МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

Національний технічний університет
«Харківський політехнічний інститут»
Кафедра КМАД

# ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ №5

Виконав студент:

Омельніцький Андрій Миколайович
Група: КН-120А

Перевірила:

Ольга Василівна Костюк

Харків, 2023 г.

# Зміст

**Глава 1**

# Мета роботи

Побудова математичної моделі двохсекторної економіки, дослідження моделі із використанням комп'ютерного моделювання.

Порядок виконання:

1. Знайти розв‘язки системи (1-21), використовуючи чисельні методи розв'язання систем диференціальних рівнянь, за допомогою вбудованих функцій пакетів прикладних програм та отримати часові характеристики основних економічних показників для заданих параметрів моделі.
2. Вивести графіки динаміки основних економічних показників для обраних початкових значень.
3. Виконати моделювання й оцінити якісні зміни основних економічних показників, варіюючи норми оподаткування та норми відрахування на пригнічення тіньового сектора.
4. За результатами моделювання надати рекомендації щодо вибору раціональних норм оподаткування.
5. Усі результати, отримані в ході виконання роботи, занести до звіту. Зробити висновки.

## Глава 2

# Основна частина

## 2.1. Пункти 1-2

Розв'яжемо систему (1-21) за завданих початковіх значень.
Значення параметрів моделі (1-21):
$\alpha = 0.5; \beta = 1.5; \gamma = 1.5; \delta = 0.1; \nu = 5; \mu = 20; \lambda = 20; \rho = 10;$
$A0 = 1; L0 = 1; D0 = 1; \tau = 0.6; \sigma = 0.5;$

Початкові умови:

$$\begin{pmatrix} \prod_1 \\ p_1 \\ w_1 \\ K_1 \\ \prod_2 \\ p_2 \\ w_2 \\ K_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0.5 \\ 0.25 \\ 0.1 \\ 0 \\ 0.5 \\ 0.25 \\ 0.1 \end{pmatrix}$$

Далі будемо розглядати приклади де на малюнку будуть представлені 4 графіка:
А) відносний обсяг виробництва у тіньовому та легальному секторі та коефіцієнт тінізації.
Б) відносні доходи тіньового та легального секторів та держави.
В) мовні ціни на товари, вироблені у кожному із секторів, та ціни залучених трудових ресурсів.
Г) відносні обсяги трудових ресурсів та відносні обсяги виробничих фондів.
За цими показниками і будемо досліджувати далі модель.

Результат роботи програми для заданих вище посатковіх значень.

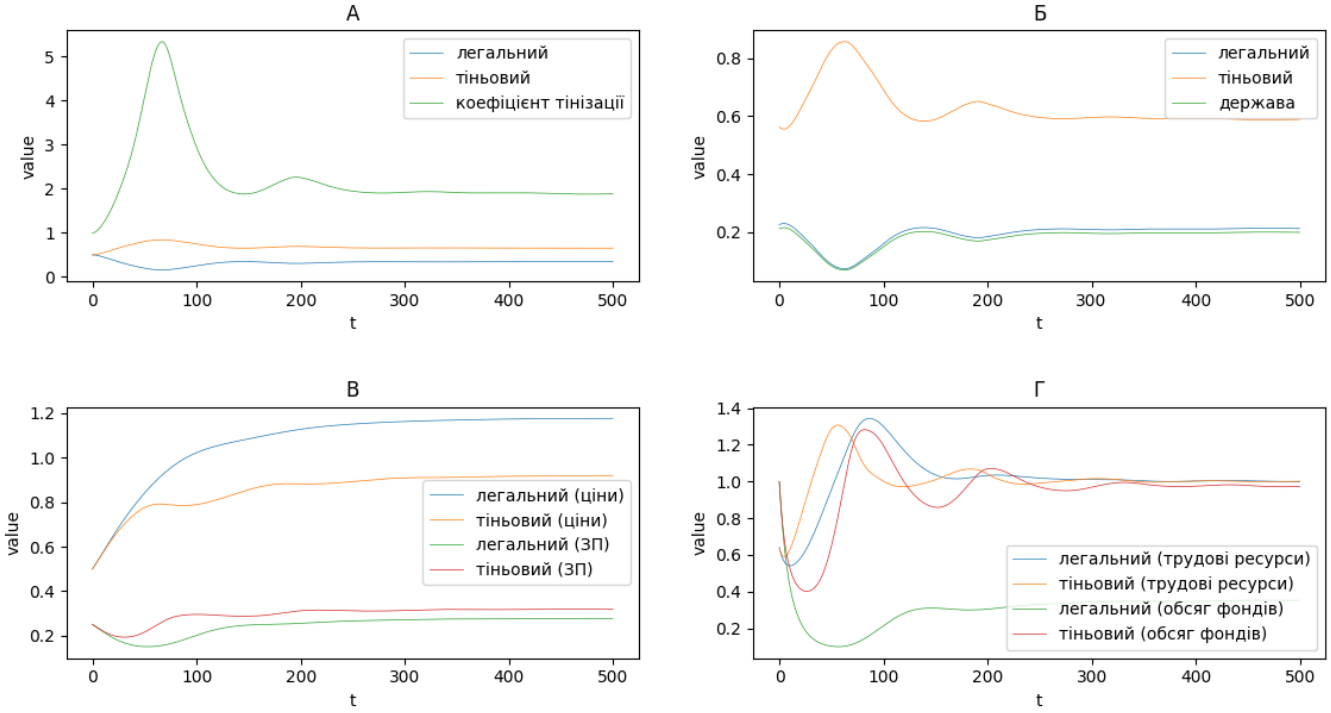


**Рис. 2.1.** Приклад

## 2.2. Пункти 3-4

Розглянемо систему за різних значень $\tau$ та $\sigma$ для того, щоб з'ясувати як покращити ситуацію. Де $\tau$ - норма оподаткування та $\sigma$ - норма відрахування на пригнічення тіньового сектора.

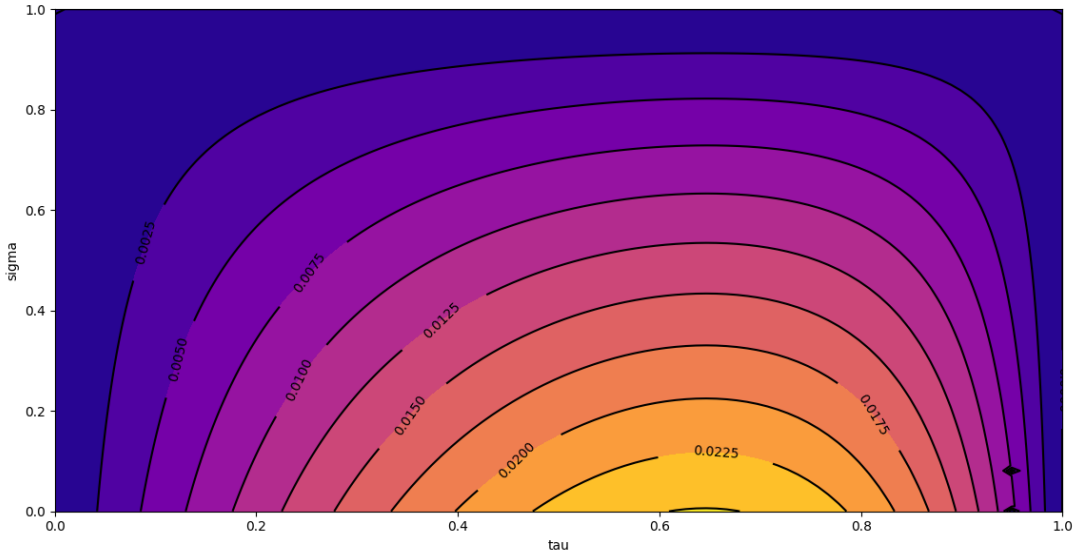


**Рис. 2.2.** Залежність G від tau и sigma

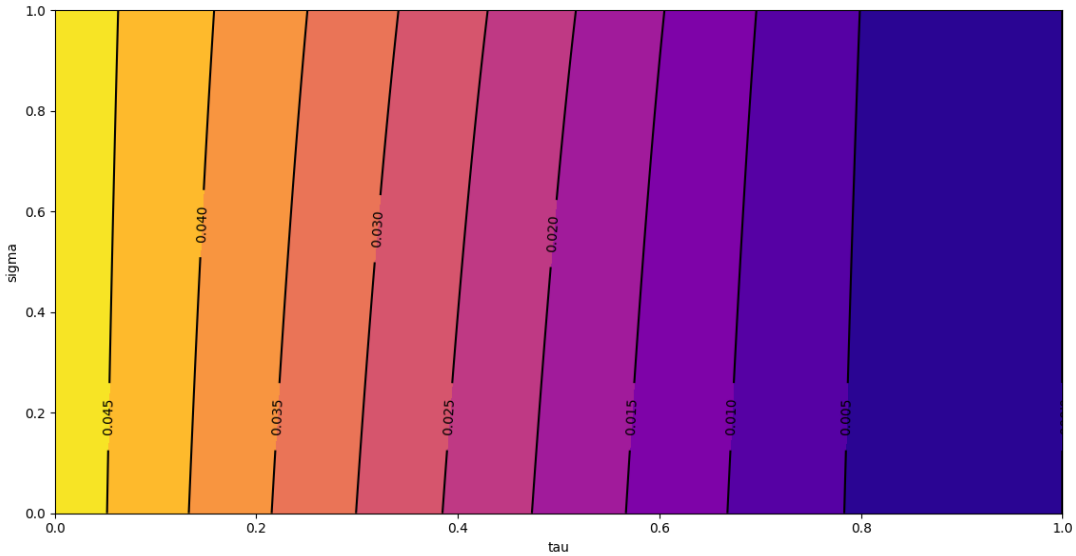


**Рис. 2.3.** Залежність G1 від tau и sigma

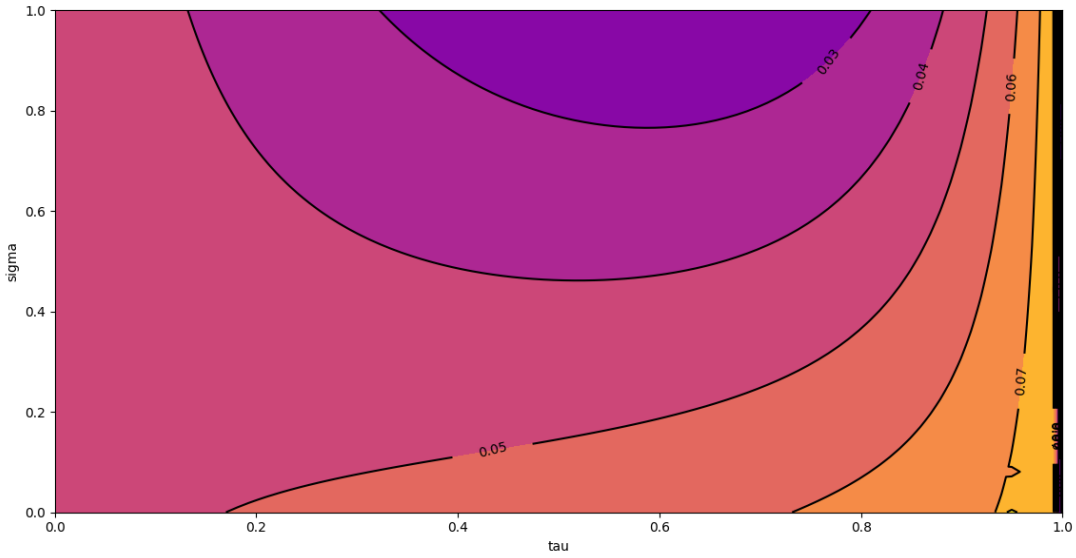


**Рис. 2.4.** Залежність G2 від tau и sigma

Як можна побачити з графіків вище, що все залежить від мети для обрання параметрів $\tau$ та $\sigma$.

Тож якщо метою є підвищення прибутку держави, то стратегія з підвишенням $\sigma$ для пригнічення тіньової частини є поганою, що могло здаватися логічним інтуітивно. А для досягнення такої мети кращем будє тримати $\sigma = 0$, а $\tau$ приблизно на рівні $\tau = 0.64$, що можна побачити на рис. 2.2.

Якщо метою є підвищення прибутку легального сектору, то краще за всього буде максимально зменшувати $\tau$, що можна побачити на рис. 2.3.

Якщо метою є пригнічення тіньового сектору, то краще за всього буде максимально підвищувати $\sigma$, а $\tau$ тримати приблизно на рівні $\tau = 0.64$. Також цікаво, що якщо $\tau = 1$, то тіньовий сектор теж буде пригнічен. Це можна побачити на рис. 2.4.

Декілька прикладів:

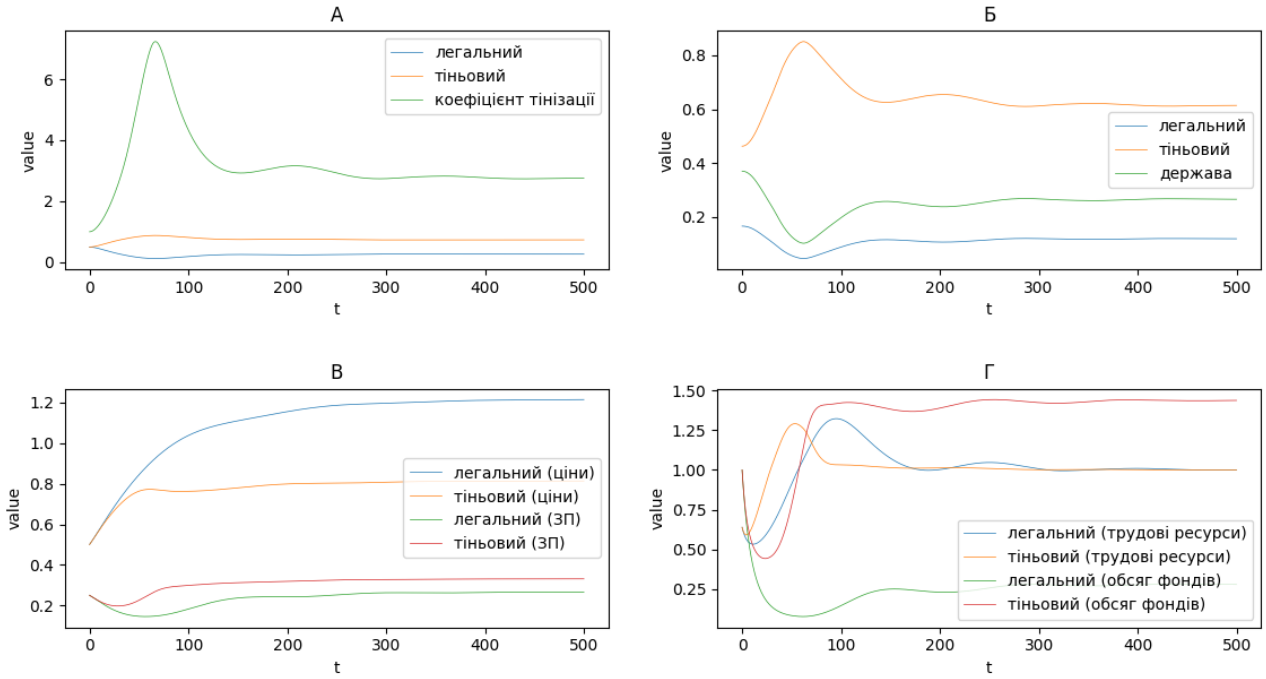


**Рис. 2.5.** Приклад для підвищення прибутку держави ($\sigma = 0$, $\tau = 0.64$).

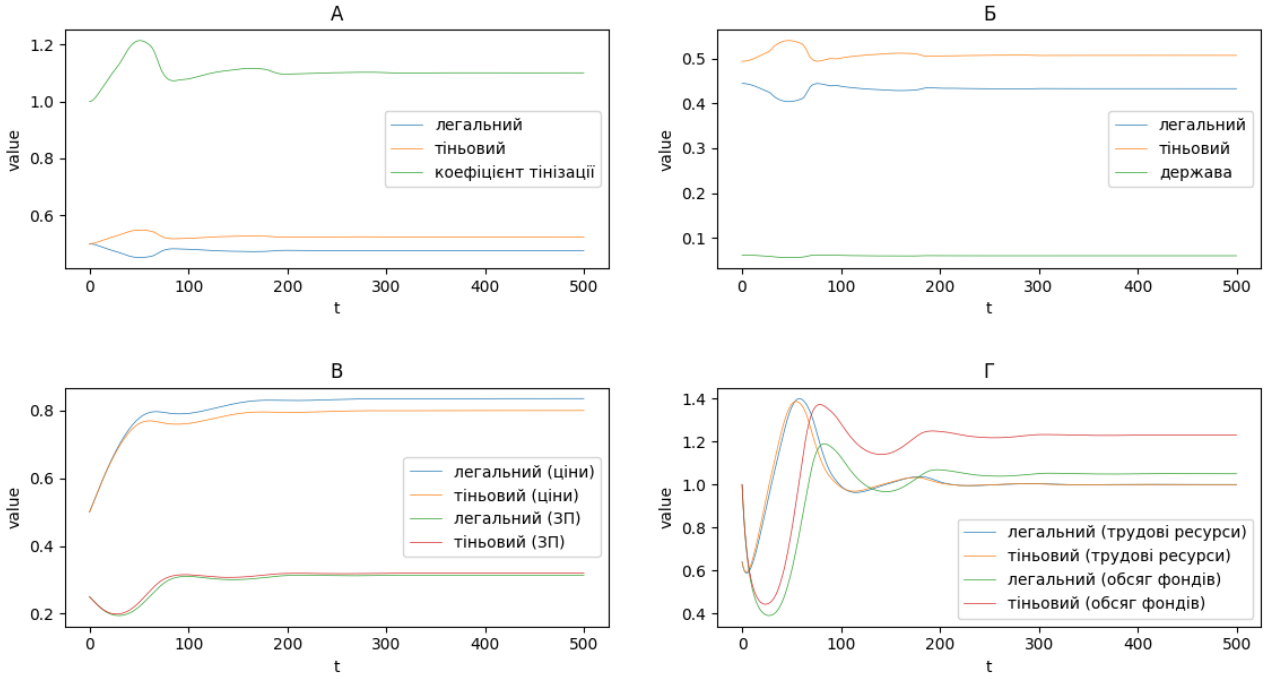


**Рис. 2.6.** Приклад для підвищення прибутку легального сектору ($\sigma = 0$, $\tau = 0.1$).

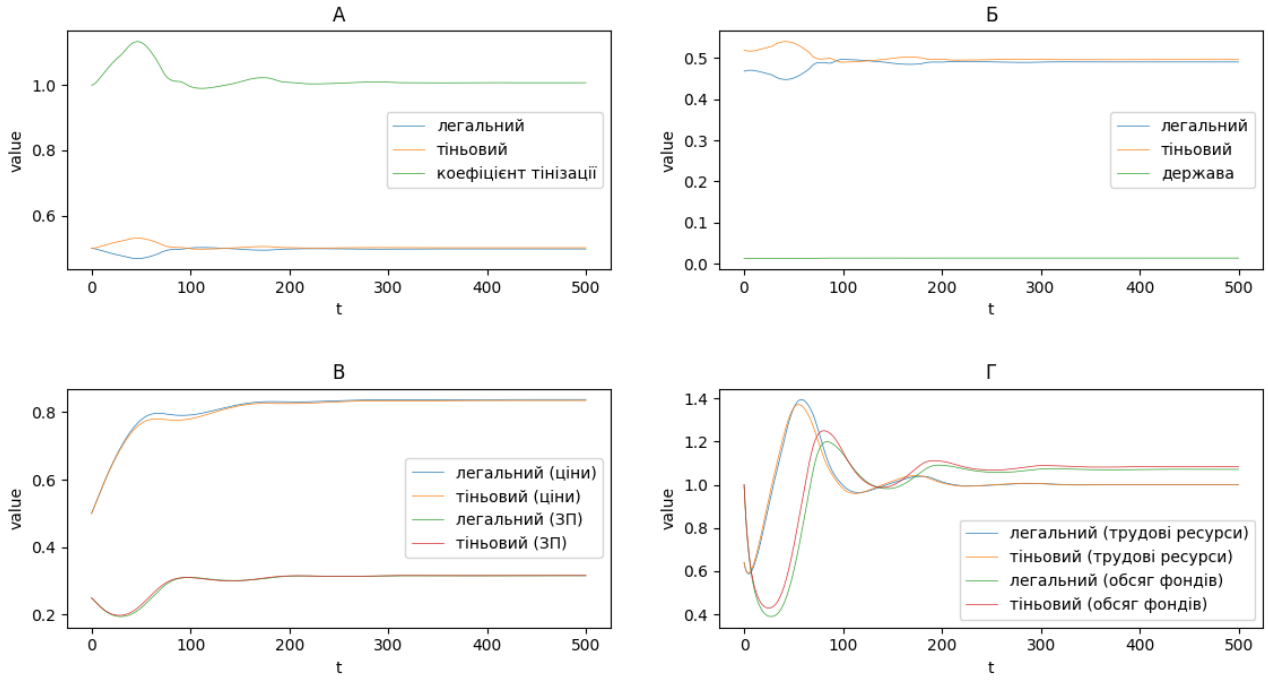


**Рис. 2.7.** Приклад для підвищення прибутку легального сектору ($\sigma = 0.8$, $\tau = 0.1$).

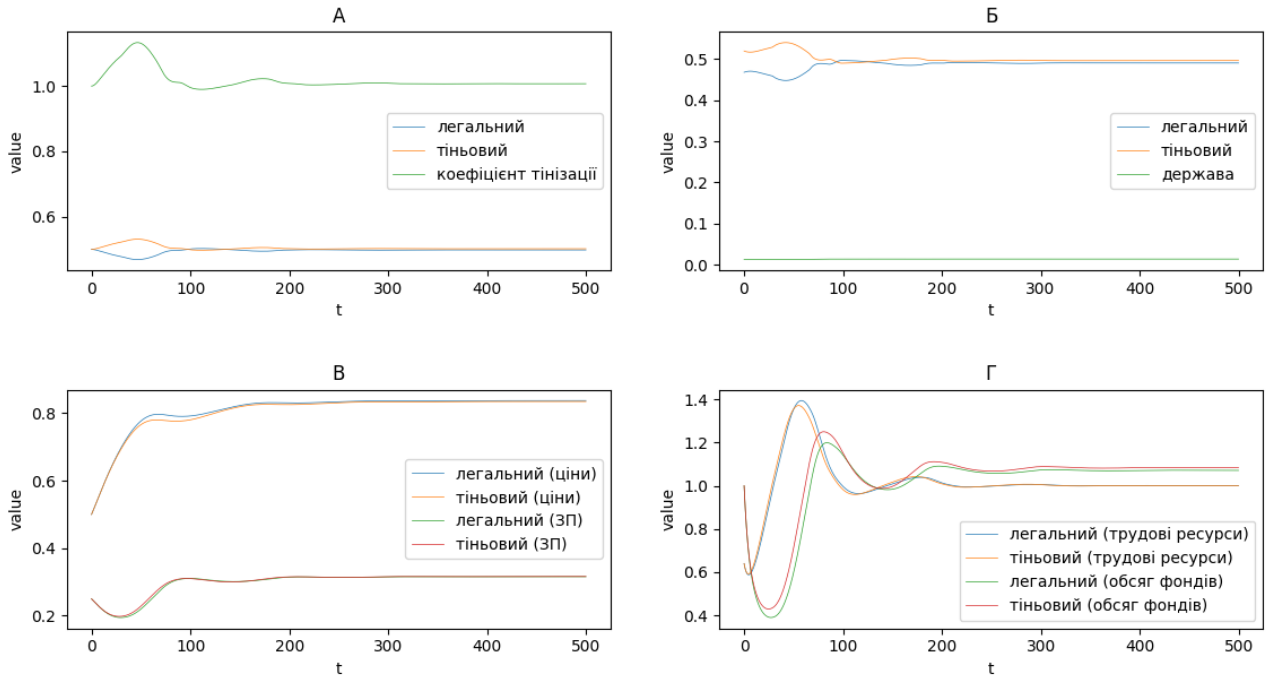


**Рис. 2.8.** Приклад для пригнічення прибутку тіньового сектору ($\sigma = 0$, $\tau = 0.64$).

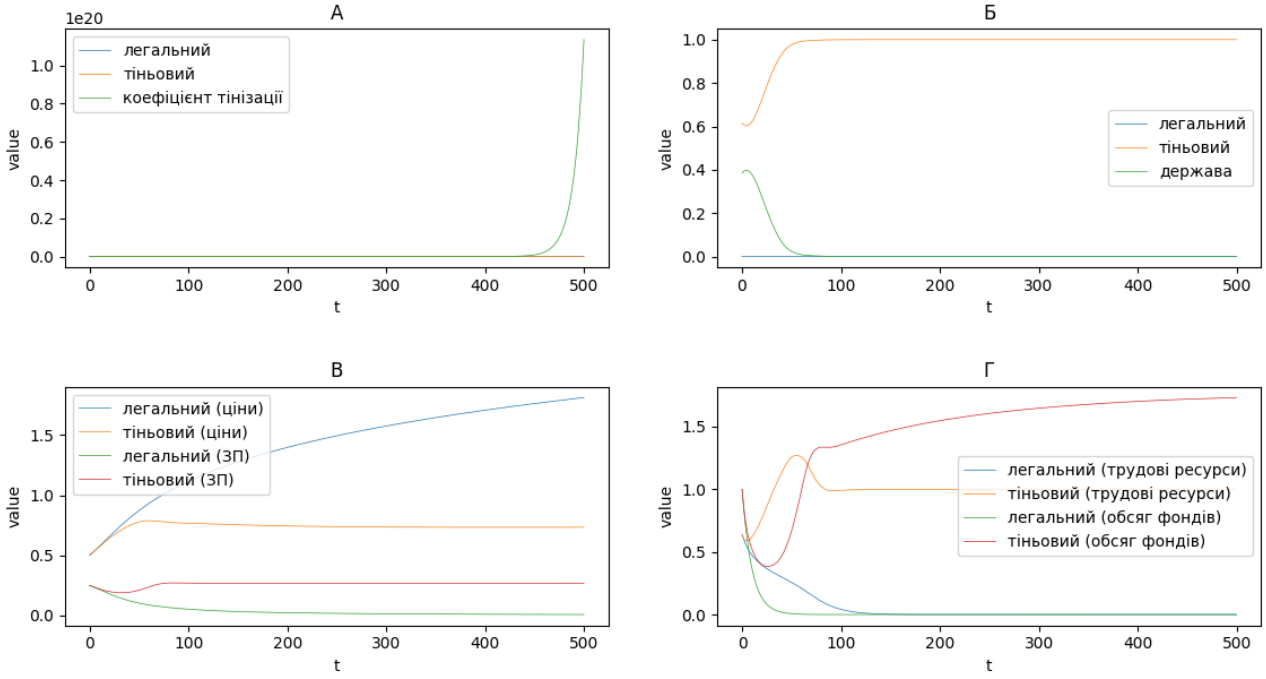


**Рис. 2.9.** Приклад для пригнічення прибутку тіньового сектору ($\sigma = 0.5$, $\tau = 1$).

Розглянемо ситуацію коли мы хочемо оптимізувати одразу декілька показників. Наприклад розглянимо таку функцію $profit(x) = 0.5 * G(x) + 0.4 * G1(x) - 0.1 * G2(x)$. Та отримаємо, що для данної системи оптимальними показниками приблизно будуть $\sigma = 0$, $\tau = 0.45$.

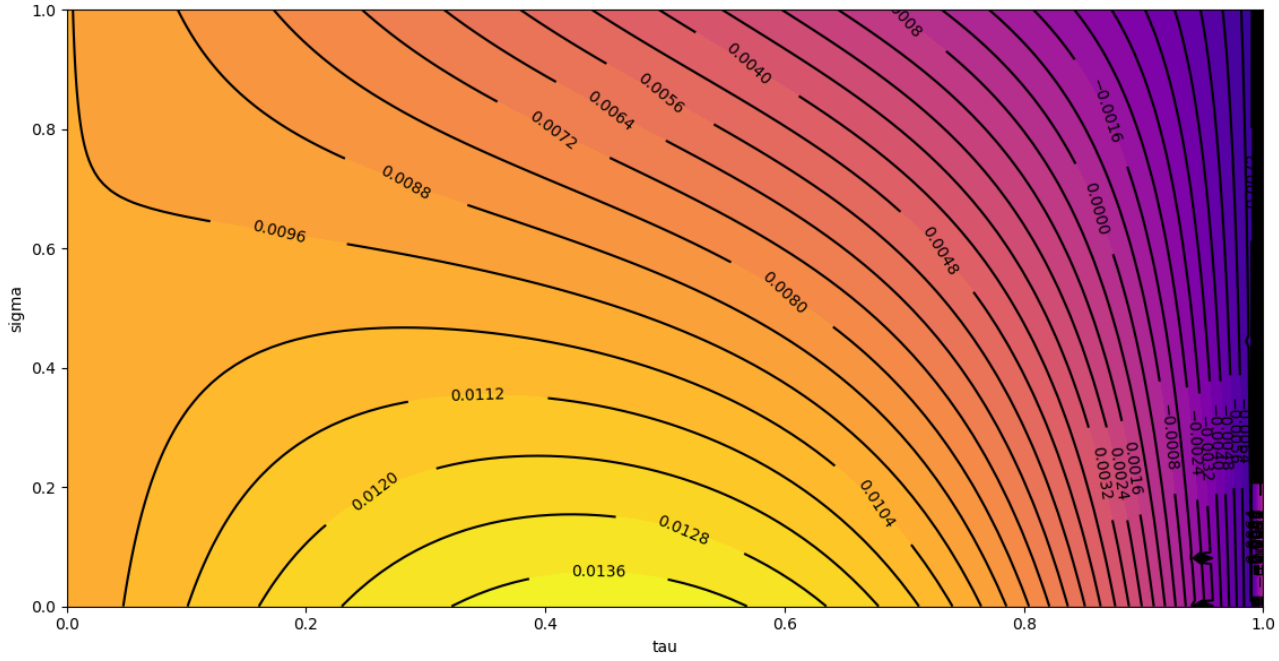


**Рис. 2.10.** Приклад ($\sigma = 0.5$, $\tau = 1$).

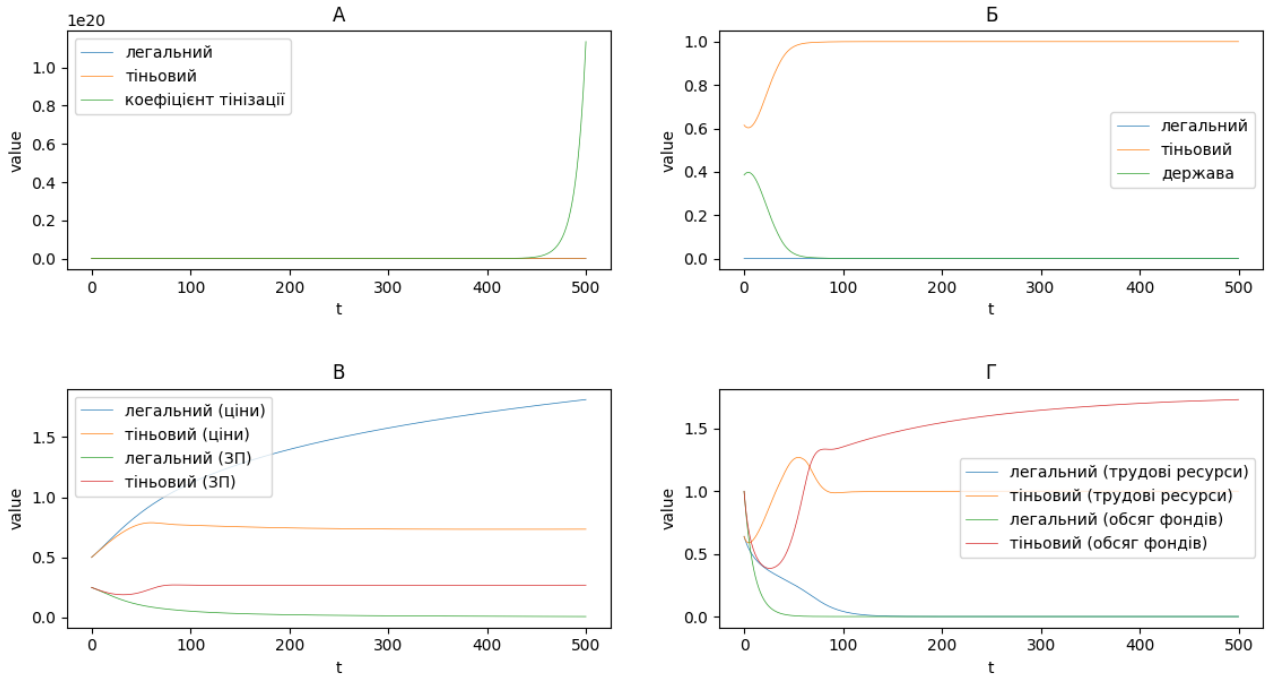


**Рис. 2.11.** Приклад ($\sigma = 0.5$, $\tau = 1$).

## Глава 3

# Висновок

У ході лабораторної роботи було побудована математична модель двохсекторної економіки, досліджена модель із використанням комп'ютерного моделювання.

Глава 4

# Код програми

## 4.1. main.py

```
from collections.abc import Callable
import matplotlib.pyplot as plt
import numpy as np
from scipy.integrate import odeint, solve_ivp
from scipy.optimize import fsolve

from services.model import EconomicModel, G_by_x


t_start = 0
t_end = 500


def count_volume(model, data):
    legal = []
    shadow = []
    ratio = []
    for current_data in data:
        legal_and_shadow = model.Q1(current_data) + model.
          Q2(current_data)
        legal_c = model.Q1(current_data) /
          legal_and_shadow
        shadow_c = model.Q2(current_data) /
          legal_and_shadow
        ratio_c = model.Q2(current_data) / model.Q1(
          current_data)

        legal.append(legal_c)
        shadow.append(shadow_c)
        ratio.append(ratio_c)

    return legal, shadow, ratio

```

```
30
31 def count_profit(model, data):
32     legal = []
33     shadow = []
34     country = []
35     for current_data in data:
36         all_profit = (model.G1(current_data) + model.G2(
               current_data) + model.G(current_data))
37         legal_c = model.G1(current_data) / all_profit
38         shadow_c = model.G2(current_data) / all_profit
39         country_c = model.G(current_data) / all_profit
40
41         legal.append(legal_c)
42         shadow.append(shadow_c)
43         country.append(country_c)
44
45     return legal, shadow, country
46
47
48 def count_price(model, data):
49     legal_p = data[:, 1]
50     shadow_p = data[:, 5]
51     legal_s = data[:, 2]
52     shadow_s = data[:, 6]
53
54     return legal_p, shadow_p, legal_s, shadow_s
55
56
57 def count_work_volume(model, data):
58     legal = []
59     shadow = []
60     for current_data in data:
61         legal_c = model.L1(current_data) / model.S1(
               current_data)
62         shadow_c = model.L2(current_data) / model.S2(
               current_data)
63
64         legal.append(legal_c)
65         shadow.append(shadow_c)
66
67     return legal, shadow
```

```
68
69
70  def count_fonds(model, data, init):
71      legal = data[:, 3] / init[3]
72      shadow = data[:, 7] / init[7]
73
74      return legal, shadow
75
76
77  def f_by_tau_sigma(
78      model: EconomicModel,
79      target_func: Callable,
80      tau: float, sigma: float,
81      init: list[float] | None = None) -> float:
82      prepared_model_call = lambda x, *args: model(x,
          changed_params={"tau": args[0], "sigma": args[1]})
83
84      #
85      if init is None:
86          init = [0, 0.5, 0.25, 0.1, 0, 0.5, 0.25, 0.1]
87
88      new_x = fsolve(prepared_model_call, x0=init, args=(tau
          , sigma))
89
90      #
                                              fsolve.    .   .
                                                                .
91      # t = np.arange(t_start, t_end, 0.1)
92      # new_x = odeint(lambda t, x, *args:
          prepared_model_call(x, *args), init, t, tfirst=True,
           args=(tau, sigma))[-1]
93
94      model.tau = tau
95      model.sigma = sigma
96      result = target_func(new_x)
97
98      #
99      model.set_default()
100     return result
```

```
101
102
103 class ABSPlotFuncByTauAndSigma:
104     def __init__(self):
105         self.levels = 10
106
107     def __call__(self):
108         x, y, z = self.count()
109         self.plot(x, y, z)
110         return self.prepared_func
111
112     def prepared_func(self, tau, sigma):
113         return f_by_tau_sigma(self.model, self.target_func
              , tau, sigma)
114
115     def count(self):
116         z = []
117         x = np.linspace(0, 1, 100)
118         y = np.linspace(0, 1, 100)
119
120         for i, sigma in enumerate(y):
121             z.append([])
122             for tau in x:
123                 res = self.prepared_func(tau, sigma)
124                 z[i].append(res)
125
126         return x, y, z
127
128     def plot(self, x, y, z):
129         ax1 = plt.subplot(1, 1, 1)
130         ax1.set_xlabel('tau')
131         ax1.set_ylabel('sigma')
132
133         ax1.contourf(x, y, z, levels=self.levels, cmap='
              plasma')
134         cs = ax1.contour(x, y, z, levels=self.levels,
              colors=np.zeros((self.levels, 3)))
135         ax1.clabel(cs)
136
137         plt.show()
138
```

```
139
140 def main():
141     line_width = 0.5
142     t = np.arange(t_start, t_end, 0.1)
143     model = EconomicModel()
144
145     p_plot = ABSPlotFuncByTauAndSigma()
146
147     #                                       G       tau
        sigma
148     # p_plot.model = model
149     # p_plot.target_func = model.G
150     # p_plot()
151
152     #                                       G1       tau
        sigma
153     # p_plot.model = model
154     # p_plot.target_func = model.G1
155     # p_plot()
156
157     #                                       G2       tau
        sigma
158     # p_plot.model = model
159     # p_plot.target_func = model.G2
160     # p_plot()
161
162     #
163     # a, b, c = 0.6, 0.3, 0.1
164     # p_plot.model = model
165     # p_plot.levels = 30
166     # p_plot.target_func = lambda x: (a * model.G(x)) + (b
          * model.G1(x)) - (c * model.G2(x))
167     # p_plot()
168
169     model.tau = 0.45
170     model.sigma = 0
171     # print(model.tau)
172     # print(model.sigma)
173
174     f = lambda t, x: model(x)
175
```

```
176     init = [0, 0.5, 0.25, 0.1, 0, 0.5, 0.25, 0.1]
177     result = odeint(f, init, t, tfirst=True)
178 
179     l_v, s_v, r_v = count_volume(model, result)
180     l_p, s_p, r_p = count_profit(model, result)
181     l_price, s_price, l_salary, s_salary = count_price(
          model, result)
182     l_wv, s_wv = count_work_volume(model, result)
183     l_f, s_f = count_fonds(model, result, init)
184 
185     ax1 = plt.subplot(2, 2, 1)
186     ax2 = plt.subplot(2, 2, 2)
187     ax3 = plt.subplot(2, 2, 3)
188     ax4 = plt.subplot(2, 2, 4)
189 
190     plt.subplots_adjust(wspace=0.2, hspace=0.5)
191     ax1.set_title('A')
192     ax1.set_xlabel('t')
193     ax1.set_ylabel('value')
194 
195     ax2.set_title('  ')
196     ax2.set_xlabel('t')
197     ax2.set_ylabel('value')
198 
199     ax3.set_title('  ')
200     ax3.set_xlabel('t')
201     ax3.set_ylabel('value')
202 
203     ax4.set_title('  ')
204     ax4.set_xlabel('t')
205     ax4.set_ylabel('value')
206 
207     ax1.plot(t, l_v, label='                    ', linewidth
          =line_width)
208     ax1.plot(t, s_v, label='                  ', linewidth=
          line_width)
209     ax1.plot(t, r_v, label='                          ␣
                        ', linewidth=line_width)
210 
211     ax2.plot(t, l_p, label='                    ', linewidth
          =line_width)
```

```
212     ax2.plot(t, s_p, label='                    ',linewidth=
          line_width)
213     ax2.plot(t, r_p, label='                  ',linewidth=
          line_width)
214
215     ax3.plot(t, l_price, label='                      ␣(
                )', linewidth=line_width)
216     ax3.plot(t, s_price, label='                    ␣(
          )', linewidth=line_width)
217     ax3.plot(t, l_salary, label='                      ␣(     )
          ', linewidth=line_width)
218     ax3.plot(t, s_salary, label='                     ␣(     )',
           linewidth=line_width)
219
220     ax4.plot(t, l_wv, label='                      ␣(
                        ␣                )', linewidth=
          line_width)
221     ax4.plot(t, s_wv, label='                    ␣(
                        ␣                )', linewidth=
          line_width)
222     ax4.plot(t, l_f, label='                      ␣(
          ␣              )', linewidth=line_width)
223     ax4.plot(t, s_f, label='                    ␣(            ␣
                      )', linewidth=line_width)
224
225     ax1.legend()
226     ax2.legend()
227     ax3.legend()
228     ax4.legend()
229
230     plt.show()
231
232
233 if __name__ == '__main__':
234     main()
```

## 4.2. model.py

```
1 import math
2 from scipy.optimize import fsolve
```

```
3  from collections.abc import Iterable
4
5
6  class EconomicModel(object):
7      def __init__(self):
8          self.set_default()
9
10     def set_default(self):
11         self.alpha = 0.5
12         self.beta = 1.5
13         self.gamma = 1.5
14         self.delta = 0.1
15         self.nu = 5
16         self.mu = 20
17         self.lambda_ = 20
18         self.rho = 10
19         self.A0 = 1
20         self.L0 = 1
21         self.D0 = 1
22         self.tau = 0.6
23         self.sigma = 0.5
24         self.theta = (1 + self.alpha * (self.beta - 1)) **
               (-1)
25
26     def L1(self, x):
27         return x[3] * ((1 - self.alpha) * self.A0 * x[1] /
               x[2]) ** (1 / self.alpha)
28
29     def Q1(self, x):
30         return self.A0 * x[3] ** self.alpha * self.L1(x)
             ** (1 - self.alpha)
31
32     def D1(self, x):
33         return self.D0 * math.exp(-self.beta * x[1]) * x
             [5] / (x[1] + x[5])
34
35     def S1(self, x):
36         return self.L0 * (1 - math.exp(-self.gamma * x[2])
             ) * x[2] / (x[2] + x[6])
37
38     def I1(self, x):
```

```
39          return (1 - self.tau) * (1 - self.theta) * x[0]
40
41      def L2(self, x):
42          return x[7] * ((1 - self.alpha) * self.A0 * x[5] /
                x[6]) ** (1 / self.alpha)
43
44      def Q2(self, x):
45          return self.A0 * x[7] ** self.alpha * self.L2(x)
              ** (1 - self.alpha)
46
47      def D2(self, x):
48          return self.D0 * math.exp(-self.beta * x[5]) * x
              [1] / (x[1] + x[5])
49
50      def S2(self, x):
51          return self.L0 * (1 - math.exp(-self.gamma * x[6])
              ) * x[6] / (x[2] + x[6])
52
53      def I2(self, x):
54          return (1 - self.theta) * x[4]
55
56      def T(self, x):
57          return self.tau * x[0]
58
59      def G(self, x):
60          """                                          """
61          return (1 - self.sigma) * self.tau * x[0]
62
63      def G1(self, x):
64          """
              """
65          return (1 - self.tau) * self.theta * x[0]
66
67      def G2(self, x):
68          """                                                """
69          return self.theta * x[4]
70
71      def count_model(self, x):
72          P1 = (x[1] * min(self.Q1(x), self.D1(x))\
73              - x[2] * min(self.L1(x), self.S1(x)) - x[0]) /
                  self.nu
```

```
74
75          p1 = (self.D1(x) - self.Q1(x)) / self.mu
76
77          w1 = (self.L1(x) - self.S1(x)) / self.lambda_
78
79          K1 = -self.delta * x[3] + self.I1(x)
80
81          P2 = (math.exp(-self.rho * self.sigma * self.T(x))
              \
82              * x[5]\
83              * min(self.Q2(x), self.D2(x)) - x[6]\
84              * min(self.L2(x), self.S2(x)) - x[4]) / self.
                nu
85
86          p2 = (self.D2(x) - self.Q2(x)) / self.mu
87
88          w2 = (self.L2(x) - self.S2(x)) / self.lambda_
89
90          K2 = -self.delta * x[7] + self.I2(x)
91
92          result = [P1, p1, w1, K1, P2, p2, w2, K2]
93          return result
94
95      def __call__(self, x, changed_params={}):
96          for param_name in changed_params:
97              if not hasattr(self, param_name):
98                  raise ValueError(param_name)
99              setattr(self, param_name, changed_params[
                  param_name])
100
101         return self.count_model(x)
102
103
104 def G_by_x(model: EconomicModel, tau: float, init: list[
      float] | None = None) -> float:
105     if isinstance(tau, Iterable):
106         return list(map(lambda x: G_by_x(model, x), list(
              tau)))
107
108     prepared_model_call = lambda x, *args: model(x,
          changed_params={"tau": args[0]})
```

```
109
110     #
111     model.set_default()
112
113     #
114     if init is None:
115         init = [0, 0.5, 0.25, 0.1, 0, 0.5, 0.25, 0.1]
116     new_x = fsolve(prepared_model_call, x0=init, args=(tau
            ,))
117
118     model.tau = tau
119     return model.G(new_x)
```