Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ МАТЕМАТИКИ И МЕХАНИКИ ИМ.
Н.И.ЛОБАЧЕВСКОГО
КАФЕДРА МАТЕМАТИЧЕСКОГО АНАЛИЗА

Направление: 01.03.01 – «Математика»

Профиль: общий

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

"АВТОМАТИЧЕСКАЯ СИСТЕМА ОТКЛЮЧЕНИЯ ПРИБОРА С ИСПОЛЬЗОВАНИЕМ РАССТОЯНИЯ МАХАЛАНОБИСА".

Студент 4 курса
Группы 05-403
«__» июня 2018 г. _____ Ф.Р. Фархшатов

Научный руководитель
к.ф.–м. н., н.с. регионального
научно-образовательного математического центра
«__» июня 2018 г. _____ А.А. Новиков

Заведующий кафедрой математического анализа
д.ф.–м. н., профессор
«__» июня 2018 г. _____ С. Р. Насыров

КАЗАНЬ-2018

Содержание

1. Введение **3**

2. Основная часть работы **4**

2.1. Расстояние Махаланобиса 4

2.2. Описание имеющихся данных и постановка задачи 6

2.3. Получение матрицы корреляций с учетом сдвигов 8

2.4. Модифицированное расстояние Махаланобиса 11

3. Заключение **12**

Список литературы **13**

4. Приложение **14**

1. Введение

Целью данной работы является использование расстояния Махаланобиса для диагностики неисправностей технических систем.

Одним из перспективных направлений в области анализа многомерных систем является теория распознавания образов в многомерном пространстве параметров наблюдения. Характерным примером такого направления является использование системы Махаланобиса-Тагучи(СМТ). СМТ уже широко используется рядом иностранных фирм, например Mitsubishi Heavy Indastries, для анализа технического состояния газо-турбинных двигателей [2]. Однако, использования этого метода в отечественном производстве пока не наблюдается.

В статье [3] рассмотрено использование расстояния Махаланобиса для диагностики газо-турбинных двигателей. В статье [4] - для диагностики програмного обеспечения. Применение СМТ также рассмотрено в статьях [6] и [7].

2. Основная часть работы

2.1. Расстояние Махаланобиса

Пусть $X = (x_{ij})$ - матрица данных размера $m \times n$. При этом мы считаем, что $i-$ый столбец этой матрицы является $i-$ым параметром рассматриваемого устройста, а $j-$ая строка - одним наблюдением в момент времени j. Таким образом мы можем представить данные в виде набора вектор-столбцов $(x_1, x_2, \ldots, x_n)$, где каждый $x_i \in \mathbb{R}^m$ $(i = 1, 2, \ldots, n)$ представляет собой некоторый параметр устройства.

Пусть $\overline{x}_i$ - среднее, а σ_i - стандартное отклонение i-го параметра набора данных X. То есть

$$\overline{x}_i = \frac{1}{m}\sum_{j=1}^{m} x_{ij}, \qquad \sigma_i = \sqrt{\frac{1}{m-1}\sum_{j=1}^{m}(x_{ij} - \overline{x}_i)^2}.$$

Любой вектор $y = (y_1, y_2, \ldots, y_n) \in \mathbb{R}^n$ можно привести к виду

$$y' = \Big(\frac{y_1 - \overline{x}_1}{\sigma_1}, \frac{y_2 - \overline{x}_2}{\sigma_2}, \ldots, \frac{y_n - \overline{x}_n}{\sigma_n}\Big)$$

путем преобразования

$$y'_i = \frac{y_i - \overline{x}_i}{\sigma_i} \qquad (i = 1, 2, \ldots, n).$$

Будем называть такое преобразование центрированием вектора y относительно набора данных X, а полученный вектор y' - центрированным относительно X.

Также будем рассматривать матрицу корреляций $C = (c_{ij})$ набора данных X, которая определяется следующим образом: $c_{ij} = cor(x_i, x_j)$. То есть элементами этой матрицы являются коэффициенты корреляции между параметрами набора данных X. Ясно, что данная матрица имеет размерность $n \times n$.

Далее определим расстояние Махаланобиса для точки $x \in \mathbb{R}^n$ и матрицы X. Оно является обобщенным расстоянием от этой точки до среднего

набора данных X.

Определение.Расстоянием Махаланобиса точки $\mathbf{y} \in \mathbb{R}^n$ для матрицы X называется число $MD(y)$, вычисляемое по следующей формуле:

$$MD(y) = \frac{1}{n} z C^{-1} z^t \qquad \text{или}$$

$$MD(y) = \frac{1}{n} \begin{bmatrix} z_1 & z_2 & \dots & z_n \end{bmatrix} C^{-1} \begin{bmatrix} z_1 \\ z_2 \\ \dots \\ z_n \end{bmatrix},$$

где вектор $z = (z_1, z_2, \dots, z_n)$ получен путем центрирования вектора $y = (y_1, y_2, \dots, y_n)$, а C^{-1} - матрица, обратная к C.

Пусть X - набор данных за период времени, в который устройство работало исправно, а Y - матрица данных этого же устройства размера $k \times n$ за время, в которое могли произойти неполадки. Расстояние Махаланобиса $MD(y)$ является показателем степени "ненормальности"системы в мемент испытания y. То есть чем больше значение $MD(y)$, тем больше оснований судить о неисправности устройства.

2.2. Описание имеющихся данных и постановка задачи

Устройством, рассмотренным в данной работе является колонна дегазации ПВХ(поливинилхлорид). То есть резервуар с низким далением, куда подается суспензия ПВХ. Дегазация происходит вследствие перепада давлений. Имеются 2 набора данных этого утройства: "нормальные"данные X, то есть записанные во время исправной работы прибора и "поломочные"данные Y, то есть такие, в период записи которых были замечены неисправности. Причем набор X содержит данные работы прибора в течение одного дня с частотой записи одно имерение в минуту, а Y - в течение четырех недель с одним измерением в час. Таким образом X имеет размерность 1440×11, а Y - 672×11.

Имеютя 11 параметров работы колонны(упорядочены в порядке их нумерации в X и Y):

1) Температура в средней части К301-1

2) Перепад давления в К301-1

3) Расход суспензии ПВХ к К301-1

4) Уровень суспензии ПВХ в К301-1

5) Температура в кубе К301-1

6) Расход пара к К301-1

7) Температура суспензии ПВХ в К301-1

8) Давление в Р1303-1

9) Температура ПВХ в Х302-1

10) Давление входного газа в С304-1

11) Температура верха К301-1

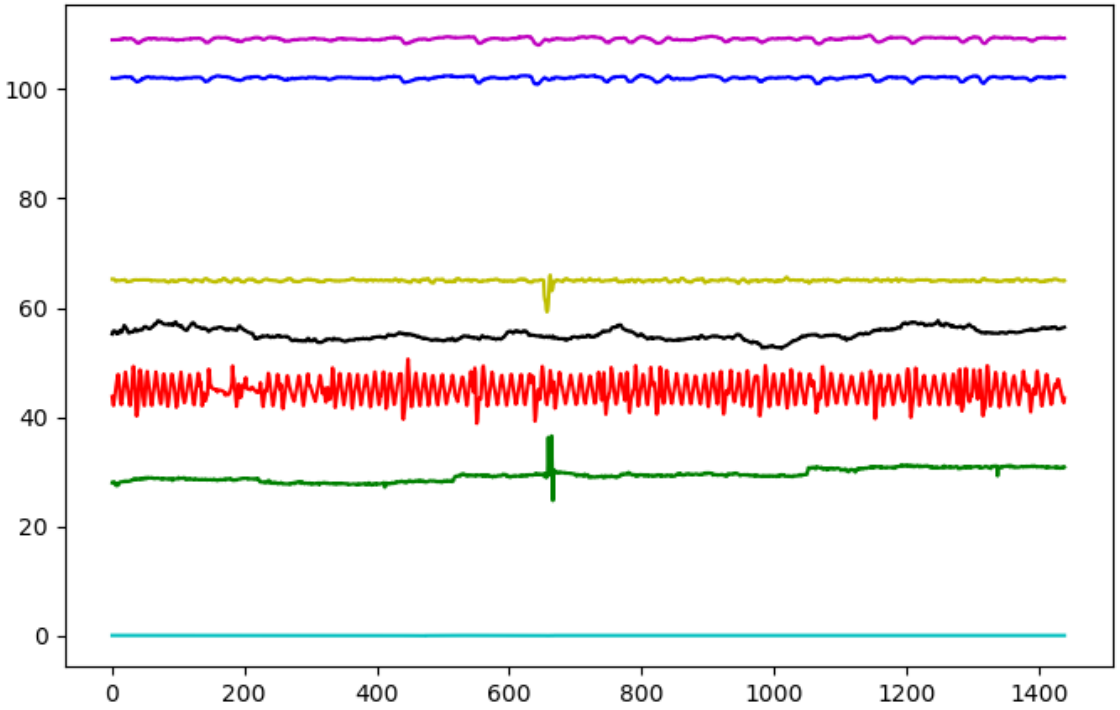

Рис. 1. График некоторых параметров "нормальных"данных X

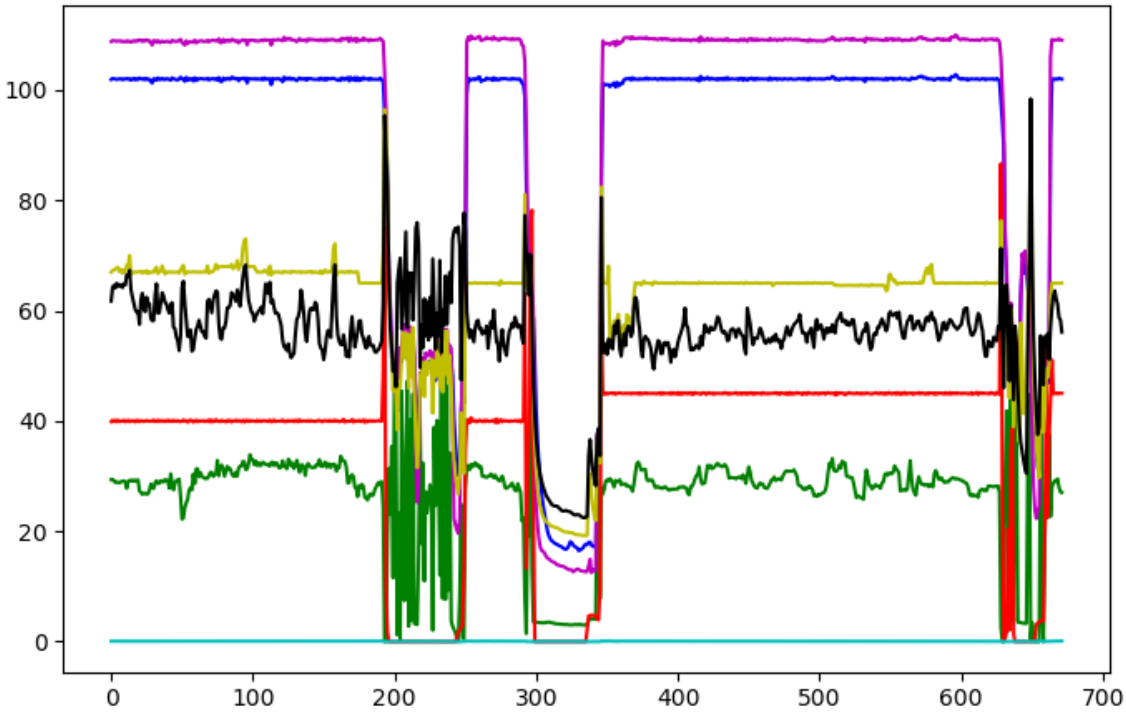

Рис. 2. График некоторых параметров "поломочных"данных Y

2.3. Получение матрицы корреляций с учетом сдвигов

Для вычисления расстояния Махаланобиса $MD(y)$ необходимо получить матрицу корреляций $C = (cor(\mathbf{x}_i, \mathbf{x}_j))$ нашего набора нашего данных $X = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$. Это можно сделать путем непосредственного вычисления. Однако, вместо этого мы модифицируем матрицу корреляций C и получим более точное расстояние Махаланобиса.

Обозначим через $\mathbf{x}_i^k$ вектор, который является частью вектора-столбца $\mathbf{x}_i$, с фиксированным размером $h = 40$, имеющий в качестве первого элемента k-й элемент параметра $\mathbf{x}_i$. Другими словами, $\mathbf{x}_i^k$ - это интервал i-го параметра в сорок минут, начинающийся от k-ой минуты. Таким образом:

$$\mathbf{x}_i^k = \begin{pmatrix} x_{ki} \\ x_{k+1,i} \\ \cdots \\ x_{k+39,i} \end{pmatrix}.$$

Будем вычислять значениия $cor(\mathbf{x}_i^k, \mathbf{x}_j^k)$ для каждого k. Теоретически эти числа должны быть равны между собой и равны $cor(\mathbf{x}_i, \mathbf{x}_j)$. Однако, после вычилений выясняется, что значения имеют сильный разброс. Например, на рисунке 3 показан график коэффициентов корреляции для интервалов параметров 11 и 6.

Таким образом, возникает проблема нахождения значимых коэффициентов корреляции для всех параметров системы, а значит и проблема нахождения матрицы корреляций C.

Поскольку мы имеем дело с физическими параметрами, такими как температура, давление, расход, то зависимости между ними проявляются не сразу, а с некоторым запаздыванием. Найдем такие сдвиги по времени, при которых коэффициент корреляции является наиболее значимым. Будем искать их в пределах от нуля до шестидесяти минут, поскольку наличие бóльших запаздываний маловероятно.

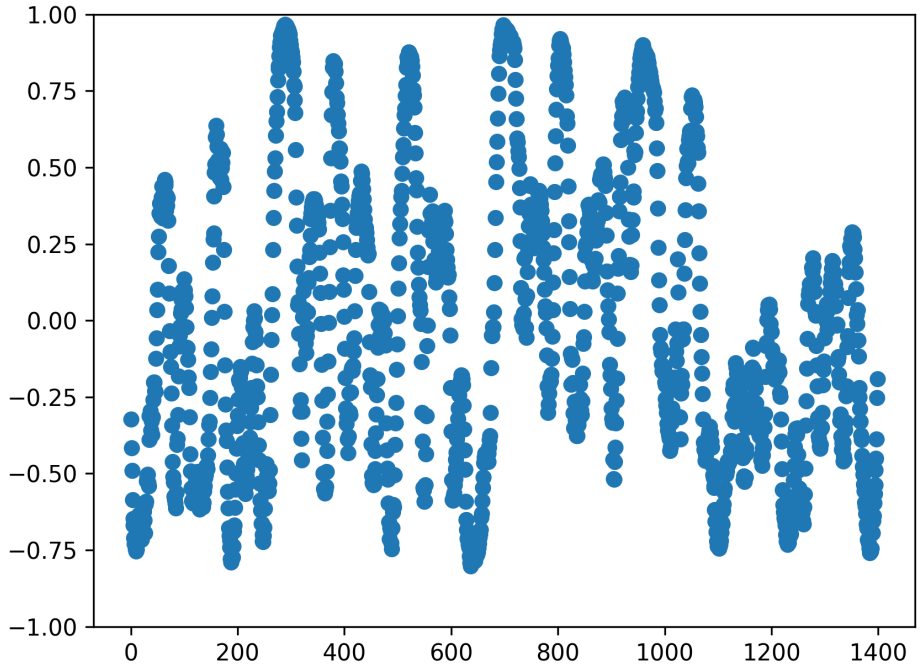

Рис. 3. Распределение коэффициентов корреляции для параметров 11 и 6

Обозначим через $cor^s(\mathbf{x}_i^k, \mathbf{x}_j^k)$ коэффициент корреляции между $\mathbf{x}_i^k$ и $\mathbf{x}_j^k$ взятый со сдвигом s. То есть $cor^s(\mathbf{x}_i^k, \mathbf{x}_j^k) = cor^s(\mathbf{x}_i^k, \mathbf{x}_j^{k+s})$. А через $Corr_{ij}^s$ - вектор значений $cor^s(\mathbf{x}_i^k, \mathbf{x}_j^k)$ для всех $k = 1, 2, \ldots, m - h - s$. То есть

$$Corr_{ij}^s = \begin{pmatrix} cor^s(\mathbf{x}_i^1, \mathbf{x}_j^1) \\ cor^s(\mathbf{x}_i^2, \mathbf{x}_j^2) \\ \ldots \\ cor^s(\mathbf{x}_i^{m-h-s}, \mathbf{x}_j^{m-h-s}) \end{pmatrix}.$$

Для любых параметров i и j будем искать такие сдвиги s, что выборочная дисперсия $Var(Corr_{ij}^s)$ является минимальной. Таким образом получаем следующую задачу оптимизации:

$$Var(Corr_{ij}^s) \xrightarrow[0 \le s < 60]{} min.$$

Решая ее, находим набор оптимальных сдвигов s_{ij}. Среди двух значений s_{ij} и s_{ji} выбираем тот, который достигается при меньшей дисперсии.

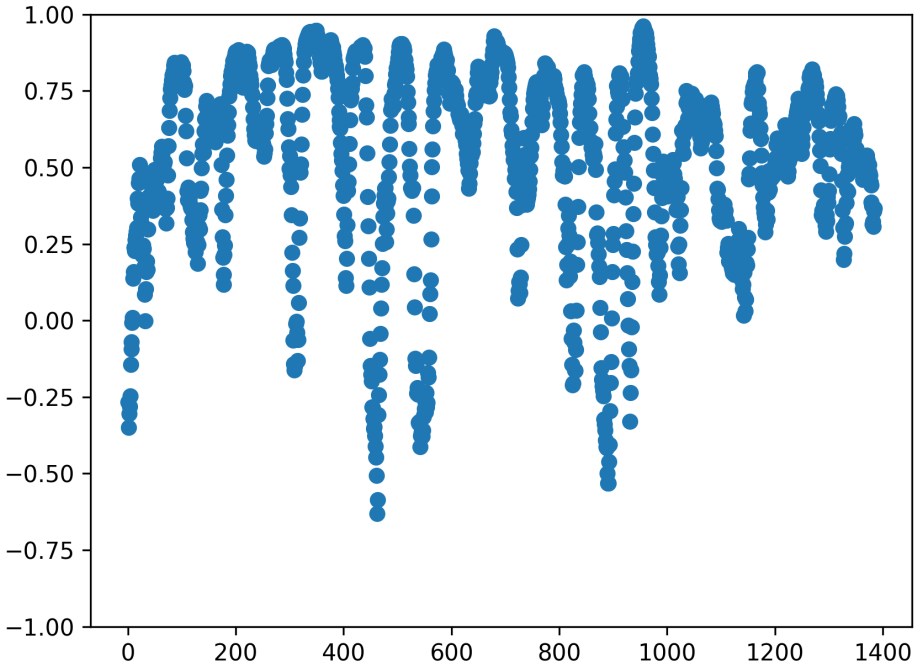

Рис. 4. Распределение коэффициентов корреляции для параметров 11 и 6 со сдвигом в 14 минут

Как видно из рисунков 3 и 4, после вычисления коэффициентов корреляции с учетом оптимальных сдвигов, расброс этих значений существенно уменьшился.

2.4. Модифицированное расстояние Махаланобиса

Для измерения степени "ненормальности"системы будем вычислять расстояние Махаланобиса для наблюдений набора данных Y, используя не матрицу корреляций C, а матрицу корреляций R, построенную с учетом оптимальных сдвигов s_{ij}. То есть:

$$MD(y) = \frac{1}{n} z R^{-1} z^t,$$

$$\text{где} \qquad z_i = \frac{y_i - \overline{x}_i}{\sigma_i} \qquad (i = 1, 2, \ldots, n).$$

Вычисляя таким образом $MD(y)$ для всех наблюдений матрицы данных Y, получаем график, расположенный на рисунке 5. Из графика следует, что, полученное таким образом расстояние, диагностирует все неисправности системы, произошедшие во время записи набора данных Y(см. рис.2).

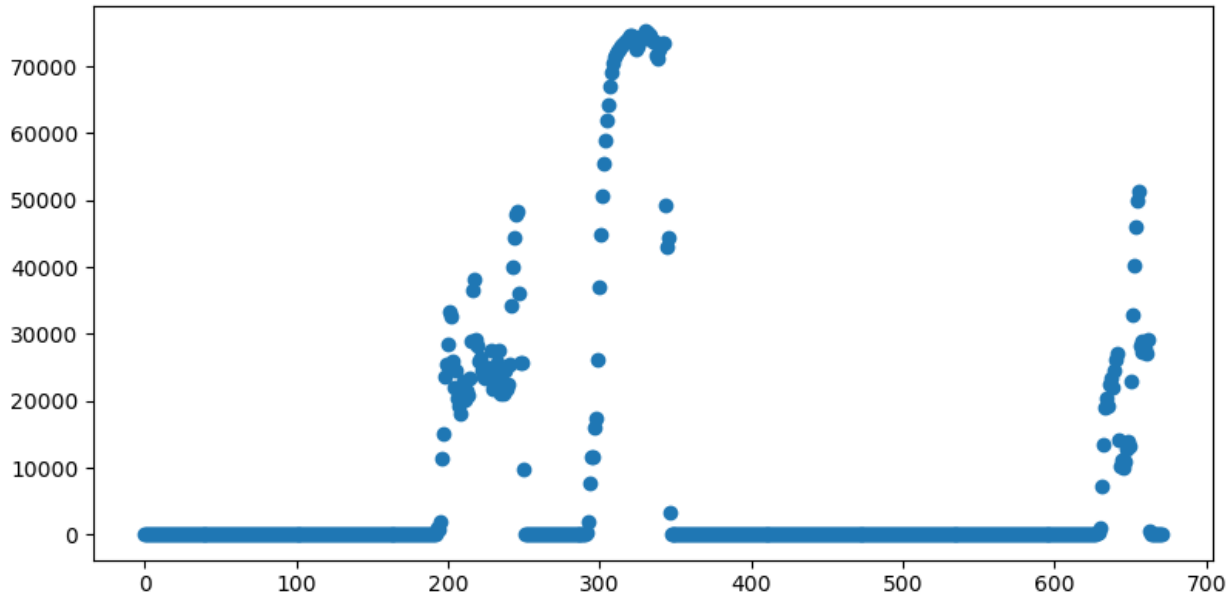

Рис. 5. Расстояние Махаланобися для наблюдений "поломочных"данных Y

3. Заключение

Для того, чтобы использовать метод Махаланобиса-Тагучи для диагностики технической системы необходимо вычислить коэффициенты корреляции для всех параметров этой системы и составить из них матрицу. Однако, поскольку мы имеем дело с процессами протекающими во времени, то зависимости между велечинами проявляются не мгновенно, а с некоторым запаздыванием (временным сдвигом). Таким образом возникает проблема определения значимых коэффициентов корреляции, и,следовательно, матрицы корреляций. В данной работе эта проблема решена путем нахождения оптимальных сдвигов при помощи минимизации выборочной дисперсии коэффициентов корреляции между всеми параметрами. Значения коэффициентов корреляции посчитанные с учетом оптимальных сдвигов являются наиболее значимыми.

Для всех наблюдений данной технической системы вычислено расстояние Махаланобиса, учитывающее оптимальные сдвиги. Что дает более точный способ диагностики неисправностей этой системы.

Список литературы

[1] Taguchi G, Jugulum R. The Mahalanobis-Taguchi Strategy: A Pattern Technology System.- John Wiley and Sons, Inc, 2002.

[2] Kutano S, Mikami N, Aoyama K. Advanced gas turbine diagnostics using pattern recognition. - Proceedings of ASME Turbo Expo GT2011-45670, 2011.

[3] Нерубасский В. Использование системы Махаланобиса-Тагути в задачах распознавания неисправностей ГТД. - Авиационно-космическая техника и технология,No 9, 108-112, 2015.

[4] Liparas D, Angelis L, Feldt R. Applying the Mahalanobis-Taguchi strategy for software defect diagnosis. - Autom Softw Eng,No 19, 141–165, 2012.

[5] Adebolaji A Jobi-Taiwo. Data classiцcation and forecasting using the Mahalanobis-Taguchi method. - Master Theses, 2014.

[6] Elizabeth A. Cudney, Kioumars Paryani, Kenneth M. Ragsdel. Applying the Mahalanobis–Taguchi System to Vehicle Handling. - CONCURRENT ENGINEERING: Research and Applications, Vol.14 No6, 2006.

[7] John B, Kadadevarmath R. S. A methodology for quantitatively managing the bug fixing process using Mahalanobis Taguchi system. - Management Science Letters 5, 1081–1090, 2015.

4. Приложение

Ниже приведен код каждой программы, использовавшийся для вычисления всех результатов работы.

Файл Visualization for Correlations.py:

```
#Рисование графиков корреляций
file = 'X.xlsx'
import openpyxl as opx
import numpy as np
book=opx.load_workbook(file)
list=book.get_sheet_by_name('Sheet')
import matplotlib.pyplot as ris
ax=3
ay=2
n=11
m=1440
ax=ax-1
def get(k): #возвращает к-ый столбец матрицы данных X
            #k принимает от 1 до 11 включительно
    M = np.zeros(m)
    for i in range(0, m):
        M[i] = float(list.cell(row=i + ay, column=k + ax).value)
    return M
output=opx.Workbook()
ship=output.active
def ArrayToXL(M, s="Output",arraysize=m):
    st=".xlsx"
    ax = 3
    ay = 2
```

```
    for i in range(0, arraysize):
        ship.cell(row=i + ay, column=5 + ax).value = M[i]
    output.save(s+st)
def Lift(k,l,h=40): #Возвращает массив коэффициентов корреляции
    A=get(k)        #на интервалах велечиной h
    B=get(l)        #между параметрами k и l
    X=np.zeros(h)
    Y=np.zeros(h)
    C=np.zeros(m-h)
    for i in range(0,m-h):
        for j in range(0,h):
            X[j]=A[i+j]
            Y[j]=B[i+j]
        C[i]=np.corrcoef(X,Y)[0,1]
    return C
def smris1(k,l,h=40): #рисует график корреляций
    Y=Lift(k,l,h)
    X=np.zeros(Y.size)
    for i in range(0,Y.size):
        X[i]=i+1
    ris.scatter(X, Y)
    name=str(k)+' and '+str(l)
    ris.title(name)
    ris.savefig(name+".png",format='png',dpi=300)
    ris.clf()
for i in range(1,12):
    for j in range(1, 12):
        if i<j:
            smris1(i,j)
```

Файл ShiftsSearching.py:

```
#Поиск оптимальных сдвигов
file = 'X.xlsx'
import openpyxl as opx
import numpy as np
book=opx.load_workbook(file)
list=book.get_sheet_by_name('Sheet')
import matplotlib.pyplot as ris
output=opx.Workbook()
ship=output.active
ax=3
ay=2
n=11
m=1440
ax=ax-1
def get(k): #k принимает от 1 до 11 включительно
    M = np.zeros(m)
    for i in range(0, m):
        M[i] = float(list.cell(row=i + ay, column=k + ax).value)
    return M
def WriteToXL(M,i,j,h,d,n=1):
    ax = 3
    ay = 2
    n=2*n-1
    arraysize=M.shape[0]
    ship.cell(row=n+ay,column=ax-2).value=str(i)+' and '+str(j)
    ship.cell(row=n+ay, column=ax-1).value ='s'
    ship.cell(row=n+1+ay, column=ax-1).value ='Var'
```

```
    for i in range(0, arraysize):
        ship.cell(row=n + ay, column=i + ax).value = M[i,0]
        ship.cell(row=n+1+ay, column=i + ax).value = M[i,1]
def SaveToXL(h,d):
    s = 'h=' + str(h) + ", d=" + str(d)
    st = ".xlsx"
    output.save(s + st)
def Lift(k,l,h=40,s=0): #возвращает массив корреляций
    A=get(k)            #со сдвигом s
    B=get(l)
    X=np.zeros(h)
    Y=np.zeros(h)
    C=np.zeros(m-h-s)
    for i in range(0,m-h-s):
        for j in range(0,h):
            X[j]=A[i+j]
            Y[j]=B[i+j+s]
        C[i]=np.corrcoef(X,Y)[0,1]
    return C
def Var(k,l,h=40,d=60): #возвращает массив дисперсий
    D = np.zeros(d)     #при различных сдвигах параметров i и j
    for s in range(0,d):
       D[s]=np.var(Lift(k,l,h,s))
    return D
def sort(D,d=60):
    SD = np.sort(D)
    ArgSD = np.argsort(D)
    M = np.zeros((d, 2))
    M[:, 0] = ArgSD
```

```
    M[:, 1] = SD
    return M
def smris2(Y,k,l,h=40,show=True,d=60,n=0):
    X=np.zeros(Y.size)
    for i in range(0,Y.size):
        X[i]=i
    ris.scatter(X, Y)
    name=str(n)+'.'+str(k)+' and '+str(l)+".png"
    ris.savefig(name,format='png',dpi=300)
    ris.clf()
def main(h=40,d=60):
    n=0
    for i in range(1, 12):
        for j in range(1, 12):
            if i < j:
                n=n+1
                D=Var(i,j,h=h,d=d)
                M=sort(D,d)
                WriteToXL(M, i,j,h,d,n=n)
                smris2(D,i,j,h,show=0,d=d,n=n)
                n = n + 1
                D = Var(j, i, h=h, d=d)
                M = sort(D, d)
                WriteToXL(M, j, i, h, d, n=n)
                smris2(D, j, i, h, show=0, d=d,n=n)
    SaveToXL(h, d)

main()
```

Файл Preparation.py:

```
#Подготовка промежуточных данных для
#вычисления матрицы корреляций с учетом сдвигов
file = 'X.xlsx'
import openpyxl as opx
import numpy as np
book=opx.load_workbook(file)
list=book.get_sheet_by_name('Sheet')
output=opx.Workbook()
ship=output.active
ax=3
ay=2
n=11
m=1440
ax=ax-1
def get(k): #k принимает от 1 до 11 включительно
    M = np.zeros(m)
    for i in range(0, m):
        u=i + ay
        v=k + ax
        M[i] = float(list.cell(row=u, column=v).value)
    return M
def MatrixToXL(C, s="CorrOutput"):
    st=".xlsx"
    ax = 3
    ay = 2
    n = 11
    m = 1440
```

```
    for j in range(0, n):
        for i in range(0, n):
            u=i + ay
            v=j + ax
            ship.cell(row=u, column=v).value = C[i, j]
    output.save(s+st)
def Lift(k,l,h=40,s=0):
    A=get(k)
    B=get(l)
    X=np.zeros(h)
    Y=np.zeros(h)
    C=np.zeros(m-h-s)
    for i in range(0,m-h-s):
        for j in range(0,h):
            X[j]=A[i+j]
            Y[j]=B[i+j+s]
        C[i]=np.corrcoef(X,Y)[0,1]
    return C
def Var(k,l,h=40,d=60):
    D = np.zeros(d)
    for s in range(0,d):
       D[s]=np.var(Lift(k,l,h,s))
    return D
def sort(D,d=60):
    SD = np.sort(D)
    ArgSD = np.argsort(D)
    M = np.zeros((d, 2))
    M[:, 0] = ArgSD
    M[:, 1] = SD
```

```
    return M
def matrix(s='Matrix'):
    C=np.zeros((11,11))
    M=np.zeros((11, 11))
    for i in range(1, 12):
        for j in range(1, 12):
            if i < j:
                D=Var(i,j)
                n=np.argmin(D)
                C[i-1,j-1]=n
                K = Var(j, i)
                m=np.argmin(K)
                M[i-1,j-1]=m
    MatrixToXL(C,s+'1')
    MatrixToXL(M,s+'2')
def matrix2(s='Matrix'):
    C=np.zeros((11,11))
    M=np.zeros((11, 11))
    for i in range(1, 12):
        for j in range(1, 12):
            if not i == j:
                D=Var(i,j)
                n=np.argmin(D)
                C[i-1,j-1]=n
    MatrixToXL(C,s+'(not symmetric)')
def VarMatrix(s='VarMatrix'):
    C=np.zeros((11,11))
    M=np.zeros((11, 11))
    for i in range(1, 12):
```

```
        for j in range(1, 12):
            if i < j:
                D=Var(i,j)
                n=np.min(D)
                C[i-1,j-1]=n
                K = Var(j, i)
                m=np.min(K)
                M[i-1,j-1]=m
    MatrixToXL(C,s+'1')
    MatrixToXL(M,s+'2')
matrix('ShiftPatternForCorr')
matrix2('ShiftPatternForCorr')
VarMatrix()
```

Файл ShiftedCorrMatrix.py:

```
#Вычисление матрицы корреляций с учетом оптимальных сдвигов
#Вычисление массива средних для параметров X
#Вычисление стандартного отклонения для каждого параметра X
import openpyxl as opx
import numpy as np
import math as math
file = 'X.xlsx'
book=opx.load_workbook(file)
list=book.get_sheet_by_name('Sheet')
output=opx.Workbook()
ship=output.active
ax=3
ay=2
n=11
```

```
m=1440
def from_Xl_to_Matrix(s='ShiftPatternForCorr'):
    st='.xlsx'
    ax = 3
    ay = 2
    n = 11
    C=np.zeros((11,11))
    boook = opx.load_workbook(s+st)
    page = boook.get_sheet_by_name('Sheet')
    for j in range(0, n):
        for i in range(0, n):
            u=i + ay
            v=j + ax
            C[i, j]= float(page.cell(row=u, column=v).value)
    return C
def MatrixToXL(C, s="Output"):
    st=".xlsx"
    ax = 3
    ay = 2
    n = 11
    for j in range(0, n):
        for i in range(0, n):
            u=i + ay
            v=j + ax
            ship.cell(row=u, column=v).value = C[i, j]
    output.save(s+st)
def ArrayToXL(M, s="ArrayOutput",arraysize=11):
    st=".xlsx"
    ax = 3
```

```
    ay = 2
    for i in range(0, arraysize):
        ship.cell(row=ay, column=i + ax).value = M[i]
    output.save(s+st)
def get(k): #k принимает от 0 до 10 включительно
    M = np.zeros(m)
    for i in range(0, m):
        u=i + ay
        v=k + ax
        M[i] = float(list.cell(row=u, column=v).value)
    return M
def shiftedCorr(k,l,s=0): #Возвращает коэффициент корреляции
    A=get(k)              #для параметроа k и l со сдвигом s
    B=get(l)
    s=int(s)
    X=np.zeros(m-s)
    Y=np.zeros(m-s)
    for i in range(0, m - s):
        X[i] = A[i]
        Y[i] = B[i+s]
    c=np.corrcoef(X, Y)[0, 1]
    return c
def shiftedCorrMatrix(): #Возвращает матрицу корреляций
    C=np.zeros((11,11))  #с учетом оптимальных сдвигов
    S=np.zeros((11,11))
    N=from_Xl_to_Matrix('ShiftPatternForCorr1')
    M=from_Xl_to_Matrix('ShiftPatternForCorr2')
    V=from_Xl_to_Matrix('VarMatrix1')
    B=from_Xl_to_Matrix('VarMatrix2')
```

```
for i in range(0, 11):
    for j in range(0, 11):
        if i < j:
            if V[i,j]<B[i,j]:
                C[i,j]=shiftedCorr(i,j,s=N[i,j])
                S[i,j]=N[i,j]
            else:
                C[i,j]=shiftedCorr(j,i,s=M[i,j])
                S[i,j]=-M[i,j]
S[1-1, 9-1] = 29 #необходимые поправки к матрице корреляций
S[4-1, 7-1] = 16
S[4-1, 11-1] = 1
S[6-1, 10-1] = 17
S[7-1, 9-1] = 26
S[9-1, 11-1] = 34
C[1-1, 9-1] = shiftedCorr(1-1, 9-1, s=29)
C[4-1, 7-1] = shiftedCorr(4-1, 7-1, s=16)
C[4-1, 11-1] = shiftedCorr(4-1, 11-1, s=1)
C[6-1, 10-1] = shiftedCorr(6-1, 10-1, s=17)
C[7-1, 9-1] = shiftedCorr(7-1, 9-1, s=26)
C[9-1, 11-1] = shiftedCorr(9-1, 11-1, s=34)
S[2-1,9-1] = -36
S[2-1,10-1] = -22
S[4-1,10-1] = -37
S[8-1,9-1] = -13
C[2-1,9-1] = shiftedCorr(9-1,2-1, s=36)
C[2-1,10-1] = shiftedCorr(10-1,2-1, s=22)
C[4-1,10-1] = shiftedCorr(10-1,4-1, s=37)
C[8-1,9-1] = shiftedCorr(9-1,8-1, s=13)
```

```
    for i in range(0, 11):
        C[i,i]=1
        for j in range(0, 11):
            if i < j:
                C[j,i]=C[i,j]
                S[j,i]=S[i,j]
    MatrixToXL(C, s="ShiftedCorrMatrix(symmetric)")
    MatrixToXL(S, s="Shifts(symmetric)")
def means(): #возвращает массив средних для параметров X
    M=np.zeros(11)
    for i in range(0,11):
        M[i]=np.mean(get(i))
    ArrayToXL(M, s="Means")
def StandartDeviations(): #возвращает массив стандартных
    M=np.zeros(11)        #отклонений для параметров X
    for i in range(0,11):
        M[i]=math.sqrt(np.var(get(i)))
    ArrayToXL(M, s="SD")
means()
StandartDeviations()
shiftedCorrMatrix()
```

Файл MahalanobisDistance.py:

```
#Вычисление расстояния Махаланобиса для матрицы данных Y
#и отрисовка соответствующего графика
import openpyxl as opx
import numpy as np
file = 'Y.xlsx'
book=opx.load_workbook(file)
```

```
list=book.get_sheet_by_name('Sheet')
import matplotlib.pyplot as ris
output=opx.Workbook()
ship=output.active
ax=3
ay=2
n=11
m=672
def smris(Y):
    X=np.zeros(Y.size)
    for i in range(0,Y.size):
        X[i]=i
    ris.scatter(X, Y)
    ris.savefig("Mahalanobis.png", format='png', dpi=300)
    ris.show()
def Xl_to_Matrix(s='ShiftPatternForCorr'):
    st='.xlsx'
    ax = 3
    ay = 2
    n = 11
    C=np.zeros((11,11))
    boook = opx.load_workbook(s+st)
    page = boook.get_sheet_by_name('Sheet')
    for j in range(0, n):
        for i in range(0, n):
            u=i + ay
            v=j + ax
            C[i, j]= float(page.cell(row=u, column=v).value)
    return C
```

```
def MatrixToXL(C, s="Output"):
    st=".xlsx"
    ax = 3
    ay = 2
    n = 11
    for j in range(0, n):
        for i in range(0, n):
            u=i + ay
            v=j + ax
            ship.cell(row=u, column=v).value = C[i, j]
    output.save(s+st)
def ArrayToXL(M, s="ArrayOutput",arraysize=11):
    st=".xlsx"
    ax = 3
    ay = 2
    for i in range(0, arraysize):
        ship.cell(row=ay, column=i+ax).value=M[i]
    output.save(s+st)
def XL_To_Array(s="ArrayInput",arraysize=11):
    st=".xlsx"
    ax = 3
    ay = 2
    M=np.zeros(arraysize)
    boook = opx.load_workbook(s + st)
    page = boook.get_sheet_by_name('Sheet')
    for i in range(0, arraysize):
        M[i]= float(page.cell(row=ay, column=i+ax).value)
    return M
def geton(k):     #Возвращает к-ый параметр Y
```

```
                    # k от нуля до 671(m-1)
    M = np.zeros(n)
    for i in range(0, n):
        M[i] = float(list.cell(row=k + ay, column=i + ax).value)
    return M
def Mahalanobis():
    C=Xl_to_Matrix(s='CorrMatrix(this)')
    M=XL_To_Array('Means')
    S=XL_To_Array('SD')
    C=np.linalg.inv(C)
    MD=np.zeros(m)
    for k in range(0,m):
        x=geton(k)
        for i in range(0,n):
            x[i]=(x[i]-M[i])/S[i]
        A=np.dot(C,x)
        B=np.dot(x,A)
        MD[k]=(1/11)*B
    return MD
M=Mahalanobis()
ArrayToXL(M, s="Mahalanobis",arraysize=m)
smris(M)
```