**New approach of Valuation Generation**

**Notation of Parameters:**

**T**: the length of Time Epoch

**R**: the bouncing range of mean valuation per time epochs

**N**: customer number

**sig**: the sigma in Gaussian function. Smaller the sigma is, closer valuations are.

Case 1: **T** = 50 time slots**, R** = 2.0

/Users/konglingkun/Desktop/Bancor-Simulator/Figures/Bancor/Valuation-TE-50BG-2.0CN-1000Sig-1.0Seed-6.pdf

Case 2: **T** = 200 time slots, **R** = 2.0 (1000/200 = 5, therefore only five "stairs")

Bancor-Simulator/Figures/Bancor/Valuation-TE-200BG-2.0CN-1000Sig-1.0Seed-4.pdf

Case 3: **T** = 200 time slots, **R** = 5.0 (the **market valuation of smart token per time epoch** is bouncing more significantly due to larger **R**)

Bancor-Simulator/Figures/Bancor/Valuation-TE-200BG-5.0CN-1000Sig-1.0Seed-4.pdf

**Details about Mean Valuation Generation:**

To begin with, based on the price of smart token, we generate the market valuation of smart token per time epoch as **Vtp**, by which the mean valuations **Vt**, i.e., blue point in above figures can be generated. After we get Vt, every customer's valuation is generated based on **Vt** (mu in Gaussian) and **sig** (sigma in Gaussian).

For instance, we first generate **Vtp** = 20 ETH, which indicates in this time epoch, customers generally regard the smart tokens' value as 20 ETH; while in different time slots of this time slot, the specific valuation **Vt** can be 19.2, 21.4, 20.8, 19.5 and so on. Then, in every time slot, **N** customers' valuations are made by **Vt**, such as 19.2, 21.4, 20.8, and etc.

The reason we use the price of smart token **P** to generate **Vtp** is that we try to simulate the equal chance for market craze and market crisis, i.e., 50% probability of **Vtp** < **P -- Vtp** selected from **(P/R, P),** and 50% probability of **Vtp** > **P -- Vtp** selected from **(P, P\*R).**

**The Selection of Vtp & the Generation of Vt and every customers' valuation:**

Vt\_list = []

for j in range(TimeSlotNum):

  '''

  First of all, we randomize mean valuations in time epochs from (P/R, P\*R),

      and save in valuation\_Epoch list.

  For instance, 0-49 time slot comprise the first time epoch.

  If the mean valuation is 20 ETH, in 0 - 49 time slots,

      customers generate their orders based on 19.4 ETH, 21.2 ETH ...

  '''

  #

  P = KennyCoin.getPrice()

  if j % T == 0: # T is the length of time epoch

      valuation\_Epoch = []

      if bool(random.getrandbits(1)):

**Vtp** = random.uniform(P/R, P)

      else:

**Vtp** = random.uniform(P, P\*R)

      # generate a random series of valuations in timeEpoch

      Vt\_temp = np.random.normal(**Vtp**, 1, T).tolist()

**Vt\_list**.extend(Vt\_temp)

    valuation\_mu = Vt\_list[j % T]

    custValuation\_list = np.random.normal(valuation\_mu, sigma, custNum)

    for i in range(custNum):

      # lauching transaction orders based on custVuation\_list[i]

      ...

Basically, what we do is to randomly select **Vtp** from **(P/R, P\*R),** and then based on **Vtp** generate **Vt**s for every time slot and save them in **Vt\_list**. After we get **Vt**, every customer's valuation of smart tokens can be generated.

Exp for Vt\_temp = np.random.normal(**Vtp**, 1, T).tolist()

Assuming **Vtp** = 20 ETH, **T** = 5 time slots:

>> Vt\_temp = [20.801, 20.978, 20.161, 19.074, 19.756]