НАВЧАЛЬО-НАУКОВИЙ КОМПЛЕКС

«ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ»

ПРИ НАЦІОНАЛЬНОМУ ТЕХНІЧНОМУ УНІВЕРСИТЕТІ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

Практична робота №5

з системного аналізу

**«Виявлення пріоритетностей галузей і технологій військово-промислового комплексу на основі методу морфологічного аналізу»**

Бригада № 3

Виконали:

студенти 4 курсу

групи КА-71

Бойко Павло  
Зінченко Світлана

групи КА-77

Худіков Павло

Київ 2021

**Завдання**

Приймається рішення щодо пріоритетності галузей і технологій військово-промислового комплексу, опис якого у вигляді морфологічної таблиці заданий в табл. 2. Рішення приймається в умовах різних потенційних обставин у світі і в країні (табл. 1).

Табл. 1. Ситуація в країні і в світі

|  |  |  |  |
| --- | --- | --- | --- |
| 1. Напруженість в світі | 2. Напруженість в країні | 3. Стан економіки | 4. Терористична загроза |
| 1.1. Відносна стабільність | 2.1. Затишшя | 3.1. Оптимістичний | 4.1. Сильна |
| 1.2. Холодна війна | 2.2. Заморожений конфлікт, збройні провокації | 3.2. Стабільний | 4.2. Помірна |
| 1.3. Локальні конфлікти | 2.3. Збройне протистояння | 3.3. Спад | 4.3. Слабка |
| 1.4. Локальні війни | 2.4. Повномасштабна агресія іншої держави | 3.4. Криза/дефолт |

Таблиця 2. Морфологічна таблиця рішень щодо ВПК

|  |  |  |
| --- | --- | --- |
| 5. Галузь | 6. Технології | 7. Експорт/використання |
| 5.1. Піхотне озброєння | 6.1. Приладобудування і електроніка | 7.1. Переважно внутрішнє використання |
| 5.2. Наземна техніка | 6.2. Машинобудування | 7.2. Переважно експорт |
| 5.3. Повітряна техніка | 6.3. Новітні матеріали |
| 5.4. Протиповітряна оборона, ракетні війська | 6.4. Оптика |
| 5.5. Розвідка | 6.5. Технології зв’язку |
| 5.6. Зв’язок і радіоелектронна боротьба |

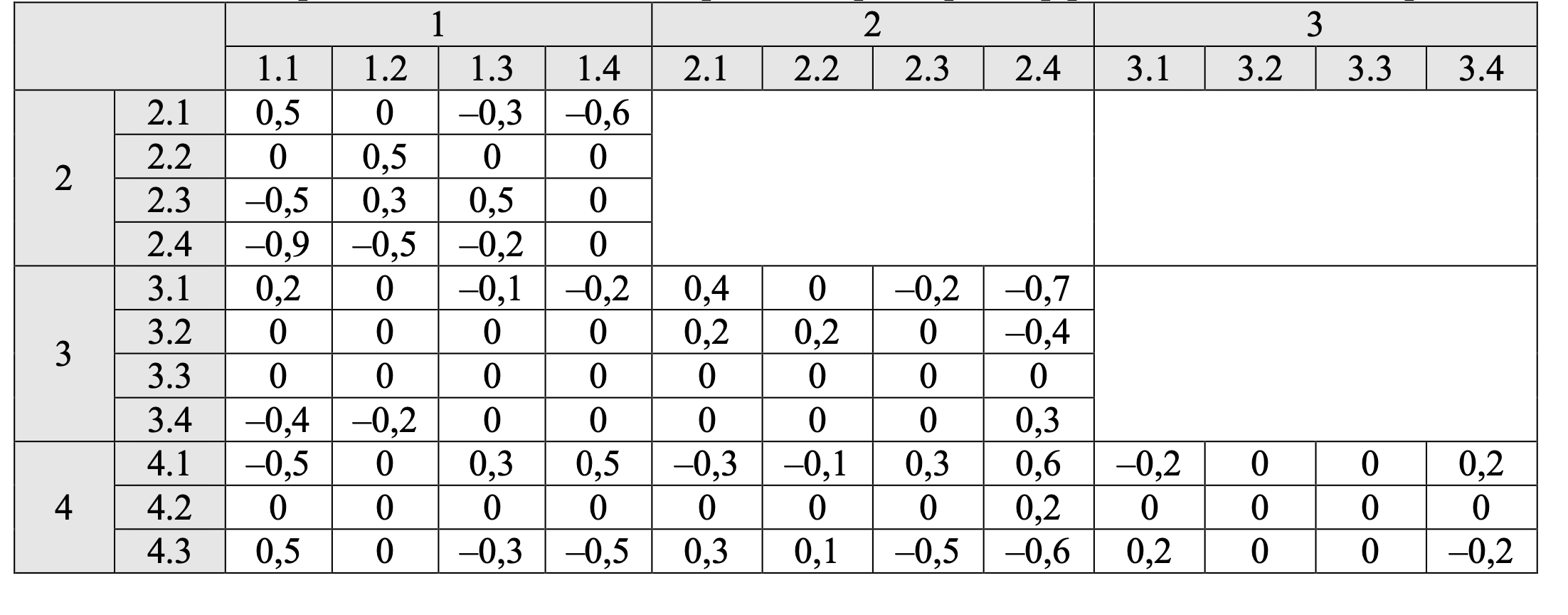
Попередні оцінки ймовірності від експертів задані в табл. 3.

Табл. 3. Попередні оцінки ймовірності альтернатив параметрів

|  |  |  |  |
| --- | --- | --- | --- |
| 1. Напруженість в світі | 2. Напруженість в країні | 3. . Стан економіки | 4. Терористична загроза |
| 0,05 | 0,2 | 0,3 | 0,35 |
| 0,35 | 0,7 | 0,65 | 0,8 |
| 0,65 | 0,4 | 0,25 | 0,2 |
| 0,2 | 0,1 | 0,05 |

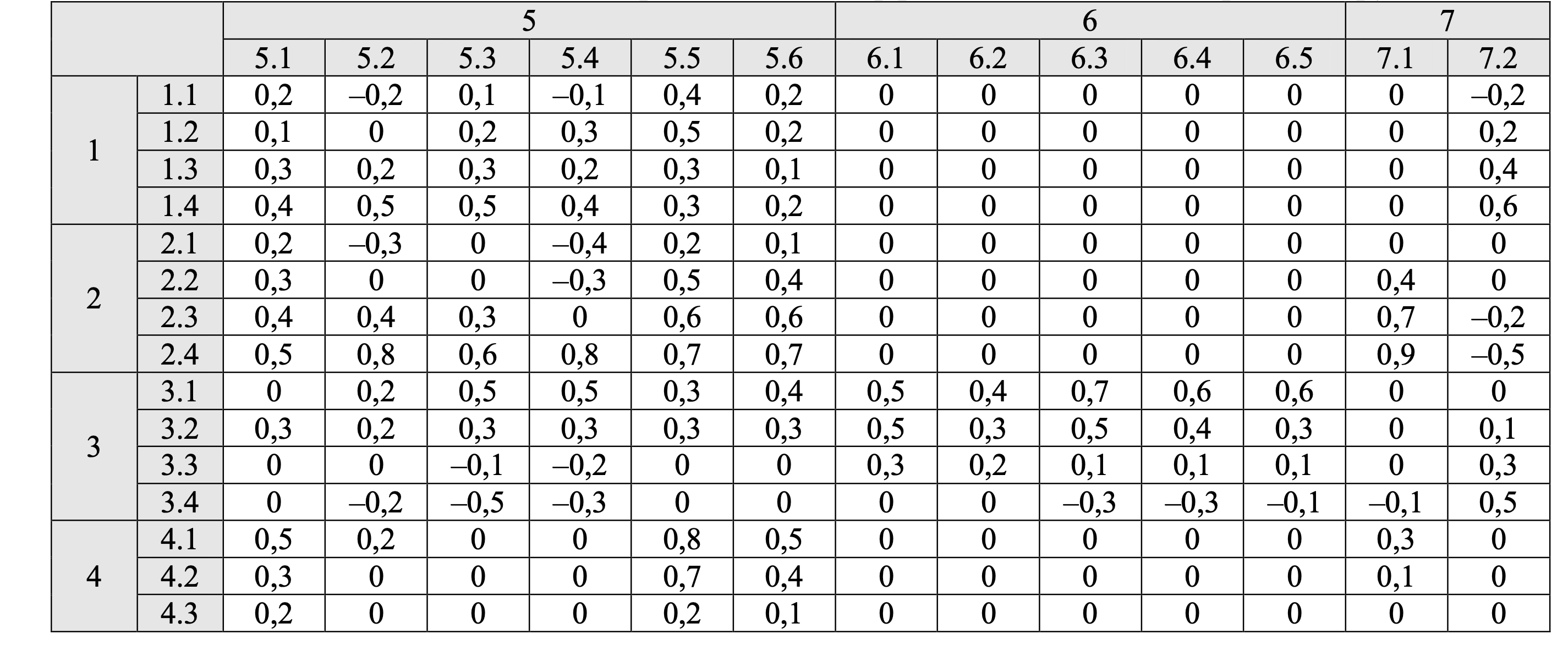
Матриця взаємозв’язків альтернатив параметрів морфологічної таблиці першого етапу задана в табл. 4.

Табл. 4. Матриця взаємозв’язків альтернатив параметрів морфологічної таблиці першого етапу



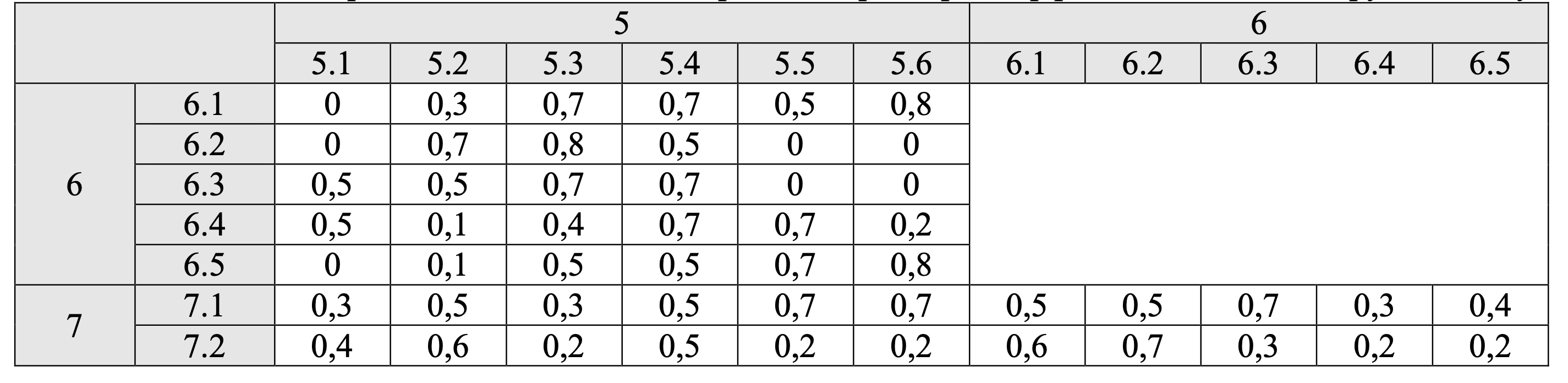
Матриця зв’язків морфологічних таблиць першого і другого етапів задана в табл. 5.

Табл. 5. Матриця зв’язків морфологічних таблиць першого і другого етапів



Крім того, параметри морфологічної таблиці другого етапу пов’язані між собою матрицею взаємозв’язків (табл. 6).

Табл. 6. Матриця взаємозв’язків альтернатив параметрів морфологічної таблиці другого етапу



Потрібно:

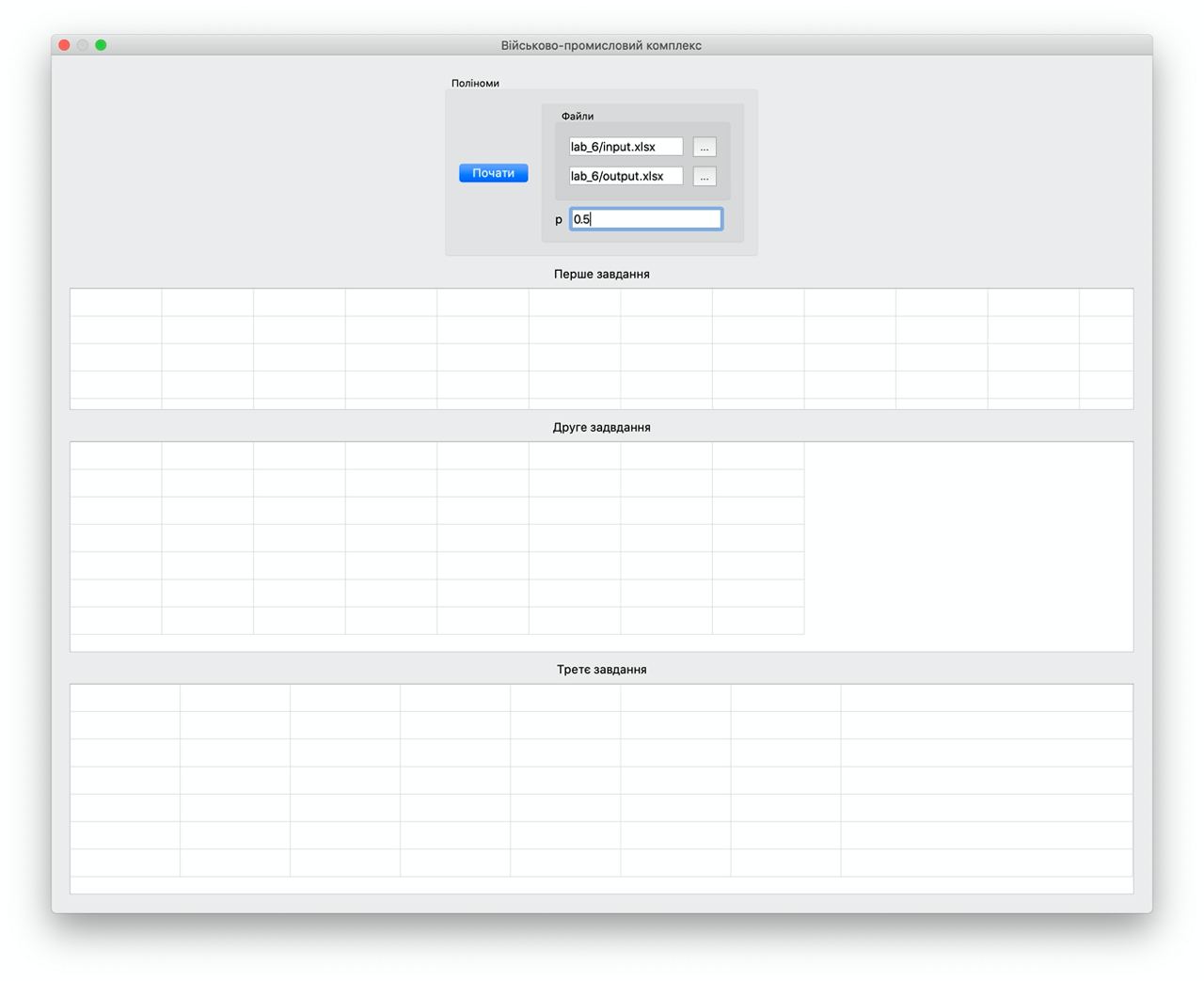
•  Знайти розподіл ймовірностей альтернатив параметрів морфологічної таблиці для ситуації в країні і в світі, якщо задані матриця попередніх оцінок альтернатив (табл. 3) і матриця взаємозв’язків (табл. 4).

•  Розрахувати незалежні очікувані результативності альтернативи рішень щодо пріоритетності галузей ВПК, виходячи з результатів розв'язку попередньої задачі і матриці зв’язків (табл. 5).

•  Розрахувати очікувані результативності альтернатив рішень з урахуванням залежностей між параметрами, заданих матрицею взаємозв’язків (табл. 6), використовуючи в якості незалежних оцінок результати розв'язку попередньої задачі.

**Інтерфейс програми**

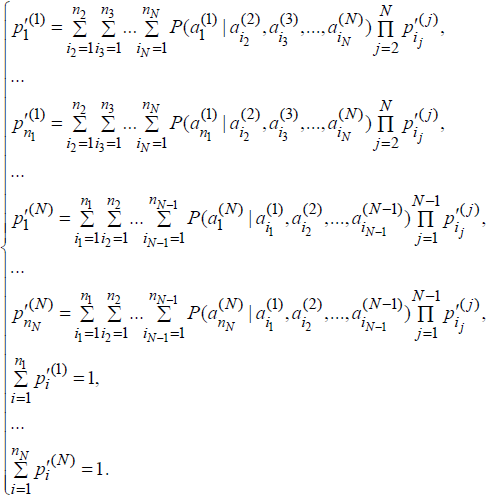
1. Користувач повинен увести файл, з якого буде зчитуватись вхідна інформація (файл з варіантом нашої задачі за замовчанням лежить у lab\_6 та називаеться input.xlsx).
2. Користувач повинен увести файл, куди запишуться результати програми. Також результати можна буде побачити у самих таблицях.
3. Користувач повинен увести значення p (від 0 до 1), яке використувується у подальших формулах.

****

**Завдання 1**

Знайти розподіл ймовірностей альтернатив параметрів морфологічної таблиці для ситуації в країні і в світі, якщо задані матриця попередніх оцінок альтернатив (табл. 3) і матриця взаємозв’язків.

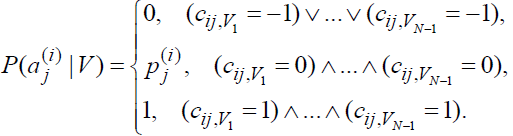
Перерахунок будемо виконувати з використання формули повної імовірності, або схеми Байєса:



Ми отримали систему нелінійних рінянь (в нашому випадку з 15 невідомими та 19 рівняннями).

Невідомими тут є величини , де K=, nj – кількість альтернатив.

Умовні імовірності будемо апроксимувати поліномом 3-го ступеня. Коефіцієнти полінома будемо знаходити виходячи з наступних міркувань:



Де с знаходимо наступним чином:

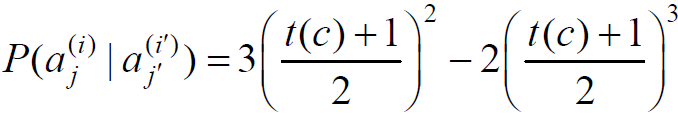


У випадку більше ніж двох характеристичних параметрів умовна ймовірність залежить від декількох значень з таблиці взаємозв’язків. Вони зводяться до одного за допомогою такої процедури:

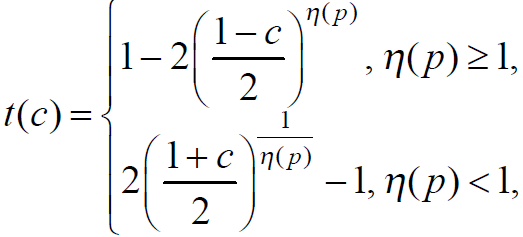
начення з матриці взаємозв’язків, від яких залежить умовна ймовірність, відобразимо на множину за допомогою перетворення.

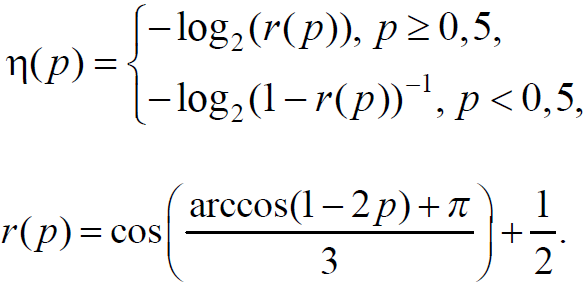
Далі обернено відображаємо отримані значення .

Апроксимацію будуємо на основі полінома:



Де



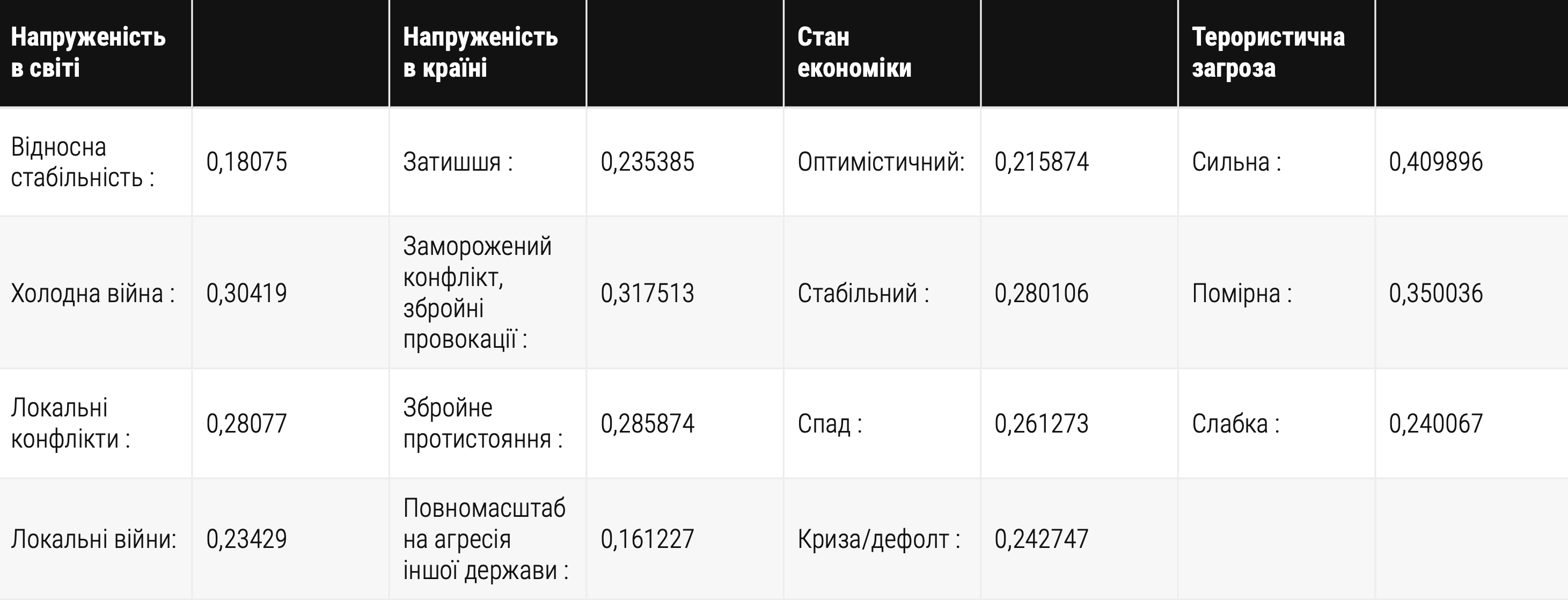


Ці формули підібрані так, щоб виконувалися інтерполяція полінома в точці 0.

Підсталяючи отримані значення умовних імовірностей ми отримали можливість вирішувати системи.

За початкові значення ми взяли імовірності з Таблиці 1. Потім ітераційно підставляючи до системи рівнянь (перших 15), отримували нові значення. Після цього ми їх нормували, щоб виконувалися останні 4 рівняння.

Отже, ми отримали такі нові значення імовірностей з урахуванням взаємозв’язків.

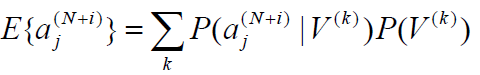


Представимо порівняльний аналіз нових і старих значень імовірностей:

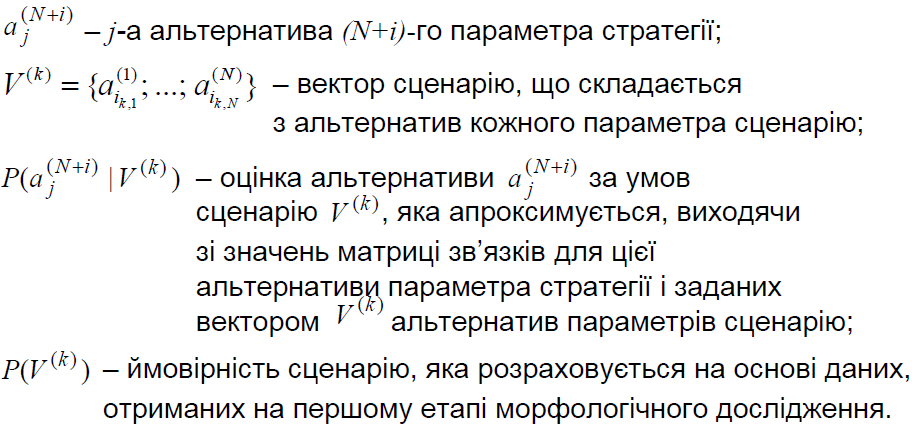
**Завдання 2**

Розрахувати незалежні очікувані результативності альтернативи рішень щодо пріоритетності галузей ВПК, виходячи з результатів розв'язку попередньої задачі і матриці зв’язків (табл. 5).

Результативність рішень будемо знаходити з наступної формули:



Де



При підрахунку умовних ймовірностей, ми будемо користуватись тими же методами апроксимації, що і в першому завданні. В точці 0 ми беремо значення результативності за 0.25 з розрахунку, що якщо альтернативи не мають ніякого впливу, то ми можемо взяти їх рівними.

Отже, ми отримали наступну таблицу:

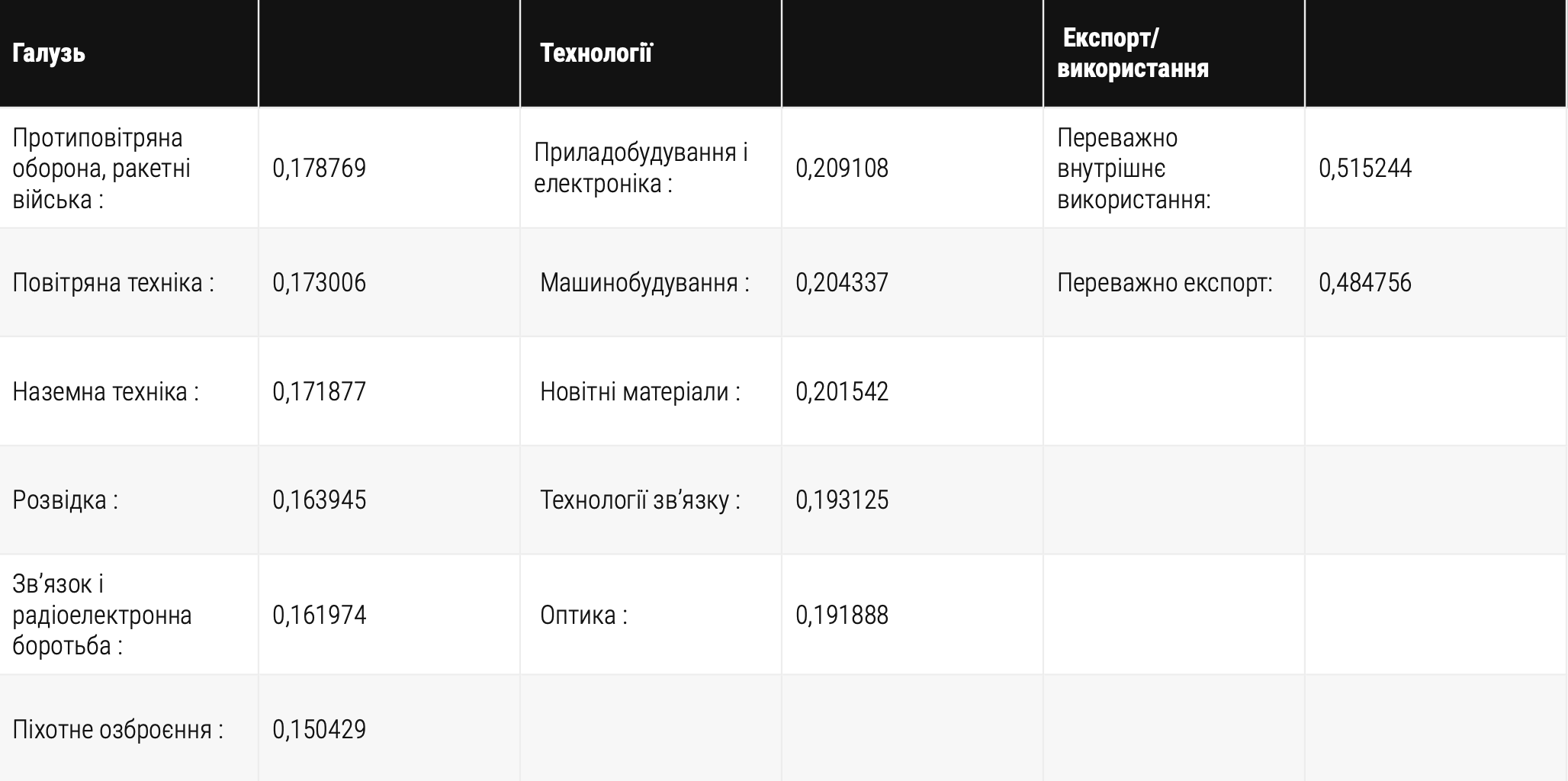


**Завдання 3**

Розрахувати очікувані результативності альтернатив рішень з урахуванням залежностей між параметрами, заданих матрицею взаємозв’язків (табл. 6), використовуючи в якості незалежних оцінок результати розв'язку попередньої задачі.

Таким самим методом, як і в першому завданні ми розрахуємо результативність, маючи на увазі зв’язки між альтернативами.

Результат:



**Висновки:**

Приймається рішення щодо пріоритетності галузей ВПК в умовах майбутньої ситуації в країні і в світі. З морфологічного аналізу ми отримали, що найбільш пріорітетними при існуючих ймовірностях впливу зовнішніх факторів та взаємозв’язків є :

Для галузей – Протиповітряна оборона, ракетні війська

Для технологій – Приладобудування і електроніка

Для експорту/використання – Переважно внутрішнє використання

Список літератури:

1. Панкратова Н.Д., Савченко І.О. Застосування методу морфологічного аналізу до задач технологічного передбачення // Наукові праці / Миколаївський держ. гуманітарний ун-т ім. Петра Могили комплексу НаУКМА. Сер. Комп’ютерні технології, системний аналіз, моделювання. — 2008. — 90, вип. 77.
2. Morphological Analysis – A Problem Solving Method By An Astrophysicist Who Discovered Dark Matter <https://northstar.greyb.com/morphological-analysis/>
3. Ritchey T. Strategic Decision Support using Computerised Morphological Analysis // 9th International Command and Control Research and Technology Symposium, Copenhagen, September 2004. — Copenhagen, 2004.
4. Using Morphological Analysis for Innovation and Resource and Development: An Invaluable Tool for Entrepreneurship https://www.scmspune.ac.in/chapter/pdf/Chapter%203.pdf
5. The Norris Brothers Ltd. morphological approach to engineering design – an early example of applied morphological analysis http://www.amg.swemorph.com/pdf/amg-3-2-2014.pdf

Лістинг:

**Main**

import logging

import os

import sys

from lab\_6.solver import Solver

import xlrd

import math

import openpyxl

from copy import deepcopy

from PyQt5.uic import loadUiType

import sys

from PyQt5.QtCore import pyqtSlot, pyqtSignal, QTimer

from PyQt5.QtWidgets import QApplication,QTableWidgetItem, QDialog, QFileDialog, QMessageBox

app = QApplication(sys.argv)

app.setApplicationName('LABA 6')

form\_class, base\_class = loadUiType('lab\_6/main\_window\_2.ui')

data = {'col1':['1','2','3','4'],

'col2':['1','2','1','3'],

'col3':['1','1','2','1']}

class MainWindow\_2(QDialog, form\_class):

# signals:

input\_changed = pyqtSignal('QString')

output\_changed = pyqtSignal('QString')

def \_\_init\_\_(self, parent = None):

super(MainWindow\_2, self).\_\_init\_\_(parent)

# setting up ui

self.setupUi(self)

self.setWindowTitle('Військово-промисловий комплекс')

self.input\_path = self.line\_input.text()

if len(self.input\_path) == 0:

self.input\_path = os\_path

self.output\_path = './output.xlsx'

#set tablewidget

self.tablewidget.verticalHeader().hide()

self.tablewidget\_first.verticalHeader().hide()

self.tablewidget\_second.verticalHeader().hide()

self.tablewidget.horizontalHeader().hide()

self.tablewidget\_first.horizontalHeader().hide()

self.tablewidget\_second.horizontalHeader().hide()

#self.tablewidget.setRowCount(0)

'''

column\_size = [60, 70, 100, 100,200, 60, 200, 80]

for index, size in enumerate(column\_size):

self.tablewidget.setColumnWidth(index,size)

'''

return

@pyqtSlot()

def input\_clicked(self):

filename = QFileDialog.getOpenFileName(self, 'Відкрити данні', '.', 'Data file (\*.xlsx)')[0]

if filename == '':

return

if filename != self.input\_path:

self.input\_path = filename

self.input\_changed.emit(filename)

return

@pyqtSlot('QString')

def input\_modified(self, value):

if value != self.input\_path:

self.input\_path = value

return

@pyqtSlot()

def output\_clicked(self):

filename = QFileDialog.getSaveFileName(self, 'Save data file', '.', 'Spreadsheet (\*.xlsx)')[0]

if filename == '':

return

if filename != self.output\_path:

self.output\_path = filename

self.output\_changed.emit(filename)

return

@pyqtSlot('QString')

def output\_modified(self, value):

if value != self.output\_path:

self.output\_path = value

return

@pyqtSlot()

def exec\_clicked(self):

self.exec\_button.setEnabled(False)

try:

prob = 0.5

p\_in = xlrd.open\_workbook(self.input\_path)

p\_out = openpyxl.Workbook()

s = Solver(prob)

s.load\_data(p\_in)

tmp, output1 = s.solve\_task1(p\_out.create\_sheet('Task 1'), self.tablewidget\_first)

tmp, output2 = s.solve\_task2(p\_out.create\_sheet('Task 2'), tmp,self.tablewidget\_second)

tmp, output3 = s.solve\_task3(p\_out.create\_sheet('Task 3'), tmp, self.tablewidget)

p\_out.save(self.output\_path)

except Exception as e:

QMessageBox.warning(self,'Error!','Error happened during execution: ' + str(e))

self.exec\_button.setEnabled(True)

return

**Solver**

import xlrd

import math

import openpyxl

from copy import deepcopy

import numpy as np

from PyQt5.QtWidgets import QTableWidgetItem

class Solver:

def \_\_init\_\_(self, p):

self.AlternativesNames = []

self.Alternatives = []

self.OutcomeNames = []

self.Outcomes = []

self.Alt\_Expert\_Prob = []

self.Alternatives\_Dependencies = []

self.Outcomes\_Dependencies = []

self.Alternatives\_Outcomes\_Crossdependencies = []

self.p = p

def solve\_task1(self, outsheet, table):

for i in range(len(self.Alternatives)):

summ = 0

for j in range(len(self.Alternatives[i])):

summ = summ + self.Alt\_Expert\_Prob[i][j]

for j in range(len(self.Alternatives[i])):

self.Alt\_Expert\_Prob[i][j] = self.Alt\_Expert\_Prob[i][j] / summ

alt\_prob = self.modify\_probabilities(self.Alt\_Expert\_Prob, self.Alternatives\_Dependencies)

row = 0

column = 0

for i in range(len(self.Alternatives)):

outsheet.cell(1, 3\*i+1).value = self.AlternativesNames[i]

for j in range(len(self.Alternatives[i])):

outsheet.cell(2+j, 3\*i+1).value = self.Alternatives[i][j] + ' :'

outsheet.cell(2+j, 3\*i+2).value = str(alt\_prob[i][j])

if 2 + j > row:

row = 2 + j

if 3 \* i + 2 > column:

column = 3 \* i + 2

result = np.empty((row, column), dtype='object')

for i in range(len(self.Alternatives)):

newitem = QTableWidgetItem(str(self.AlternativesNames[i]))

table.setItem(0, 3\*i,newitem)

result[0, 3\*i] = self.AlternativesNames[i]

for j in range(len(self.Alternatives[i])):

newitem = QTableWidgetItem(str(self.Alternatives[i][j] + ' :'))

table.setItem(1+j, 3\*i,newitem )

newitem = QTableWidgetItem(str(alt\_prob[i][j]))

table.setItem(1+j, 3\*i+1, newitem)

result[1+j, 3\*i] = self.Alternatives[i][j] + ' :'

result[1+j, 3\*i+1] = str(alt\_prob[i][j])

table.resizeColumnsToContents()

table.resizeRowsToContents()

return alt\_prob, result

def solve\_task2(self, outsheet, alt\_prob, table):

outcome\_prob = self.calculate\_outcome\_prob(alt\_prob)

outcomes\_c = [None for k in range(len(self.Outcomes))]

for i in range(len(self.Outcomes)):

outcomes\_c[i] = deepcopy(self.Outcomes[i])

sort = sorted(zip(outcome\_prob[i], outcomes\_c[i]), reverse=True)

outcomes\_c[i] = [x for y, x in sort]

outcome\_prob[i] = [y for y, x in sort]

row = 0

column = 0

for i in range(len(self.Outcomes)):

outsheet.cell(1, 3\*i+1).value = self.OutcomeNames[i]

for j in range(len(self.Outcomes[i])):

outsheet.cell(2+j, 3\*i+1).value = outcomes\_c[i][j] + ' :'

outsheet.cell(2+j, 3\*i+2).value = str(outcome\_prob[i][j])

if 2 + j > row:

row = 2 + j

if 3 \* i + 2 > column:

column = 3 \* i + 2

result = np.empty((row, column), dtype='object')

for i in range(len(self.Outcomes)):

newitem = QTableWidgetItem(str(self.OutcomeNames[i]))

table.setItem(0, 3\*i,newitem)

result[0, 3\*i] = self.OutcomeNames[i]

for j in range(len(self.Outcomes[i])):

newitem = QTableWidgetItem(str(outcomes\_c[i][j] + ' :'))

table.setItem(1+j, 3\*i,newitem )

newitem = QTableWidgetItem(str(outcome\_prob[i][j]))

table.setItem(1+j, 3\*i+1, newitem)

result[1+j, 3\*i] = outcomes\_c[i][j] + ' :'

result[1+j, 3\*i+1] = str(outcome\_prob[i][j])

table.resizeColumnsToContents()

table.resizeRowsToContents()

return outcome\_prob, result

def solve\_task3(self, outsheet, outcome\_prob, table):

outcome\_mod\_prob = self.modify\_probabilities(outcome\_prob, self.Outcomes\_Dependencies)

outcomes\_c = [None for k in range(len(self.Outcomes))]

for i in range(len(self.Outcomes)):

outcomes\_c[i] = deepcopy(self.Outcomes[i])

sort = sorted(zip(outcome\_mod\_prob[i], outcomes\_c[i]), reverse=True)

outcomes\_c[i] = [x for y, x in sort]

outcome\_mod\_prob[i] = [y for y, x in sort]

row = 0

column = 0

for i in range(len(self.Outcomes)):

outsheet.cell(1, 3 \* i + 1).value = self.OutcomeNames[i]

for j in range(len(self.Outcomes[i])):

outsheet.cell(2 + j, 3 \* i + 1).value = outcomes\_c[i][j] + ' :'

outsheet.cell(2 + j, 3 \* i + 2).value = str(outcome\_mod\_prob[i][j])

outsheet.cell(2, 3\*i+1).font = openpyxl.styles.Font(bold=True)

outsheet.cell(2, 3\*i+2).font = openpyxl.styles.Font(bold=True)

if 2 + j > row:

row = 2 + j

if 3 \* i + 2 > column:

column = 3 \* i + 2

result = np.empty((row, column), dtype='object')

for i in range(len(self.Outcomes)):

newitem = QTableWidgetItem(str(self.OutcomeNames[i]))

table.setItem(0, 3\*i,newitem)

result[0, 3\*i] = self.OutcomeNames[i]

for j in range(len(self.Outcomes[i])):

newitem = QTableWidgetItem(str(outcomes\_c[i][j] + ' :'))

table.setItem(1+j, 3\*i,newitem )

newitem = QTableWidgetItem(str(outcome\_mod\_prob[i][j]))

table.setItem(1+j, 3\*i+1, newitem)

result[1+j, 3\*i] = outcomes\_c[i][j] + ' :'

result[1+j, 3\*i+1] = str(outcome\_mod\_prob[i][j])

table.resizeColumnsToContents()

table.resizeRowsToContents()

return outcome\_mod\_prob, result

def modify\_probabilities(self, starting\_prob, dep\_matrix):

cur = 0

l = len(starting\_prob)

res = [None for k in range(2)]

res[0] = [None for k in range(l)]

res[1] = [None for k in range(l)]

for i in range(l):

res[0][i] = [None for k in range(len(starting\_prob[i]))]

res[1][i] = [None for k in range(len(starting\_prob[i]))]

for i in range(l):

for j in range(len(starting\_prob[i])):

res[1][i][j] = starting\_prob[i][j]

for iteration in range(500):

for i in range(l):

for j in range(len(starting\_prob[i])):

n = [0 for k in range(l)]

res[cur][i][j] = 0

while True:

# calculating of probabilities

cur\_dep = [None for k in range(l - 1)]

for k in range(l):

if k > i:

cur\_dep[k - 1] = dep\_matrix[i][k][j][n[k]]

elif k < i:

cur\_dep[k] = dep\_matrix[i][k][j][n[k]]

p\_a = self.calculate\_p(cur\_dep)

for k in range(l):

if k == i:

continue

p\_a = p\_a \* res[1 - cur][k][n[k]]

res[cur][i][j] = res[cur][i][j] + p\_a

# modification of index array

cn = l - 1

while cn >= 0:

n[cn] = n[cn] + 1

if n[cn] == len(starting\_prob[cn]):

n[cn] = 0

cn = cn - 1

else:

break

if cn < 0:

break

# norming the probabilities

summ = 0

for j in range(len(starting\_prob[i])):

summ = summ + res[cur][i][j]

for j in range(len(starting\_prob[i])):

res[cur][i][j] = res[cur][i][j] / summ

cur = 1 - cur

return res[cur]

def calculate\_outcome\_prob(self, alt\_prob):

res = [None for k in range(len(self.Outcomes))]

for i in range(len(res)):

res[i] = [0 for k in range(len(self.Outcomes[i]))]

for i in range(len(self.Outcomes)):

for j in range(len(self.Outcomes[i])):

n = [None for k in range(len(self.Alternatives))]

for k in range(len(self.Alternatives)):

n[k] = 0

while True:

# calculating of probabilities

cur\_dep = [None for k in range(len(self.Alternatives))]

for k in range(len(self.Alternatives)):

cur\_dep[k] = self.Alternatives\_Outcomes\_Crossdependencies[k][i][n[k]][j]

p\_a = self.calculate\_p(cur\_dep)

for k in range(len(self.Alternatives)):

if k == i:

continue

p\_a = p\_a \* alt\_prob[k][n[k]]

res[i][j] = res[i][j] + p\_a

#modification of index array

cn = len(self.Outcomes) - 1

while cn >= 0:

n[cn] = n[cn] + 1

if n[cn] == len(self.Alternatives[cn]):

n[cn] = 0

cn = cn - 1

else:

break

if cn < 0:

break

summ = 0

for j in range(len(self.Outcomes[i])):

summ = summ + res[i][j]

for j in range(len(self.Outcomes[i])):

res[i][j] = res[i][j] / summ

return res

def calculate\_p(self, a\_i):

c = 1

for i in range(len(a\_i)):

c = c \* (2 / (1 - a\_i[i]) - 1)

c = 1 - 2 / (c + 1)

rp = math.cos((math.acos(1 - 2 \* self.p) + math.pi) / 3) + 0.5

if rp <= 0.5:

np = -1 \* math.log(rp, 2)

else:

np = -1 / math.log(1 - rp, 2)

if np >= 1:

tc = 1 - 2 \* math.pow((1 - c) / 2, np)

else:

tc = 2 \* math.pow((1 + c) / 2, 1 / np) - 1

return 3 \* ((tc + 1) / 2) \* ((tc + 1) / 2) - 2 \* ((tc + 1) / 2) \* ((tc + 1) / 2) \* ((tc + 1) / 2)

def load\_data(self, file):

sheet = file.sheet\_by\_index(0)

# inputting alternatives

alt\_count, out\_count = None, None

for i in range(1000):

if str(sheet.row\_values(0)[i]) == '\*':

alt\_count = i

break

max\_row = 0

for i in range(alt\_count):

self.AlternativesNames.append(sheet.row\_values(0)[i])

max\_ind = None

for j in range(1, 1000):

if sheet.row\_values(j+1)[i] == '\*':

max\_ind = j

break

self.Alternatives.append([])

self.Alt\_Expert\_Prob.append([])

for j in range(max\_ind):

self.Alternatives[i].append(sheet.row\_values(j+1)[i])

self.Alt\_Expert\_Prob[i].append(None)

if max\_row < max\_ind:

max\_row = max\_ind

# inputting outcomes

base\_row = max\_row + 2

for i in range(1000):

if sheet.row\_values(base\_row)[i] == '\*':

out\_count = i

break

max\_row = 0

for i in range(out\_count):

self.OutcomeNames.append(sheet.row\_values(base\_row)[i])

max\_ind = None

for j in range(1000):

if sheet.row\_values(j + 1 + base\_row)[i] == '\*':

max\_ind = j

break

self.Outcomes.append([])

for j in range(max\_ind):

self.Outcomes[i].append(sheet.row\_values(j + 1 + base\_row)[i])

if max\_row < max\_ind:

max\_row = max\_ind

base\_row = base\_row + max\_row + 3

# inputting alternative probabilities

max\_row = 0

for i in range(alt\_count):

for j in range(len(self.Alt\_Expert\_Prob[i])):

self.Alt\_Expert\_Prob[i][j] = float(sheet.row\_values(base\_row + j)[i])

base\_row = 1

# inputting alt-alt dependency matrix

sheet = file.sheet\_by\_index(1)

self.Alternatives\_Dependencies = [[None for h in range(alt\_count)] for k in range(alt\_count)]

for i in range(alt\_count - 1):

base\_column = 1

for j in range(1, alt\_count):

if j <= i:

base\_column = base\_column + len(self.Alternatives[j])

continue

li = len(self.Alternatives[i])

lj = len(self.Alternatives[j])

self.Alternatives\_Dependencies[i][j] = [[None for h in range(lj)] for k in range(li)]

self.Alternatives\_Dependencies[j][i] = [[None for h in range(li)] for k in range(lj)]

for i1 in range(li):

for j1 in range(lj):

if sheet.row\_values(base\_row + i1)[base\_column + j1] == '\*':

self.Alternatives\_Dependencies[i][j][i1][j1] = 0

self.Alternatives\_Dependencies[j][i][j1][i1] = 0

else:

self.Alternatives\_Dependencies[i][j][i1][j1] = float(

sheet.row\_values(base\_row + i1)[base\_column + j1])

self.Alternatives\_Dependencies[j][i][j1][i1] = float(

sheet.row\_values(base\_row + i1)[base\_column + j1])

base\_column = base\_column + len(self.Alternatives[j])

base\_row = base\_row + len(self.Alternatives[i])

base\_row = base\_row + 2

# inputting alt\_out dependency matrix

self.Alternatives\_Outcomes\_Crossdependencies = [[None for h in range(out\_count)] for k in range(alt\_count)]

for i in range(alt\_count):

base\_column = 1

for j in range(out\_count):

li = len(self.Alternatives[i])

lj = len(self.Outcomes[j])

self.Alternatives\_Outcomes\_Crossdependencies[i][j] = [[None for h in range(lj)] for k in range(li)]

for i1 in range(li):

for j1 in range(lj):

if sheet.row\_values(base\_row + i1)[base\_column + j1] == '\*':

self.Alternatives\_Outcomes\_Crossdependencies[i][j][i1][j1] = 0

else:

self.Alternatives\_Outcomes\_Crossdependencies[i][j][i1][j1] = float(

sheet.row\_values(base\_row + i1)[base\_column + j1])

base\_column = base\_column + len(self.Outcomes[j])

base\_row = base\_row + len(self.Alternatives[i])

base\_row = base\_row + 2

# inputting out-out dependency matrix

self.Outcomes\_Dependencies = [[None for h in range(out\_count)] for k in range(out\_count)]

for i in range(out\_count):

base\_column = 1

for j in range(1, out\_count):

if j <= i:

base\_column = base\_column + len(self.Outcomes[j])

continue

li = len(self.Outcomes[i])

lj = len(self.Outcomes[j])

self.Outcomes\_Dependencies[i][j] = [[None for h in range(lj)] for k in range(li)]

self.Outcomes\_Dependencies[j][i] = [[None for h in range(li)] for k in range(lj)]

for i1 in range(li):

for j1 in range(lj):

if sheet.row\_values(base\_row + i1)[base\_column + j1] == '\*':

self.Outcomes\_Dependencies[i][j][i1][j1] = 0

self.Outcomes\_Dependencies[j][i][j1][i1] = 0

else:

self.Outcomes\_Dependencies[i][j][i1][j1] = float(

sheet.row\_values(base\_row + i1)[base\_column + j1])

self.Outcomes\_Dependencies[j][i][j1][i1] = float(

sheet.row\_values(base\_row + i1)[base\_column + j1])

base\_column = base\_column + len(self.Outcomes[j])

base\_row = base\_row + len(self.Outcomes[i])