Team 9 Final Report Code

JavaScript app:

```
function pricePayoff(size) {
  var netPVCost = Math.round(size*2.91 - size*2.91*36/100);
  setText("totalPrice", '$'+Math.round(size*2.91));
  setText("netPrice", '$'+netPVCost);
  setText("payoffTime", Math.round(10*netPVCost/totalElectrCost)/10);
  //Since Math.roud() rounds to the nearest integer, in order
  //to get one decimal place, multiply the value by 10, round,
  //and then devide it by 10.
}

var totalUsage = 0;
var totalElectrCost = 0;

onEvent("button", "click", function( ) {
  totalUsage = Math.round(getNumber("avgDailyUsage")*365);
  setNumber("annualUsage", totalUsage);
  totalElectrCost = Math.round(getNumber("avgDailyCost")*365);
  setText("annualCost", '$'+totalElectrCost);

  if (totalUsage <= 1800) {
    setText("sizePV", "1 kW");
    setText("annualPVEnergy", "1,745");
    pricePayoff(1000);
  } else if ((totalUsage > 1800) && (totalUsage <= 2700)) {
    setText("sizePV", "1.5 kW");
    setText("annualPVEnergy", "2,619");
    pricePayoff(1500);
  } else if ((totalUsage > 2700) && (totalUsage <= 3500)) {
    setText("sizePV", "2 kW");
    setText("annualPVEnergy", "3,492");
    pricePayoff(2000);
  } else if ((totalUsage > 3500) && (totalUsage <= 4500)) {
    setText("sizePV", "2.5 kW");
    setText("annualPVEnergy", "4,365");
    pricePayoff(2500);
  } else if ((totalUsage > 4500) && (totalUsage <= 5400)) {
    setText("sizePV", "3 kW");
    setText("annualPVEnergy", "5,238");
    pricePayoff(3000);
  } else if ((totalUsage > 5400) && (totalUsage <= 6300)) {
    setText("sizePV", "3.5 kW");
    setText("annualPVEnergy", "6,111");
    pricePayoff(3500);
  } else if ((totalUsage > 6300) && (totalUsage <= 7200)) {
    setText("sizePV", "4 kW");
```

```
    setText("annualPVEnergy", "6,984");
    pricePayoff(4000);
  } else if ((totalUsage > 7200) && (totalUsage <= 8100)) {
    setText("sizePV", "4.5 kW");
    setText("annualPVEnergy", "7,857");
    pricePayoff(4500);
  } else if ((totalUsage > 8100) && (totalUsage <= 9000)) {
    setText("sizePV", "5 kW");
    setText("annualPVEnergy", "8,730");
    pricePayoff(5000);
  } else if ((totalUsage > 9000) && (totalUsage <= 9900)) {
    setText("sizePV", "5.5 kW");
    setText("annualPVEnergy", "9,603");
    pricePayoff(5500);
  } else if ((totalUsage > 9900) && (totalUsage <= 10800)) {
    setText("sizePV", "6 kW");
    setText("annualPVEnergy", "10,476");
    pricePayoff(6000);
  } else if ((totalUsage > 10800) && (totalUsage <= 11700)) {
    setText("sizePV", "6.5 kW");
    setText("annualPVEnergy", "11,349");
    pricePayoff(6500);
  } else if ((totalUsage > 11700) && (totalUsage <= 12800)) {
    setText("sizePV", "7 kW");
    setText("annualPVEnergy", "12,414");
    pricePayoff(7000);
  } else if ((totalUsage > 12800) && (totalUsage <= 13700)) {
    setText("sizePV", "7.5 kW");
    setText("annualPVEnergy", "13,300");
    pricePayoff(7500);
  } else if ((totalUsage > 13700) && (totalUsage <= 14600)) {
    setText("sizePV", "8 kW");
    setText("annualPVEnergy", "14,187");
    pricePayoff(8000);
  } else if ((totalUsage > 14600) && (totalUsage <= 15500)) {
    setText("sizePV", "8.5 kW");
    setText("annualPVEnergy", "15,074");
    pricePayoff(8500);
  } else if ((totalUsage > 15500) && (totalUsage <= 16400)) {
    setText("sizePV", "9 kW");
    setText("annualPVEnergy", "15,960");
    pricePayoff(9000);
  } else if ((totalUsage > 16400) && (totalUsage <= 17300)) {
    setText("sizePV", "9.5 kW");
    setText("annualPVEnergy", "16,847");
    pricePayoff(9500);
  } else if ((totalUsage > 17300) && (totalUsage <= 18300)) {
    setText("sizePV", "10 kW");
    setText("annualPVEnergy", "17,734");
    pricePayoff(10000);
```

```
  }
  else {
    setScreen("screen2");
    onEvent("tryAgain", "click", function( ) {
      setScreen("screen1");
    });
    setText("avgDailyUsage", "");
    setText("avgDailyCost", "");
    setText("annualUsage", "");
    setText("annualCost", "");
    setText("sizePV", "");
    setText("annualPVEnergy", "");
    setText("totalPrice", "");
    setText("netPrice", "");
    setText("payoffTime", " ");
  }
});
```

JavaScript app translated to Python enhanced by a bar chart:

```
import math
import matplotlib.pyplot as plt
print()
#The max daily usage: 50!
#Small house uses about 8-10 kWh at the cost of $1-2 per day
#Large household uses about 35-50 kWh at the cost of $5.5 - 7 per day
avgDailyUsage = float(input("Your average electricity usage per day in kWh
from your Utility bill: "))
avgDailyCost = float(input("Your average electricity cost per day: "))
print()
print("Your annual electricity usage is about {:,.0f}
kWh.".format(avgDailyUsage*365))    #curly brackets indicate the
placeholder for variable, comma is used to indicate the thousands place
and .0 means that there will be no decimal values, and f means that this
variable uses float as a data type
print("Your annual electricity cost is about
${:,.0f}.".format(avgDailyCost*365))
def pricePayoff(size, prod):
 print()
 panels = math.ceil(size/275) #275 watts is produced by a standard
polycrystaline solar panel
 print("Approximate NUMBER of PANELS: "+str(panels)+';')
 print()  #2.91 is the average cost of 1 watt produced by a PV system
 print("TOTAL COST: about ${:,.0f};".format(size*2.91))
 print()
 netCost = size*2.91 - size*2.91*36/100  #26% is federal income tax credit
and 10% is state, all = 36%
 print("NET COST (after deducting 36% federal & state tax credits):
${:,.0f};".format(netCost))
 print()
 print("PAYOFF: It will take about {:.1f} years to pay off such
system".format(netCost/(avgDailyCost*365)))
 print("from what you will save on your utility bills each year.")
 print()
 months = ['Jan:', 'Feb:', 'Mar:', 'Apr:', 'May:', 'Jun:', 'Jul:', 'Aug:',
'Sep:', 'Oct:','Nov:', 'Dec:']
 print("Such PV System will produce the following amount of AC energy each
month:")
 for i in range(len(prod)):
   print(months[i],prod[i])
 print("Total:",sum(prod),'kWh')
 print()
```

```
if avgDailyUsage*365 <= 18300:
 print()
 print("Specifications of the PV System for area code = 87507:")
 print("*"*54)
 print()
 #Use PVWatts Calculator to get annual production made by 1 - 10 kW
DC(direct current) PV Systems:
 if avgDailyUsage*365 <= 1800:        #max 1 kW DC System outpt
   print("SIZE: 1 kW DC System;")
   print()
   print("RECOMMENDED TYPE: Fixed (roof mount);")
   print()
   print("ANNUAL PRODUCTION: about 1,745 kWh of AC")
   production = [122, 125, 156, 165, 179, 170, 157, 155, 146, 136, 121,
113]
   pricePayoff(1000, production)
 elif avgDailyUsage*365 > 1800 and  avgDailyUsage*365 <= 2700:   #max 1.5
kW DC System output
   print("SIZE: 1.5 kW DC System;")
   print()
   print("RECOMMENDED TYPE: Fixed (roof mount);")
   print()
   print("ANNUAL PRODUCTION: about 2,619 kWh of AC;")
   production = [183, 188, 233, 248, 269, 256, 235, 232, 219, 205, 182,
169]
   pricePayoff(1500, production)
 elif avgDailyUsage*365 > 2700 and avgDailyUsage*365 <= 3500:
   print("SIZE: 2 kW DC System;")
   print()
   print("RECOMMENDED TYPE: Fixed (roof mount);")
   print()
   print("ANNUAL PRODUCTION: about 3,492 kWh of AC;")
   production = [244, 251, 311, 330, 359, 341, 313, 310, 292, 273, 242,
226]
   pricePayoff(2000, production)
 elif avgDailyUsage*365 > 3500 and avgDailyUsage*365 <= 4500:
   print("SIZE: 2.5 kW DC System;")
   print()
   print("RECOMMENDED TYPE: Fixed (roof mount);")
   print()
   print("ANNUAL PRODUCTION: about 4,365 kWh of AC;")
   production = [305, 314, 389, 413, 448, 426, 391, 387, 365, 341, 303,
282]
```

```
    pricePayoff(2500, production)
  elif avgDailyUsage*365 > 4500 and avgDailyUsage*365 <= 5400:
    print("SIZE: 3 kW DC System;")
    print()
    print("RECOMMENDED TYPE: Fixed (roof mount);")
    print()
    print("ANNUAL PRODUCTION: about 5,238 kWh of AC;")
    production = [367, 376, 467, 495, 538, 511, 470, 465, 438, 409, 363,
339]
    pricePayoff(3000, production)
  elif avgDailyUsage*365 > 5400 and avgDailyUsage*365 <= 6300:
    print("SIZE: 3.5 kW DC System;")
    print()
    print("RECOMMENDED TYPE: Fixed (roof mount);")
    print()
    print("ANNUAL PRODUCTION: about 6,111 kWh of AC;")
    production = [428, 439, 544, 578, 628, 597, 548, 542, 511, 477, 424,
395]
    pricePayoff(3500, production)
  elif avgDailyUsage*365 > 6300 and avgDailyUsage*365 <= 7200:
    print("SIZE: 4 kW DC System;")
    print()
    print("RECOMMENDED TYPE: Fixed (roof mount);")
    print()
    print("ANNUAL PRODUCTION: about 6,984 kWh of AC;")
    production = [489, 502, 622, 660, 717, 682, 626, 620, 584, 545, 484,
452]
    pricePayoff(4000, production)
  elif avgDailyUsage*365 > 7200 and avgDailyUsage*365 <= 8100:
    print("SIZE: 4.5 kW DC System;")
    print()
    print("RECOMMENDED TYPE: Fixed (roof mount);")
    print()
    print("ANNUAL PRODUCTION: about 7,857 kWh of AC;")
    production = [550, 565, 700, 743, 807, 767, 705, 697, 657, 614, 545,
508]
    pricePayoff(4500, production)
  elif avgDailyUsage*365 > 8100 and avgDailyUsage*365 <= 9000:
    print("SIZE: 5 kW DC System;")
    print()
    print("RECOMMENDED TYPE: Fixed (roof mount);")
    print()
    print("ANNUAL PRODUCTION: about 8,730 kWh of AC;")
```

```
    production = [611, 627, 778, 826, 897, 852, 783, 774, 730, 682, 605,
565]
    pricePayoff(5000, production)
 elif avgDailyUsage*365 > 9000 and avgDailyUsage*365 <= 9900:
    print("SIZE: 5.5 kW DC System;")
    print()
    print("RECOMMENDED TYPE: Fixed (roof mount);")
    print()
    print("ANNUAL PRODUCTION: about 9,603 kWh of AC;")
    production = [672, 690, 856, 908, 986, 938, 861, 852, 803, 750, 666,
621]
    pricePayoff(5500, production)
 elif avgDailyUsage*365 > 9900 and avgDailyUsage*365 <= 10800:
    print("SIZE: 6 kW DC System;")
    print()
    print("RECOMMENDED TYPE: Fixed (roof mount);")
    print()
    print("ANNUAL PRODUCTION: about 10,476 kWh of AC;")
    production = [733, 753, 933, 991, 1076, 1023, 940, 929, 876, 818, 726,
678]
    pricePayoff(6000, production)
 elif avgDailyUsage*365 > 10800 and avgDailyUsage*365 <= 11700:
    print("SIZE: 6.5 kW DC System;")
    print()
    print("RECOMMENDED TYPE: Fixed (roof mount);")
    print()
    print("ANNUAL PRODUCTION: about 11,349 kWh of AC;")
    production = [794, 816, 1011, 1073, 1166, 1108, 1018, 1007, 949, 886,
787, 734]
    pricePayoff(6500, production)
 elif avgDailyUsage*365 > 11700 and avgDailyUsage*365 <= 12800:
    print("SIZE: 7 kW DC System;")
    print()
    print("RECOMMENDED TYPE: Fixed (open rack);")
    print()
    print("ANNUAL PRODUCTION: about 12,414 kWh of AC;")
    production = [867, 889, 1107, 1172, 1274, 1214, 1117, 1104, 1040, 969,
859, 802]
    pricePayoff(7000, production)
 elif avgDailyUsage*365 > 12800 and avgDailyUsage*365 <= 13700:
    print("SIZE: 7.5 kW DC System;")
    print()
    print("RECOMMENDED TYPE: Fixed (open rack);")
    print()
```

```
    print("ANNUAL PRODUCTION: about 13,300 kWh of AC;")
    production = [928, 953, 1186, 1255, 1365, 1300, 1197, 1183, 1114, 1038,
921, 859]
    pricePayoff(7500, production)
 elif avgDailyUsage*365 > 13700 and avgDailyUsage*365 <= 14600:
    print("SIZE: 8 kW DC System;")
    print()
    print("RECOMMENDED TYPE: Fixed (open rack);")
    print()
    print("ANNUAL PRODUCTION: about 14,187 kWh of AC;")
    production = [990, 1016, 1265, 1339, 1456, 1387, 1276, 1262, 1189,
1108, 982, 916]
    pricePayoff(8000, production)
 elif avgDailyUsage*365 > 14600 and avgDailyUsage*365 <= 15500:
    print("SIZE: 8.5 kW DC System;")
    print()
    print("RECOMMENDED TYPE: Fixed (open rack);")
    print()
    print("ANNUAL PRODUCTION: about 15,074 kWh of AC;")
    production = [1052, 1080, 1345, 1423, 1547, 1474, 1356, 1341, 1263,
1177, 1043, 974]
    pricePayoff(8500, production)
 elif avgDailyUsage*365 > 15500 and avgDailyUsage*365 <= 16400:
    print("SIZE: 9 kW DC System;")
    print()
    print("RECOMMENDED TYPE: Fixed (open rack);")
    print()
    print("ANNUAL PRODUCTION: about 15,960 kWh of AC;")
    production = [1114, 1143, 1424, 1507, 1638, 1561, 1436, 1420, 1337,
1246, 1105, 1031]
    pricePayoff(9000, production)
 elif avgDailyUsage*365 > 16400 and avgDailyUsage*365 <= 17300:
    print("SIZE: 9.5 kW DC System;")
    print()
    print("RECOMMENDED TYPE: Fixed (open rack);")
    print()
    print("ANNUAL PRODUCTION: about 16,847 kWh of AC;")
    production = [1176, 1207, 1503, 1590, 1729, 1647, 1516, 1499, 1411,
1315, 1166, 1088]
    pricePayoff(9500, production)
 elif avgDailyUsage*365 > 17300 and avgDailyUsage*365 <= 18300:
    print("SIZE: 10 kW DC System;")
    print()
    print("RECOMMENDED TYPE: Fixed (open rack);")
```

```
    print()
    print("ANNUAL PRODUCTION: about 17,734 kWh of AC;")
    production = [1238, 1270, 1582, 1674, 1819, 1734, 1595, 1577, 1486,
1385, 1228, 1145]
    pricePayoff(10000, production)
 #Plotting a bar chart to display monthly production of AC energy by PV
Solar System:
 #List comprehension used for x coordinates
 x = [i for i in range(1, 13)]
 months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep',
'Oct', 'Nov', 'Dec']
 plt.bar(x, production, tick_label = months, width = 0.8, color = ['red',
'green', 'blue'])
 #Labels for x and y axis:
 plt.xlabel('Month')
 plt.ylabel('kWh')
 #Title of the graph:
 plt.title('Monthly AC Energy produced by your PV Solar System')
 plt.show()

else:
 print()
 print("This app was designed for residential houses that use up to 50kWh
electricity a day")
```

Computational Model written using Python to simulate a solar tracker:

```
from scipy.optimize import curve_fit
from numpy import vectorize
import math
import matplotlib.pyplot as plt
#===== Helper functions for manipulating data=====
def Deg2Rad(x):  return (math.pi*x/180.0) # Deg2Rad convert
degrees -> radians
def Sine(x): return math.sin(Deg2Rad(x)) # Sine function is
math.sin of degrees
def RoundN(x,n):  return ( round(x*10.0**n)/10.0**n) # Round to
N decimal places
#-------------------------------------------------
# ===== Function to find angle for max response in lab data
=====
#.   This scans the data and finds largest value
#.       then returns the index pointing to largest value
#.   Useful to find the angle closest to the maximum response in
the data
#.       "curve_fit" needs guesses for values, like the angle
for the peak
#        response at 90 degrees.  If we know this angle of
maximum in lab
#        data we can compute best guess angle offset to start
the curve_fit
def GetMaxAngle(angle,data):
  max_value = max(data)
  max_index = data.index(max_value)
  return angle[max_index]
#-------------------------------------------------
#=====        Enter the data from panel     =====
LabAngle = [i*10.0 for i in range(0,10)] #. 0 to 90 degrees in
10 degree increments
LabDataClear =
[66.7,80.6,90.6,97.3,97.6,94.7,88.6,79.2,64.7,46.1] # actual lab
data
NLabData = len(LabDataClear) # how many data points?
#-------------------------------------------------------
#=====    Define the parameterized model and intial   =====
def pFitModel(x, Efficiency, Scatter, AngleShift):
  return Efficiency*Sine(x + AngleShift)*((Scatter*Sine(x +
AngleShift) + 1.0)/(Scatter + 1.0))*100
#.  We need to define our realistic function to fit to data
#.  This function has the ideal function plus a scatter function
```

```
#.  The overall Efficiency and amount of Scatter are parameters
in this function
#.  It also computes AngleShift, which is how far we have to
shift the data angles
#.     to center the lab data at 90 degrees (straight at sun)
#.  Parameters this function needs: Efficiency, Scatter,
AngleShift
#. "curve_fit" function needs the model function to be
'vectorized': the function
#.    must be able to take a list as input and generate a list
as output.
#
#  Define the vectorized version
vFitModel = vectorize(pFitModel) # this syntax came from the doc
notes
# "curve_fit" works more reliably if we have initial guesses for
parameters
#
MaxAngleGuess = GetMaxAngle(LabAngle, LabDataClear) # If we find
the angle in the
#      data that gives largest output, that is close to pointing
at sun, so we use
#      that value to compute the offset to make the lab angles
centered so that
#.     max response is near 90 degrees.  The curve fit procedure
will adjust this
#.     to make it precise.
#
guess = [ 1.0, 0.0, 90-MaxAngleGuess] # Make following guesses:
#.     1) Efficiency is 1.0 ... a perfect panel
#.     2) Scatter is zero ... panel uses all the light it gets
#.     3) AngleShift is 90-MaxAngleGuess ... this shifts data so
max is at 90 deg
#-----------------------------------------------------------------
---
#=====    Invoke the curve fit procedure according to docs
=====
#.  Routine returns the parameters in the 'params' list
#.     and an error term in the 'covariance'
#.  We send the parameterized model function, the x,y data, and
the guesses
params, covariance =
curve_fit(f=vFitModel,xdata=LabAngle,ydata=LabDataClear,p0=guess
,bounds=(-10000,10000))
#.  Now we extract the parameters of interest
FM_efficiency=RoundN(params[0],3)
```

```
FM_scatter=RoundN(params[1],3)
FM_angleshift=RoundN(params[2],3)
#.  and print them out to include in plotting program
print(" FM means Fitted Model ")
print("FM_efficiency = ",FM_efficiency)
print("FM_scatter = ",FM_scatter)
print("FM_angleshift = ",FM_angleshift)

LabAngleShifted = [LabAngle[i]+FM_angleshift for i in
range(0,10)]
print("Lab Angle Shifted = ", LabAngleShifted)
print("Lab Data ", LabDataClear)
#===== Define the simplified Fit Model for plotting
#.  and sample at 5 degree angle increments
#.  Zero out the FM_angleshift parameter here because its now
included in the
#.     lab angle data.
def FitModel(x):
   return pFitModel(x,FM_efficiency,FM_scatter,0.0)
Angle = [(i*5) for i in range(37) ]
IdealModel = [100*Sine(a) for a in Angle]
FitModelSampled = [FitModel(a) for a in Angle]
#-----------------------------------------------------
#=====   Plot Ideal, Lab Data, and FitModel Data  ======
plt.plot(Angle,IdealModel,label="Ideal PV Panel Model",
color='red',linestyle='solid',marker='o',markerfacecolor='red',m
arkersize=4)# plot lab data
plt.plot(Angle,FitModelSampled,label="LMS Fit PV Panel Model",
color='grey',linestyle='solid',marker='o',markerfacecolor='black
',markersize=4)# plot lab data
plt.plot(LabAngleShifted,LabDataClear,label="Lab Data -Clear
Skies",color='green',linestyle='dashed',marker='v',markerfacecol
or='green',markersize=8)
plt.xlabel('Degrees')
plt.ylabel('Percent')
plt.grid()
plt.legend()
#Title of the graph:
plt.title('Ideal Model of PV Panel vs Incident Angle')
plt.show()
plt.savefig('PVPanelPlot.png')
```