturity, interest_rate, compounding_frequency_yr)

dsp.display_answer(price_i)
```

```
Answer: $2145.48
```

ii. An 8% discount continuous annual yield, compounded semi-annually.

```
[4]: # Question ii
interest_rate = 0.08

price_ii = bond.price_zero_coupon_bond_continuous(
    face_value, years_to_maturity, interest_rate)

dsp.display_answer(price_ii)
```

```
Answer: $2018.97
```

iii. A nonconstant yield of $y(t) = 0.06 + 0.2te^{-t^2}$.

```
[5]: # Question iii
q = "iii"
def yield_function(t): return 0.06 + 0.2 * t * math.exp(-t**2)

price_iii = bond.price_zero_coupon_bond_nonconstant_yield(
    face_value, years_to_maturity, yield_function)

dsp.display_answer(price_iii)
```

```
Answer: $2725.32
```

1.1.2 Part B (3 marks)

A 10 year \$10,000 government bond has a coupon rate of 5% payable quarterly and yields 7%. Calculate the price.

```
[6]: # Part b (3 marks)
face_value = 10_000
years_to_maturity = 10
coupon_rate = 0.05
interest_rate = 0.07
compounding_frequency_yr = 4

price_b = bond.price_coupon_bearing_bond_discrete(
    face_value, years_to_maturity, coupon_rate, interest_rate,␣
 ↪compounding_frequency_yr)

dsp.display_answer(price_b)
```

```
Answer: $8570.29
```

1.2 Question 2 (6 marks)

Consider the cash flow

$$C_0 = -3x, \quad C_1 = 5, \quad C_2 = x$$

(at periods 0, 1, 2 respectively) for some $x > 0$.

1.3 Part A (3 marks)

Apply the discount process $d(k) = (1+r)^{-k}$ so that the present value is

$$P = \sum_{k=0}^{2} d(k) C_k$$

What is the range of x such that $P > 0$ when $r = 5\%$?

$$\begin{align}
0 &< P \tag{1}\\
0 &< \sum_{i=0}^{2} d(k) C_k \tag{2}\\
0 &< \left[(1+0.05)^{-0} \cdot -3x\right] + \left[(1+0.05)^{-1} \cdot 5\right] + \left[(1+0.05)^{-2} \cdot x\right] \tag{3}\\
0 &< [1 \cdot -3x] + \left[\frac{5}{1+0.05}\right] + \left[\frac{x}{(1+0.05)^2}\right] \tag{4}\\
0 &< -3x + \frac{5}{1.05} + \frac{x}{1.05^2} \tag{5}\\
0 &< -3 \cdot 1.05^2 x + 5 \cdot 1.05 + x \tag{6}\\
0 &< -2.3075x + 5.25 \tag{7}\\
2.3075x &< 5.25 \tag{8}\\
x &< \frac{5.25}{2.3075} \tag{9}\\
x &\lessapprox 2.275 \tag{10}\\
& \tag{11}
\end{align}$$

Therefore, when $r = 5\%$, $P > 0$ holds when $x \lessapprox 2.275$.

We can verify this in code numerically as follows.

```
[7]: r = 0.05
cash_flow = lambda k, x: -3*x if k == 0 else (5 if k == 1 else x)

accept_condition = lambda p: p > 0

# Function to calculate present value based on some value of x
def present_value(x: float) -> float:
    present_value = sum(cash_flow(k, x) * interest.
 ↪discrete_compound_interest_discounted(r, k, 1) \
                        for k in range(2+1))

    return present_value

cash_flows_x: Dict[float, float] = {}
x_min = -100.0
```

```
x_max = 100.0
x_step = 0.01
accept_min_x = 0
accept_max_x = None

# Loop over a wide range of x
for x in np.arange(x_min, x_max, x_step):
    cash_flow_x = present_value(x)

    cash_flows_x[x] = cash_flow_x

    if accept_condition(cash_flow_x):
        # Min
        if not accept_min_x:
            accept_min_x = x
        elif x < accept_min_x:
            accept_min_x = x

        # Max
        if not accept_max_x:
            accept_max_x = x
        elif x > accept_max_x:
            accept_max_x = x

if accept_min_x == x_min:
    accept_min_x = 0

if accept_max_x == x_max:
    accept_max_x = math.inf

print(
    f"Range of x such that P > 0 when r = {r * 100}%: {accept_min_x} < x <␣
 ↪{accept_max_x}")
```

```
Range of x such that P > 0 when r = 5.0%: 0 < x < 2.2700000000523204
```

1.4 Part B (3 marks)

The IRR (internal rate of return) is r such that $P = 0$. For what range of x will there be a unique, strictly positive IRR?

$$\begin{aligned}
P &= \sum_{k=0}^{2} d(k) \cdot C_k && (12)\\
0 &= [d(0) \cdot -3x] + [d(1) \cdot 5] + [d(2) \cdot x] && (13)\\
&= -3x(1+r)^{-0} + 5(1+r)^{-1} + x(1+r)^{-2} && (14)\\
&= -3x \cdot 1 + \frac{5}{1+r} + \frac{x}{(1+r)^2} && (15)\\
&= -3x(1+r)^2 + 5(1+r) + x && (16)\\
&= -3x(1^2 + 2r + r^2) + 5 + 5r + x && (17)\\
&= -3x - 6xr - 3xr^2 + 5 + 5r + x && (18)\\
&= -3xr^2 + (5-6x)r + (x - 3x + 5) && (19)\\
&= -3xr^2 + (5-6x)r + (5-2x) && (20)
\end{aligned}$$

As we want $r > 0$, then solve for $\Delta > 0$

$$\begin{aligned}
0 &< \Delta && (21)\\
0 &< (5-6x)^2 - 4 \cdot (-3x) \cdot (5-2x) && (22)\\
0 &< (25 - 60x + 36x^2) + 12x \cdot (5-2x) && (23)\\
0 &< 36x^2 - 60x + 25 + 60x - 24x^2 && (24)\\
0 &< 12x^2 + 25 && (25)
\end{aligned}$$

This will always hold $\forall x \in \mathbb{R}^+$

Therefore, if $x > 0$, then IRR $= \{r | P = 0\} > 0$.

1.5 Question 3 (8 marks)

Cashflows (C_i)	Times (t_i)
2.3	1.0
2.9	2.0
3.0	3.0
3.2	4.0
4.0	5.0
3.8	6.0
4.2	7.0
4.8	8.0
5.5	9.0
105	10.0

Table 1: Bond Cashflows

In this question, consider a bond with a set of cashflows given in Table 1. Here, note that the face value F is already included in the last cashflow. Let y be the yield to maturity, t_i be the time of the i^{th} cashflow C_i, and $PV = 100$ be the market price of the bond at $t = 0$. Assume continuous compounding. Then y solves

$$PV = \sum_i C_i e^{-yt_i}$$

1.6 Part A (3 marks)

Write out the Newton iteration to compute y_{n+1} from y_n (see L2.49). Specifically, clearly indicate the functions $f(y)$ and $f'(y)$.

In this question

$$f(y) = \left[\sum_{t=1}^{10} C_t \cdot \beta(y,t)\right] - P = \left[\sum_{t=1}^{10} C_t \exp\{-y \cdot t\}\right] - P$$

and

$$f'(y) = \sum_{t=1}^{10} C_t \cdot \beta'(y,t) = \sum_{t=1}^{10} C_t \cdot \frac{d}{dx}\exp\{-y \cdot t\} = -\sum_{t=1}^{10} tC_t \exp\{-y \cdot t\}$$

Using these definitions, perform the following

1. Choose an intial value of x_0, say $x_0 = 0.05$
2. Compute the following until the termination condition

$$x_{n+1} \approx x_n - \frac{f(x_n)}{f'(x_n)} \tag{26}$$

$$\approx x_n + \frac{\left[\sum_{t=1}^{10} C_t \exp\{-x_n \cdot t\}\right] - P}{-\sum_{t=1}^{10} tC_t \exp\{-x_n \cdot t\}} \tag{27}$$

3. Terminate when $|x_{n+1} - x_n| < \epsilon$ or $|f(x_{n+1})| < \epsilon$

1.7 Part B (5 marks)

Implement the above Newton iteration in Matlab (I'm using Python) using the stopping criteria

$$|y_{n+1} - y_n| < 10^{-8}.$$

Fill in Table 2 for $y_0 = 0.05$ (add rows as necessary).

In addition, try with larger values for y_0 and observe the accuracy and convergence speed. How does the performance change?

```
[8]: # Part b (5 marks)
cashflows = [2.3, 2.9, 3.0, 3.2, 4.0, 3.8, 4.2, 4.8, 5.5, 105]

# Market price of the bond at t = 0
PV = 100

def f(y): return sum(
    cashflows[t-1] * interest.continuous_compound_interest_discounted(y, t) for␣
 ↪t in range(1, len(cashflows)+1)) - PV

def f_prime(y): return - \
    sum(((t * cashflows[t]) / ((1 + y)**(t + 1))
        for t in range(len(cashflows)))


eps = 1e-8

# Set initial y value
x_0 = 0.05

# Print table header
col_heads = ["$$n$$", "$$y_n$$" , "$$|y_{n}-y_{n-1}|$$"]
col_spaces = [3, 11, 17]

md_table = ""
header_row = ""
format_row = ""

for i, col_head in enumerate(col_heads):
    space = col_spaces[i]
    part = f"|{col_head:^{space}}"
    header_row += part
    middle = '-'*(max(1, len(part) - 2 - 2))
    format_row += f"| :{middle}: "

header_row += "|"
format_row += "|"

# Solve y using Newton's method given f and PV as inputs

approx, table_rows, _ = newtons.newtons_method(f, f_prime, x_0, eps, 9999999,␣
 ↪generate_table=True, log=False,
                                               col_spaces=col_spaces,␣
 ↪precision=10)

md_table += header_row + "  \n"
md_table += format_row + "  \n"
md_table += "  \n".join(table_rows)

def printmd(string):
    display(Markdown(string))

printmd(md_table)
```

n	y_n	$\|y_n - y_{n-1}\|$
0	0.0345487861	0.0154512139
1	0.0372734985	0.0027247125
2	0.0369704389	0.0003030597
3	0.0370078067	3.73678e-05
4	0.0370032489	4.5578e-06
5	0.0370038056	5.567e-07
6	0.0370037376	6.8e-08
7	0.0370037459	8.3e-09

1.7.1 Part ii: Larger values of y_0

[11]:
```
increment = 0.01
y_0_vals = [x for x in np.arange(0.05, 0.25+increment, increment)]

for y_0 in y_0_vals:
    print(
        f"{Fore.CYAN}y_0 = {Fore.LIGHTRED_EX}{round(y_0, 2)}{Style.RESET_ALL}",␣
 ↪end=": ")
    approx, _, num_iterations = newtons.newtons_method(
        f, f_prime, y_0, eps, 9999999, generate_table=False, log=False)

    ans = f" {Fore.LIGHTGREEN_EX}{approx:.10f}{Style.RESET_ALL} in {Fore.
 ↪LIGHTMAGENTA_EX}{num_iterations} iterations{Style.RESET_ALL}."

    print(ans)
```

```
y_0 = 0.05:  0.0370037459 in 7 iterations.
y_0 = 0.06:  0.0370037448 in 8 iterations.
y_0 = 0.07:  0.0370037447 in 8 iterations.
y_0 = 0.08:  0.0370037447 in 8 iterations.
y_0 = 0.09:  0.0370037448 in 8 iterations.
y_0 = 0.1:  0.0370037452 in 8 iterations.
y_0 = 0.11:  0.0370037459 in 8 iterations.
y_0 = 0.12:  0.0370037448 in 9 iterations.
y_0 = 0.13:  0.0370037447 in 9 iterations.
y_0 = 0.14:  0.0370037447 in 9 iterations.
y_0 = 0.15:  0.0370037440 in 6 iterations.
y_0 = 0.16:  0.0370037457 in 9 iterations.
```

```
y_0 = 0.17:  0.0370037447 in 10 iterations.
y_0 = 0.18:  0.0370037447 in 10 iterations.
y_0 = 0.19:  0.0370037451 in 9 iterations.
y_0 = 0.2:  0.0370037448 in 11 iterations.
y_0 = 0.21:  0.0370037447 in 11 iterations.
y_0 = 0.22:  0.0370037453 in 11 iterations.
y_0 = 0.23:  0.0370037447 in 12 iterations.
y_0 = 0.24:  0.0370037458 in 12 iterations.
y_0 = 0.25:  0.0370037444 in 12 iterations.
```

As seen above, as the initial estimate, y_0 is increased beyond 0.05 (up to 0.25), more iterations of Newton's method are required in order to achieve the same level of accuracy. In other words, for the same iteration number, a solution starting with a higher y_0 has a lower accuracy.

The difference in the time taken to solve is negligable here, however, so is not reported.

1.8 Question 4

In the Constant Growth DDM model, the present value of the share is

$$PV = \sum_{t=1}^{\infty} \frac{D_t}{(1+k)^t}$$

where $D_1, D_2, ...$ are (non-random) dividends and $k > 0$ is the required rate of return

Suppose $D_0 > 0$, $k > 0$ and $g > 0$

Derive the formula for the present value (2) when

$$D_t = D_0(1+g)^{\lceil \frac{t}{2} \rceil}, \quad t = 1, 2, ...$$

where $\lceil x \rceil$ is the smallest integer greater than or equal to x. What is the condition of g so that PV is finite? To get full marks, you will need to write an explicit expression (without summation).

First substitute for D_t as defined in the question and expand to see the pattern

$$\begin{aligned}
PV &= \sum_{t=1}^{\infty} \frac{D_t}{(1+k)^t} && (28)\\
&= \sum_{t=1}^{\infty} \frac{D_0(1+g)^{\lceil \frac{t}{2} \rceil}}{(1+k)^t} && (29)\\
&= \left[D_0 \cdot \frac{(1+g)^{\lceil \frac{1}{2} \rceil}}{(1+k)^1}\right] + \left[D_0 \cdot \frac{(1+g)^{\lceil \frac{2}{2} \rceil}}{(1+k)^2}\right] + \left[D_0 \cdot \frac{(1+g)^{\lceil \frac{3}{2} \rceil}}{(1+k)^3}\right] + \left[D_0 \cdot \frac{(1+g)^{\lceil \frac{4}{2} \rceil}}{(1+k)^4}\right] + \cdots && (30)\\
&= \left[D_0 \cdot \frac{(1+g)^1}{(1+k)^1}\right] + \left[D_0 \cdot \frac{(1+g)^1}{(1+k)^2}\right] + \left[D_0 \cdot \frac{(1+g)^2}{(1+k)^3}\right] + \left[D_0 \cdot \frac{(1+g)^2}{(1+k)^4}\right] + \cdots && (31)
\end{aligned}$$

Now consider, splitting up the geometric series into sub series where

- the exponent on the numerator is half the exponent on the denominator; and
- the above is not the case

We then have

$$PV = \left\{\left[D_0 \cdot \frac{(1+g)^1}{(1+k)^2}\right] + \left[D_0 \cdot \frac{(1+g)^2}{(1+k)^4}\right] + \cdots\right\} + \left\{\left[D_0 \cdot \frac{(1+g)^1}{(1+k)^1}\right] + \left[D_0 \cdot \frac{(1+g)^2}{(1+k)^3}\right] + \cdots\right\} \tag{32}$$

$$= D_0 \left\{\left[\frac{(1+g)^1}{(1+k)^2}\right] + \left[\frac{(1+g)^2}{(1+k)^4}\right] + \cdots\right\} + D_0 \left\{\left[\frac{(1+g)^1}{(1+k)^1}\right] + \left[\frac{(1+g)^2}{(1+k)^3}\right] + \cdots\right\} \tag{33}$$

$$= D_0 \left\{\left[\frac{(1+g)^1}{(1+k)^2}\right] + \left[\frac{(1+g)^2}{(1+k)^4}\right] + \cdots\right\} + D_0(1+k) \left\{\left[\frac{(1+g)^1}{(1+k)^2}\right] + \left[\frac{(1+g)^2}{(1+k)^4}\right] + \cdots\right\} \tag{34}$$

$$= \sum_{t=1}^{\infty} D_0 \left[\frac{1+g}{(1+k)^2}\right]^t + \sum_{t=1}^{\infty} D_0(1+k) \left[\frac{1+g}{(1+k)^2}\right]^t \tag{35}$$

$$= \sum_{t=1}^{\infty} D_0 \frac{1+g}{(1+k)^2} \left[\frac{1+g}{(1+k)^2}\right]^{t-1} + \sum_{t=1}^{\infty} D_0(1+k) \frac{1+g}{(1+k)^2} \left[\frac{1+g}{(1+k)^2}\right]^{t-1} \tag{36}$$

$$= \sum_{t=1}^{\infty} D_0 \frac{1+g}{(1+k)^2} \left[\frac{1+g}{(1+k)^2}\right]^{t-1} + \sum_{t=1}^{\infty} D_0 \frac{1+g}{1+k} \left[\frac{1+g}{(1+k)^2}\right]^{t-1} \tag{37}$$

$$\tag{38}$$

These are both valid geometric series. Now apply the infinite geometric series formula

$$\sum_{n=1}^{\infty} ar^{n-1} = S_n = \frac{a}{1-r}$$

$$PV = \frac{D_0 \frac{1+g}{(1+k)^2}}{1 - \frac{1+g}{(1+k)^2}} + \frac{D_0 \frac{1+g}{1+k}}{1 - \frac{1+g}{(1+k)^2}} \tag{39}$$

$$= \frac{D_0 \frac{1+g}{(1+k)^2} + D_0 \frac{1+g}{1+k}}{1 - \frac{1+g}{(1+k)^2}} \tag{40}$$

$$= \frac{D_0 \frac{1+g}{1+k} \left[\frac{1}{1+k} + 1\right]}{1 - \frac{1+g}{(1+k)^2}} \tag{41}$$

$$\tag{42}$$

Condition of g for when PV is finite.

From the geometric series, PV will be finite when $|r| < 1$, where $r = \frac{1+g}{(1+k)^2}$. So solve

$$|r| < 1 \tag{43}$$

$$\left|\frac{1+g}{(1+k)^2}\right| < 1 \tag{44}$$

$$-(1+k)^2 < 1+g < (1+k)^2 \tag{45}$$

$$-(1+2k+k^2) - 1 < g < (1+2k+k^2) - 1 \tag{46}$$

$$-(2+2k+k^2) < g < k^2 + 2k \tag{47}$$

$$-2 - 2k - k^2 < g < k(k+2) \tag{48}$$

$$\tag{49}$$

Since $g > 0$ and $k > 0$, we do not need to consider the lower bound. Thus for PV to be finite ($PV < \infty$), the following condition must hold

$$g < k(k+2)$$