# 2 Competitive Auctions

Lesson objectives:

- Use what was learned about auctions to build a competitive bid function that attempts to maximize your revenue

Over the last few weeks we have been studying auctions, their applications and the math that powers them. In this case you will be using that knowledge to build a bidder in an attempt to maximize your own revenues. There are two major components of this assignment: the case below, which should be turned in normally, and an in-class auction exercise.

The purpose of this case is to build a bidding function, and then simulate a number of different situations. Before beginning, we'll outline the following data structures:

- **Price format:** The price format variable is an integer which is equal to the price that should be paid by the highest bidder upon winning (E.g. 2 for 2nd-price or 7 for 7th-price auction).
- **Auction format:** The auction format is one of two types: "SB" for sealed bid or "OO" for open-outcry.
- **Bid history:** The bid history is an ordered list of dictionaries. There are two types depending on if the auction format. In the case of a sealed bid auction, the history will look like the following:

  ```
  BH =  [{"Round": 1, "Value": None ,  "Bids": [1.25, 0, .85] }].
  ```

  In this case, the first bidder bid 1.25 for the item, the second bidder bid zero and the third bidder bid .85. When the auction type is "OO" the bid history should take the following form:

  ```
  BH = [
          { "Round" : 1, "Value": .25,  "Bids": [1, 1, 1]}
          , { "Round" : 2, "Value": .5,  "Bids": [1, 1, 0]}
          , { "Round" : 3, "Value": .75,  "Bids": [1, 0, 0]} ].
  ```

  In this case, the first round of the auction was held at a price of .25 and all 3 bidders decided to bid on. Since there was no winner a second round was held, this time at .5. Raising the price caused the third bidder to drop out, but as there were still two bidders another round was held. In the third and final round, which was held at .75 only the first bidder bid and thus won the item. If this was a second price auction, the winner would have paid .5 while, if this was a first price auction, the winner would pay .75.

- **Bidder information:** The bidder information is a list of the following format:

```
BI = [
        {"BidderName" : "Bidder 1", "BidFunction" : B1 }
        , {"BidderName" : "Bidder 2", "BidFunction" : B2 }
        , {"BidderName" : "Bidder 3", "BidFunction" : B3 }
        , {"BidderName" : "Bidder 4", "BidFunction" : B4 }
]
```

  The functions B1, B2, B3 and B4 are all bid functions, as discussed below.

- **Bid Functions:** Bid functions should take both a bid history and a current bid value as arguments; though both or either may not be used.

  In the case of the sealed bid auction, the function should return the bid of that bidder. Note that the bid function does not take the bidder's value of the item as an input, as that value should be part of the function. For example, in the sealed bid 2nd price auction that we've discussed in class, we have said that the bid function is equal to the value that the bidder has for the item. If a person has a value of $v = .3$, then their bid function would be:

```
def SP(currentBid, bidHistory):
        return .3
```

  If we wanted to initialize 10 number bidders, each with a value drawn from an independent, uniform [0,1] random variable, we could do the following:

```
from numpy import random
BidF = []
for rnd in random.uniform(0,1, size = [1,10])[0]:
        f = lambda currentBid, bidHist, rnd2=rnd: rnd2
        BidF.append( f )
```

  Or, in the case of the first price auction in a similar setting:

```
from numpy import random
BidF = []
for rnd in random.uniform(0,1, size = [1,10])[0]:
        f = lambda currentBid, bidHist, rnd2=rnd: 9.0 * rnd2 / 10.0
        BidF.append( f )
```

  In the case of the "OO" auction, the bid function should return a 1 or zero, depending on if the user wants to bid on that item. For example, the following function would only bid on an item if less than half of the possible bidders bid on the item in the last round and the current bid is less than 5:

```
def bh2(currentBid, bidHistory):
        if len( bidHistory ) > 1:
                ## Handle the first round which would have no bidHistory
```

```
                if 1.0*sum(bidHistory[-1]['Bids']) / len(bidHistory[-1]['Bids']) < .5
                                and currentBid < 5:
                        return 1
                else:
                        return 0
        else:
                ### If there is no bid history then don't bid.
                return 0
```

Some important facts about the bidding:

- Each bid list within the bid history should have the same number of items; bidders should not leave and reappear.
- Bidders are allowed to skip rounds when they bid. In other words, a bidder may abstain in round 3 (bid zero), but then decide to bid in round 4.
- An "OO" auction ends once there is only a single bidder on an item at a value. In the case of a second price open outcry, the bidder will pay the value on the previous round.
- In the case of an "OO" tie (where round "n" has zero bids, but round "n-1" has more than 1 bid), a random number is chosen determining which bidder wins the item. In this case, the bidder will pay the value on round "n-1".
- For the case of the "OO" auction, pretend that everyone yells their bid at the same time during each round, so that at the time of making a bid each bidder has the same information.

A helpful programming hint: Python is totally okay with having program names as values in key-value pairs and they can be accessed in a simple manner. For example, if bidF is defined as above the following code snippet will run as expected.

```
XDict = [ {"name" : "one", "bt": BidF[0]}]
print XDict[0]["bt"](3,2)
```

### Case

Please complete the following under the assumptions listed above.

1. First, we will need to build an auction evaluator. The auction evaluator runs the auction and determines if there is a winner or runs another round of the auction. The input of the auction evaluator is as follows:
   - Price Format
   - Auction Format

- Bid History (possibly empty)
- Bidder information
- Value to bid on (only used in the "OO" auction)
- Value to increment bid on (only used in the "OO" auction)

In the case of the sealed bid auction the function should exit after a single round while the open-outcry auction will require the function to call itself in a recursive manner.

In the case of an "OO" auction, the function should do the following

i Check to see if there is a winner in the case where there are no current bidders, but were bidders in the previous round

ii Pass the current bid history to each bidders bid function, with an incremented value to bid on. In other words, if the current bid passed is .5 and the increment value is .05 then the new current bid should be .55.

iii Create an updated bid history

iv Check to see if there is only a single bidder left at this bid value (is there a single winner?) and, if so, end. If there is no winner, then call the auction evaluation function again, this time with the updated bid history and updated current bid.

Test your function in the following situations and report the results:

(a) Sealed bid, second price auction with 10 bidders. The bidders have valuations equal to $0, .1, .2, .3, .., .9$ and bid according to the optimal, uniform auction bid function as derived in class. Who is the winning bidder and what do they pay? Note that there is no simulation or randomness in this example, the purpose is to test your auction function.

(b) Sealed bid, first price auction with 10 bidders. The bidders have valuations equal to $0, .1, .2, .3, .., .9$ and bid according to the optimal, uniform auction bid function as derived in class. Who is the winning bidder and what do they pay? Note that there is no simulation or randomness in this example, the purpose is to test your auction function.

(c) Using the same value distribution as above, run a second price "OO" auction, with the bidding starting at 0 and incrementing by .05. Each bidder behaves by bidding only if the current bid is less than their value. Who is the winning bidder and what did they pay? How many rounds of the auction were needed?

2. Answer the following:[4]

   - Using your auction evaluator, estimate the expected revenue of a second price auction, where values are drawn from an exponential distribution with parameters $\lambda = \{1, 10\}$. Complete the analysis for 2, 5 and 10 bidders.
   - In class we derived the following expression for the equilibrium bid function in a first price auction:

     $$b^I(v) = v - \int_0^v \frac{F^{n-1}(x)dx}{F^{n-1}(v)}$$

     Using this expression, solve for the equilibrium bid function of the first price auction with an exponential distribution with with parameters $\lambda = \{1, 10\}$ and 2, 5 and 10 bidders. Feel free to use mathematical software, such as Mathematica's free online integrator to compute each one.
   - Using the previously computed bid function, estimate the expected value of the first price auction under $\lambda = \{1, 10\}$ and 2, 5 and 10 bidders.

## In-Class Auction

We will be running an in class auction and it is up to you to build a bidder to beat your classmates. The format of the auction is a bit special in that we'll be using the second-price open outcry format described above. In particular:

- The value of the item is $\mu$, which is unknown to each team. $\mu$ will be drawn from a mean zero, unit standard deviation normal random variable.
- Each team will receive a unique number, which represents a signal of the value of the item, which was drawn from a multivariate normal distribution with mean $\mu$. The variance covariance matrix on this distribution is the identity matrix.
- I'll begin the bidding at a small number and you'll put the bid value into your bid function with an empty bid history. Each team will then tell me if they have decided to bid or not at that value (1/0).
- Assuming that there are no winners we'll increment the bid function upwards and re-do the bidding until there is a winner.
- You will know the entire bid history for each item.

[4] Estimate does not mean "run once." It means use simulation to get multiple values and average them together. For each, please report the number of times you decided to simulate each and why you choose that number.

Each group will be evaluated based on the profit generated by their bidding function. We will (hopefully) do enough of them to get a good sense of which of your algorithm worked best. For the in-class assignment, if you simply bid b(v) = your signal signal, you will receive 75% of the credit. Bidding your signal isn't optimal, your goal is to try to find something that is performant.

For the in-class assignment please turn in a short ($<$ 10 slides) presentation describing the algorithm that you used, how you tested it and what you think that its strengths and weaknesses are.