New Results

Kenny 10/23/2017

Notations:

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.

P: the price of smart token at the beginning of every time slot

Bancor Market:

The whole simulating time is comprised of **1000** time slots.

In every time slot, Bancor Market processes orders launched by **N** customers.

Valuation Making -> Transaction Launching -> Transaction Processing

Valuation Making in Bancor:

At the beginning of every time slot, **N** customers will be announced about the price of smart token **P.**

Based on this price, we generate the market valuation of smart token per time epoch as **Vtp**. E.g. 20 ETH. (By random pick)

By **Vtp**, valuation in every time slot of this time epoch **Vt** can be generated. E.g. 19.2, 21.4, 20.8, 19.5 …

-- Vt_list = np.random.normal(**Vtp**, 1, **T**) # here 1 is the sigma

Customers according to **Vt** as mu in Gaussian, generate their valuations.

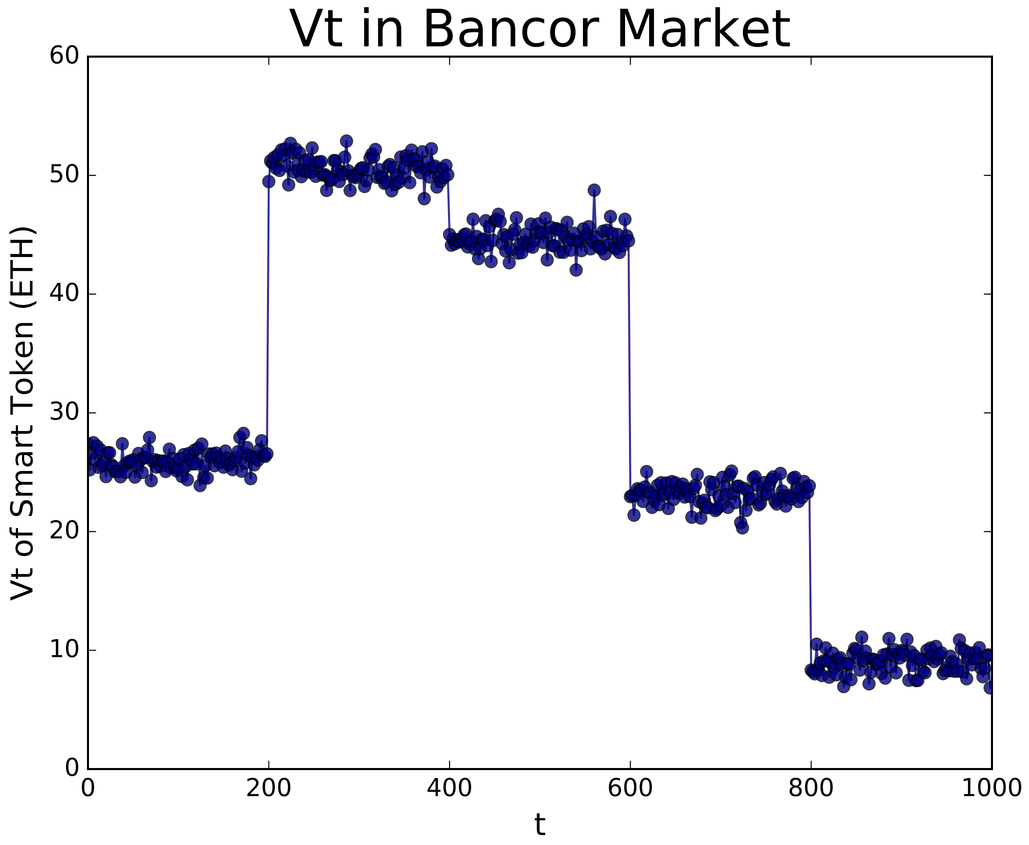
-- custValuation_list = np.random.normal(**Vt, sig**, **N**)

Code of Random Pick:

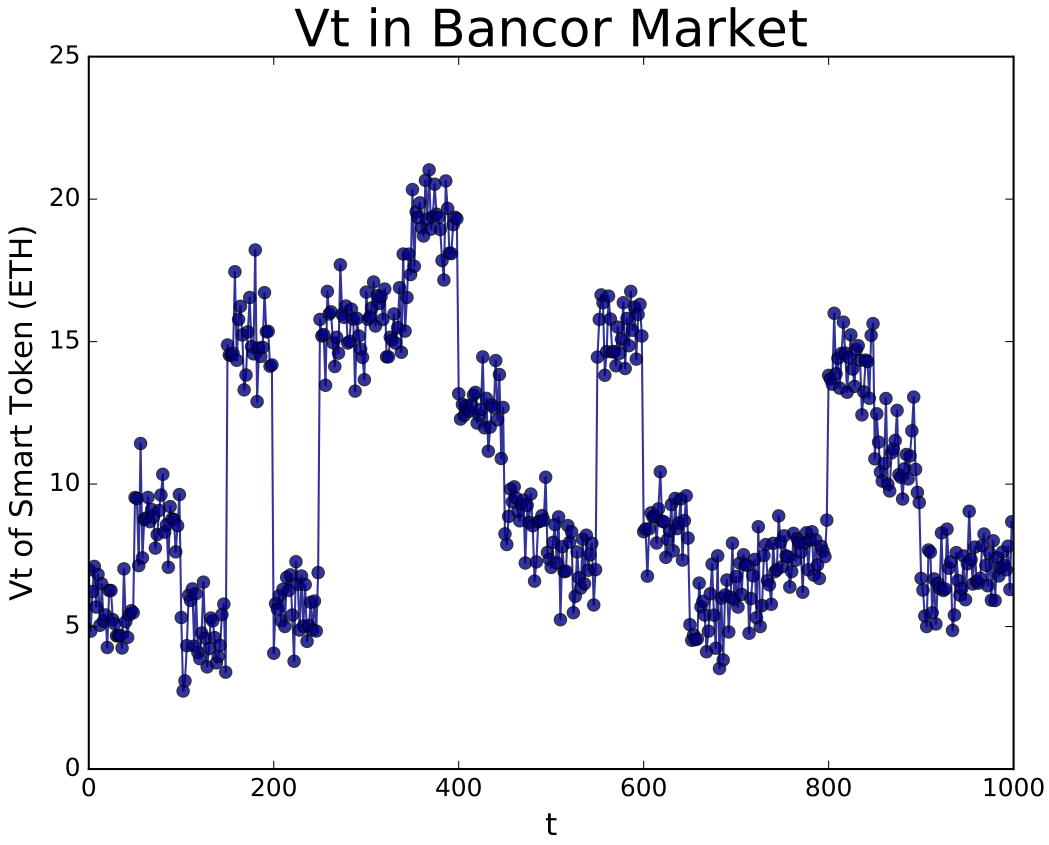
```
# getrandbits(1) return False or True randomly
if bool(random.getrandbits(1)):
    Vtp = random.uniform(P/R, P)
else:
    Vtp = random.uniform(P, P*R)
```

By code, the valuation has 50% probability to be larger than P; while 50% probability to be smaller.
We try to simulate the equal chance for **market craze** and **market crisis.**

Figures about Valuation Making:

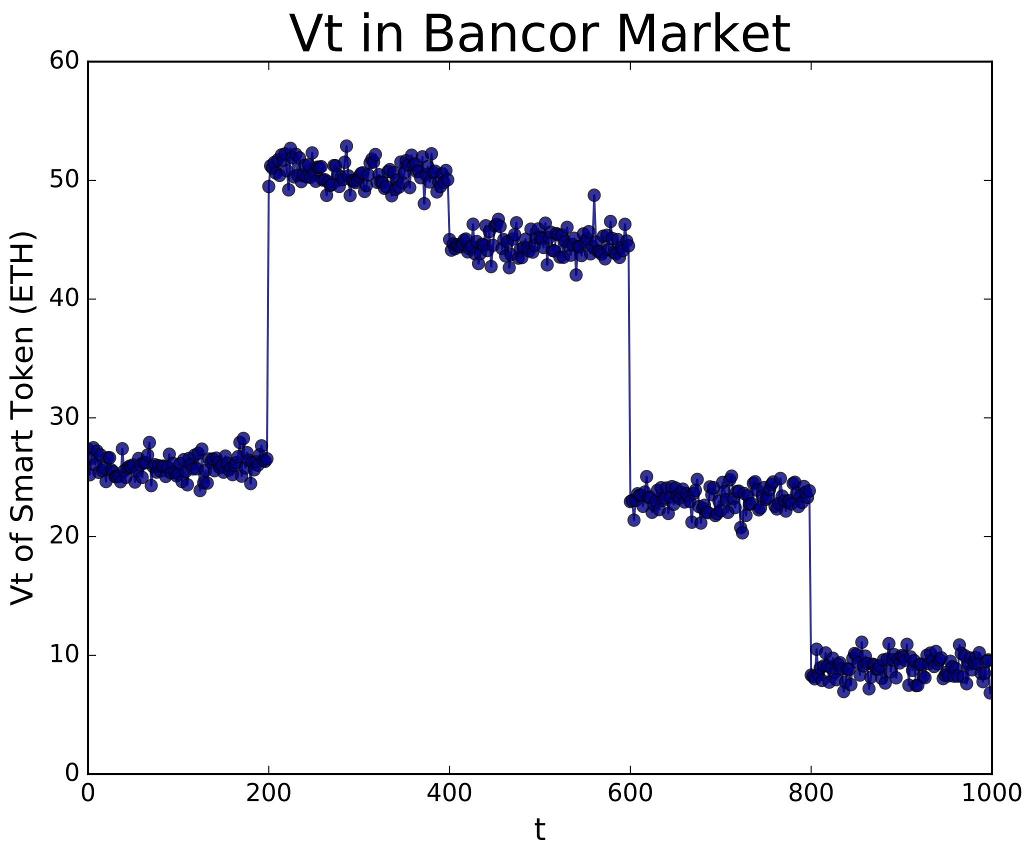


Length of Time Epoch = 200 time slots
Bouncing Range = 5.0

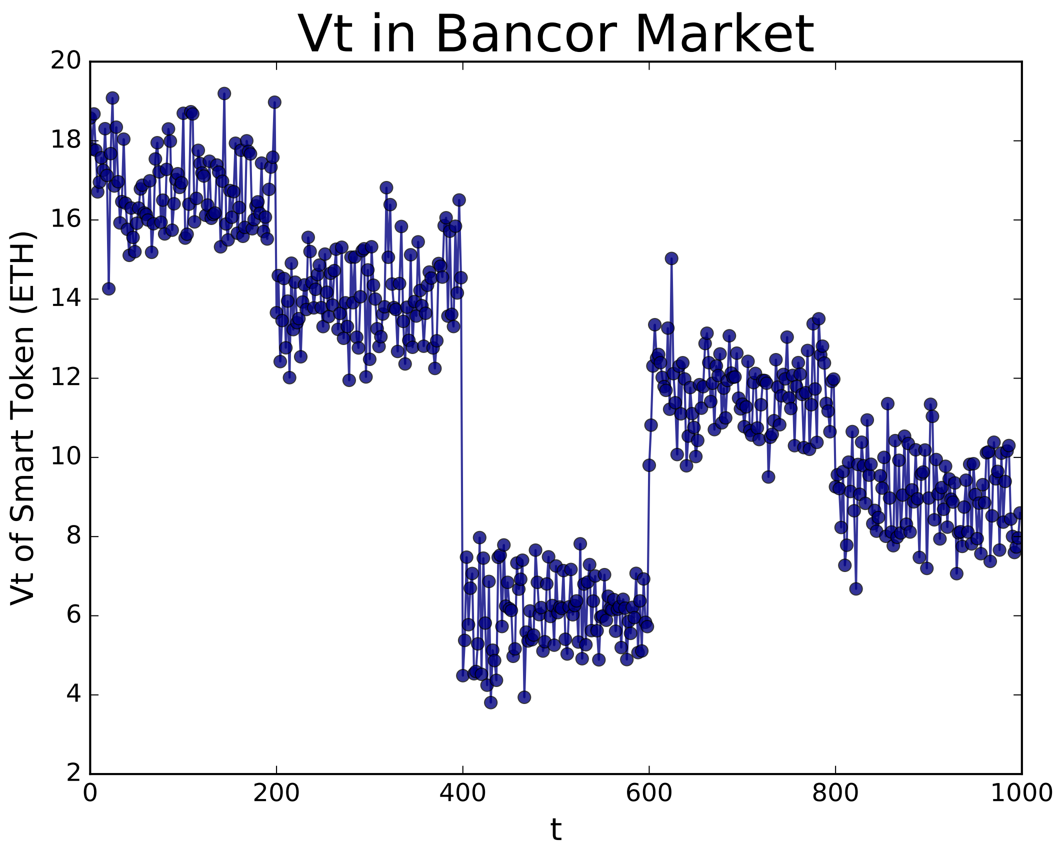


Length of Time Epoch = 50 time slots
Bouncing Range = 5.0

Figures about Valuation Making:



Length of Time Epoch = 200 time slots
Bouncing Range = 5.0



Length of Time Epoch = 200 time slots
Bouncing Range = 2.0

Transaction Generating in Bancor:

After customers making their valuations of smart token, they will launch transaction orders with several stipulations:

1. If valuation > **Psc,** customers will launch transaction orders to buy the smart token; otherwise when valuation < **Psc** , sell orders will be generated.

2. If no reserve tokens in hand, customer will not launch orders to buy smart token, though their valuations might be higher than **Psc.** Ditto for sell orders.

3. Customers make valuations and launch orders in every time slot; while in one time slot, one customer can only try to generate one order.
   -- 1000 time slots, totally **<=1000N** transaction orders will be made

4. Customer uses all of their reserve tokens or smart tokens to buy or to sell if they have money in hand.

Code about Transaction Generation:

```
# self represents a customer (customer class). we name him/her as XXX
if self._valuation > marketPrice and self._reserveBalance > 0:
    # XXX issues a buy order
    self._market.buy(self, self._reserveBalance) # all-in policy
elif self._valuation < marketPrice and self._tokenBalance> 0:
    # XXX issue a sell order
    self._market.sell(self, self._tokenBalance)
else:
    # nothing to do
    pass
```

After customers generating their transaction orders by valuations and money they hold, they wait to see whether market could match their orders.

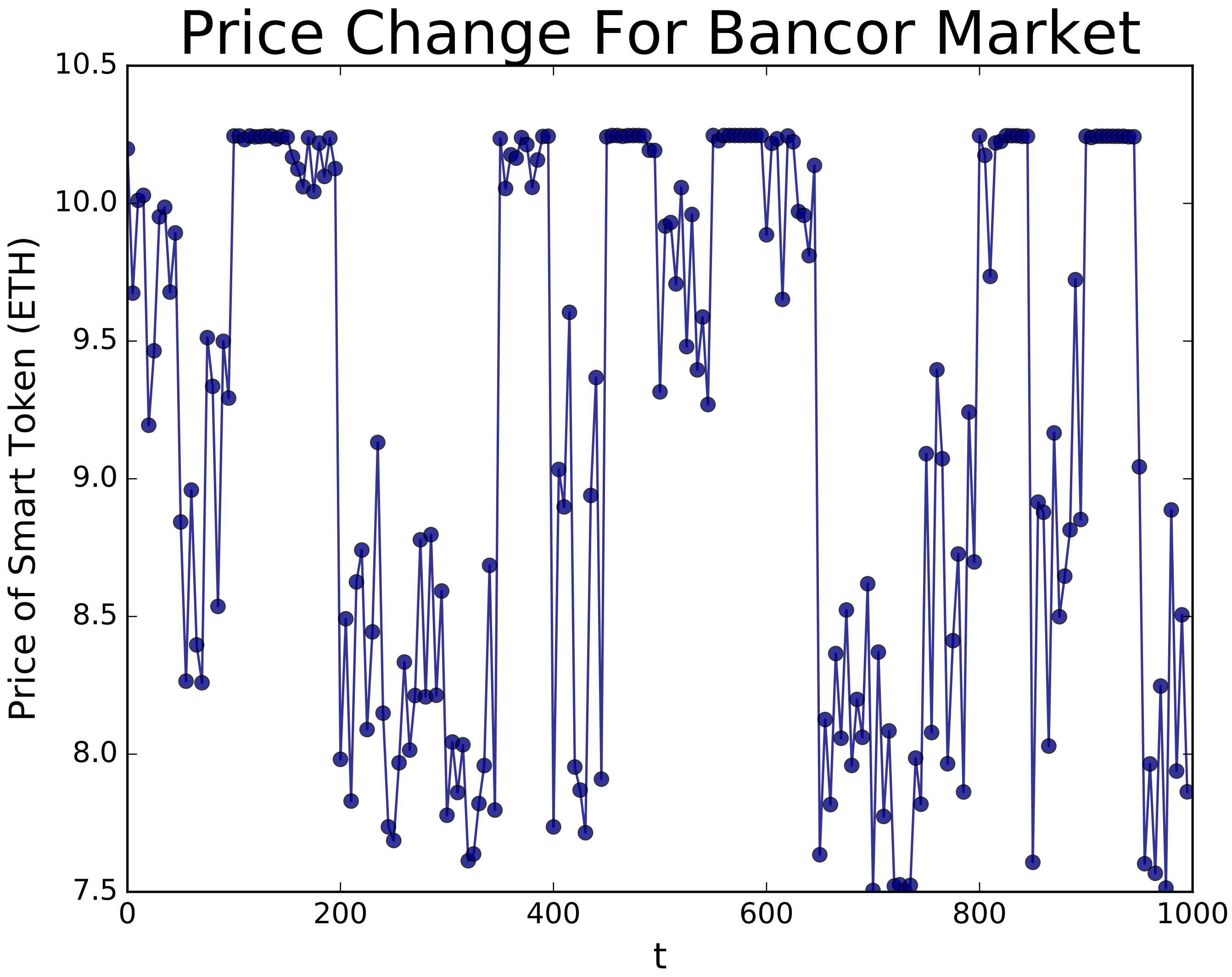
Transaction Processing in Bancor:

Bancor Market processes customers' transaction orders **one by one**.

In market's eyes:
When dealing with one of customers' transaction orders,
if the real-time price of smart token does not meet this customer's valuation, the market will announce the customer to cancel this order and skip this transaction order to try to deal with the next customer's order.

In customers' eyes:
Customers generate valuations of product first, and then accord to the product's real-time price in market to decide whether to cancel the transaction.

Price fluctuation in Bancor:



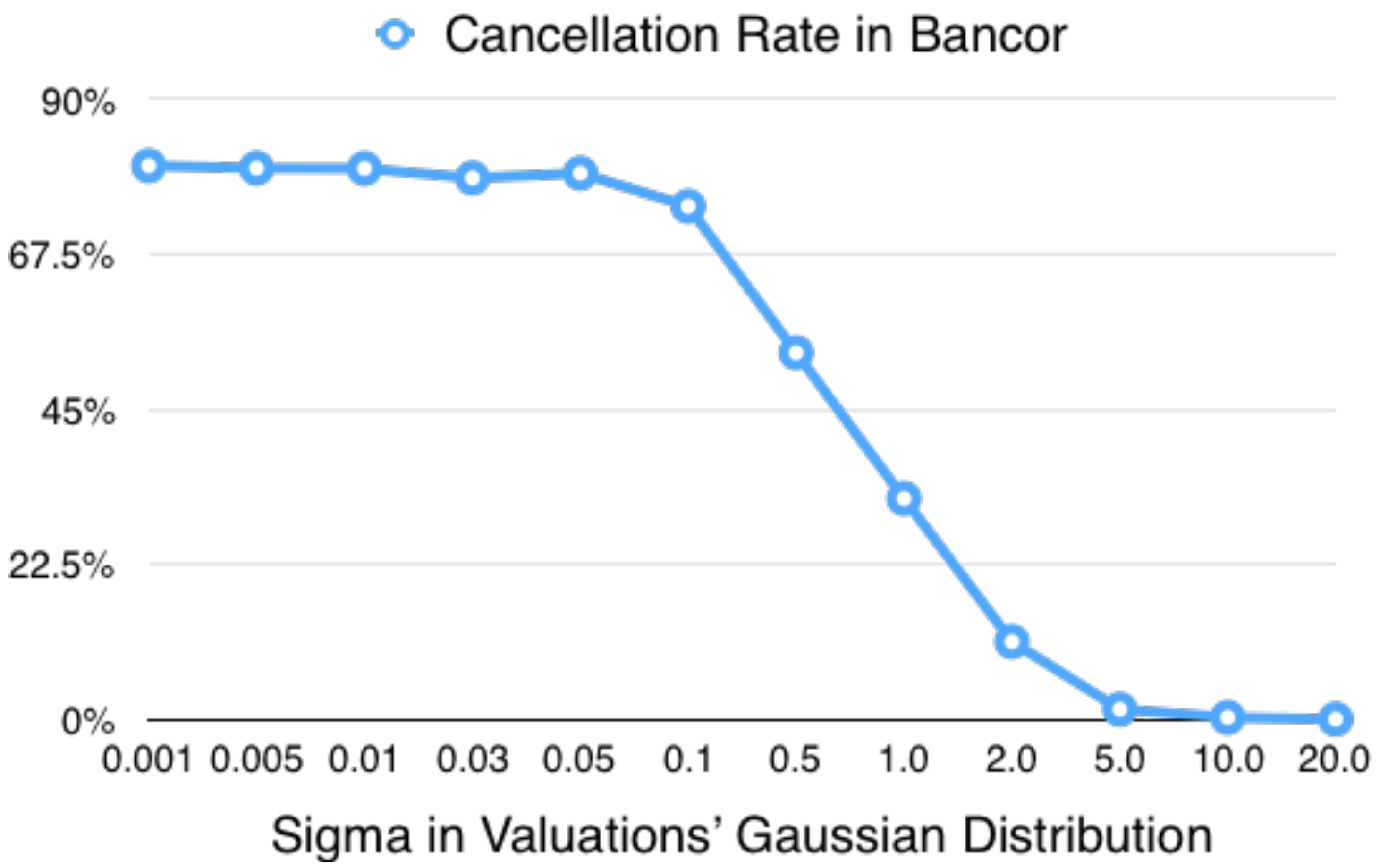
Y-axis: Price of smart token (ETH)
X-axis: time of the simulation

The price change in Bancor Market when T = 50, R = 2.0, N=1000, sig=2.0.

Cancellation Rate in Bancor Market:

Further, since transaction orders might be canceled in market, we also track the cancellation rate of transaction orders successfully launched by customers.

Bancor Cancellation Rate when sigma changes:



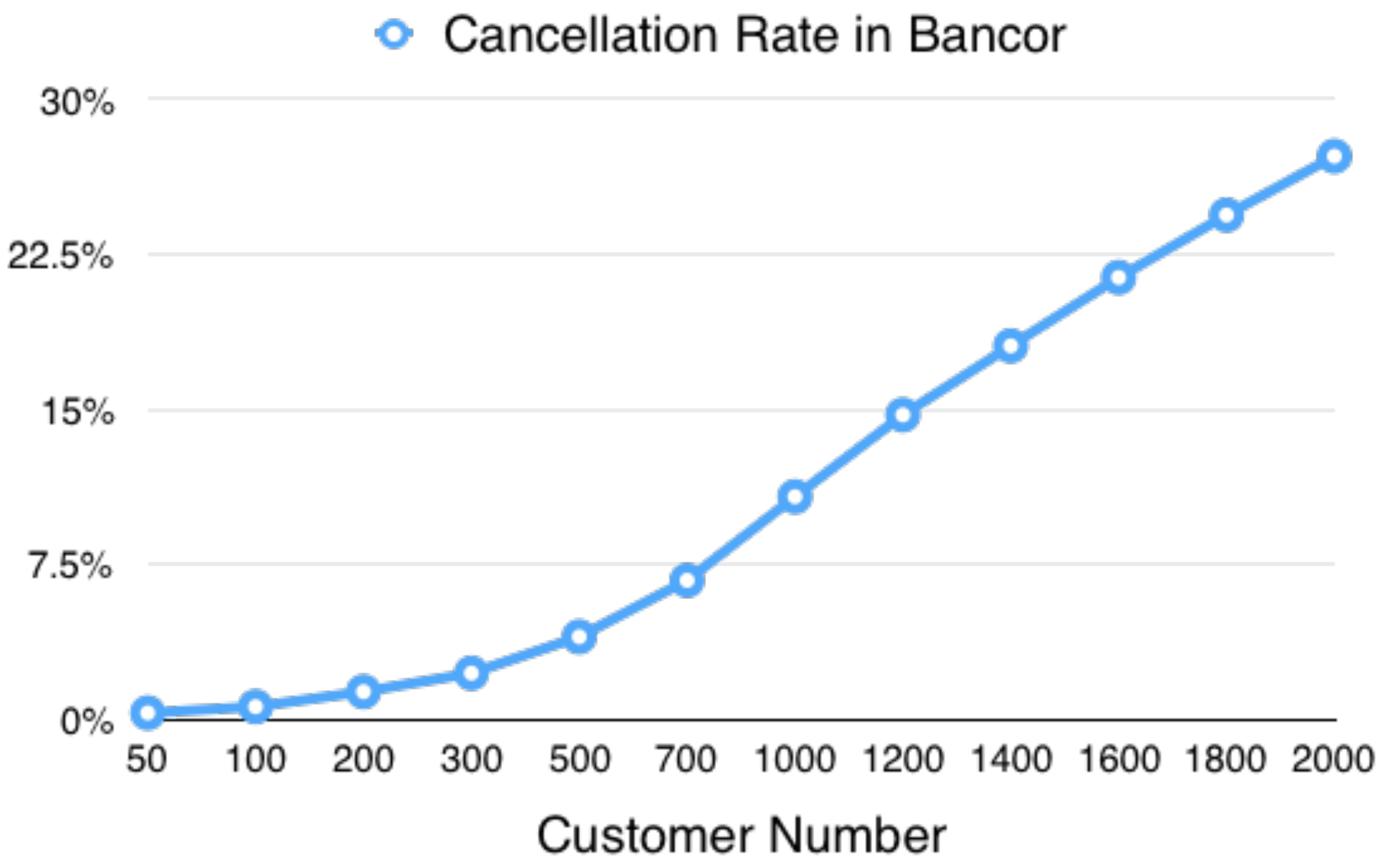
T = 50,
R = 2.0,
N=1000,
sig changes

Bancor Cancellation Rate when sigma changes:

With smaller sigma, the transactions' cancellation rate is much higher, which indicates more tightly customers make their valuations, more likely they need to cancel their transaction orders.

To view mathematical proof, see slides from annex 1 to annex 9.

Bancor Cancellation Rate when Customers' Number changes:



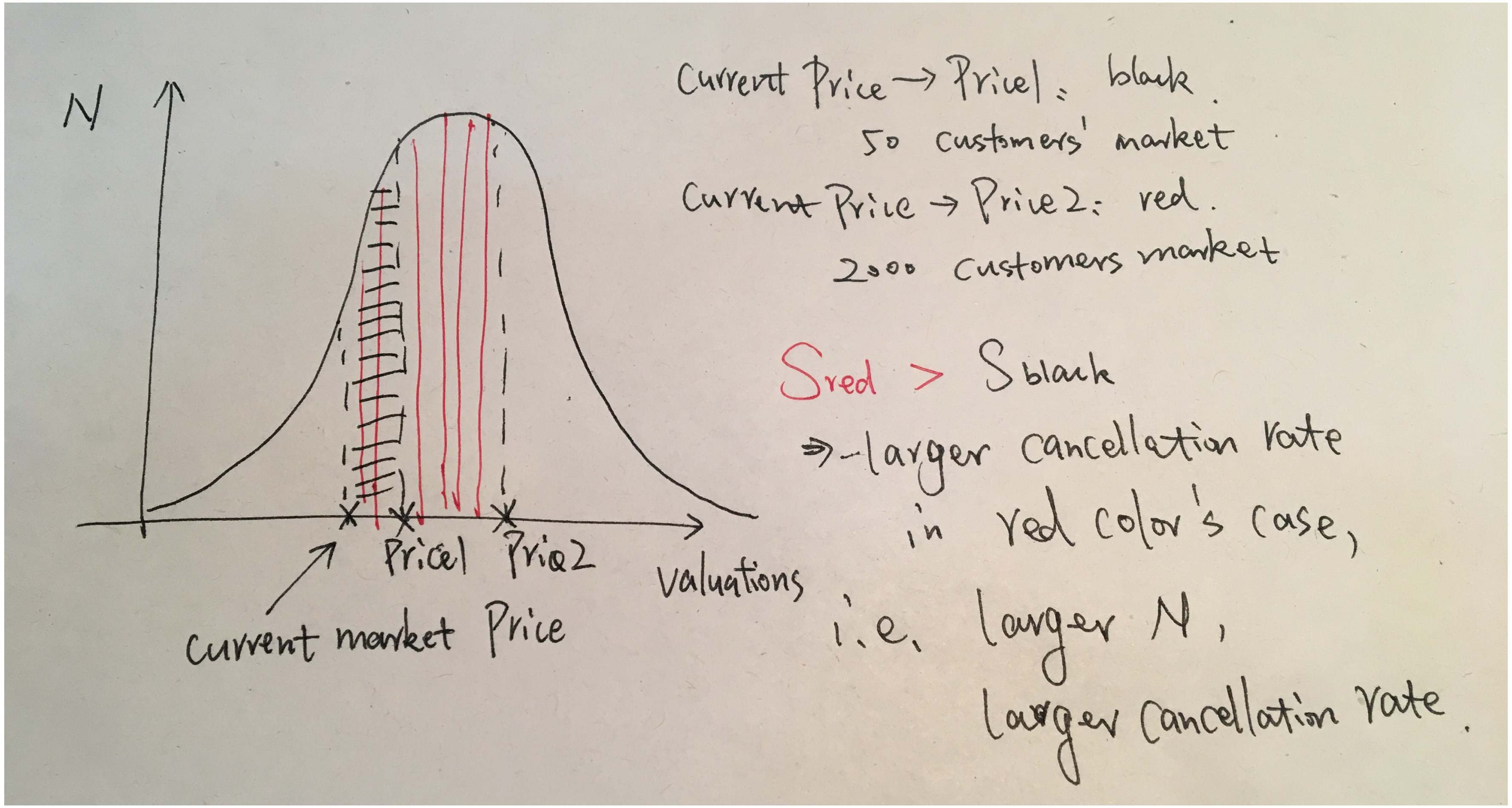
T = 50,
R = 2.0,
N changes
sig = 2.0

Bancor Cancellation Rate when Customers’ Number changes:

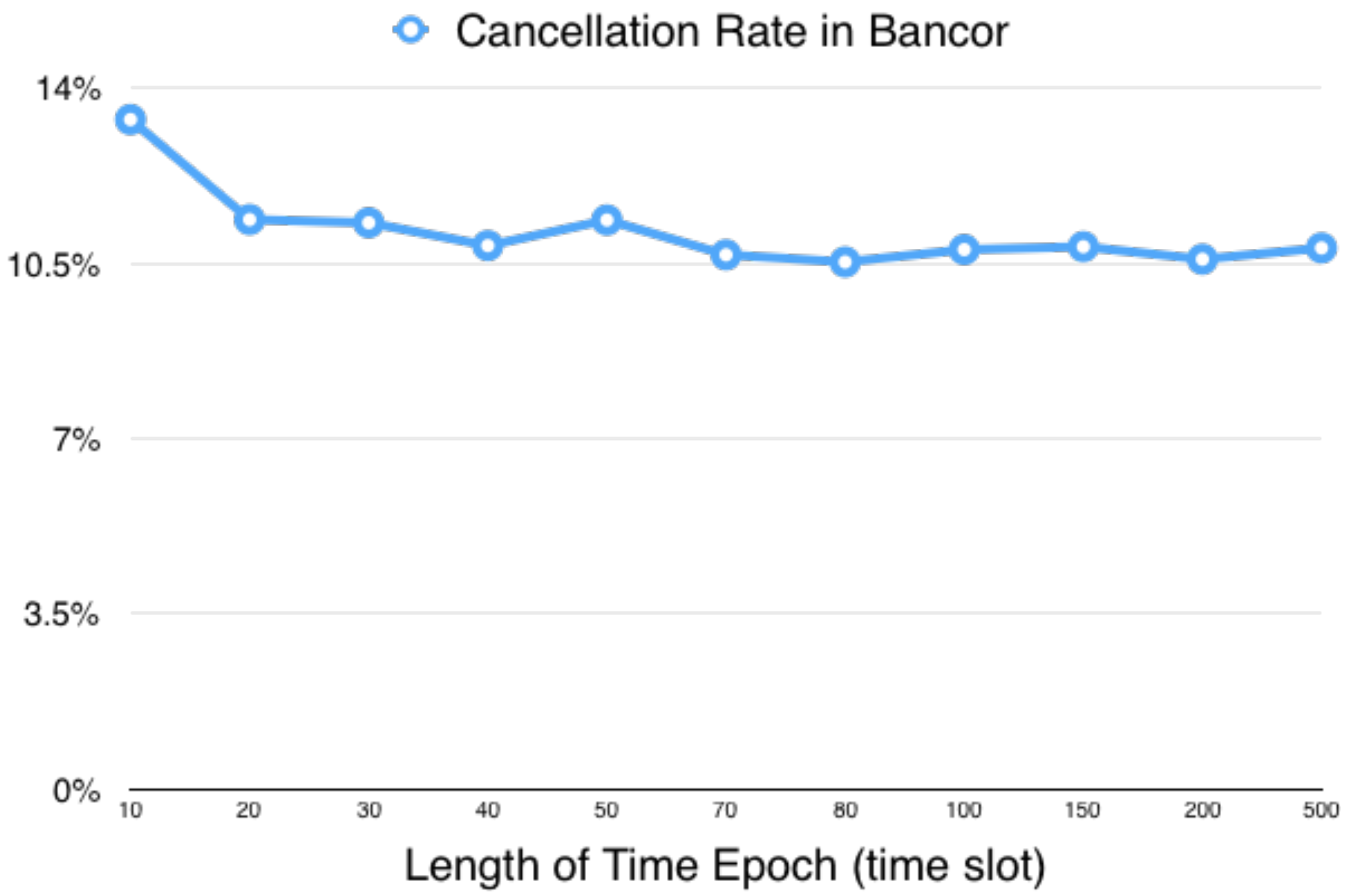
With larger customer number, the transactions’ cancellation rate is much higher.

The reason is that we set every customer initially have 200 reserve tokens and 200 smart tokens.
Assuming customers all buy smart tokens:
In 50 customers’ market, 200*50 reserve tokens are converted to smart tokens.
In 2000 customers’ market, 200*2000 reserve tokens are converted to smart tokens.
Therefore, the price of smart token fluctuates more fiercely in 2000 customers’ market.

Bancor Cancellation Rate when Customers' Number changes:

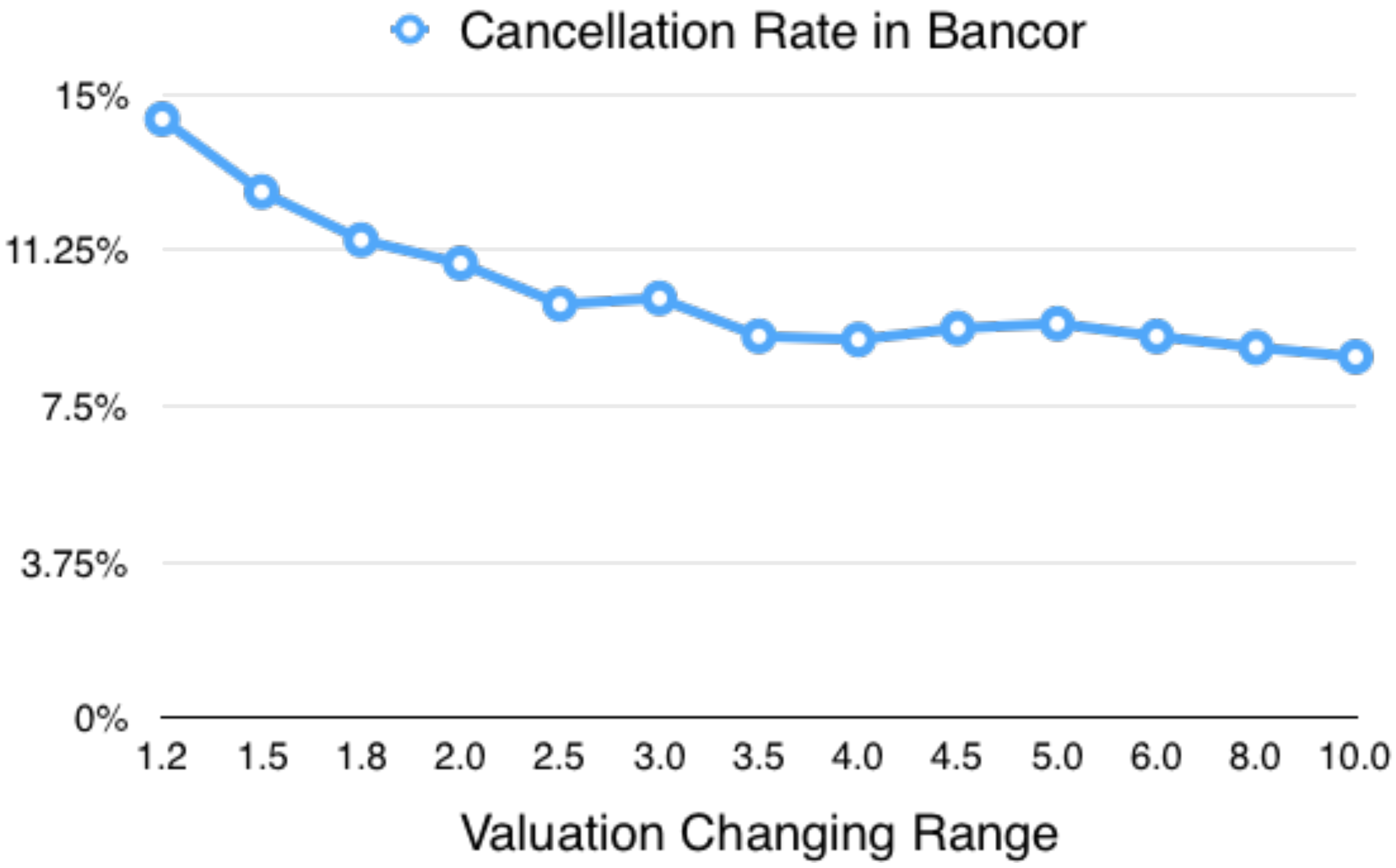


Bancor Cancellation Rate when Length of Time Epoch changes:



T changes,
R = 2.0,
N = 1000
sig = 2.0

Bancor Cancellation Rate when Bouncing Range changes:



T = 50,
R changes,
N = 1000
sig = 2.0

Classic Market:

The whole simulating time is comprised of **1000** time slots.

In every time slot, Classic Market processes the orders launched by **N** customers by managing the order list in the market.

Valuation Making -> Transaction Launching -> Transaction Processing

Valuation Making in Classic:

The valuation making in Classic Market is similar with Bancor Market.

However, since smart tokens in classic market are not created or destroyed, the price of the smart token is a constant.

Transaction Generating in Classic:

1~4 stipulations are similar with Bancor Market.

5. A customer will not launch new order if his order has not been fulfilled.

For instance, in a certain time slot, a customer launches a sell order at valuation 5 ETH to sell all of his 200 smart tokens.
However, in the next time slot, he finds that only 120 smart tokens have been sold. Therefore, he will not launch new order and continue to wait for his remaining 80 smart tokens being sold at valuation 5 ETH.
In the end of simulation, i.e., 1000 time slots have passed, if this transaction order is still unfinished in market, we say this order should be canceled.

Transaction Processing in Classic:

Classic Market manages an order list to process all transaction orders launched by customers.

In short, all transaction orders from customers will be separated into two sub-lists, named as sell list and buy list. In each list, orders will be sorted by the valuation of these orders.

Transaction Processing in Classic:

Transaction Processing in Classic:

Both Partially Failed orders and Totally Failed orders will be remained in order list and expect in the next time slot they can be finished.

However, if these orders are never finished -- after 1000 time slot, they still are remained in the market, we then say these orders should be canceled.

Thus, we calculate the **cancellation rate** by:

# of orders remained in market / # of all launched orders.

Comparison Between Bancor and Classic's Cancellation Rate :

The review of Cancel Rate:

In Bancor:

Why: The price of smart token cannot meet customer's valuation in order.

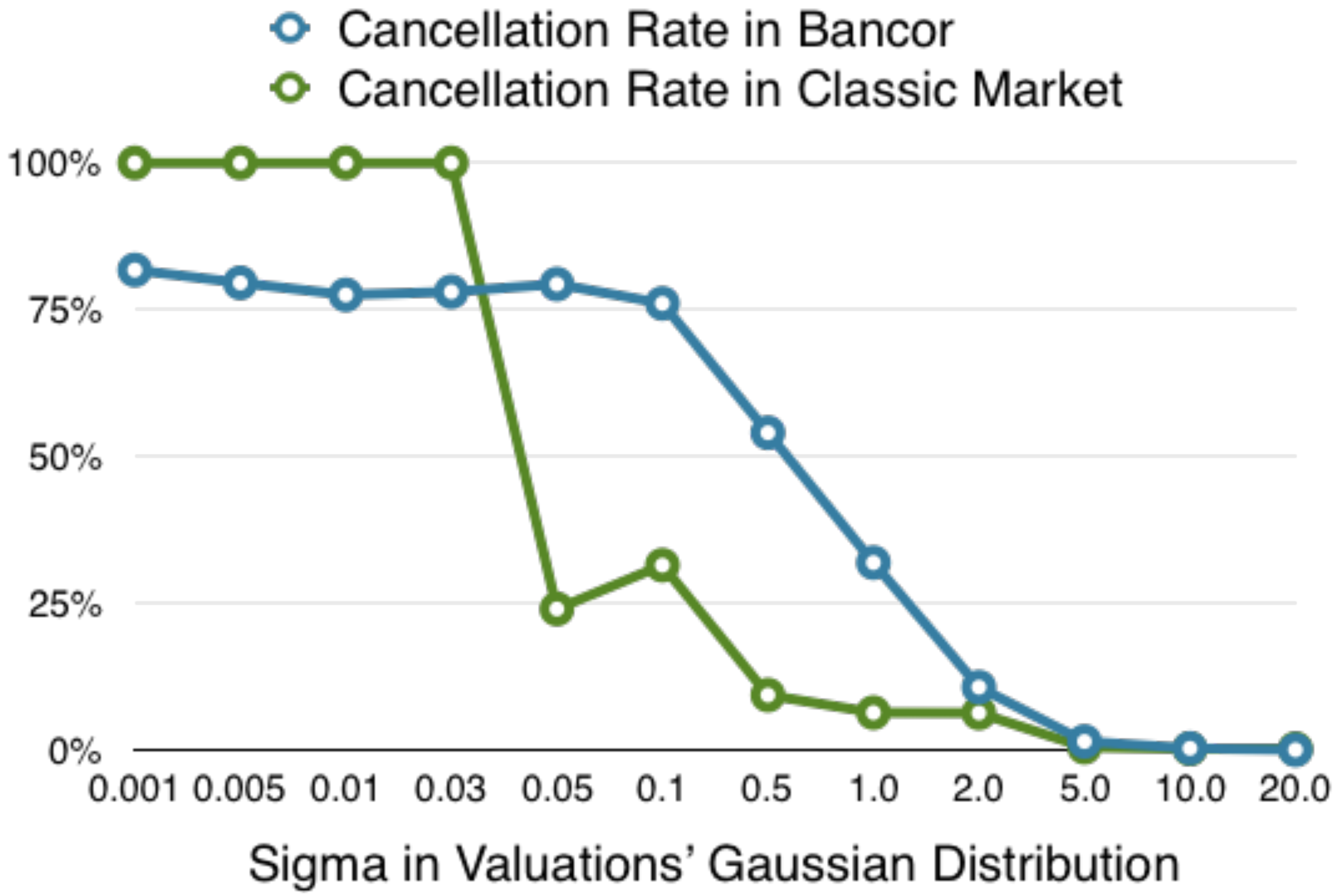
Cal: cancellation rate = # of all canceled orders / # of all launched orders

In Classic:

Why: Customer's order cannot be finished before the end of simulation.

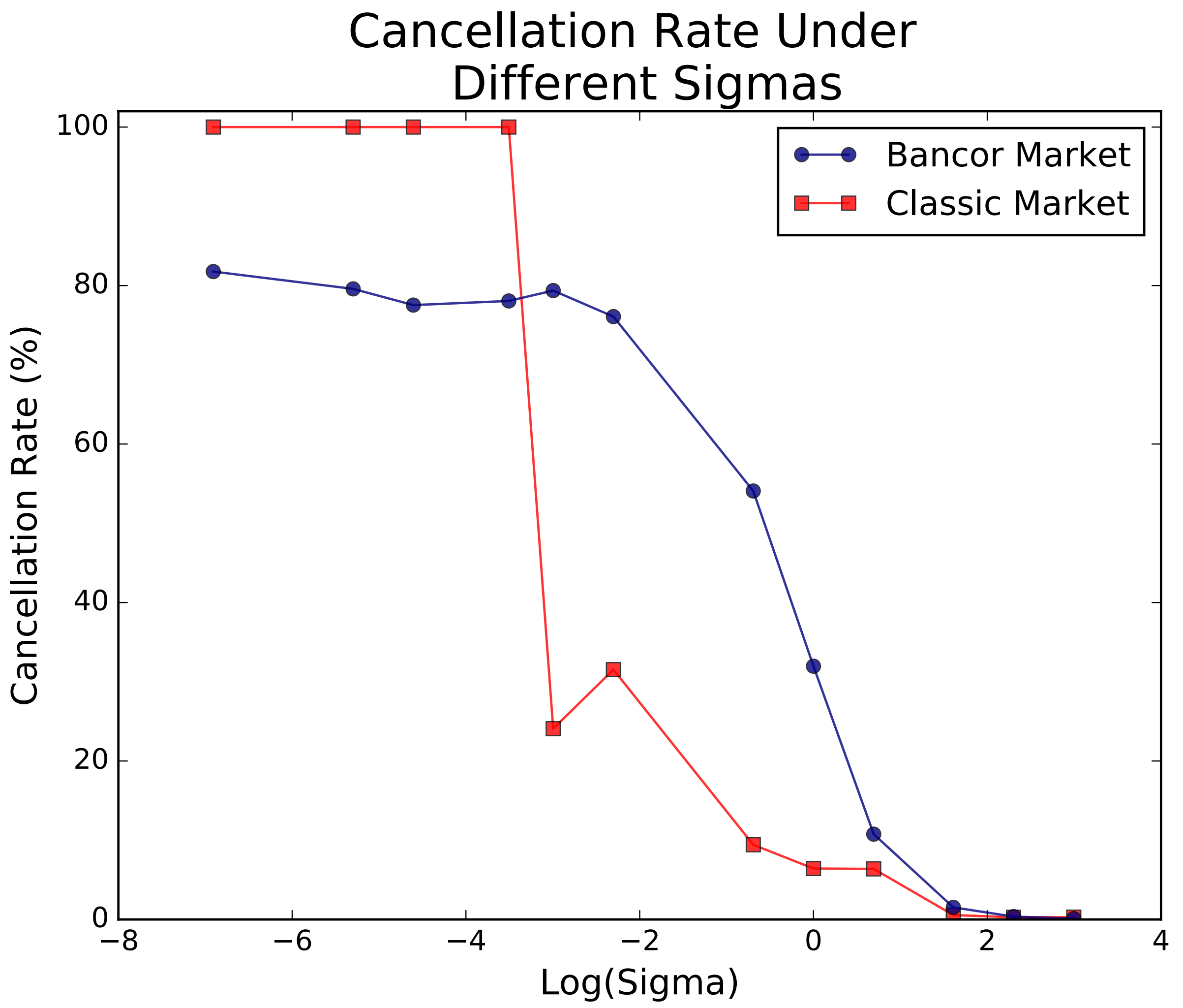
Cal: cancellation rate = # of orders unfinished / # of all launched orders.

Comparison between Bancor and Classic when sig changes:



T = 50,
R = 2.0,
N=1000,
sig changes

Comparison between Bancor and Classic when sig changes (log):



T = 50,
R = 2.0,
N=1000,
sig changes

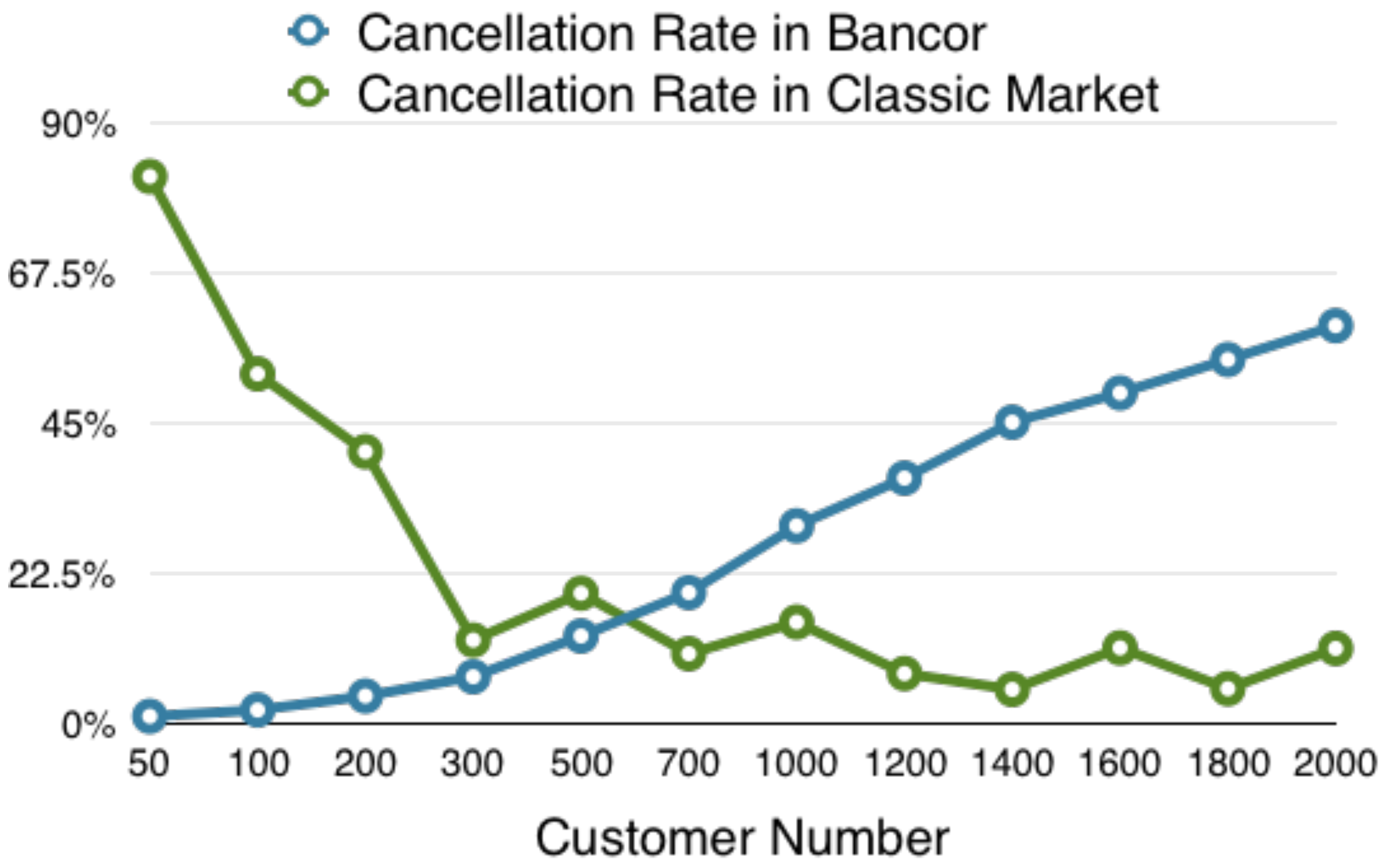
Comparison between Bancor and Classic when sig changes:

The reason for 100% cancellation rate is:
under small sigma such as 0.001, 0.005 and so on, every customer almost makes the same valuation.

For instance, every one makes a valuation at 20 ETH when the price of smart token is 15 ETH.
Due to our simulating rules – “larger to buy, smaller to sell”, every customer wants to buy smart tokens.
Then, all customers are stuck in market as no matched orders in market will release their launched orders.
In this case, the problem of “Co-incidence of Double wants” does exist.

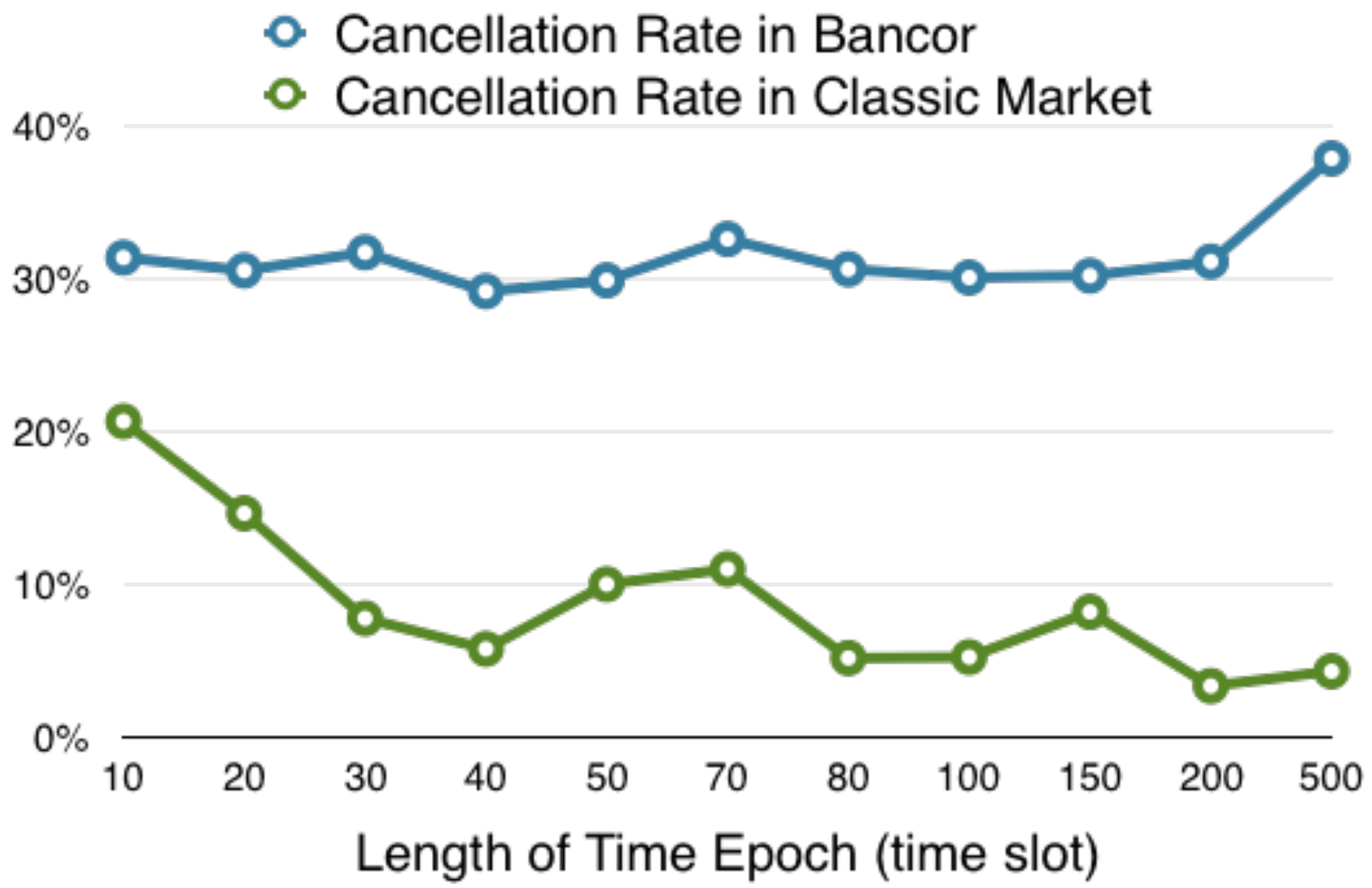
However, when sigma is large, 10 for instance.
When the price of smart token is 15 ETH, customers’ valuations can be 5 ETH, 18 ETH, 29 ETH and etc.
In this case, there are always buyers and sellers in the market.
Therefore, the cancellation rate in classic market is low.

Comparison between Bancor and Classic when N changes:



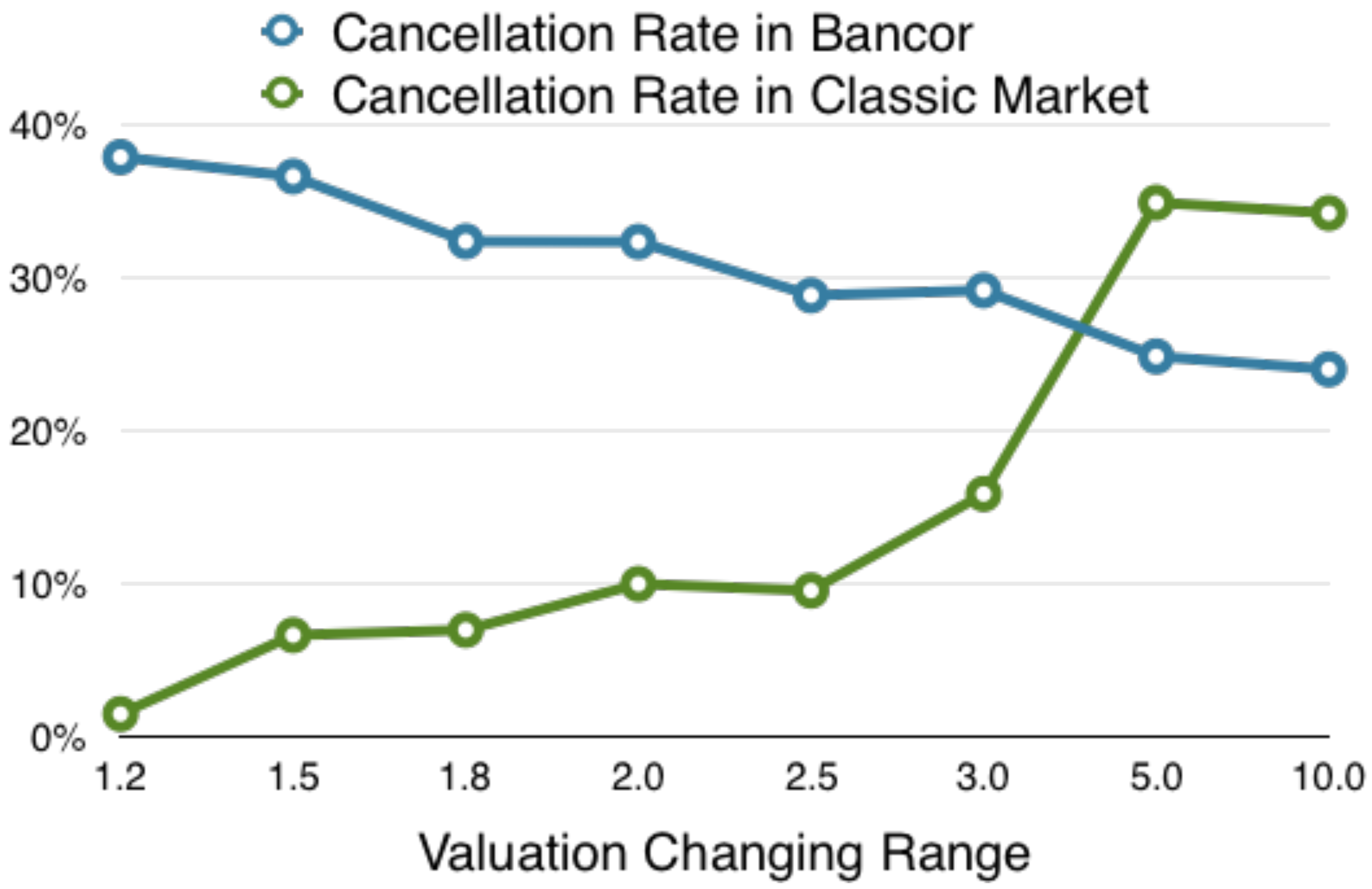
T = 50,
R = 2.0,
N changes
sig = 1.0

Comparison between Bancor and Classic when T changes:



T changes,
R = 2.0,
N = 1000
sig = 1.0

Comparison between Bancor and Classic when R changes:



T = 50,
R changes,
N = 1000
sig = 1.0

Summary:

1. Silde 11: The market craze or market crisis can lead to price of smart token bouncing fiercely between different time epochs.

2. Silde 13 & 14: the order's cancel rate in Bancor Market can be quite high -- reaching beyond 20% in several parameter settings, especially when customers' valuations are made tightly, i.e., sigma is small, and customers' number is large.

3. Slide 26: The low classic cancellation rate in Figure indicates the "co-incidence of double wants" might not be a problem in Classic Market when customers' valuations vary.

4. Slide 26 -- 31: The cancellation rate in Bancor Market in many cases is much higher than in Classic Market.

Summary:

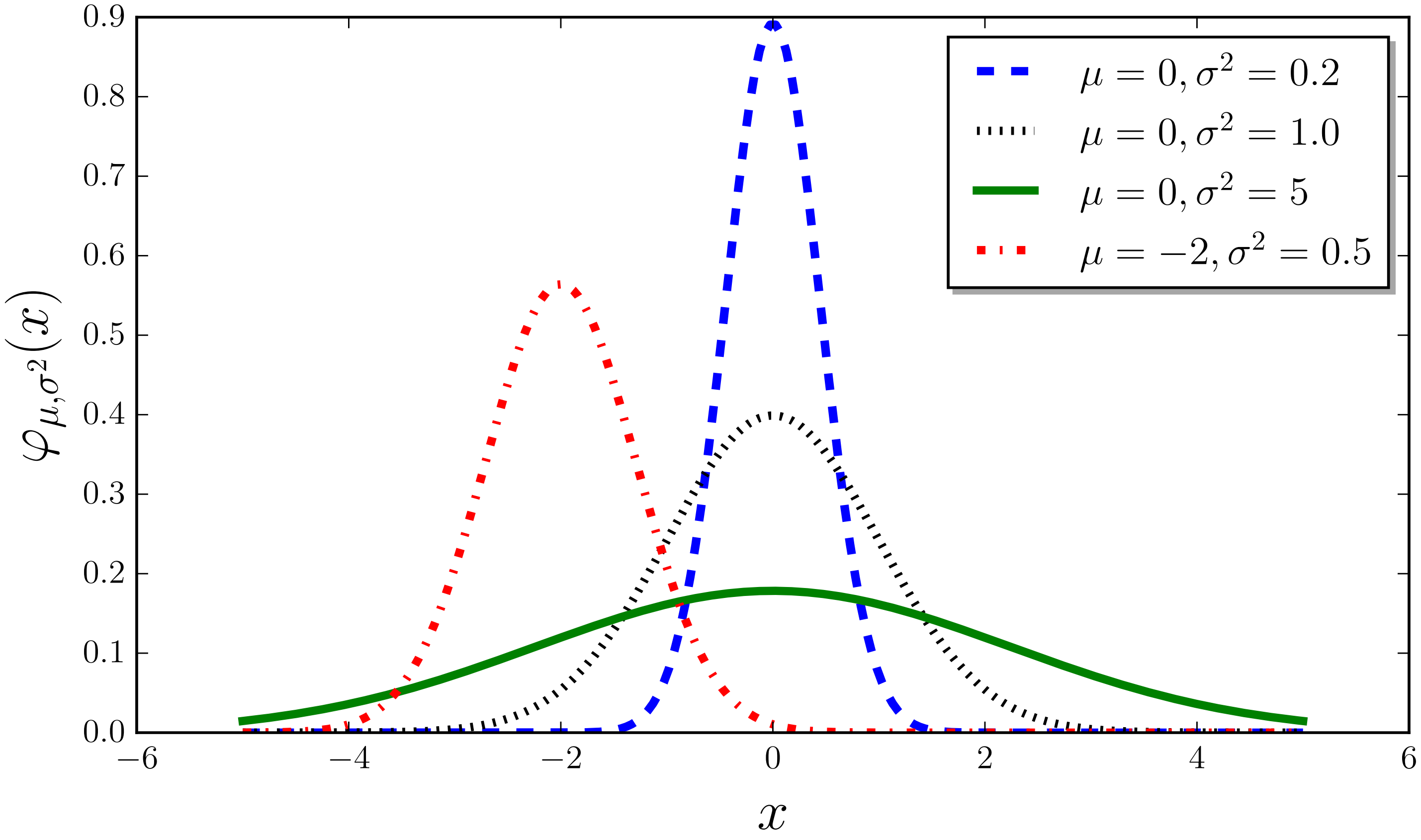
Recently, I have cleaned my code and added many comments.
They are presented on:
https://github.com/Ohyoukillkenny/Bancor-Simulator

Flaw and Future Work:

Up to now, what we have fully discussed in Bancor market is based on **limited order**.

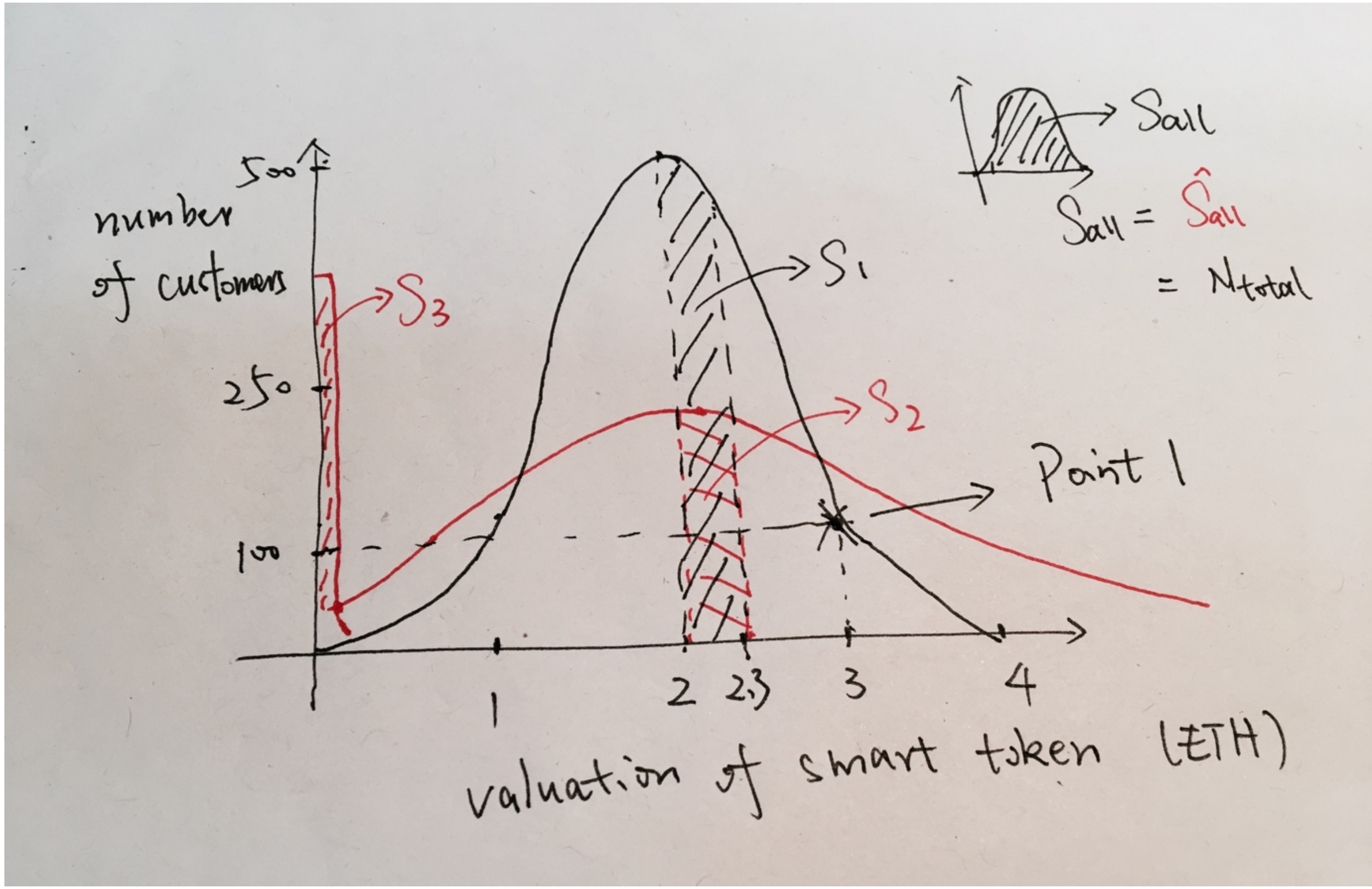
Now, I am still working on modifying our simulating model in order to make it feasible in more general cases.

Proof of smaller sigma, higher cancel rate:

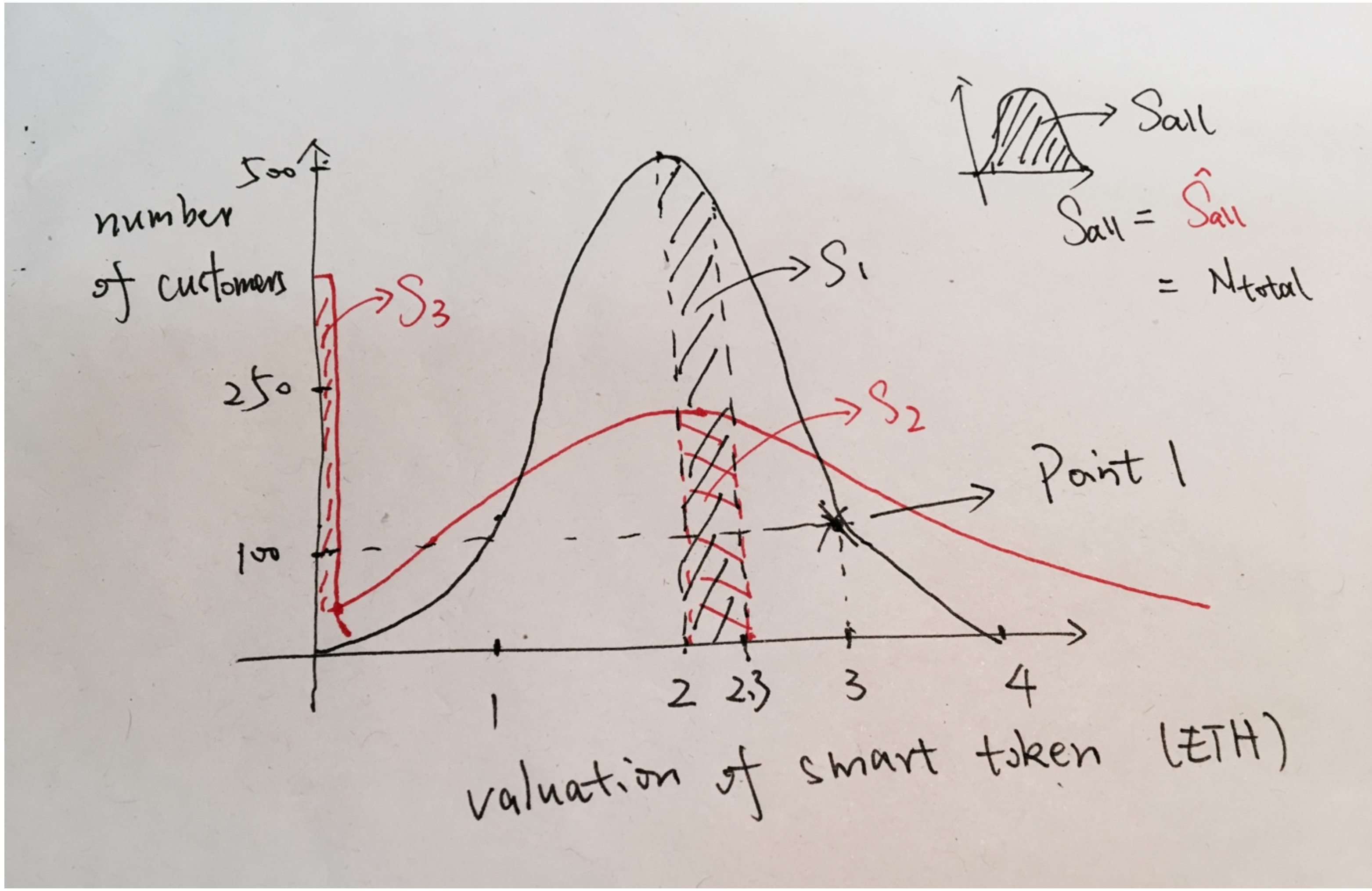


The Gaussian function in different sigma settings. Smaller the sigma is, steeper the Gaussian curve is.

Proof of smaller sigma, higher cancel rate:

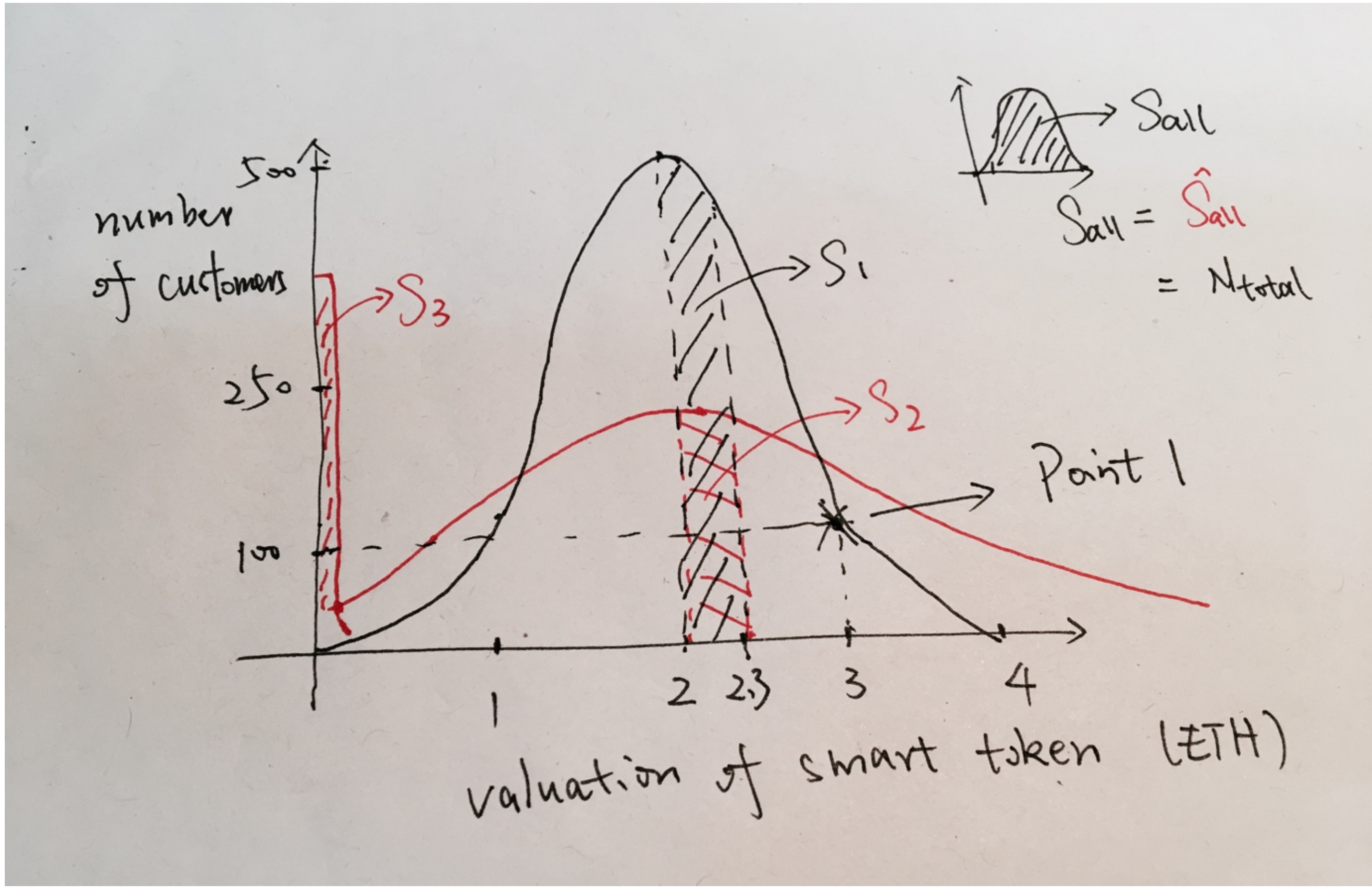


The black curve shows the valuation distribution with **sig1**, the red curve shows with **sig2**. By the previous slide, we know **sig2** > **sig1.** The mean valuation is 2.



The point 1 means there are 100 customers making valuation as 3 ETH.

Thus, by doing a simple calculus, like small plot in the right-top corner of Figure, we know that the total area rounded up by x-axis and Gaussian curve is the total number of customers.

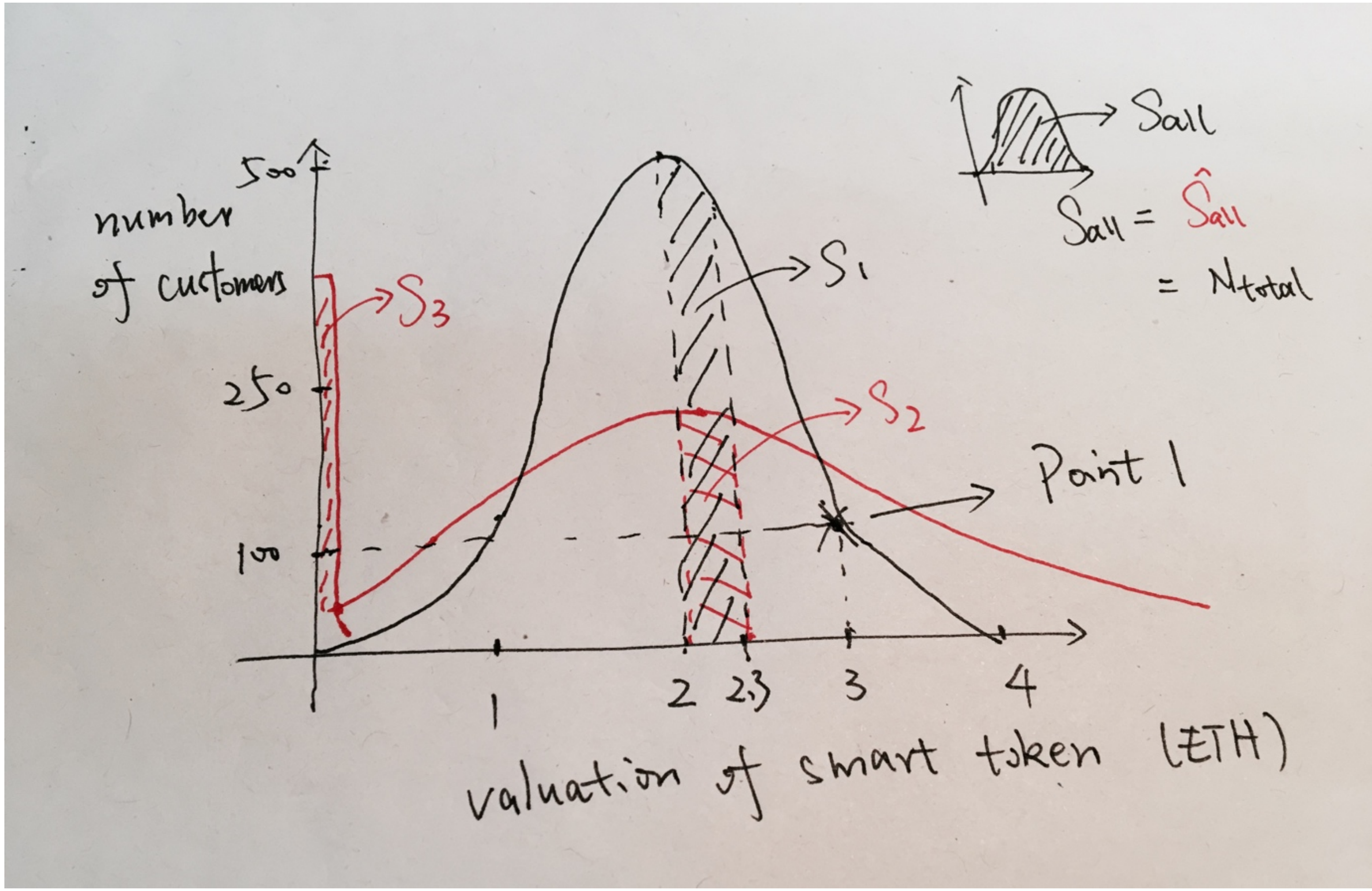


Since both in red curve and black curve, there are totally 2000 customers coming into market,

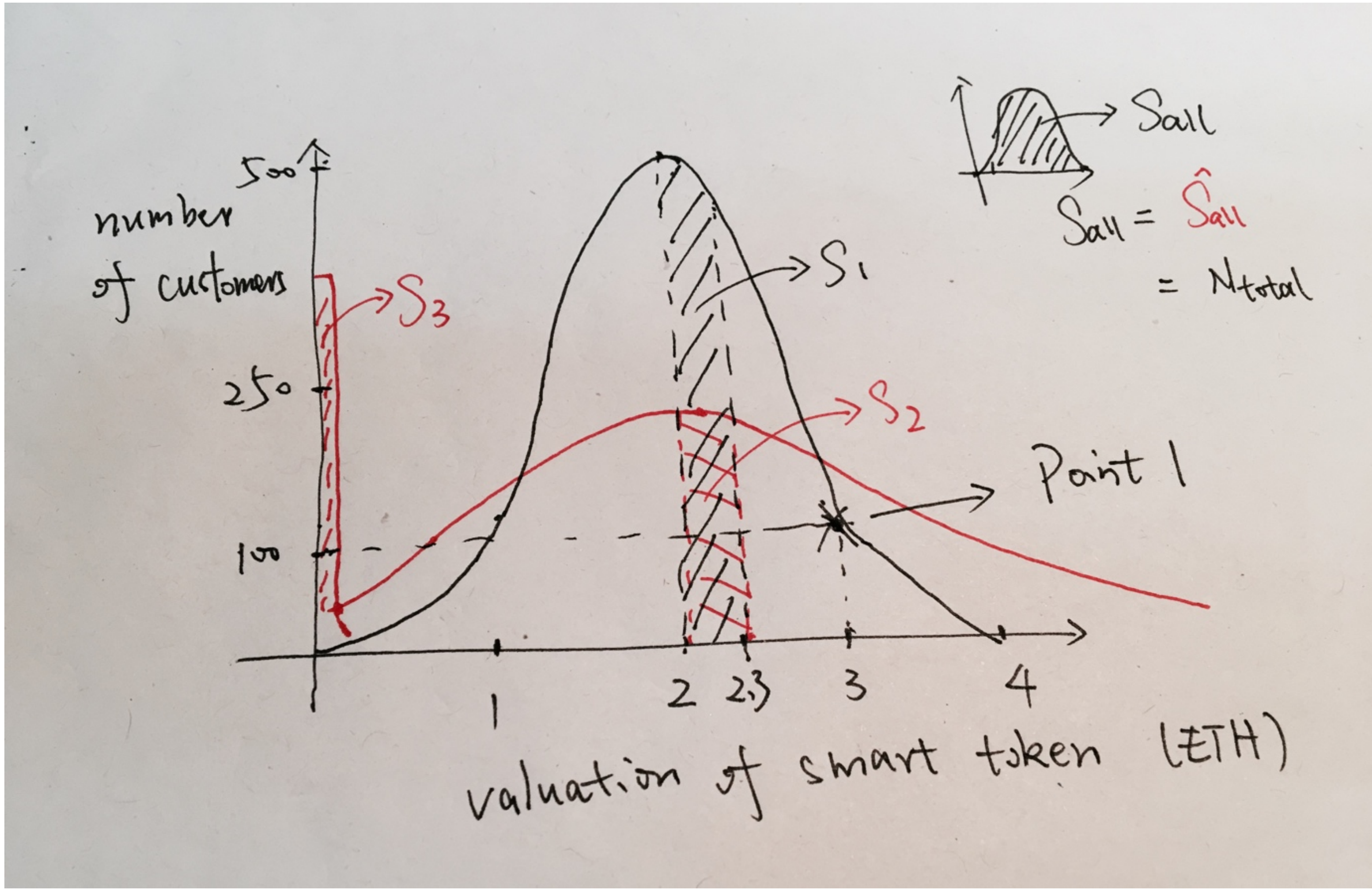
we know that:

S all = **S all** = **N total**

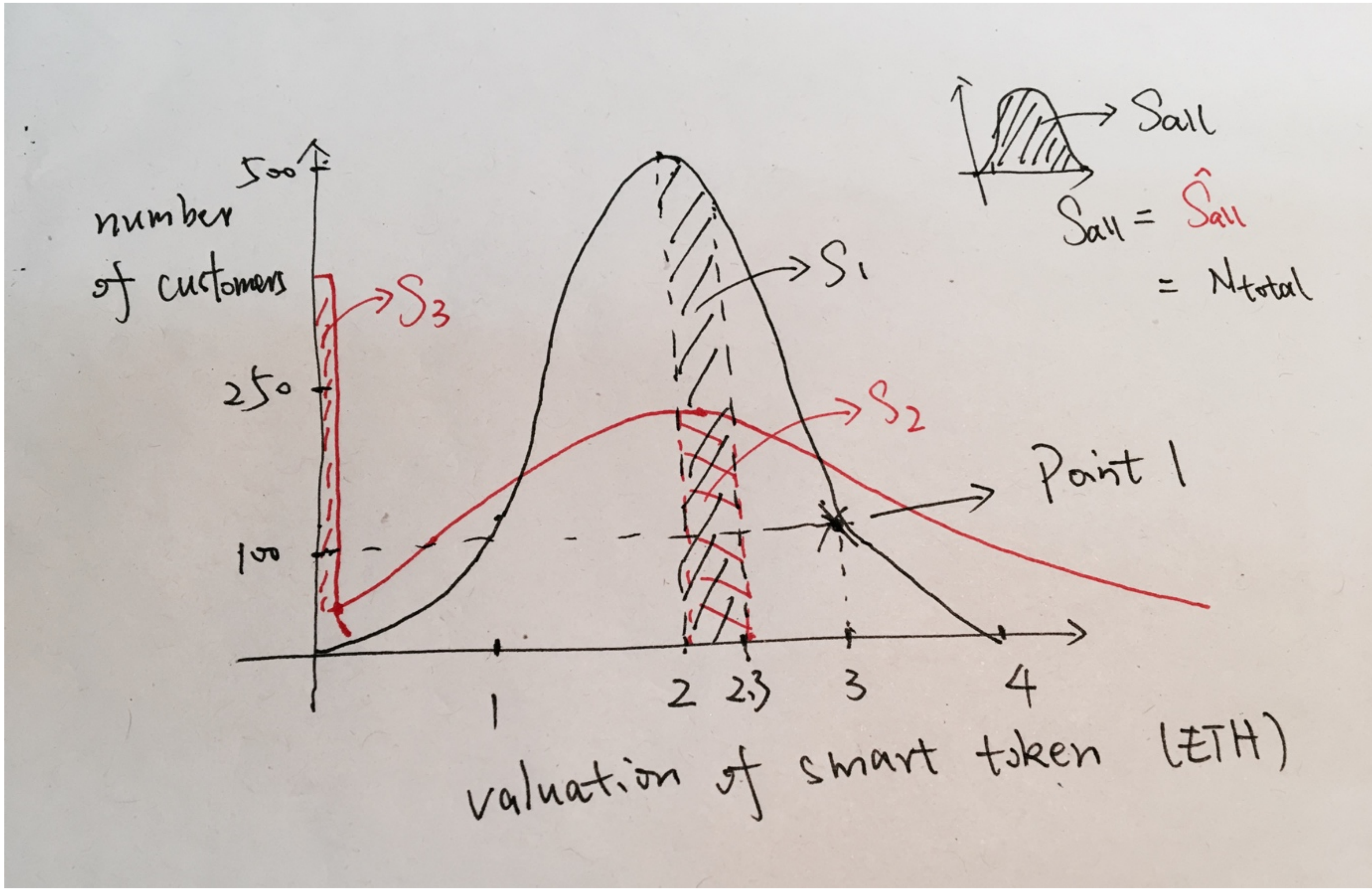
= 2000 (# of customers)



When one customer successfully making his transaction, the price of smart token in market will fluctuate, from 2 ETH to 2.3 ETH.



In this case, the **blacked shadowed area S1** or **the red shadowed area S2** presents the number of customers who now cannot make transactions. (buy-order valuation smaller than current price of smart token)



Hence, the current probability of customers' order being canceled in black-curve distribution:

Pr = S1/S all = S1/N total.

Similarly,

Pr = S2/S all = S2/ N total.

Apparently, **S1 > S2**.

Therefore, in current state,

Pr > Pr.

Proof of smaller sigma, higher cancel rate:

In fact, whether the price of smart token is increasing or decreasing, **Pr** is **always** larger than **Pr**. This indicates, **at every time**, the probability of transaction being canceled in Black curve Gaussian distribution (with **sig1**) is larger than it in red curve (with **sig2**).

Combining with the fact that **sig2** > **sig1**, the proof of smaller sigma causing higher cancel rate is done.

Proof of smaller sigma, higher cancel rate:

If you are careful enough, you might notice the weird S3 area.
This is because when the valuation by Gaussian function is smaller than 0, we set this valuation to be 0.001 * mean valuation (in our example is 0.002). Therefore, S3 actually equals with number of customers who generate valuation smaller than 0.

```
for i in range(custNum):
    if custValuation_list[i] < 0:
        # Customer does not want to sell their token in free.
        # Here we give them a small valuation when valuation < 0
        custList[i].changeValuation(0.001*currentMarketPrice)
    else:
        custList[i].changeValuation(custValuation_list[i])
```

Actually, S3 does not disturb the proof at all, since larger the S3 is, smaller the S2 is.