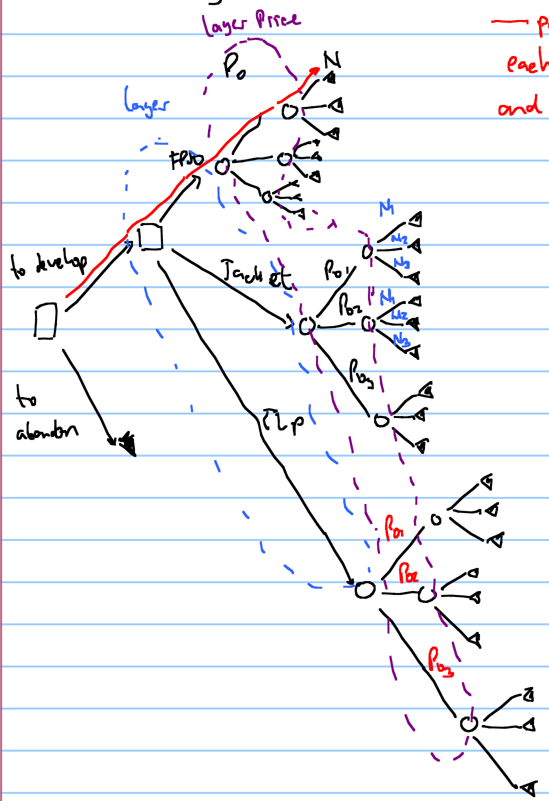
(Cont.) Probability / decision trees

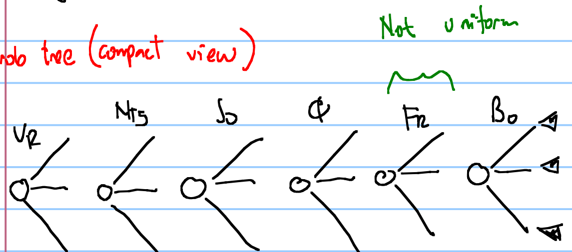


— path
each path has an economic value and a probability

$$P_{path} = P_{P_o} \cdot P_N$$

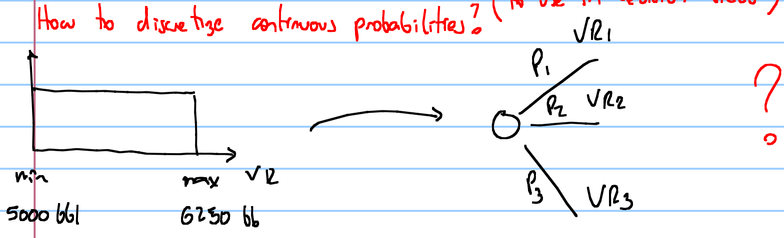
lets solve the previous problem $\left(TRR = \frac{VR \cdot Ntg \cdot So\,(\phi)\,Fr}{Bo}\right.$ using probability trees:

prob tree (compact view)



$3 \times 3 \times 3 \times 3 \times 3 \times 3 = 3^6 = 729$ paths

How to discretize continuous probabilities? (to use in decision trees)



there are many methods:

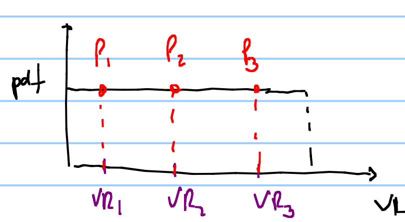
① Value discretization

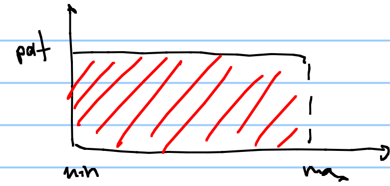
a) define "arbitrarily" "n" values of variable (e.g. uniformly spaced)

b) read from pdf its associated probability

$P_1, P_2, P_3 = 8E\text{-}4$

c) normalize probability





$$P_o (max - min) = 1$$

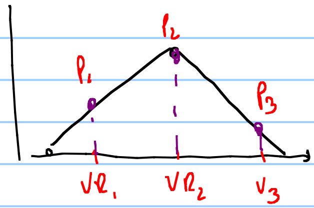
$$P = \frac{1}{(6250 - 5000)} = 8E\text{-}4$$

c) $P_1 = \frac{P_{old}}{\Sigma P_i} = \frac{8E\text{-}4}{3.8E\text{-}4} = 0.333$ ✓

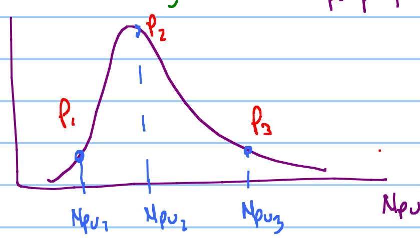
$P_2 = 0.333$ ✓

$P_3 = 0.333$ ✓

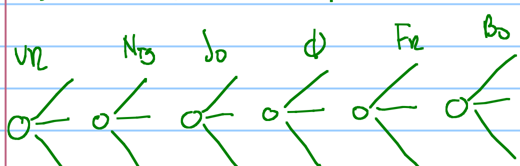
what if the distribution was not uniform pdf



we can also apply this method to discretize our "Npu" pdf



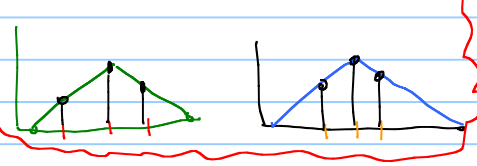
How to solve the tree?



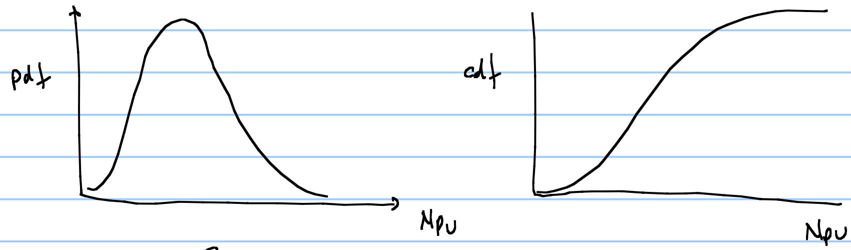
1) in excel

Case	VR	NTG	So	Φ	FR	Bo	Npu	P
1	VR_1	NTG_1	So_1	Φ_1	FR_1	Bo_1		
2	VR_1	NTG_1	So_1	Φ_1	FR_1	Bo_2		
3	VR_1	NTG_1	So_1	Φ_1	FR_1	Bo_3	A	B
4	VR_1	NTG_1	So_1	Φ_1	FR_2	Bo_1		
5	VR_1	NTG_1	So_1	Φ_1	FR_2	Bo_2		
6	VR_1	NTG_1	So_1	Φ_1	FR_2	Bo_3		

if all input probabilities are uniform, then the probability of each case (path) is the same, but, if not, they could be different



with MC (or LHS) we obtain a distribution



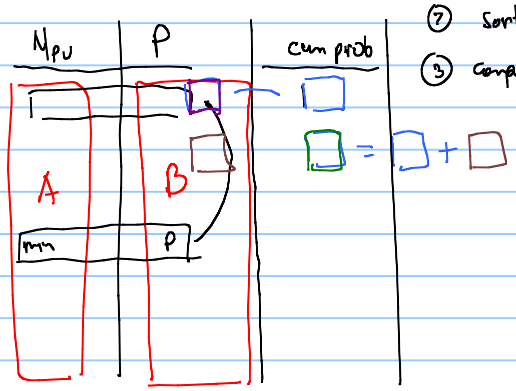
BUT with decision trees?

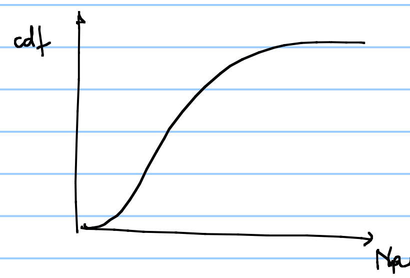
if there are many paths on the tree, we can also compute a distribution:

① take column A and B from above

② Sort values using column "A", e.g from min to max

③ compute cummulative probability



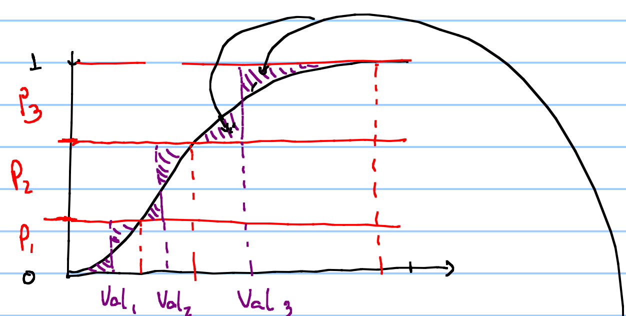


if not enough cases i just report all discrete cases with their associated probability

Npu	p

there are other methods to discretize continuous distributions

1) • on the cdf
• define desired probabilities (e.g. P_1, P_2, P_3)

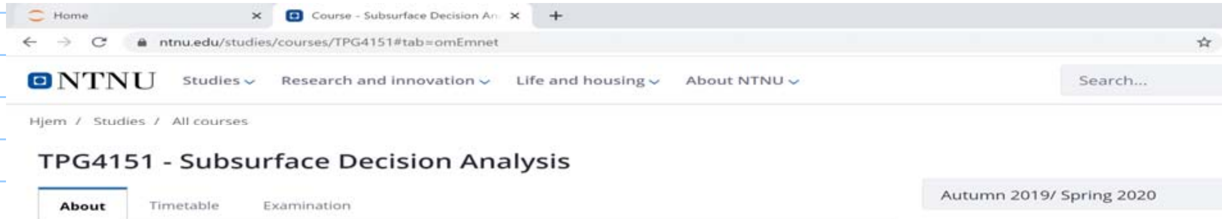


2) change the location of Val_1, Val_2, Val_3 until areas are the same

the discrete distribution (can be done graphically, with the eye or analytically)

Val_1	P_1
Val_2	P_2
Val_3	P_3

for more information about uncertainty quantification, MC simulation, decision trees, take the course:



Several discretization methods are available.

- 3-Point Shortcuts
  - Extended Pearson-Tukey
  - McNamee-Celona
  - Extended Swanson-Megill
- Moment Matching
- CDF Discretization
  - Bracket Mean ←
  - Bracket Median
- Value Discretization (High Resolution Probability Tree) ←

2019 HØST IØ8810 Vinterskole - Planlegging under usikkerhet 24

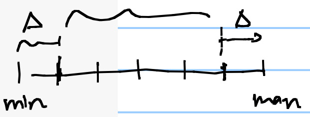
Prof Reidar Bratvold
Aojie Hong

Example, solving the TRR problem with probability tree in python:

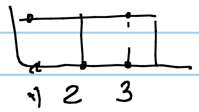
```
In [8]: #importing needed libraries
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as scpstat
```

```
In [9]:  #declaring necessary functions
def Npu(a):
    #returns N in [stb or Sm3]
    #input:
    #por, porosity, [-]
    #RV, rock volume, [m3 or bbl]
    #NTG, net to gross, [-]
    #So, oil saturation, [-]
    #Bo, oil formation volume factor [m3/Sm3]
    por=a[0]
    RV=a[1]
    NTG=a[2]
    So=a[3]
    Bo=a[4]
    Fr=a[5]
    TRR=por*RV*NTG*So*Fr/Bo
    return TRR
```

```
def total_prob(a):
    #calculates the probability of a particular combination of branches
    prob_1=a[0]
    prob_2=a[1]
    prob_3=a[2]
    prob_4=a[3]
    prob_5=a[4]
    prob_6=a[5]
    b=prob_1*prob_2*prob_3*prob_4*prob_5*prob_6
    return b
def branches(min_val,max_val,nr):
    delta=(max_val-min_val)*0.5/nr
    a=np.linspace(min_val+delta,max_val-delta,3)
    return a
def discrete_prob_uniform(min_val,max_val,val):
    #discretizes a uniform probability function using the value discretization method
    a=scpstat.uniform.pdf(val,loc=min_val,scale=(max_val-min_val))
    a=a/np.sum(a)
    return a
```



$$\Delta = \frac{max - min}{n} \cdot 0.5$$



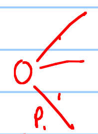
normalization

gets pdf

```
#input data
#porosity
por_min=0.18
por_max=0.3
```

```
#input data
#porosity
por_min=0.18
por_max=0.3
#Rock Volume [1E06 bbl]
RV_min=5000
RV_max=6250
#Net to gross
NTG_min=0.3
NTG_max=0.5
#Oil saturation
So_min=0.8
So_max=0.9
#Oil formation volume factor [bbl/stb]
Bo_min=1.35
Bo_max=1.6
#recovery factor [-]
Fr_min=0.18
Fr_max=0.35
```

```
#nr. branches per variable
nb=3
#calculating branches
por=branches(por_min,por_max,nb)
RV=branches(RV_min,RV_max,nb)
NTG=branches(NTG_min,NTG_max,nb)
So=branches(So_min,So_max,nb)
Bo=branches(Bo_min,Bo_max,nb)
Fr=branches(Fr_min,Fr_max,nb)
#calculating probabilities of each branch (using value discretization)
prob_por=discrete_prob_uniform(por_min,por_max,por)
prob_RV=discrete_prob_uniform(RV_min,RV_max,RV)
prob_NTG=discrete_prob_uniform(NTG_min,NTG_max,NTG)
prob_So=discrete_prob_uniform(So_min,So_max,So)
prob_Bo=discrete_prob_uniform(Bo_min,Bo_max,Bo)
prob_Fr=discrete_prob_uniform(Fr_min,Fr_max,Fr)
```

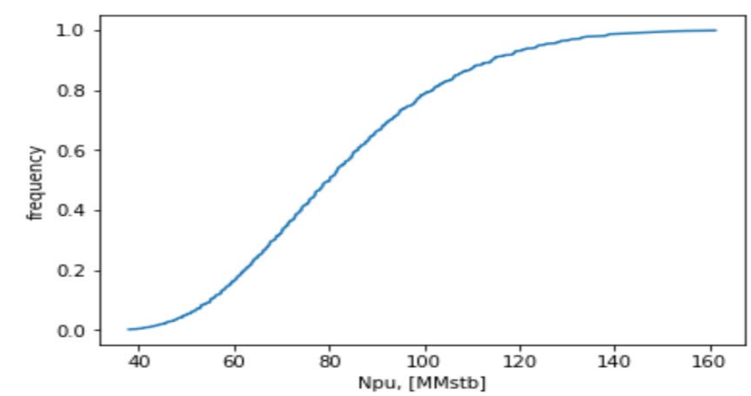


```
nr_variables=6
#create an element-wise combination of all branches
combination_vector=np.array(np.meshgrid(por,RV,NTG,So,Bo,Fr)).T.reshape(-1,nr_variables)
combination_prob=np.array(np.meshgrid(prob_por,prob_RV,prob_NTG,prob_So,prob_Bo,prob_Fr)).T.reshape(-1,nr_variables)
N_comb=len(combination_vector)
results_val=[]
results_prob=[]
for i in range(0,N_comb-1):
    results_val.append(Npu(combination_vector[i]))
    results_prob.append(total_prob(combination_prob[i]))
results=np.vstack((results_val,results_prob))
results=results.T
results=np.sort(results,axis=0)
cdf=np.cumsum(results[:,1])
```

- results_val=[] → vector to gather results
- results_val.append(...) → applying Npu function on each line
- results_prob.append(...) → calculating prob of each path
- results=np.sort(...) → sort from min to max
- cdf=np.cumsum(...) → calculate cummulative prob

$$\begin{bmatrix} (VR_1, & NTG_1, & S_{o1}, & \Phi_1, & Fr_1, & Bo_1) \\ VR_1 & NTG & S_{o1} & \Phi_1 & Fr_1 & Bo_2 \\ \vdots \end{bmatrix} \quad \begin{bmatrix} P_{VR_1} & P_{NTG_1} & P_{S_1} & P_{\Phi_1} & P_{Fr_1} & P_{Bo_1} \\ P_{VR_1} & P_{NTG_1} & P_{S_1} & P_{\Phi_1} & P_{Fr_1} & P_{Bo_2} \end{bmatrix}$$

```
#plot cdf
plt.xlabel('Npu, [MMstb]')
plt.ylabel('frequency')
plt.plot(results[:,0],cdf,label="cdf")
plt.show()
```



pending topics:
- offshore structures for oil and gas production
- important part of CAPEX → affect NPV
- technical constraints
- flow assurance { wax, hydrate, scale, emulsion, asphalthenes, corrosion, erosion, vibration, slugging
- Electric submersible pumps (ESPs) } tentative
- production optimization

Monday | Friday

13.03
offshore struc

16.03
offshore struc

20.03
offshore
flow assurance

23.03
flow ass.

27.03
flow assure
LedaFlow (transient multiphase flow simulator)

30.03
LedaFlow (transient multiphase flow simulator)
Production optimization

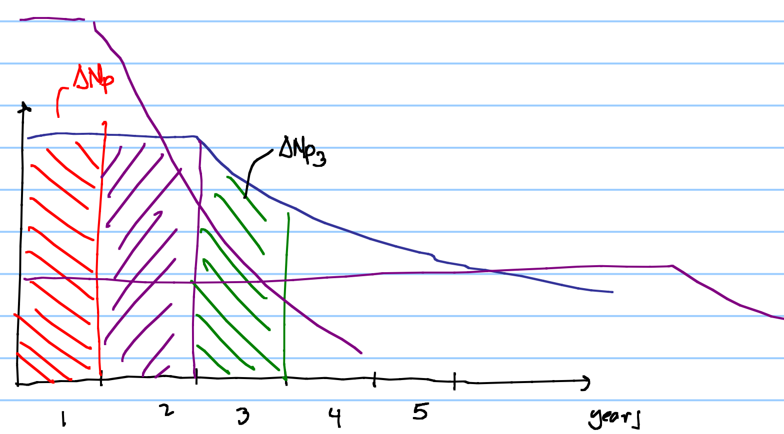
04.04
Exercise session

17.04
prod. optimization

20.04
- Consultation
- Course summary

- Exercise 2 will be merged with exercise set 1 with delivery date after Easter
  (please keep working on it ☹ ≥) !
  There will be one exercise using LedaFlow.

Clarifications problem 6, exercise set 1



$$NPV_{revenue} = \sum_{i=1}^{N} \frac{\Delta Q_{p_i} \cdot P_o}{(1+d_f)^i}$$

$$N_{pu} \cdot P_o \cdot F_D = NPV_{revenue} = \frac{\Delta N_{p_1} \cdot P_o}{(1+d_f)^1} + \frac{\Delta N_{p_2} \, P_o}{(1+d_f)^2} + \frac{\Delta N_{p_3} \cdot P_o}{(1+d_f)^3} + \ldots$$

$$N_{pu} = \sum_{i=1}^{N} \Delta N_{p_i}$$

Use data from your Snohvit exercise to see the range of variation of Fd!!!