## ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА КОМП'ЮТЕРНА ТЕХНІКА

https://doi.org/10.31649/1997-9266-2024-175-4-66-74

УДК [004.6:519.216]:519.87(045)

**Б. І. Мокін**[1]
**О. О. Войцеховська**[1]
**Н. В. Собчук**[1]
**О. В. Бондарчук**[1]

# ЕКВІВАЛЕНТНІ МОДЕЛІ ФУНКЦІЙ РОЗПОДІЛУ ДАНИХ ВИПАДКОВОГО ХАРАКТЕРУ

[1]Вінницький національний технічний університет

*Представлено результати дослідження, виконаного авторами в процесі формування доповіді «Еквівалентування щільностей розподілу даних випадкового характеру», виголошеній на Міжнародній науково-практичній конференції «Інформаційні технології та комп'ютерне моделювання (ІТКМ-24)», що відбулася в Івано-Франківську в період з 21 по 24 травня 2024 року, в якій запропоновано спосіб синтезу статистичної оцінки еквівалентної щільності розподілу даних випадкового характеру, який не вимагає «вирівнювання гістограм» з використанням $\chi^2$-розподілу Пірсона. На першому етапі запропонований авторами спосіб використовує подрібнення гістограми, побудованої з використанням відносно невеликої вибірки даних випадкового характеру, з подальшим кумулятивним підсумовуванням для визначення еквівалентної функції розподілу цих даних на множині наявних їх значень. На другому етапі запропонованого авторами способу здійснюється інтерполяція побудованої еквівалентної функції розподілу даних випадкового характеру кубічними сплайнами за алгоритмом, що використовує виведені авторами математичні співвідношення. А на третьому етапі шляхом диференціювання еквівалентної математичної моделі розподілу даних випадкового характеру, отриманої на попередніх етапах реалізації цього способу, синтезується еквівалентна математична модель щільності розподілу даних випадкового характеру, з використанням яких створено первинну базову гістограму. В роботі приведено або згадано усі Python-програми, необхідні для реалізації усіх етапів запропонованого способу синтезу еквівалентної моделі щільності розподілу даних випадкового характеру, а також наведено приклад розв'язання конкретної практичної задачі з роз'ясненнями усіх дій, необхідних до виконання на кожному із етапів реалізації запропонованого способу.*

**Ключові слова:** дані випадкового характеру, гістограма, подрібнення гістограми, кумулятивна сума, функція розподілу, еквівалентування, інтерполяція, кубічні сплайни, щільність розподілу, Python-програми.

### Вихідні передумови та постановка задачі

В задачах оброблення даних випадкового характеру часто виникає необхідність в побудові статистичних оцінок диференціальних законів розподілу цих даних, або, що те ж саме, в побудові статистичних оцінок їхньої щільності розподілу, для реалізації яких в класичній математичній статистиці пропонується спосіб, що починається з формування гістограм з подальшим їхнім «вирівнюванням» з використанням відомих теоретичних моделей для щільностей розподілу випадкових величин та $\chi^2$-розподілу Пірсона як критерію адекватності процесу «вирівнювання». При достатній кількості експериментальних даних цей спосіб «вирівнювання гістограм», як показано в роботах [1], [2], приводить до адекватних статистичних оцінок, але за їхніх малих вибірок, зазвичай, мають місце або довірчі ймовірності зі значеннями, нижчими за поріг довіри, або виникає неоднозначність у тому, в бік якого теоретичного розподілу схилиться статистична його оцінка в припущенні збільшення кількості цих даних.

В нашій доповіді «Еквівалентні закони розподілу випадкових величин», виголошеній на уні-

верситетській науково-технічній конференції 2024 року і опублікованій у вигляді статті в роботі [3] та у вигляді тез в роботі [4], запропоновано інший спосіб побудови статистичних оцінок функції розподілу випадкової величини, який не використовує процес «вирівнювання гістограм» із застосуванням $\chi^2$-розподілу Пірсона як критерію адекватності процесу «вирівнювання». За цим способом ми запропонували синтезувати еквівалентні функції розподілу, які більшою мірою віддзеркалюють властивості конкретного набору експериментальних даних у порівнянні з їхніми «вирівняними за Пірсоном» еквівалентами.

Суть запропонованого способу еквівалентування полягає в подрібненні базової гістограми, зображеної на рис. 1, з подальшим кумулятивним підсумовуванням за програмою, написаною мовою Python [5], для отримання східчастої еквівалентної функції розподілу, показаної на рис. 2 після трикратного подрібнення з подальшим підсумовуванням. При цьому потрібно не забути кожну ординату на цьому рисунку поділити на три.

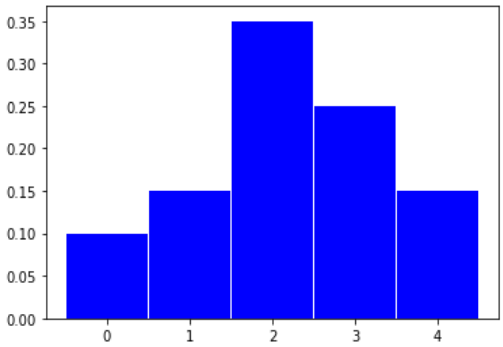


Рис. 1. Гістограма випадкових даних

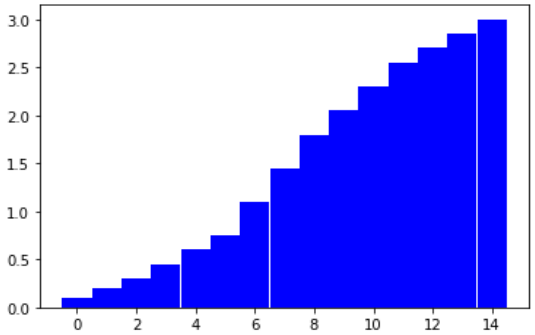


Рис. 2. Подрібнення і кумуляція

А в іншій нашій доповіді «Еквівалентування щільностей розподілу даних випадкового характеру», виголошеній на Міжнародній науково-практичній конференції «Інформаційні технології та комп'ютерне моделювання (ІТКМ-24)», що відбулася в Івано-Франківську в період з 21 по 24 травня 2024 року, тези якої опубліковані в роботі [6], запропоновано на другому етапі еквівалентування здійснити інтерполяцію останньої версії статистичної оцінки кумулятивного варіанта еквівалентної східчастої функції розподілу випадкової величини, за допомогою кубічних сплайнів, скориставшись методикою сплайн-інтерполяції, описаною в роботі [7]. А на третьому етапі цього способу еквівалентування запропоновано прямим диференціюванням сплайн-інтерпольованої функції розподілу випадкової величини синтезувати еквівалентну математичну модель щільності розподілу даних випадкового характеру.

Оскільки перший етап запропонованого способу еквівалентування математичних моделей, що характеризують випадкові величини до чотирьох Python-програм реалізації включно, детально розкритий у роботі [3], то далі зосередимося лише на розкритті другого та третього етапів цього способу.

## Розв'язання поставленої задачі

Почнемо розв'язання поставленої задачі із синтезу сплайн-моделі еквівалентної функції розподілу випадкової величини, гістограма якої показана на рис. 1 і до граничних точок розрядів якої ми будемо «прив'язувати» відповідні кубічні сплайни, пам'ятаючи при цьому, що після трикратного подрібнення гістограми і переходу до її еквіваленту, представленому на рис. 2, ординаті з абсцисою 1 на цьому рисунку буде відповідати ордината з абсцисою 3, ординаті з абсцисою 2 буде відповідати ордината з абсцисою 6, ординаті з абсцисою 3 буде відповідати ордината з абсцисою 9, ординаті з абсцисою 4 буде відповідати ордината з абсцисою 12, а ординаті з абсцисою 5 буде відповідати ордината з абсцисою 15.

Усі ці абсциси і ординати, визначні за рис. 2, зведені в табл. 1.

Таблиця 1

| $I$ | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| $x_i$ | $x_0$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
| $y_0$ | 0,03 | | | | | |
| $y_1$ | | 0,07 | | | | |

Продовження табл. 1

| | | | | | | |
|---|---|---|---|---|---|---|
| $y_2$ | | | 0,15 | | | |
| $y_3$ | | | | 0,40 | | |
| $y_4$ | | | | | 0,70 | |
| $y_5$ | | | | | | 1,0 |

У процесі сплайн-інтерполяції необхідно розв'язати систему із 20 рівнянь з 20 невідомими, складену з дотриманням умов рівності в дотичних точках базових для сусідніх відрізків сплайнів, умов рівності їхніх перших та других похідних у цих точках, а також умов рівності нулю других похідних на кінцях поля базової гістограми, для розв'язання якої розроблено програму мовою Python.

Запишемо цю систему рівнянь відносно сплайнів

$$s_j = \sum_{i=0}^{3} c_{ij} x_j^i, \ \ j = 1,\ 2,...,5 \tag{1}$$

та похідних від них

$$f_j = \sum_{i=1}^{3} i c_{ij} x_j^{i-1}, \ j = 1,2,...,5 \tag{2}$$

з урахуванням вищевикладених умов. В результаті отримаємо

$$\begin{cases} c_{01} = 0,03, \\ c_{01} + c_{11} + c_{21} + c_{31} = 0,07, \\ c_{02} + c_{12} + c_{22} + c_{32} = 0,07, \\ c_{02} + 2c_{12} + 4c_{22} + 8c_{32} = 0,15, \\ c_{03} + 2c_{13} + 4c_{23} + 8c_{33} = 0,15, \\ c_{03} + 3c_{13} + 9c_{23} + 27c_{33} = 0,40, \\ c_{04} + 3c_{14} + 9c_{24} + 27c_{34} = 0,40, \\ c_{04} + 4c_{14} + 16c_{24} + 64c_{34} = 0,70, \\ c_{05} + 4c_{15} + 16c_{25} + 64c_{35} = 0,70, \\ c_{05} + 5c_{15} + 25c_{25} + 125c_{35} = 1,0 \\ c_{11} + 2c_{21} + 3c_{31} = c_{12} + 2c_{22} + 3c_{32}, \\ c_{12} + 4c_{22} + 12c_{32} = c_{13} + 4c_{23} + 12c_{33}, \\ c_{13} + 6c_{23} + 27c_{33} = c_{14} + 6c_{24} + 27c_{34}, \\ c_{14} + 8c_{24} + 48c_{34} = c_{15} + 8c_{25} + 48c_{35}, \\ 2c_{21} + 6c_{31} = 2c_{22} + 6c_{32}, \\ 2c_{22} + 12c_{32} = 2c_{23} + 12c_{33}, \\ 2c_{23} + 18c_{33} = 2c_{24} + 18c_{34}, \\ 2c_{24} + 24c_{34} = 2c_{25} + 24c_{35}, \\ 2c_{25} + 30c_{35} = 0, \\ 2c_{21} = 0. \end{cases} \tag{3}$$

Для розв'язання системи алгебраїчних рівнянь (3) використаємо Python-програму 1.

*Python-програма* 1:

```
In [1]: import sympy
In [2]: from sympy import*
In [3]: c01,c11,c21,c31=symbols('c01 c11 c21 c31')
In [4]: c02,c12,c22,c32=symbols('c02 c12 c22 c32')
In [5]: c03,c13,c23,c33=symbols('c03 c13 c23 c33')
In [6]: c04,c14,c24,c34=symbols('c04 c14 c24 c34')
In [7]: c05,c15,c25,c35=symbols('c05 c15 c25 c35')
```

```
In [8]: x=symbols('x')
In [9]: s1=Function('s1')(x)
In [10]: s2=Function('s2')(x)
In [11]: s3=Function('s3')(x)
In [12]: s4=Function('s4')(x)
In [13]: s5=Function('s5')(x)
In [14]: s1=c01+c11*x+c21*x**2+c31*x**3
In [15]: s2=c02+c12*x+c22*x**2+c32*x**3
In [16]: s3=c03+c13*x+c23*x**2+c33*x**3
In [17]: s4=c04+c14*x+c24*x**2+c34*x**3
In [18]: s5=c05+c15*x+c25*x**2+c35*x**3
In [19]: eq1=c01-0.03
In [20]: eq2=c01+c11+c21+c31-0.07
In [21]: eq3=c02+c12+c22+c32-0.07
In [22]: eq4=c02+2*c12+4*c22+8*c32-0.15
In [23]: eq5=c03+2*c13+4*c23+8*c33-0.15
In [24]: eq6=c03+3*c13+9*c23+27*c33-0.40
In [25]: eq7=c04+3*c14+9*c24+27*c34-0.40
In [26]: eq8=c04+4*c14+16*c24+64*c34-0.70
In [27]: eq9=c05+4*c15+16*c25+64*c35-0.70
In [28]: eq10=c05+5*c15+25*c25+125*c35-1.0
In [29]: eq11=c11+2*c21+3*c31-c12-2*c22-3*c32
In [30]: eq12=c12+4*c22+12*c32-c13-4*c23-12*c33
In [31]: eq13=c13+6*c23+27*c33-c14-6*c24-27*c34
In [32]: eq14=c14+8*c24+48*c34-c15-8*c25-48*c35
In [33]: eq15=2*c21+6*c31-2*c22-6*c32
In [34]: eq16=2*c22+12*c32-2*c23-12*c33
In [35]: eq17=2*c23+18*c33-2*c24-18*c34
In [36]: eq18=2*c24+24*c34-2*c25-24*c35
In [37]: eq19=2*c25+30*c35
In [38]: eq20=2*c21
In [39]: solve([eq1,eq2,eq3,eq4,eq5,eq6,eq7,eq8,eq9,eq10,eq11,eq12,eq13,\
           eq14,eq15,eq16,eq17,eq18,eq19,eq20],c01,c11,c21,c31,c02,c12,c22,\
           c32,c03,c13,c23,c33,c04,c14,c24,c34,c05,c15,c25,c35)
Out[39]:
          {c01: 0.0300000000000000, c11: 0.0405263157894737,c21: 0.0,
          c31: -0.000526315789473684,c02: -0.0131578947368421,
         c12: 0.170000000000000,c22: -0.129473684210526,
         c32: 0.0426315789473684,c03: 0.647894736842105,
         c13: -0.821578947368421,c23: 0.366315789473684,
         c33: -0.0400000000000000,c04: -0.361052631578947,
         c14: 0.187368421052632, c24: 0.0300000000000000,
         c34: -0.00263157894736842,c05: -0.563157894736842,
         c15: 0.338947368421053,c25: -0.00789473684210526,
         c35: 0.000526315789473684}
In [40]: d1={c01: 0.0300000000000000, c11: 0.0405263157894737,c21: 0.0,
         c31: -0.000526315789473684,c02: -0.0131578947368421,
           c12: 0.170000000000000,c22: -0.129473684210526,
           c32: 0.0426315789473684,c03: 0.647894736842105,
         c13: -0.821578947368421,c23: 0.366315789473684,
         c33: -0.0400000000000000,c04: -0.361052631578947,
         c14: 0.187368421052632, c24: 0.0300000000000000,
         c34: -0.00263157894736842,c05: -0.563157894736842,
           c15: 0.338947368421053,c25: -0.00789473684210526,
         c35: 0.000526315789473684}
```

```
In [41]: s11=s1.subs([(c01,d1[c01]),(c11,d1[c11]),(c21,d1[c21]),(c31,\
                d1[c31])]).evalf(3);s11
Out[41]:
         -0.000526*x**3 + 0.0405*x + 0.03
In [42]: s22=s2.subs([(c02,d1[c02]),(c12,d1[c12]),(c22,d1[c22]),(c32,\
                d1[c32])]).evalf(3);s22
Out[42]:
         0.0426*x**3 - 0.129*x**2 + 0.17*x - 0.0132
In [43]: s33=s3.subs([(c03,d1[c03]),(c13,d1[c13]),(c23,d1[c23]),(c33,\
                d1[c33])]).evalf(3);s33
Out[43]:
         -0.04*x**3 + 0.366*x**2 - 0.822*x + 0.648
In [44]: s44=s4.subs([(c04,d1[c04]),(c14,d1[c14]),(c24,d1[c24]),(c34,\
                d1[c34])]).evalf(3);s44
Out[44]:
         -0.00263*x**3 + 0.03*x**2 + 0.187*x - 0.361
In [45]: s55=s5.subs([(c05,d1[c05]),(c15,d1[c15]),(c25,d1[c25]),(c35,\
                d1[c35])]).evalf(3);s55
Out[45]:
         0.000526*x**3 - 0.00789*x**2 + 0.339*x - 0.563
In [46]: ds1=s11.diff();ds1
Out[46]:
         0.0405 - 0.00158*x**2
In [47]: ds2=s22.diff();ds2
Out[47]:
         0.128*x**2 - 0.259*x + 0.17
In [48]: ds3=s33.diff();ds3
Out[48]:
         -0.12*x**2 + 0.733*x - 0.822
In [49]: ds4=s44.diff();ds4
Out[49]:
         -0.00789*x**2 + 0.06*x + 0.187
In [50]: ds5=s55.diff();ds5
Out[50]:
         0.00158*x**2 - 0.0158*x + 0.339
In [51]: y,z,u,v=symbols ('y z u v')
In [52]: expr1=s11
In [53]: expr2=s22.subs(x,y)
In [54]: expr3=s33.subs(x,z)
In [55]: expr4=s44.subs(x,u)
In [56]: expr5=s55.subs(x,v)
In [57]: expr11=ds1
In [58]: expr22=ds2.subs(x,y)
In [59]: expr33=ds3.subs(x,z)
In [60]: expr44=ds4.subs(x,u)
In [61]: expr55=ds5.subs(x,v)
In [62]: import numpy as np
In [63]: import matplotlib
In [64]: import matplotlib.pyplot as plt
In [65]: Fr1=lambdify(x,expr1,"numpy")
In [66]: x=np.linspace(0,1,21)
In [67]: Fr1(x)
Out[67]:
array([0.03      , 0.03202493, 0.03404947, 0.03607322, 0.03809579,
       0.04011678, 0.0421358 , 0.04415245, 0.04616634, 0.04817707,
       0.05018425, 0.05218749, 0.05418638, 0.05618055, 0.05816958,
       0.06015309, 0.06213069, 0.06410197, 0.06606655, 0.06802402,
       0.069974  ])
```

```
In [68]: Fr2=lambdify(y,expr2,"numpy")
In [69]: y=np.linspace(1,2,21)
In [70]: Fr2(y)
Out[70]:
array([0.0704    , 0.07239233, 0.0744106 , 0.07648678, 0.0786528 ,
       0.08094063, 0.0833822 , 0.08600947, 0.0888544 , 0.09194892,
       0.095325  , 0.09901457, 0.1030496 , 0.10746203, 0.1122838 ,
       0.11754688, 0.1232832 , 0.12952473, 0.1363034 , 0.14365118,
       0.1516    ])
In [71]: Fr3=lambdify(z,expr3,"numpy")
In [72]: z=np.linspace(2,3,21)
In [73]: Fr3(z)
Out[73]:
array([0.148  , 0.15641, 0.16542, 0.175  , 0.18512, 0.19575, 0.20686,
       0.21842, 0.2304 , 0.24277, 0.2555 , 0.26856, 0.28192, 0.29555,
       0.30942, 0.3235 , 0.33776, 0.35217, 0.3667 , 0.38132, 0.396  ])
In [74]: Fr4=lambdify(u,expr4,"numpy")
In [75]: u=np.linspace(3,4,21)
In [76]: Fr4(u)
Out[76]:
array([0.39899   , 0.413805  , 0.42864967, 0.44352205, 0.45842016,
       0.47334203, 0.48828569, 0.50324916, 0.51823048, 0.53322767,
       0.54823875, 0.56326176, 0.57829472, 0.59333566, 0.60838261,
       0.62343359, 0.63848664, 0.65353978, 0.66859103, 0.68363843,
       0.69868   ])
In [77]: Fr5=lambdify(v,expr5,"numpy")
In [78]: v=np.linspace(4,5,21)
In [79]: Fr5(v)
Out[79]:
array([0.700424  , 0.71547652, 0.73052155, 0.74555947, 0.76059069,
       0.77561559, 0.79063458, 0.80564805, 0.82065638, 0.83565999,
       0.85065925, 0.86565457, 0.88064634, 0.89563495, 0.9106208 ,
       0.92560428, 0.94058579, 0.95556572, 0.97054447, 0.98552243,
       1.0005    ])
In [80]: plt.plot(x,Fr1(x),'-r',y,Fr2(y),'-g',z,Fr3(z),'-c',u,Fr4(u),'-b',\
             v,Fr5(v),'-r',linewidth=3)
```

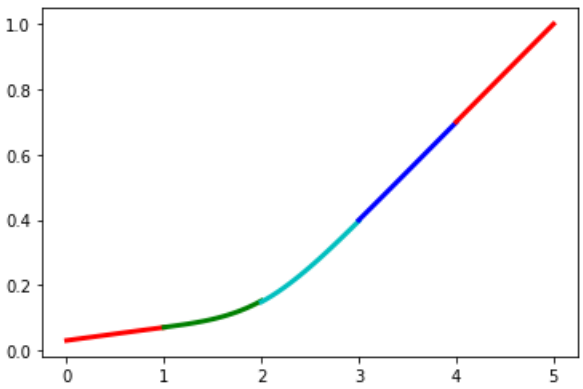


Рис. 3. Графік еквівалентної функції розподілу, синтезований Python-програмою на основі кубічних сплайнів

```
In [81]: x=symbols('x')
In [82]: f1=lambdify(x,expr11,"numpy")
In [83]: x=np.linspace(0,1,21)
In [84]: f1(x)
Out[84]:
array([0.0405    , 0.04049605, 0.0404842 , 0.04046445, 0.0404368 ,
       0.04040125, 0.0403578 , 0.04030645, 0.0402472 , 0.04018005,
       0.040105  , 0.04002205, 0.0399312 , 0.03983245, 0.0397258 ,
       0.03961125, 0.0394888 , 0.03935845, 0.0392202 , 0.03907405,
       0.03892   ])
```

```
In [85]: y=symbols('y')
In [86]: f2=lambdify(y,expr22,"numpy")
In [87]: y=np.linspace(1,2,21)
In [88]:f2(y)
Out[88]:
array([0.039  , 0.03917, 0.03998, 0.04143, 0.04352, 0.04625, 0.04962,
       0.05363, 0.05828, 0.06357, 0.0695 , 0.07607, 0.08328, 0.09113,
       0.09962, 0.10875, 0.11852, 0.12893, 0.13998, 0.15167, 0.164  ])
In [89]: z=symbols('z')
In [90]: f3=lambdify(z,expr33,"numpy")
In [91]: z=np.linspace(2,3,21)
In [92]: f3(z)
Out[92]:
array([0.164  , 0.17635, 0.1881 , 0.19925, 0.2098 , 0.21975, 0.2291 ,
       0.23785, 0.246  , 0.25355, 0.2605 , 0.26685, 0.2726 , 0.27775,
       0.2823 , 0.28625, 0.2896 , 0.29235, 0.2945 , 0.29605, 0.297  ])
In [93]: u=symbols('u')
In [94]: f4=lambdify(u,expr44,"numpy")
In [95]: u=np.linspace(3,4,21)
In [96]: f4(u)
Out[96]:
array([0.29599  , 0.29660327, 0.2971771 , 0.29771148, 0.2982064 ,
       0.29866188, 0.2990779 , 0.29945447, 0.2997916 , 0.30008927,
       0.3003475 , 0.30056627, 0.3007456 , 0.30088548, 0.3009859 ,
       0.30104687, 0.3010684 , 0.30105047, 0.3009931 , 0.30089627,
       0.30076  ])
In [97]: v=symbols('v')
In [98]: f5=lambdify(v,expr55,"numpy")
In [99]: v=np.linspace(4,5,21)
In [100]: f5(v)
Out[100]:
array([0.30108  , 0.30092595, 0.3007798 , 0.30064155, 0.3005112 ,
       0.30038875, 0.3002742 , 0.30016755, 0.3000688 , 0.29997795,
       0.299895 , 0.29981995, 0.2997528 , 0.29969355, 0.2996422 ,
       0.29959875, 0.2995632 , 0.29953555, 0.2995158 , 0.29950395,
       0.2995   ])
In [101]: fig=plt.figure(facecolor='white')
<Figure size 432x288 with 0 Axes>
In [102]: plt.plot(x,f1(x),'-r',y,f2(y),'-g',z,f3(z),'-c',u,f4(u),'-b',\
          v,f5(v),'-r',linewidth=3)
Out[102]:
[<matplotlib.lines.Line2D at 0x1f8ac834310>,
 <matplotlib.lines.Line2D at 0x1f8ac8343d0>,
 <matplotlib.lines.Line2D at 0x1f8ac8344c0>,
 <matplotlib.lines.Line2D at 0x1f8ac8341f0>,
 <matplotlib.lines.Line2D at 0x1f8ac834610>]
```

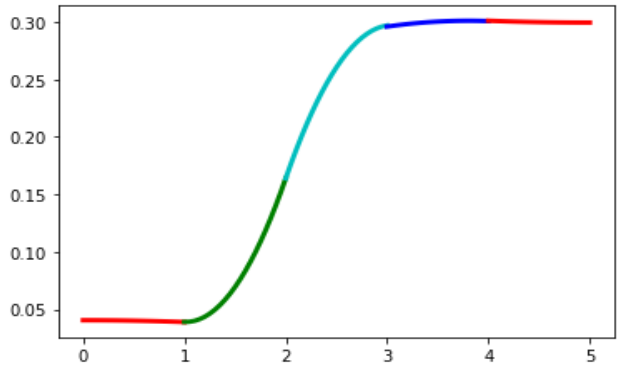


Рис. 4. Графік еквівалентної щільності розподілу, синтезований Python-програмою на основі кубічних сплайнів

### Аналіз отриманих результатів

Об'єднуючи результати, отримані після виконання команд In [41]—In [45], отримаємо еквівалентну сплайн-модель функції розподілу у вигляді

$$s=\begin{cases}-0,000526x^3+0,405x+0,03, & x\in[0,1],\\ 0,0426x^3-0,129x^2+0,17x-0,0132, & x\in[1,2],\\ -0,04x^3+0,366x^2-0,822x+0,648, & x\in[2,3],\\ -0,00263x^3+0,03x^2+0,187x-0,361, & x\in[3,4],\\ 0,000526x^3-0,00789x^2+0,339x-0,563, & x\in[4,5].\end{cases} \quad (4)$$

А об'єднуючи результати, отримані після виконання команд In [46]—In [50], отримаємо еквівалентну сплайн-модель щільності розподілу у вигляді

$$f=\begin{cases}-0,00158x^2+0,0405, & x\in[0,1],\\ 0,128x^2-0,259x+0,17, & x\in[1,2],\\ -0.12x^2+0,733x-0,822, & x\in[2,3],\\ -0.00789x^2+0,06x+0,187, & x\in[3,4],\\ 0.00158x^2-0,0158x+0,339, & x\in[4,5].\end{cases} \quad (5)$$

Графіки еквівалентних сплайн-моделей (4), (5), показаних на рис. 3, 4, підтверджують і правильність розв'язання поставленої задачі, і високу точність отриманих результатів, адже, якби точність була недостатньою, то кінці ліній об'єднаного графіка на границях відрізків не збігалися б.

### Висновки

Запропоновано метод синтезу еквівалентних моделей функцій розподілу та щільностей розподілу даних випадкового характеру, який не вимагає виконання процедури «вирівнювання» гістограм з використанням $\chi^2$-розподілу Пірсона в класичному варіанті синтезу цих моделей.

Запропонований метод синтезу базується на використанні кубічних сплайнів і їхньому лекальному припасуванні після трансформації гістограми у східчасту функцію розподілу.

Графік щільності розподілу (рис. 4), яка набула властивості класичного β-розподілу, підтверджує високу ефективність запропонованого способу синтезу, оскільки вихідну гістограму до такого розподілу класичним «вирівнюванням» за Пірсоном з високою довірчою ймовірністю привести не вдалося б.

## СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

[1] Б. І. Мокін, В. Б. Мокін, і О. Б. Мокін, *Практикум для самостійної роботи студентів з навчальної дисципліни «Методологія та організація наукових досліджень».* Частина 1: від постановки задачі до синтезу та ідентифікації математичної моделі. ВНТУ, Вінниця, 2018, 179 с. [Електронний ресурс]. Режим доступу: http://mokin.com.ua/files/articles/65/46/Mokin_SRS_MOND.pdf .

[2] Б. І. Мокін, О. Б., Мокін, і О. М. Косарук, *Ідеологія дуальності в вищій технічній освіті на основі інтеграції навчання з виробництвом,* моногр. Вінниця, Україна: ВНТУ, 2019, 224 с.

[3] Б. І. Мокін, О. Б. Мокін, О. О. Войцеховська, Д. О. Шалагай, і О. В. Бондарчук, «Еквівалентні моделі законів розподілу випадкових величин,» *Вісник Вінницького політехнічного інституту,* № 2, с. 25-35, 2024. https://doi.org/10.31649/1997-9266-2024-173-2-53-60 .

[4] Б. І. Мокін, О. О. Войцеховська, Д. О. Шалагай, і О. В. Бондарчук, «Еквівалентування законів розподілу,» *на LIII науково-технічній конференції факультету інтелектуальних інформаційних технологій та автоматизації Вінницького національного технічного університету*, м. Вінниця, 20-22 березня 2024 р., с. 952-956. [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2024/paper/view/20844/17254 .

[5] *Python*. [Electronic resource]. Available: https://www.python.org//downloads/ .

[6] Б. І. Мокін, О. О. Войцеховська, Н. В. Собчук, і О. В. Бондарчук, «Еквівалентування щільностей розподілу даних випадкового характеру,» на *Міжнародній науково-практичній конференції «Інформаційні технології та комп'ютерне моделювання»*, м. Івано-Франківськ, 21-24 травня 2024 р., с.103-104. [Електронний ресурс]. Режим доступу: https://journal.comp-sc.if.ua/test/index.php/ITCM .

[7] Р. Н. Квєтний, Я. В. Іванчук, І. В. Богач, О. Ю. Софина, і М. В. Барабан, *Методи та алгоритми комп'ютерних обчислень. Теорія і практика,* підруч. Вінниця, Україна: ВНТУ, 2023, 280 с.

Рекомендована кафедрою системного аналізу та інформаційних технологій ВНТУ

Стаття надійшла до редакції 29.03.2024

***Мокін Борис Іванович*** — академік НАПН України, д-р техн. наук, професор, професор кафедри системного аналізу та інформаційних технологій, e-mail: borys.mokin@gmail.com ;

***Войцеховська Ольга Олександрівна*** — д-р філософії, старший викладач кафедри системного аналізу та інформаційних технологій, e-mail: olgav1085@gmail.com ;

***Собчук Наталія Валеріївна*** — канд. тех. наук, доцент, доцент кафедри електричних станцій та систем, e-mail: natashasobchuk37@gmail.com ;

***Бондарчук Олексій Валерійович*** — аспірант кафедри системного аналізу та інформаційних технологій, e-mail: alexey.bondarchuk@aleax.me .

Вінницький національний технічний університет, Вінниця

**B. I. Mokin**[1]
**O. O. Voitsekhovska**[1]
**N. V. Sobchuk**[1]
**O. V. Bondarchuk**[1]

# Equivalent Models of Random Data Distribution Functions

[1]Vinnytsia National Technical University

*This paper presents the results of the research carried out by the authors in the process of forming the report "Equivalence of random data distribution densities", delivered at the International Scientific and Practical Conference "Information Technologies and Computer Modeling (ITCM-24)", held in Ivano-Frankivsk in the period from May 21 to 24, 2024, in which the method for the synthesis of a statistical estimate of the equivalent density of a random data distribution is proposed, which does not require "histogram alignment" using the Pearson $\chi^2$ -distribution. At the first stage, the method proposed by the authors uses the crunching of a histogram constructed using a relatively small sample of random data, followed by cumulative summation to determine the equivalent distribution function of these data on a set of their available values. At the second stage of the method proposed by the authors, the constructed equivalent distribution function of random data is interpolated by cubic splines according to the algorithm that uses the mathematical relations derived by the authors. And at the third stage of the method proposed by the authors, by differentiating the equivalent mathematical model of the distribution of random data, obtained at the previous stages of the implementation of this method, an equivalent mathematical model of the density of the distribution of random data is synthesized, using which the primary base histogram was created. The paper contains or mentions all the Python programs necessary for the implementation of all stages of the proposed method of synthesis of the equivalent density model of the random data distribution , as well as the example of solving a specific practical problem with explanations of all actions necessary to perform at each of the stages implementation of the proposed method.*

**Keywords:** random data, histogram, histogram shredding, cumulative sum, distribution function, equivalence, interpolation, cubic splines, distribution density, Python programs.

***Mokin Borys I.*** — Academician of NAPS of Ukraine, Dr. Sc. (Eng.), Professor, Professor with the Chair of System Analysis and Information Technologies, e-mail: borys.mokin@gmail.com ;

***Voitsekhovska Olha O.*** — PhD, Senior Lecturer with the Chair of System Analysis and Information Technologies, e-mail: olgav1085@gmail.com ;

***Sobchuk Nataliia V***. — Cand. Sc. (Eng.), Associate Professor, Associate Professor with the Chair of Electrical Power Stations and Systems, e-mail: natashasobchuk37@gmail.com ;

***Bondarchuk Oleksii V.*** — Post-Graduate Student with the Chair of System Analysis and Information Technologies, e-mail: alexey.bondarchuk@aleax.me