double maxCost- the maximum cost per kWh in the highest period of the day, Note calculate this while entering price data

double minCost the minimum cost per kWh in the lowest cost period of the day, Note calculate this while entering price data

double eLoss-=.80 our energy conversion loss

xMax- maximum charging rate either capacity/4 or capacity/8 for a battery with 45%DOD for c/4 it takes 4 hours to charge, 8 for c/8,

surcharge=$3  (is the cost per kWh of usage during the peak hour each day, $3 used in paper so greedy approach is not used as much)

T is the length of the low-price period(in real time, likely to be only 1 hour on our case)

maxCapacity-usable energy storage capacity that is 50% of their average daily demand.

avgDemand-average daily demand of electricity

avgCost-average  daily electric cost

capacity=current amount of battery charged, can’t be negative or greater than maxCapacity)

energyVectPos= the position in the energy vector we need to add/subtract energy from

#define MAXCAPACITY (some number of maxCapacity)

#define MINCAPACITY (some number of minCapacity)

bool maxChargeRate = true;

/\*\*

\* charge for one hour using charge rate

\*/

Function chargeBattery(double chargeRate, int energyVectPos){

if (capacity!= MAXCAPACITY){ 🡨???

//Check to make sure capacity is not > maxCapacity

//if it is change capacity to maxCapacity

if ((capacity < MAXCAPACITY) && (!maxChargeRate) ){

charge = MAXCAPACITY;

}

// If we are using the max charge rate,

// This should only charge1/4 of the maxCapacity!

if ((capacity < MAXCAPACITY) && maxChargeRate ){

charge = MAXCAPACITY/4;

}

//add charged energy-to-energy data for that hour

//Remember loss of energy from energy conversion!

totalChargeHour += charge;

}

return totalChargeHour;

}

/\*\*

\* Discharge for one hour using charge rate

\*/

Function dischargeBattery(double dischargeRate, int energyVectPos){

//Check to make sure capacity is not < 0 if it is change capacity to 0

if ((capacity < MINCAPACITY) && (!maxChargeRate) ){

charge = 0;

}

//remember if we are using the max charge rate,

// this should only discharge1/4 of the maxCapacity!

if ((capacity < MINCAPACITY) && (!maxChargeRate) ){

charge = MAXCAPACITY/4;

}

// Subtract discharged energy from energy data from that hour

// Remember loss of energy from energy conversion!

totalChargeHour -= charge;

return totalChargeHour;

}

Function batteryBehavior(){

// each hour of data/price in our vector vectDataI, vectPricei

for (int i=0;i<72,i++){

if(vectDataI <=avgCost){

if(vectPricei<=avgDemand){

if((xMax\*eLoss\* maxCost \*T )− (xMax\* minCost \*T )> xMax\*surcharge){

//charge max rate

chargeBattery (XMax, vectDataI,);

}

else{

//charge at rate to sustain the target avgDemand

//not sure how to do, chargeBattery (?,vectDataI,);

}

}

else{

if((xMax\*eLoss\* maxCost \*T )− (xMax\* minCost \*T )> xMax\*surcharge){

//charge max rate

chargeBattery (XMax, vectDataI,);

}

else{

//discharge at rate to sustain the target avgDemand

//not sure how to do, dischargeBattery (?,vectDataI,);

}

}

}

else{

if(vectPricei<=avgDemand){

if((xMax \* eLoss \* maxCost \*T )− (xMax \* minCost \*T )> xMax \*surcharge){

//discharge max rate

dischargeBattery(XMax, vectDataI,);

}

else{

//do nothing

}

}

else{

if((xMax\*eLoss\* maxCost \*T )− (xMax\* minCost \*T )> xMax\*surcharge){

//discharge max rate

dischargeBattery(XMax, vectDataI,);

}

else{

//discharge at rate to sustain the target avgDemand

//not sure how to do, dischargeBattery (?,vectDataI,);

}

}

}

}