PLACKETT BURMAN DESIGN

GROUP MEMBERS:

1. Subhashree Bharathan – 119010090
2. Shwetha S – 119010105
3. Rachita K Kumar – 119013027

INTRODUCTION:

Plackett Burman Design is a statistical method used to identify the importance of a particular variable from a list of variables that influence any process. This method is extensively used in designing experiments for bioprocess operations like fermentation where complete knowledge of the system is not available. Design parameters that influence the system in any bioprocess could be physical variables (like temperature, pressure, agitation, aeration, etc.,), chemical variables (like the media components – yeast extract, Carbon/Nitrogen ratio, salts, etc.,), or biological (growth rate, cell density, etc.,). This leaves us with too many parameters to consider while optimising the system for maximum yield. Thus, it becomes imperative that we find the parameters that have the most effect on the desired output (say, yield in this case).

But, the major downfall of using Plackett Burman design is that it does not account for the interaction among the variables. Thus, there might be some discrepancy on asserting the most influential variable though it gives a rough idea of possible list of influencing parameters. Also, for this method, high and low values for each parameter have to be ascertained and they have to be used correspondingly for each trial.

AIM:

To determine the variable that has the maximum effect on the response of the given system with regard to the given list of input parameters.

OVERVIEW OF METHODOLOGY:

- *INPUT:*
    - Number of parameters to be considered
    - Generate matrix – import **pyDOE** and use **pbdesign** module
    - Number of dummy variables among the parameters
    - Position of these dummy variables in the list of parameters
    - List of response variable values (say, yield in a fermentation process) corresponding to all the trials that are generated by the user on the basis of the high and low conditions for each parameter as in the matrix generated
- Calculate the variance for all the parameters (including the dummy variables)
- Perform F test for all the parameters
- *OUTPUT:*
    - Arrange F values in increasing order and print the order of importance of the variables on the system

VARIABLE DESCRIPTION:

- *INPUT VARIABLES:*
  - **n** → number of parameters (Integer type)
  - **res[ ]** → List of response values (Float type)
- *OUTPUT VARIABLES:*
  - **difference[ ]** → List of arithmetic difference between the sum response variable corresponding to the high and low conditions for a given parameter in the trial (Float type)
  - **Mean_sqr[ ]** → List of variance values for each parameter (Float type)
  - **MSE** → Mean square error (Float type)
  - **F_value[ ]** → List of the F test values for each parameter (Float type)
  - **final_list[ ]** → List of tuples with the parameter and its corresponding F test value
- *ANCILLARY VARIABLES:*
  - **dummy_matrix[ ][ ]**→ List of lists with the high (+1) and low (-1) values for each parameter generated before accounting for the dummy variables
  - **dum_num** → number of dummy variables generated based on the number of input parameters (which is 'n') (Integer type)
  - **matrix[ ][ ]**→ List of lists with the high (+1) and low (-1) values for each parameter generated after accounting for the dummy variables

PACKAGES & MODULES USED:

- pyDOE → pbdesign()
  - For generation of Plackett Burman matrix
- math → pow()
- string → ascii_uppercase()
  - To label the parameters with its corresponding F values and zip it into a list of tuples
  - ascii_uppercase generates alphabets for a given range and is used to represent the pameters

PSEUDOCODE:

```
• From pyDOE import module pbdesign
• import math
• import string
• Input the number of parameters, n
• Create a matrix, dummy_matrix using pbdesign() function. Make up the parameters to one
  less than multiple of four.
    • If (len(dummy_matrix)-n) ==1, n=n+4. Set dum_num=4.
    • Else dum_num=len(dummy_matrix)-n-1 . Set n=len(dummy_matrix)-1
• Create a matrix, matrix using pbdesign() function
• Print the matrix
```

- Input the response values corresponding to each trial, as a list res
- Calculate the differences between the high and low values for each parameter
    - sum+=(matrix[row][column])*res[row]
- Calculate the variance and mean squared error for each parameter
- Determine the F-statistic for each parameter
    - F=Variance/MSE
- The parameter with highest F-statistic is determined. It is deemed the most significant for the experiment

SOURCE CODE:

```
from pyDOE import pbdesign

import math

import string

'''

To use the Plackett Burman design for the given input variables and find

the most influential variable.

'''

n=int(raw_input("Enter the number of parameters:"))

dummy_matrix=pbdesign(n)

if((len(dummy_matrix)-n)==1):

   n=n+4

   dum_num=4

else:

   dum_num=len(dummy_matrix)-n-1

 #make up the number of parameters to one less than multiple of Four

   n=len(dummy_matrix)-1

matrix=pbdesign(n)

print matrix,"\n"
```

```
#accept the response values corresponding to each experimental trial

res=[ ]

for i in range(0,len(matrix)):

    print "The yield for the",i+1,"th trial:"

    val=float(raw_input())

    res.append(val)

#list of all differences for each parameter

difference=[ ]

for column in range(0,n):

    sum=0.0

    for row in range(0,n+1):

        sum+=(matrix[row][column])*res[row]

    difference.append(round(sum,3))

print "Difference b/w high and low values:",difference,"\n"


#variance of each parameter

Mean_sqr=[ ]

for val in difference:

    MS=(math.pow(val,2))/(n+1)

    Mean_sqr.append(round(MS,3))

MSE=0.0

Mean=0.0

flag=0

#determination of mean square error

for i in range(0,dum_num):

    for j in range(dum_num,n):

        flag+=1

       Mean+=Mean_sqr[j]
    MSE=Mean/flag

print "Variance:",Mean_sqr,"\n"

print "Mean square error:",MSE,"\n"


#F test value for each parameter

F_value=[ ]

for i in range(0,n):

    fvalue=round(Mean_sqr[i]/MSE,3)

    F_value.append(fvalue)

print "F value for the given set of parameters:","\n"

final_list=zip(string.ascii_uppercase[:n],F_value)

#printing the parameters ranging from A,B,C,D and so on against their corresponding F-values

print final_list

for para in final_list:

    if(para[1]==max(F_value)):

            print "Most significant parameter for the system is:",para,"\n"

#prints the most influential parameter from amongst the input
```