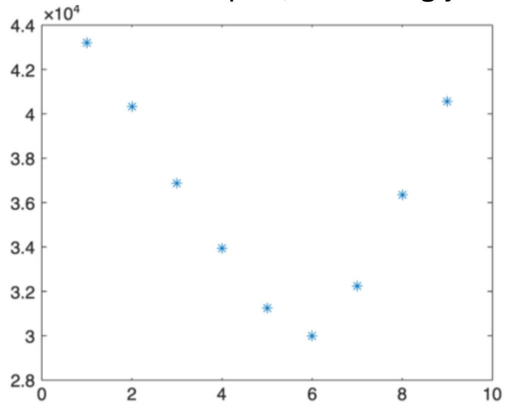
Goals: Our goal is to optimize the pool size in order to minimize expected cost for the school district.
Assumptions: We are assuming that the distribution of the data is uniform. We are also assuming that there cannot be a fractional number of substitute teachers in the pool. Another assumption is that if the need for substitutes exceeds the size of the pool, classes can be covered by regular teachers, but at a higher pay rate.
Methodology: Programming is used to run numerous simulations of each pool size in order to the plot the cost and keep limiting the range to better optimize the pool sizes. This requires many iterations while increasing the number of simulations until a smooth plot is achieved. Once a smooth plot is shown, the minimum is taken and the range is adjusted around this found minimum value.

MATLAB Code for Project 4 in Section 5.5 Part C:

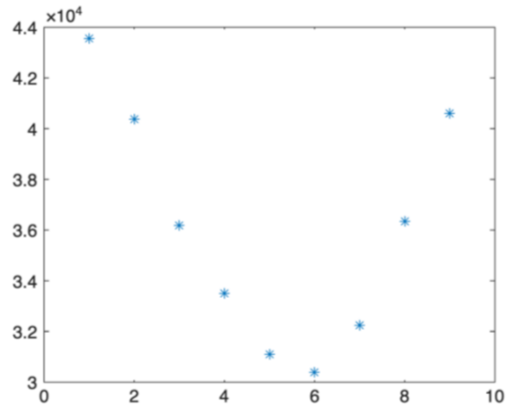
```
% project #4 in section 5.5 part c
rng(0) % to set the seed of random numbers


h=75; % interval width
a=201:h:876; % the left end point of each interval
b=275:h:950; % the right end point of each interval


% relative and cumulative percentages
rp=[2.5 2.5 5 7.5 12.5 17.5 42.5 5 2.5 2.5]/100';
cp=[2.5 5 10 17.5 30 47.5 90 95 97.5 100]/100'; % height
n=length(cp); % the number of intervals


% calculate the slopes
m(1)=cp(1)/(h/2); % first piece
m(n)=(cp(n)-cp(n-1)/(1.5*h)); % last piece
for k=2:n-1
   m(k)=(cp(k)-cp(k-1))/h;
end


% run the simulation for different pool size S for N times
% use the avg cost as the cost for that pool size
N=1000; % number of simulations for each pool size S


Cost=[]; % a vector of costs at different
p=45; % pay rate for subs
r=81; % pay rate for regular overtime


Sk=1; % for the # of entries in cost
for S=100:100:900
   C=0; % cost for each S
   for k=1:N
```

```
        rd=rand; % each rand # will produce demand x by inverse splines
          if rd<cp(1), x0=(a(1)+b(1))/2; y0=cp(1); x=x0+(rd-y0)/m(1);
          elseif rd<cp(2), x0=(a(2)+b(2))/2; y0=cp(2); x=x0+(rd-y0)/m(2);
          elseif rd<cp(3), x0=(a(3)+b(3))/2; y0=cp(3); x=x0+(rd-y0)/m(3);
          elseif rd<cp(4), x0=(a(4)+b(4))/2; y0=cp(4); x=x0+(rd-y0)/m(4);
          elseif rd<cp(5), x0=(a(5)+b(5))/2; y0=cp(5); x=x0+(rd-y0)/m(5);
          elseif rd<cp(6), x0=(a(6)+b(6))/2; y0=cp(6); x=x0+(rd-y0)/m(6);
          elseif rd<cp(7), x0=(a(7)+b(7))/2; y0=cp(7); x=x0+(rd-y0)/m(7);
          elseif rd<cp(8), x0=(a(8)+b(8))/2; y0=cp(8); x=x0+(rd-y0)/m(8);
          elseif rd<cp(9), x0=(a(9)+b(9))/2; y0=cp(9); x=x0+(rd-y0)/m(9);
          else x0=b(10); y0=cp(10); x=x0+(rd-y0)/m(10);
          end
        if x<S, Cxs=p*S; % cost for that demand
        else Cxs=p*S+(x-S)*r;
        end
      C=C+Cxs; % add up the cost for each simulation
      end
      Cost(Sk)=C/N; % use the average as the cost for that S
      Sk=Sk+1;
    end
    plot(Cost,'*')
```

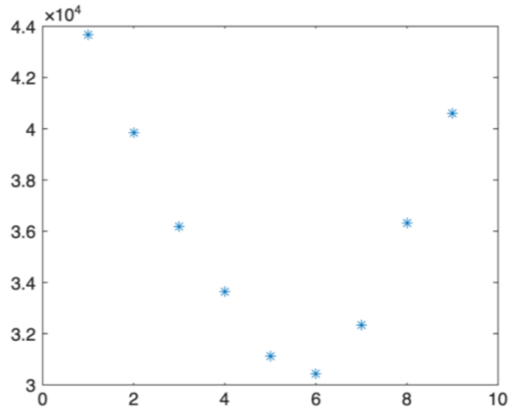
Now we change a line of code so that S=100:100:900. We also change the N values to receive a smooth plot, accordingly. For N=1000 the plot is:



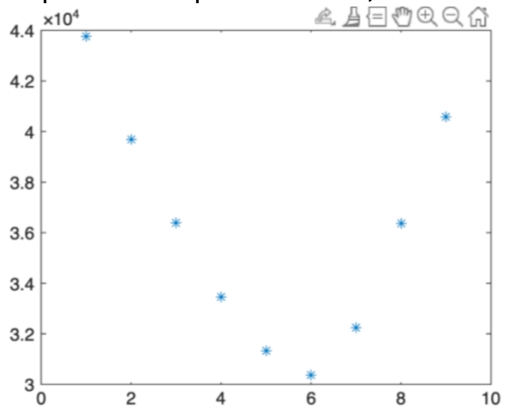
Now for N=2000:



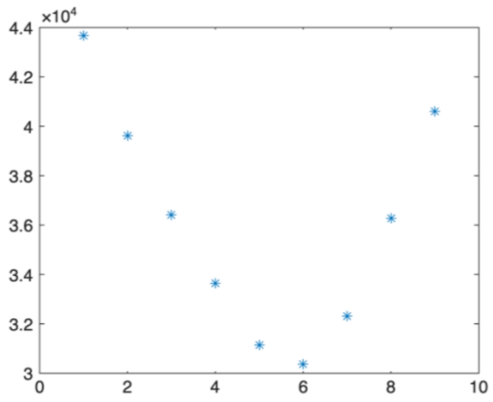
And N=3000:



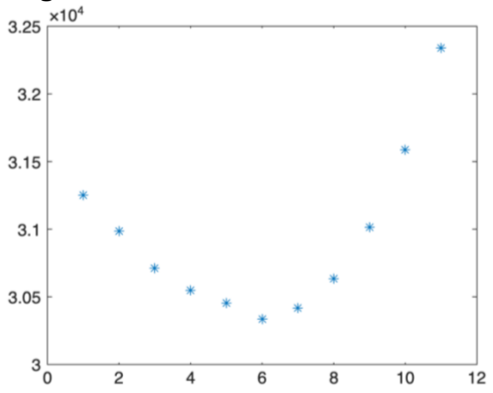
The plot looks quite smooth, but to be safe, for N=4000 the plot is:



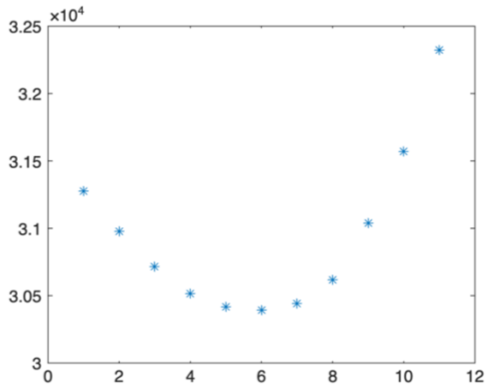
Finally, for N=5000:



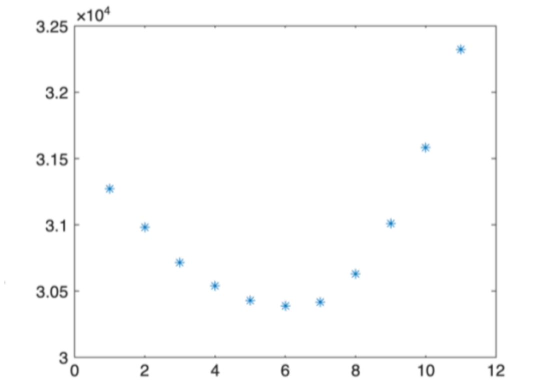
Since this plot is very smooth, we can stop here. The minimum of this graph is at X=6, or S=600. Thus, we can shift the values of S to reflect where the optimal pool size will fall. Let us change the width to 200, with intervals f=of 20. So, now S=500:20:700. Now that the range is smaller, we will start with a higher value of N. For N=50000, the plot is:



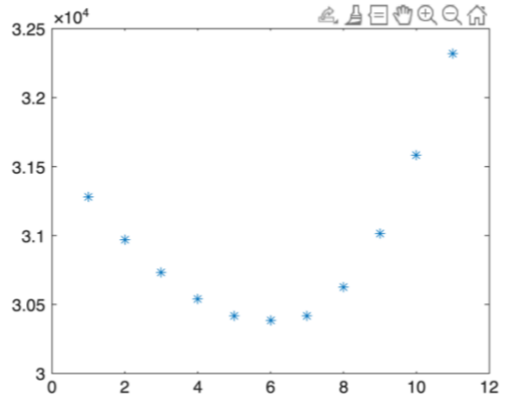
Now, for N=100000:



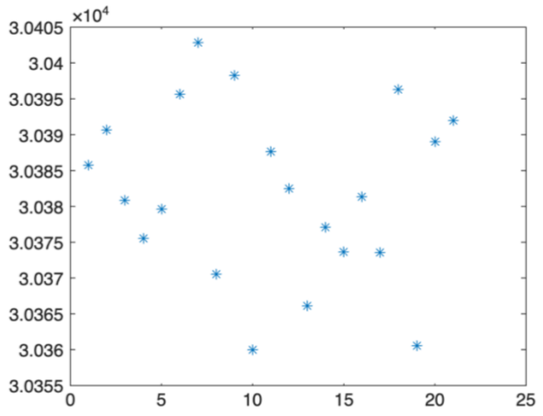
The plot is almost smooth, so now to increase to N=150000:



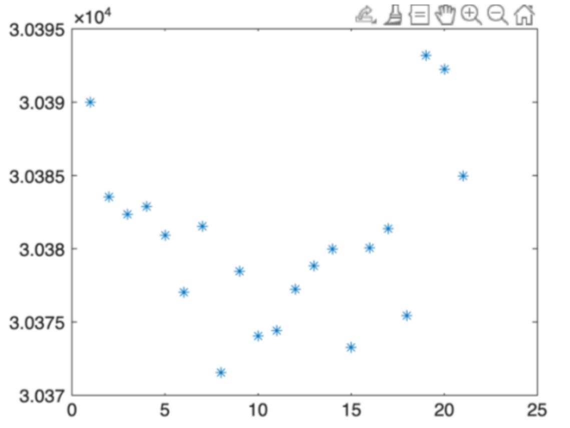
To be sure that the minimum is true, as the values on either side are quite close, compute N=200000:



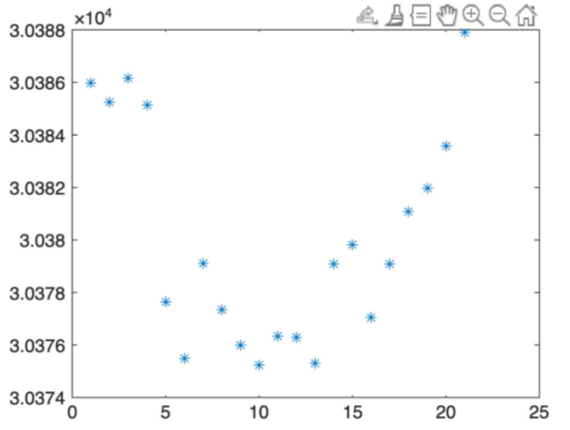
This is now totally smooth and accurate. Again, the minimum is occurring at X=6, or S=600. We now need to further constrict the range to 20 at intervals of one. We now have that S=590:1:610. Again, we need a fairly large amount of simulations so we will start with N=100000, where the plot looks like:



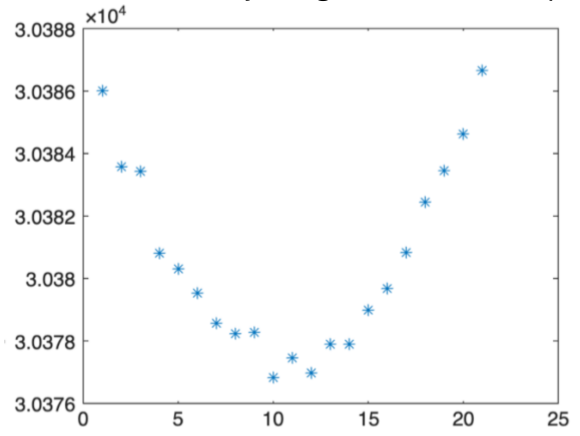
This plot is very far from smooth, so increase the number of simulations by a good amount. For N=1000000, the plot is:



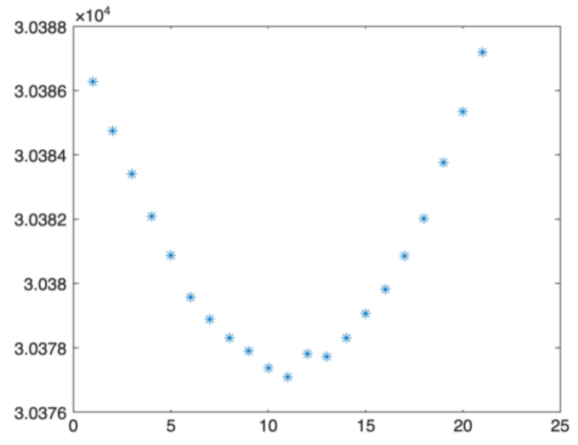
While smoother, this is still insufficient. For N=10000000:



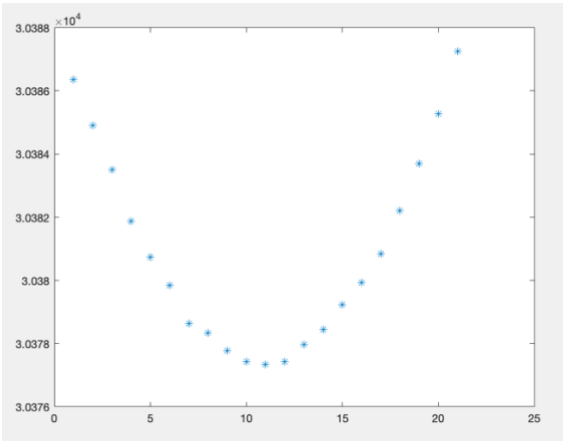
This still has a ways to go, so let us compute N=100000000:



This looks much smoother, but to be sure, compute N=1000000000:



It is still not quite there, let us compute N=1500000000:

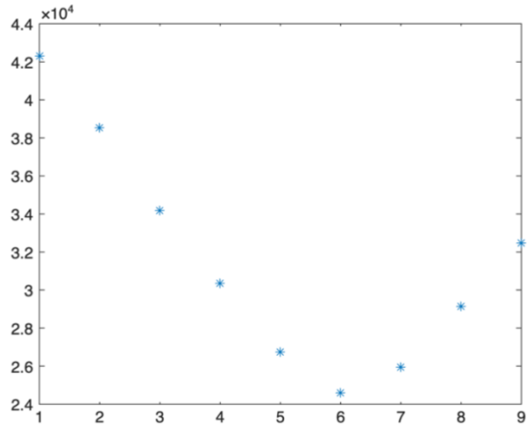


This is quite smooth, where the minimum occurs at X=11, so S=600. Thus, a pool size of 600 is the optimal pool size for a pay rate of $45 for substitutes and an overtime rate of $81.

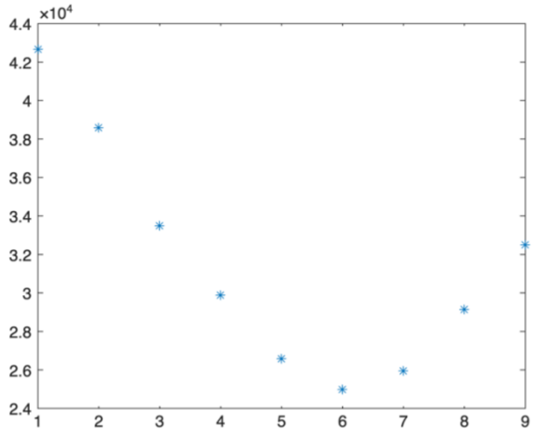
MATLAB Code for Project 4 in Section 5.5 Part D:

```
% project #4 in section 5.5 part d
rng(0) % to set the seed of random numbers

h=75; % interval width
a=201:h:876; % the left end point of each interval
b=275:h:950; % the right end point of each interval

% relative and cumulative percentages
rp=[2.5 2.5 5 7.5 12.5 17.5 42.5 5 2.5 2.5]/100';
cp=[2.5 5 10 17.5 30 47.5 90 95 97.5 100]/100'; % height
n=length(cp); % the number of intervals

% calculate the slopes
m(1)=cp(1)/(h/2); % first piece
m(n)=(cp(n)-cp(n-1)/(1.5*h)); % last piece
for k=2:n-1
    m(k)=(cp(k)-cp(k-1))/h;
end

% run the simulation for different pool size S for N times
% use the avg cost as the cost for that pool size
N=1000; % number of simulations for each pool size S

Cost=[]; % a vector of costs at different
p=36; % pay rate for subs
r=81; % pay rate for regular overtime

Sk=1; % for the # of entries in cost
for S=100:100:900
    C=0; % cost for each S
    for k=1:N
        rd=rand; % each rand # will produce demand x by inverse splines
            if rd<cp(1), x0=(a(1)+b(1))/2; y0=cp(1); x=x0+(rd-y0)/m(1);
            elseif rd<cp(2), x0=(a(2)+b(2))/2; y0=cp(2); x=x0+(rd-y0)/m(2);
            elseif rd<cp(3), x0=(a(3)+b(3))/2; y0=cp(3); x=x0+(rd-y0)/m(3);
            elseif rd<cp(4), x0=(a(4)+b(4))/2; y0=cp(4); x=x0+(rd-y0)/m(4);
            elseif rd<cp(5), x0=(a(5)+b(5))/2; y0=cp(5); x=x0+(rd-y0)/m(5);
            elseif rd<cp(6), x0=(a(6)+b(6))/2; y0=cp(6); x=x0+(rd-y0)/m(6);
            elseif rd<cp(7), x0=(a(7)+b(7))/2; y0=cp(7); x=x0+(rd-y0)/m(7);
            elseif rd<cp(8), x0=(a(8)+b(8))/2; y0=cp(8); x=x0+(rd-y0)/m(8);
            elseif rd<cp(9), x0=(a(9)+b(9))/2; y0=cp(9); x=x0+(rd-y0)/m(9);
            else x0=b(10); y0=cp(10); x=x0+(rd-y0)/m(10);
            end
    if x<S, Cxs=p*S; % cost for that demand
    else Cxs=p*S+(x-S)*r;
    end
  C=C+Cxs; % add up the cost for each simulation
  end
  Cost(Sk)=C/N; % use the average as the cost for that S
  Sk=Sk+1;
end
plot(Cost,'*')
```

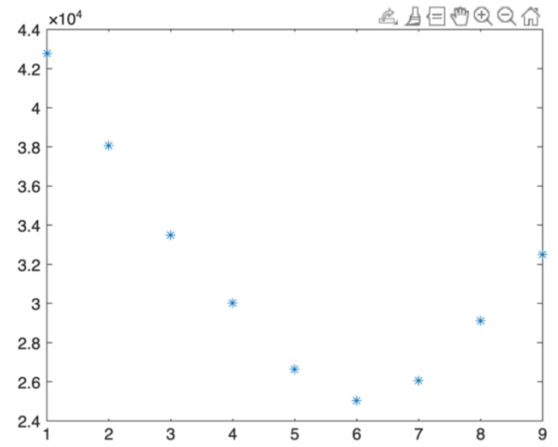
Like before, we begin with the range of 100-900 for S. Also, as before, with an interval width of 100. So, for S=100:100:900 and N=1000, the plot is:



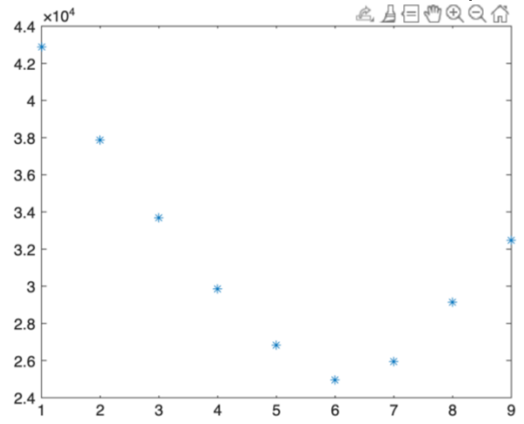
This is reasonably smooth, but we will continue, so for N=2000 the plot is:



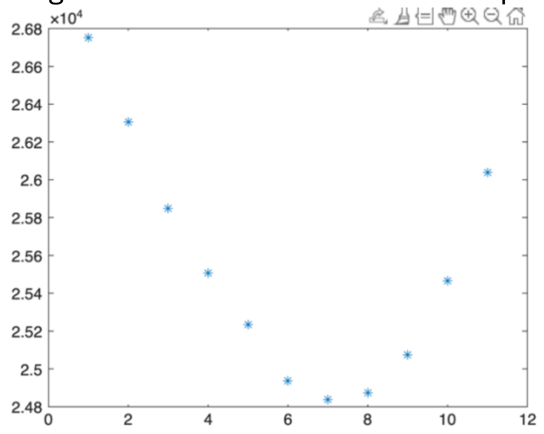
Now, we compute for N=3000:



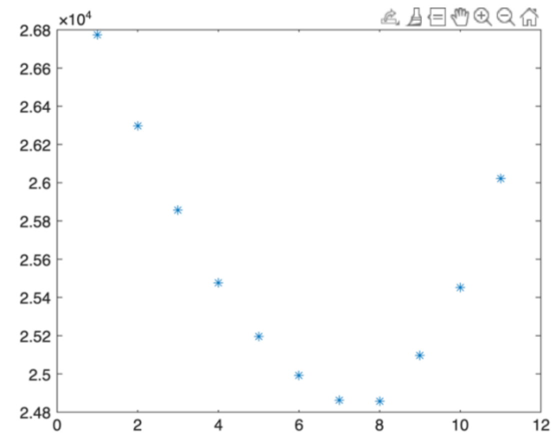
Just to be sure that the first four X points prove smooth, we compute at N=4000:



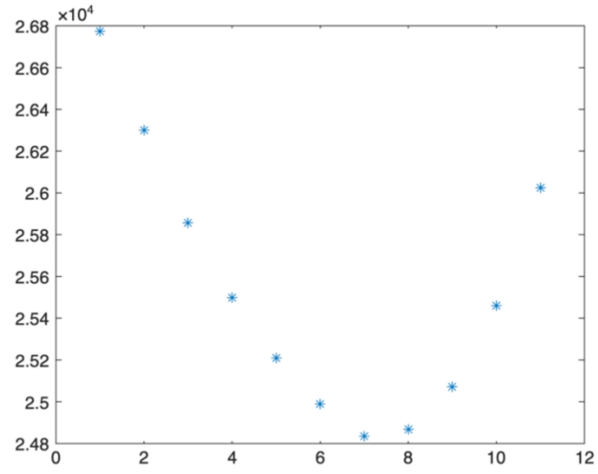
This is now quite smooth. The minimum appears at X=6, where S=600. So, we narrow the range to S=500:20:700 and first compute N=50000, for which the plot is:



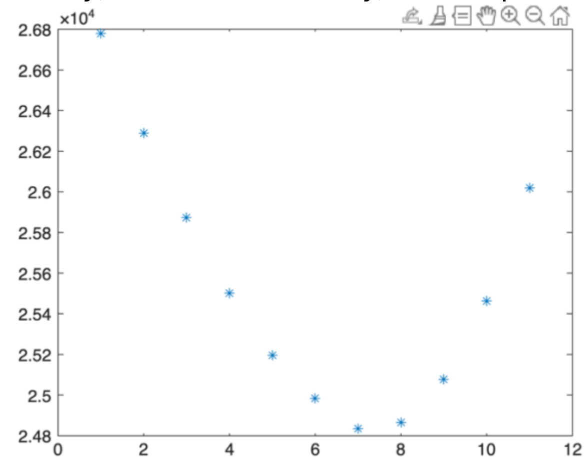
This looks smooth but we now compute for N=100000:



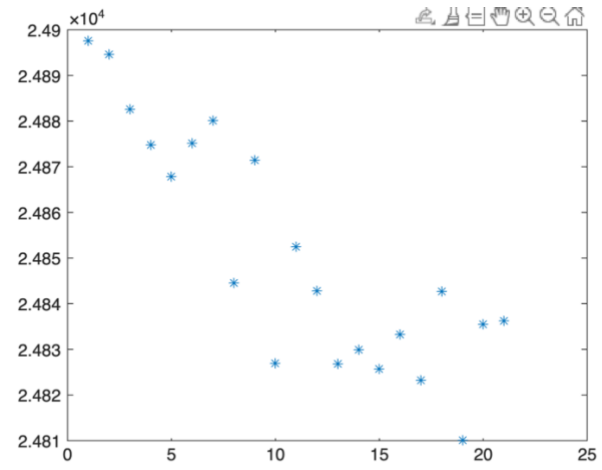
Since the values for X=6 and X=7 have flipped as minimums, we want to be sure and raise the number of simulations a few more times. So, for N=150000:



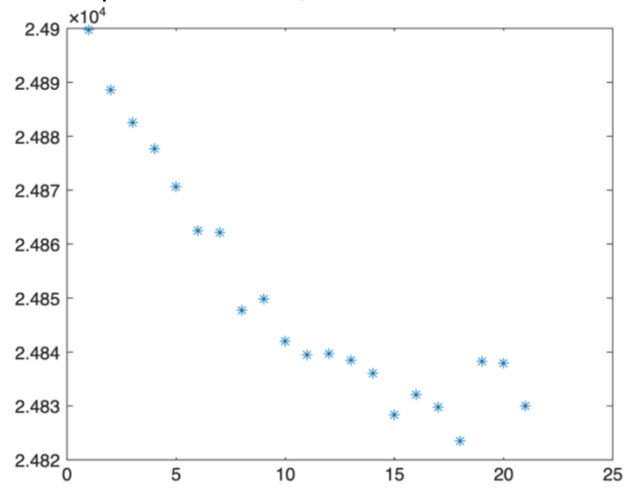
Finally, to ensure accuracy, we compute N=200000:



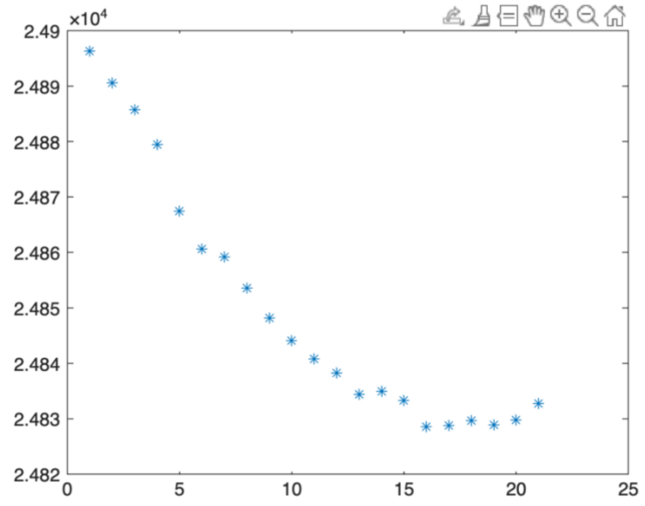
The plot is smooth and the minimum has stayed at X=7, it is reasonable to assume this is accurate. We now further limit the range to S=610:1:630. We start at N=100000 simulations, where the plot is:



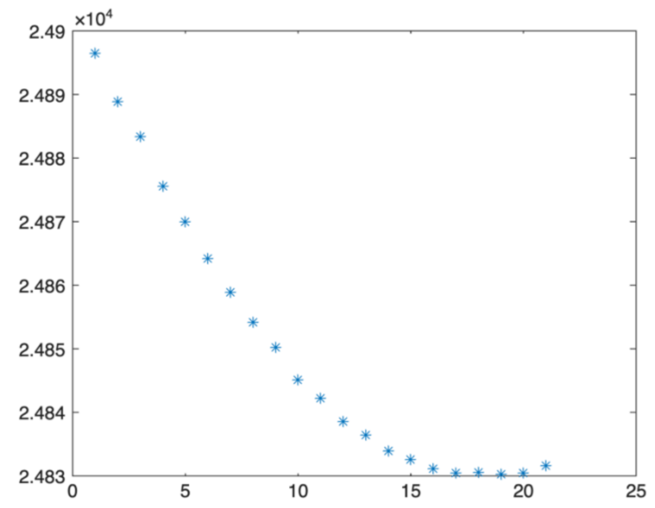
This is quite scattered, so we increase the number of simulations to N=1000000:



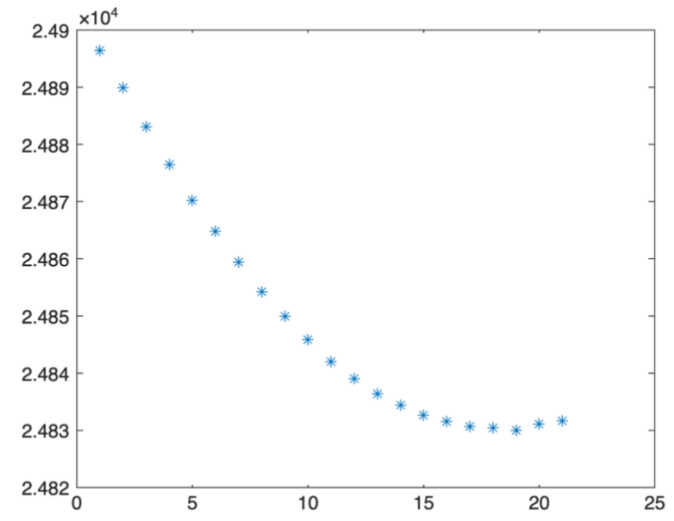
This is better, but not smooth, increase simulations to N=10000000:



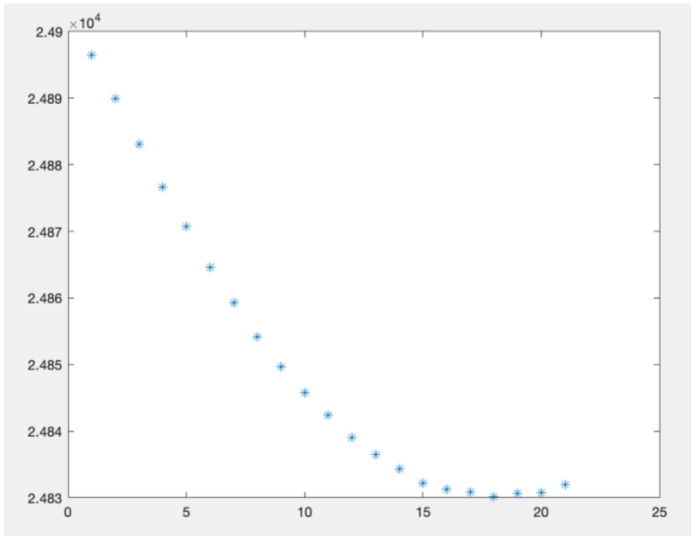
This is still improved but not smooth, now compute N=100000000:



There is still inconsistencies near the minimum, to be sure we compute N=500000000:



This is somewhat hard to tell where the minimum is, so let us compute N=700000000:



This is now quite smooth and the minimum is sure to be at X=18, where S=627. Thus, the optimal pool size is 627 substitutes for a pay rate of $36 and overtime rate of $81.

Results: For Part C, the optimal pool size is 600. This means that a pool size of 600 on Tuesdays will be the lowest expected cost to the school district. For Part C, this optimal pool size increases to 627 substitutes to have on hand on Tuesdays for the lowest expected cost to the district. The answers to Part C (S=600) and Part D (S=627) are different due to the difference in pay rate. This difference is intuitively correct. This is because when the pay rate for substitutes is lower, like in Part D, the school district can afford to keep more substitutes on hand, which ensures that classes that need the substitutes have them. The goal was to minimize costs by minimizing the pool size, so the optimal pool size can be larger when the pay rate is lower.