22 ENDÅ FLEIRE EMNE

IKKJE PENSUM I ACC421A

I FUNDAMENTAL VERDIVURDERING

Førelesing ved

Professor Kjell Henry Knivsflå,
Institutt for rekneskap, revisjon og rettsvitskap, NHH

E-post: kjell.knivsfla@nhh.no; **Twitter: @KjellKnivsfla**

NHH

INNHALD

FØRELESING 22

«TOPICS IN VALUATION»

1) MIDTÅRSJUSTERING

2) VERDIVURDERING I PYTHON

FØRELESING 22

1. FORMLAR

IKKJE PENSUM I ACC421A

NÅR FKE BLIR REALISERT MIDT I ÅRET

I verdivurdering er det vanleg å føresetje at kontantstraumen blir realisert i slutten av året slik at han skal diskonterast heile året

$VEK_0 = (FKE_1 + VEK_1)/(1 + ekk_1)$

→ **Dersom kontantstraumen blir realisert midt i året:**

$VEK_0 = FKE_1 /(1+ekk_1)^{1/2} + VEK_1/(1 + ekk_1)$

FKE_1 skal då diskonterast berre eit halvår

1.1 FKE-MODELLEN

$$VEK_0 = \sum_{t=1}^{T} \frac{FKE_t}{(1+ekk_1)\cdot\ldots\cdot(1+ekk_t)^{1/2}} + \frac{FKE_{T+1}}{\left((1+ekk_1)\cdot\ldots\cdot(1+ekk_T)^{1/2}\right)\cdot(ekk_{T+1} - ekv_{T+1})}$$

eller

$ekk_T = ekk_{T+1} = \ldots$

$$VEK_0 = \sum_{t=1}^{T} \frac{FKE_t \cdot (1+ekk_t)^{1/2}}{(1+ekk_1)\cdot\ldots\cdot(1+ekk_t)} + \frac{FKE_{T+1}\cdot(1+ekk_{T+1})^{1/2}}{(1+ekk_1)\cdot\ldots\cdot(1+ekk_T)\cdot(ekk_{T+1} - ekv_{T+1})}$$

1.2 SPE-MODELLEN

$$VEK_0 = EK_0 \cdot (1 + ekk_1)^{1/2}$$

$$+ \sum_{t=1}^{T} \frac{SPE_t \cdot (1 + ekk_t)^{1/2}}{(1 + ekk_1) \cdot \cdots \cdot (1 + ekk_t)}$$

$$+ \sum_{t=1}^{T} \frac{(((1 + ekk_{t+1})/(1 + ekk_t))^{1/2} - 1) \cdot EK_t \cdot (1 + ekk_t)^{1/2}}{(1 + ekk_1) \cdot \cdots \cdot (1 + ekk_t)}$$

$$+ \frac{SPE_{T+1} \cdot (1 + ekk_{T+1})^{1/2}}{(1 + ekk_1) \cdot \cdots \cdot (1 + ekk_T) \cdot (ekk_{T+1} - ekv_{T+1})}$$

Dersom $ekk_{t+1} = ekk_t$, er dette leddet 0

1.3 DØME

Framtidsrekneskap og krav:

	År	0	1	2	3	4
	NRE		20	11	9,6	9,792
-	ΔEK		10	10	2,4	2,448
=	**FKE**		**10**	**1**	**7,2**	**7,344**
	EK	100	110	120	122,4	124,848
➔	ekr		20%	10%	8%	8%
	ekk		**5%**	**5,5%**	**6%**	**6%**
	SPE		**15**	**4,95**	**2,4**	**2,448**
	Vekst i ss					**2%**

1) FKE
REALISERT I SLUTTEN AV ÅRET

1) FKE-modellen

VEK_0 = 10/1,05 + 1/(1,05 · 1,055) + 7,2/(1,05 · 1,055 · 1,06)

+ 7,344/(1,05 · 1,055 · 1,06 · (0,06 - 0,02))

= **<u>172,9</u>**

2) SPE-modellen

VEK_0 = 100 + 15/1,05 + 4,95/(1,05 · 1,055) + 2,4/(1,05 · 1,055 · 1,06)

+ 2,448/(1,05 · 1,055 · 1,06 · (0,06 - 0,02))

= **<u>172,9</u>**

2.1) FKE
REALISERT MIDT AV ÅRET

1) FKE-modellen

VEK_0 = $10/1{,}05^{1/2}$ + $1/(1{,}05 \cdot 1{,}055^{1/2})$

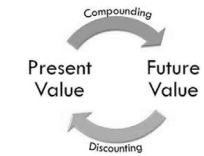
+ $7{,}2/(1{,}05 \cdot 1{,}055 \cdot 1{,}06^{1/2})$

+ $7{,}344 \cdot 1{,}06^{1/2}/(1{,}05 \cdot 1{,}055 \cdot 1{,}06 \cdot (0{,}06 - 0{,}02))$

= 9,8 + 0,9 + 6,3 + 161,0

= 9,8 + 0,9 + 167,3 (sjå Excel-dømet på seinare plansje)

= **<u>178,0</u>**



2.2) FKE
REALISERT MIDT AV ÅRET

2) SPE-modellen

$VEK_0 = 100 \cdot 1{,}05^{1/2}$

$+ 15/1{,}05^{1/2} + 4{,}95/(1{,}05 \cdot 1{,}055^{1/2}) + 2{,}4/(1{,}05 \cdot 1{,}055 \cdot 1{,}06^{1/2})$

$+ ((1{,}055/1{,}05)^{1/2} - 1) \cdot 110/1{,}05^{1/2}$

$+ ((1{,}06/1{,}055)^{1/2} - 1) \cdot 120/(1{,}05 \cdot 1{,}055^{1/2})$

$+ 2{,}448 \cdot 1{,}06^{1/2}/(1{,}05 \cdot 1{,}055 \cdot 1{,}06 \cdot (0{,}06 - 0{,}02))$

$= 102{,}5 + 14{,}6 + 4{,}6 + 2{,}1 + 0{,}3 + 0{,}3 + 53{,}7$

$= \underline{\mathbf{178{,}1}}$

1.4 KONSTANT KRAV

$$VEK_0 = \sum_{t=1}^{T} \frac{FKE_t \cdot (1 + ekk_t)^{1/2}}{(1 + ekk_1) \cdot \cdots \cdot (1 + ekk_t)} + \frac{FKE_{T+1} \cdot (1 + ekk_{T+1})^{1/2}}{(1 + ekk_1) \cdot \cdots \cdot (1 + ekk_{T+1}) \cdot (ekk_{T+1} - ekv_{T+1})}$$

Dersom ekk_t er konstant lik ekk,

$$VEK_0 = (1 + ekk)^{1/2} \cdot \left(\sum_{t=1}^{T} \frac{FKE_t}{(1 + ekk)^t} + \frac{FKE_{T+1}}{(1 + ekk)^T \cdot (ekk - ekv)} \right)$$

dvs

$$VEK_0^M = (1 + ekk)^{\frac{1}{2}} \cdot VEK_0^S$$

MIDTÅRSJUSTERING
TILNÆRMA

1) FKE-modellen

$VEK_0 = 10/1{,}05 + 1/(1{,}05 \cdot 1{,}055) + 7{,}2/(1{,}05 \cdot 1{,}055 \cdot 1{,}06)$

$+ 7{,}344/(1{,}05 \cdot 1{,}055 \cdot 1{,}06 \cdot (0{,}06 - 0{,}02))$

$= \underline{\mathbf{172{,}9}}$

2) Midtårsjustering

Gjennomsnittskravet er lat vi seie 5,5%

$VEK_0 = \underline{\mathbf{172{,}9}} \cdot 1{,}055^{1/2} = \underline{\mathbf{177{,}6 \approx 178{,}0}}$

EXCEL
XNNV(k; FKS; dato)

1) Dersom diskonteringsrenta (k) er **konstant**, så kan vi nytte

NNV(k; FKS)

til å rekne noverdien, under føresetnad av diskontering frå slutten av perioden

Justering til midt i perioden er NNV(k; FKS) · (1 + k)^(1/2)

2) Det finst også ein funksjon i Excel som brukar dei **faktiske datoane for realisering av kontantstraumen**:

XNNV(k; FKS; dato)

der datoane kan variere fritt

Engelsk:
NPV() og XNPV()

DØME
EXCELFUNKSJONAR

DATO	31.12.2021	30.06.2022	30.06.2023	30.06.2024	30.06.2025
FKE		10	1	7,2	7,344
				183,6	
	0,0	10,0	1,0	190,8	
ekk_t		5,0 %	5,5 %	6,0 %	
VEK slutten av året	172,918	9,52	0,90	162,49	
VEK midt i året	177,982	9,76	0,93	167,30	
ekk	5,49 %				
				MÅL	
NNV	172,918	172,918	kr 0,000	0	
NNV · (1+ekk)^(1/2)	177,601				
XNNV	177,615				

ekk er det konstante gjennomsnittskravet som gjev same verdi for NNV som når ekk varierer

Bruk av XNNV() avvik litt sidan talet på dagar av-vik litt mellom datoane

FØRELESING 22

2.

IKKJE PENSUM I ACC421A

VERDIVURDERING I PYTHON

Python is a clear and powerful object-oriented programming language

2.1

THE BASIC

```
# Import modules which are used in the program

import matplotlib.pyplot as plt
import numpy as np

# FINANCIAL STATEMENTS t = 0

IC_0, NFD_0 = 1000, 800

# ROIC and g from 1 to T

T = 4

roic = [None, 0.20, 0.16, 0.02, 0.08]
g_ic = [None, 0.05, 0.12, 0.06, 0.03]

print()

print("THE ASSUMPTIONS:", "T =", T)

print()

print("Current financial statements:", "...IC:", IC_0, "...NFD:", NFD_0, "...E:", IC_0 - NFD_0)

print()

print("Roic:", roic)

print()

print("Growth in IC:", g_ic)
```

Imports **modules** which are used in the program

The values are given names and the other way around

For example roic is given values over time by the use of an **array**

Selected text and variables are **printed**

```
# Start with specifying the invested capital, net operating profit
# after tax, and the free cash flow from operations from 0 to T = 4

IC, NOPAT, FCFO = [IC_0], [None], [None]

for t in range(T+1):
    if t > 0:
        IC.append((1 + g_ic[t]) * IC[t-1])
        NOPAT.append(roic[t] * IC[t-1])
        FCFO.append(NOPAT[t] - (IC[t] - IC[t-1]))
Next

print()

print("THE BUDGET:")

print()

print("IC:", IC)

print()

print("NOPAT:", NOPAT)

print()

print("FCFO:", FCFO)

print()

# The growth after T, i.e. from T+1

g = g_ic[T]
```

A **loop** fills the arrays, for example FCFO

PYTHON
PRINTS

PYTHON **PRINTS** THIS ON THE SCREEN

THE ASSUMPTIONS: T = 4

Current financial statements: ...IC: 1000 ...NFD: 800 ...E: 200

Roic: [None, 0.2, 0.16, 0.02, 0.08]

Growth in IC: [None, 0.05, 0.12, 0.06, 0.03]

THE BUDGET:

IC: [1000, 1050.0, 1176.0, 1246.5600000000002, 1283.9568000000002]

NOPAT: [None, 200.0, 168.0, 23.52, 99.72480000000002]

FCFO: [None, 150.0, 42.0, -47.04000000000018, 62.32800000000003]

CONSTANT
WACC

```
# The WACC
print("THE WACC:")
print()
rf, betaI, erp, crp = 0.02, 0.8, 0.05, 0.01
```

w = 0.6

60% EQUITY WEIGHT SEEMS **OKAY?**

```
betaD = crp/erp
kD = rf + betaD * erp
def betaE(w): return betaI + (betaI - betaD)*(1-w)/w
def kE(w): return rf + betaE(w) * erp
def wacc(w): return kE(w) * w + kD * (1 - w)
print("ASSUMPTIONS:", "rf:", rf, "erp:", erp, "crp:", crp, "betaI:", betaI)
print()
print("kE:", kE(w),"w:", w,"kD:", kD,"wacc:", wacc(w))
print()
```

THE WACC:

ASSUMPTIONS: rf: 0.02 erp: 0.05 crp: 0.01 betaI: 0.8

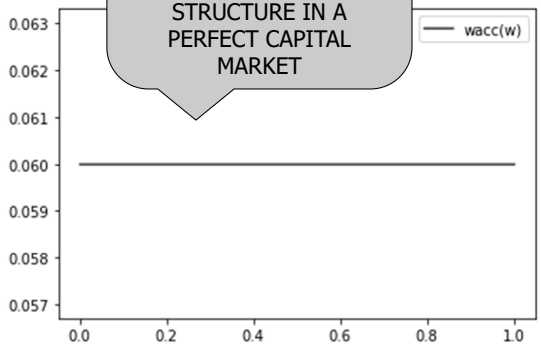
kE: 0.08000000000000002 w: 0.6 kD: 0.03 wacc: 0.06000000000000001

PLOT
A FIGURE

```
# Import modules which are used in the program

import matplotlib.pyplot as plt
import numpy as np

# Make a plot of wacc(w)

x = np.linspace(0.001, 0.999)

plt.plot(x, wacc(x), "red", label="wacc(w)")

plt.legend()

plt.show()
```

MODIGLIANI – MILLER

WACC IS INDEPENDENT OF THE CAPITAL STRUCTURE IN A PERFECT CAPITAL MARKET



VALUATION

```
print("VALUATION")
# The continuing value as a function of w
def CV(w): return FCFO[T]*(1+g)/(wacc(w) - g)
# The enterprise value as a function of w
def EV(w): return FCFO[1]/(1+wacc(w)) + FCFO[2]/(1+wacc(w))**2 + FCFO[3]/(1+wacc(w))**3
                + FCFO[T]/(1+wacc(w))**T + CV(w)/(1+wacc(w))**T

print()
print("The enterprise value is:", EV(w))
print()
print("The value of (net financial) debt is:", NFD_0)
print()
print("The Equity value is:", EV(w) - NFD_0)
print()
```

THE VALUATION:

The enterprise value is: 1883.7866157969324

The value of (net financial) debt is: 800

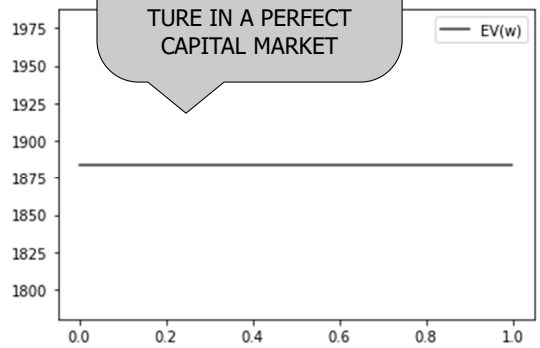
The Equity value is: 1083.7866157969324

PLOT
EV(w)

```
# Make a plot of EV(w)

x = np.linspace(0.001, 0.999)

plt.plot(x, EV(x), "red", label="EV(w)")

plt.legend()

plt.show()
```

MODIGLIANI – MILLER

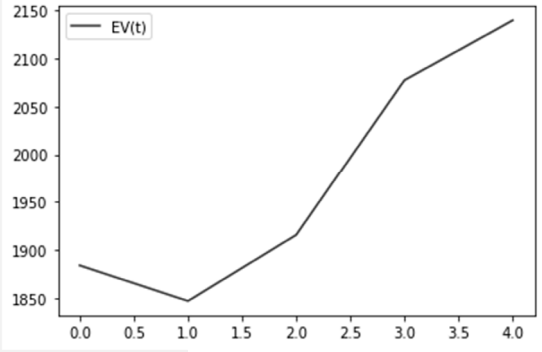
EV IS INDEPENDENT OF THE CAPITAL STRUC-TURE IN A PERFECT CAPITAL MARKET



EV OVER TIME

```
# Calculate the enterprise value over time

E_V = [EV(w)]

for t in range(T+1):

    if t > 0 and t < T+1:

        E_V.append(E_V[t-1] * (1 + wacc(w)) - FCFO[t])

    if t > T:

        E_V.append(E_V[t-1] * (1+g))

next

print()

print("Enterprise value over time:", E_V)

print()

# Figure: EV over time

x = range(0, T+1, 1)

plt.plot(x, E_V, "red", label="EV(t)")

plt.legend()

plt.show()
```

Enterprise value over time: [1883.7866157969324, 1846.8138127447485, 1915.6226415094334, 2077.5999999999995, 2139.9279999999994]



2.2 ANALYSIS

What **should we do** to let the financing have an effect on the wacc and EV

What about uncertainty?

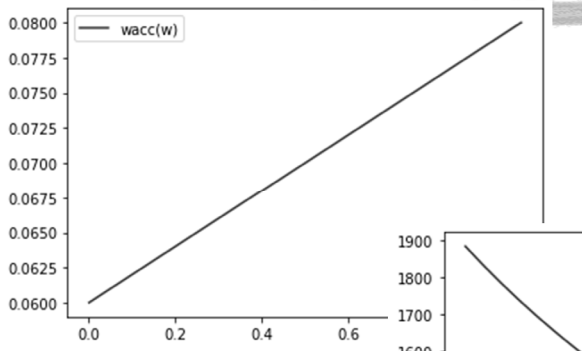
1) IMPERFECT CAPITAL MARKET

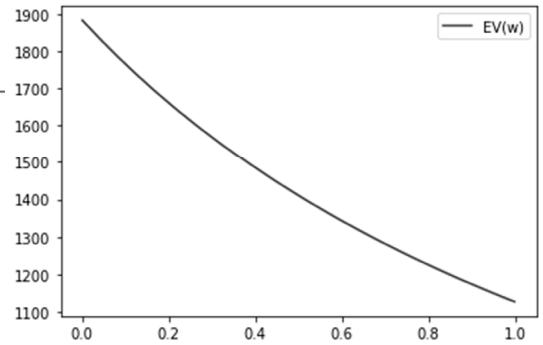
def kE(w): return rf + betaE(w) * erp + orp

CAPM holds in a perfect capital market. Adding an orp (= other risk premium) makes the cost of equinity imperfect and we observe that the capital structure matter!

Assume: orp = 1%

WACC AND EV DEPEND ON w





It becomes optimal to set w* = 0:

NO EQUITY, 100% DEBT

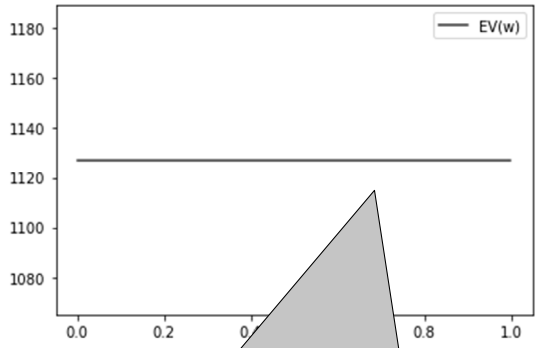
FIRM-SPECIFIC CRP

$$kD = rf + betaD * erp + (1 - mrs) * crp$$

Firm-specific crp

Assume: mrs = 0 and crp is increased to 2%

TRADEOFF



The two effects are balancing each other

2) SIMULATION

EXAMPLE OF ANALYSIS

-> SIMULATION

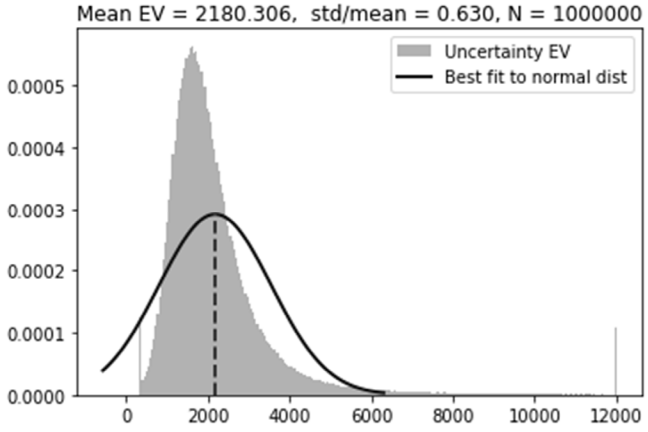
-> UNCERTAINTY OF EV

THE CODE

```
import scipy.stats as sta
from scipy.stats.mstats import winsorize

# SIMULATION
N = 1000000
mean = [rf, roic[T], g_ic[T]]
sd = [0.01, 0.02, 0.01]
rho = [0.7, 0.5, 0.6]
cov  = [[sd[0]**2, rho[0]*sd[0]*sd[1], rho[1]*sd[0]*sd[2]],
        [rho[0]*sd[0]*sd[2], sd[1]**2, rho[2]*sd[1]*sd[2]],
        [rho[1]*sd[0]*sd[2], rho[2]*sd[1]*sd[2], sd[2]**2]]
data = sta.multivariate_normal.rvs(mean, cov, size=N)
rf = data[:, 0]
roic[T] = data[:, 1]
g_ic[T] = data[:, 2]
IC[T] = (1 + g_ic[T]) * IC[T-1]
NOPAT[T] = roic[T] * IC[T-1]
FCFO[T] = (NOPAT[T] - (IC[T] - IC[T-1]))
betaD = mrs*crp/erp
kD = rf + betaD * erp + (1 - mrs) * crp
betaE = betaI + (betaI - betaD)*(1-w)/w
kE = rf + betaE * erp + orp
wacc = kE * w + kD * (1 - w)
CV = FCFO[T]*(1+g)/(wacc - g_ic[T])
if any(wacc > g_ic[T]) and any(roic[T] - wacc > -0.03):
    EV = FCFO[1]/(1+wacc) + FCFO[2]/(1+wacc)**2 +
        FCFO[3]/(1+wacc)**3 + FCFO[T]/(1+wacc)**T + CV/(1+wacc)**T
else:
    EV = Non
EV = list(EV)
j = len(EV) - EV.count(None)
# Fit a normal distribution to the data:
EV = winsorize(EV, (0.005, 0.005))
m_EV, sd_EV = sta.norm.fit(EV)
```

```
# Fit a normal distribution to the data:
EV = winsorize(EV, (0.005, 0.005))
m_EV, sd_EV = sta.norm.fit(EV)
# Plot the histogram.
plt.hist(EV, bins=250, density=True, alpha=0.6, color='gray', label='Uncertainty EV')
# Plot the PDF.
lowlim = m_EV-2*sd_EV
upplim = m_EV+3*sd_EV
x = np.linspace(lowlim, upplim, 100
def p(x): return sta.norm.pdf(x, m_EV, sd_EV)
fig1 = plt.plot(x, p(x), 'blue', linewidth=2, label='Best fit to normal dist')
title = "Mean EV = %.3f,  std/mean = %.3f, N = %.0f" % (m_EV, sd_EV/m_EV, j)
plt.title(title)
plt.legend()
ymax = plt.ylim()
pmax = (0, p(m_EV))
fig2 = plt.plot([m_EV, m_EV], pmax, 'red', linestyle='dashed', linewidth=2)
plt.show()
```

UNCERTANTY



2.3 CONCLUSION

Python is well suited for valuation!