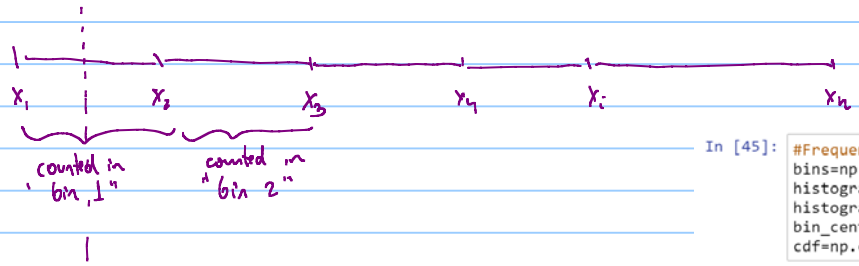
a comment in python about bins



for plotting

$x_{1.5}$ $x_{2.5}$ $x_{3.5}$

```
In [45]: #Frequency analysis of results
bins=np.linspace(TRR.min(),TRR.max(),20)
histogram,bins=np.histogram(TRR,bins=bins)
histogram=histogram/n
bin_centers=0.5*(bins[1:]+bins[:-1])
cdf=np.cumsum(histogram)
```

```
In [46]: #plotting pdf
plt.xlabel('Npu, [MMstb]')
plt.ylabel('frequency')
plt.plot(bin_centers,histogram)
plt.show()
```

```
bin_centers=0.5*(bins[1:]+bins[:-1])
```

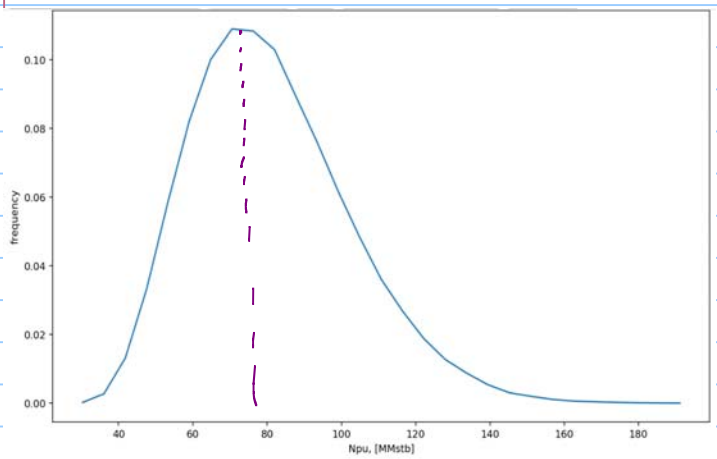
index : 0 1 2 3

bins $[a_1\ a_2\ a_3\ a_4]$

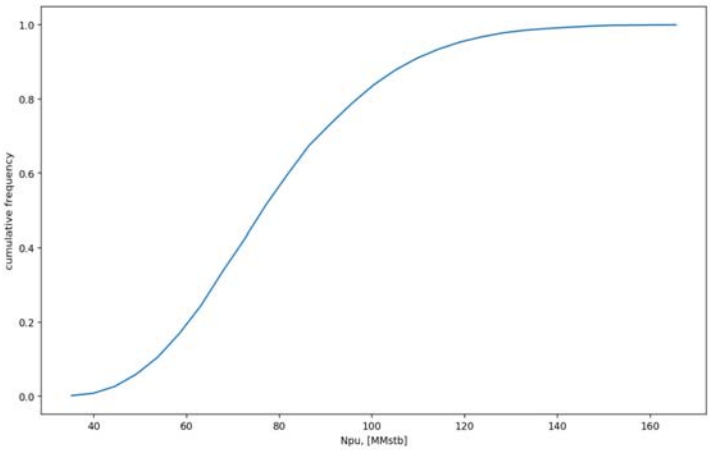
bins[1:] $\therefore [a_2\ a_3\ a_4]$

bins[:-1] $\therefore [a_1\ a_2\ a_3]$

bin centers : $\frac{a_1+a_2}{2}$ $\frac{a_3+a_3}{2}$ $\frac{a_4+a_3}{2}$



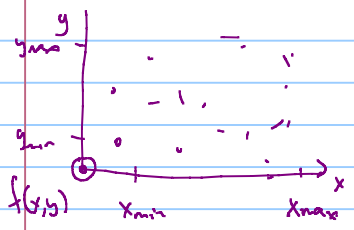
pdf

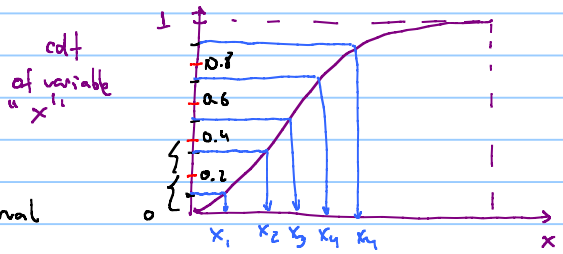



```
In [47]: #plotting cdf
plt.xlabel('Npu, [MMstb]')
plt.ylabel('cumulative frequency')
plt.plot(bin_centers,cdf)
plt.show()
```

Other methods to quantify uncertainty:

Latin Hypercube Sampling (LHS) like Monte Carlo but instead of random sampling, a more intelligent sampling is made





1: divide cdf in "n" intervals

2: sample randomly from each interval

0 → 0.2
0.2 → 0.4
0.4 → 0.6
0.6 → 0.8
0.8 → 1

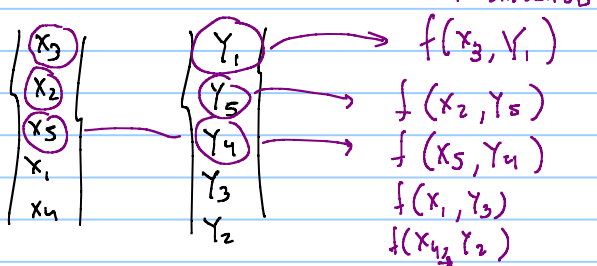
3: find corresponding "x"s

x_1
x_2
x_3
x_4
x_5

4. Shuffle (random) $\{x_3, x_2, x_5, x_1, x_4\}$

5: Repeat steps 1 - 4 for each variable

LHS Simulations



6: do frequency analysis on results

interval_start : x_1, x_2, x_3, x_4

interval_end : x_2, x_3, x_4, x_5

- x_5 end
- x_4
- x_3
- x_2
- x_1 start

function vectorization

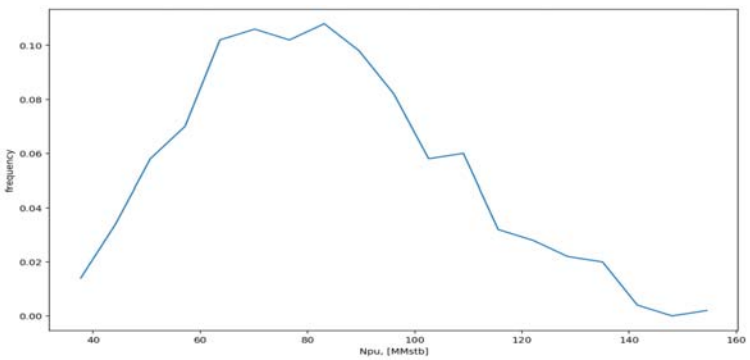
np.random.uniform(x_1, x_2, , x_2, x_3, x_4, x_4) ⟷ np.random.uniform(x_1, x_2) → x^*

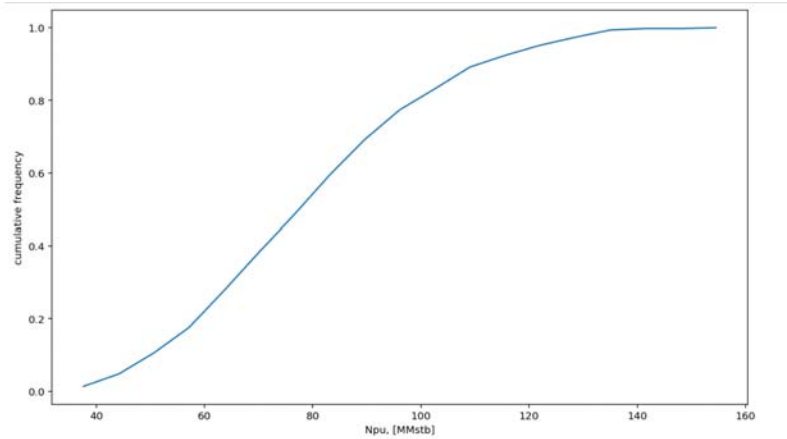
np.random.uniform(x_2, x_3) →

In [5]:
```
#declare functions
def Npu(por,RV,NTG,So,Bo,Fr):
    #por porosity in fraction
    #RV rock volume in stb or Sm3
    #NTG net to gross, in fraction
    #So oil saturation, in fraction
    #Bo oil formation volume factor, bbl/stb or m3/Sm3
    #Fr recovery factor, in fraction
    TRR=por*RV*NTG*So*Fr/Bo
    return TRR
def LHS_samples_uniform(min_val,max_val,n):
    #returns a list of random values from a uniform distribution using LHS
    #n: number of samples to generate
    #min_val minimum value of the variable
    #max_val maximum value of the variable
    len_int=1/n  #delta in cdf
    interval_start=np.linspace(0,1-len_int,n)
    interval_end=np.linspace(len_int,1,n)
    cdf_samples=np.random.uniform(interval_start,interval_end)
    samples=scpstat.uniform.ppf(cdf_samples,loc=min_val,scale=(max_val-min_val))
    np.random.shuffle(samples)
    return samples
```

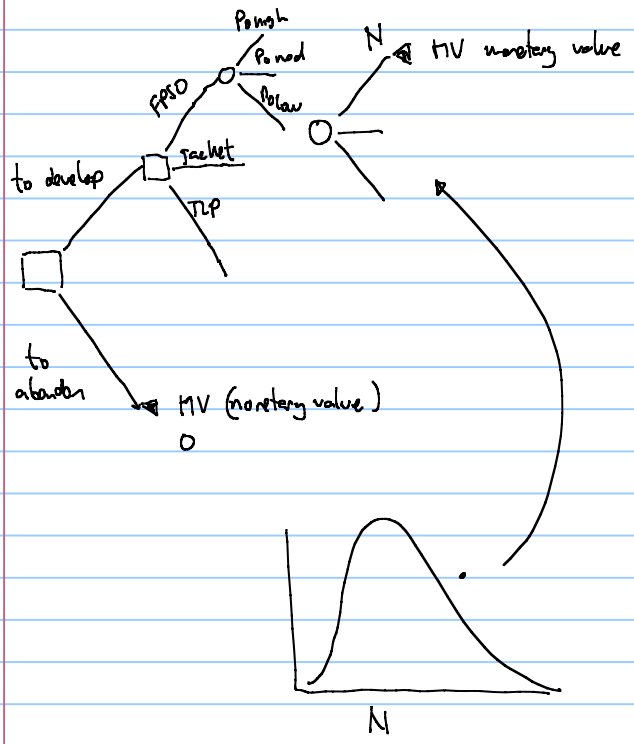
In [6]:
```
#input data
por_min=0.18
por_max=0.3
RV_min=5e3   # in million bbl
RV_max=6.25e3
NTG_min=0.3
NTG_max=0.5
So_min=0.8
So_max=0.9
Bo_min=1.35
Bo_max=1.6
Fr_min=0.18
Fr_max=0.35
```

In [12]:
```
#LHS simulation
n=500
por=LHS_samples_uniform(por_min,por_max,n)
RV=LHS_samples_uniform(RV_min,RV_max,n)
NTG=LHS_samples_uniform(NTG_min,NTG_max,n)
So=LHS_samples_uniform(So_min,So_max,n)
Bo=LHS_samples_uniform(Bo_min,Bo_max,n)
Fr=LHS_samples_uniform(Fr_min,Fr_max,n)
#LHS simulation, compute Npu for all samples
TRR=Npu(por,RV,NTG,So,Bo,Fr)
```





- to reduce even further computational time, and to include integer / discrete variables it is possible to use probability trees



□ decision node

○ chance node

◀ end node

			P
P_1	$P_{o\,low}$	30 USD/bbl	0.4
P_2	$P_{o\,high}$	100 USD/bbl	0.1
P_3	$P_{o\,med}$	70 USD/bbl	0.5

Case nr.	topside type	P_o	N	MV	Probability
1	FPSO	HIGH	SMALL	( )	$P_{o\,high} \cdot P_{N\,small}$
2	FPSO	HIGH	MED	( )	
3	FPSO	HIGH	HIGH		
4	FPSO	MED	SMALL		
5	FPSO	MED	MED		
6	FPSO	MED	HIGH		
7	FPSO	LOW	SMALL		
8	FPSO	LOW	MED		
9	FPSO	LOW	HIGH		

and similar for Jacket and TLP (tension leg platform)

oil price N

9 : 3 × 3

cases cases