ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА КОМП'ЮТЕРНА ТЕХНІКА

https://doi.org/10.31649/1997-9266-2025-180-3-90-100

УДК [004.42:517.443:681.518.3](045)

О. О. Войцеховська[1]
Б. І. Мокін[1]
О. Б. Мокін[1]

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ РЕАЛІЗАЦІЇ ФУР'Є-ІНТЕГРАЛЬНОГО МЕТОДУ ІДЕНТИФІКАЦІЇ ДЛЯ ВІДНОВЛЕННЯ ВХІДНИХ СИГНАЛІВ ІНФОРМАЦІЙНО-ВИМІРЮВАЛЬНИХ СИСТЕМ

[1]Вінницький національний технічний університет

Розроблено інформаційну технологію реалізації Фур'є-інтегрального методу ідентифікації для відновлення вхідних сигналів інформаційно-вимірювальних систем за їхніми вихідними сигналами, створеного у 80 роках минулого сторіччя Б. І. Мокіним та узагальненого О. Б Мокіним. В основу розробленої інформаційної технології покладено комп'ютерну програму, створену на мові Python. Перша частина цієї Python-програми відновлює вхідний сигнал інформаційно-вимірювальної системи за її вихідним сигналом у вигляді зрізаного ряду Фур'є, що зумовлює появу на графіку цього відновленого вхідного сигналу перепадів, зумовлених скінченною кількістю відновлених гармонічних складових. Друга частина цієї Python-програми трансформує зрізаний ряд Фур'є у ряд Фур'є з нескінченною кількістю гармонічних складових, тобто формує еквівалентну модель вхідного ряду, очищену від перепадів, зумовлених скінченною кількістю відновлених гармонічних складових. З відповідним обґрунтуванням трансформацію зрізаного ряду Фур'є у ряд Фур'є з нескінченною кількістю гармонічних складових здійснено з використанням нелінійного варіанта методу найменших квадратів. В процесі реалізації Python-програми продемонстровано як впливають динамічні характеристики інформаційно-вимірювальних систем на вихідні сигнали цих систем, що підтверджено і розрахунками і графічно, і є підтвердженням необхідності доповнення інформаційно-вимірювальних систем інформаційними технологіями відновлення їхніх вхідних сигналів за їхніми вихідними сигналами зі структурою, запропонованою в цій статті. Здійснено аналіз отриманих результатів і запропоновані рішення, згідно з якими структуру запропонованої інформаційної технології та структуру Python-програми, покладеної в основу цієї інформаційної технології, можна привести у відповідність іншим умовам функціонування інформаційно-вимірювальної системи, вхідний сигнал якої відновлюється за її вихідним сигналом.

Ключові слова: інформаційна технологія, відновлення вхідних сигналів інформаційно-вимірювальних систем по вихідним сигналам, Фур'є-інтегральний метод ідентифікації, комп'ютерна програма, Python

Вихідні передумови та постановка задачі

В статтях, опублікованих Б. І. Мокіним ще в наукових журналах Радянського Союзу, таких, як «Изв. Вузов «Электромеханика» (1974 г., № 12; 1976 г., № 12), «Электронная техника» (1975 г., вып. 5), «Метрология» (1975 г., № 10; 1979 г., № 9), «Автоматизированные системы управления и приборы автоматики»(1977 г., вып. 41, вып. 42), було зауважено на суттєвий вплив, який здійснюють на вихідні сигнали різноманітних інформаційно-вимірювальних систем (ІВС) динамічні характеристики цих систем, і продемонстровано які похибки вносяться у вихідні сигнали під час практичного використання цих ІВС.

В цих же статтях запропоновано розробити метод «очищення» вихідних сигналів ІВС від динамічних спотворень, зумовлених динамічними характеристиками цих ІВС.

І такий метод, названий його автором Фур'є-інтегральним методом ідентифікації динамічних

систем та сигналів (ФІМІ), розроблено Б. І. Мокіним і представлено у другому розділі його докторської дисертації «Параметрический синтез эргатических систем управления технологическими объектами», захищеної ним у 1984 році в спецраді Д 068.14.03 Київського політехнічного інституту.

Але, незважаючи на те, що розрахункові співвідношення і алгоритми реалізації ФІМІ у 1990 році були подані в монографії [1], доповнені О. Б. Мокіним в роботі [2], а в 2010 році уже увійшли і в навчальний посібник з грифом міністерства [3], широкого застосування в практиці досліджень цей метод не набув, як внаслідок скептичного ставлення багатьох дослідників до проблеми відновлення вхідних сигналів за спотвореними динамічними похибками вихідних сигналів ІВС, так і внаслідок небажання дослідників створювати комп'ютерні програми реалізації ФІМІ, які, на думку автора методу, вони повинні були створювати самостійно під свої задачі у вигляді власних інформаційних технологій (ІТ).

Але оскільки проблема відновлення вхідних сигналів ІВС за їхніми вихідними сигналами є не тривіальною і, як нами буде показано далі, не малозначущою, то автори вирішили «оживити» ФІМІ шляхом створення ІТ його реалізації з використанням комп'ютерної програми, розробленої на мові Python [4], і продемонструвати з її використанням, яку інформаційну небезпеку несуть в собі вихідні сигнали ІВС, якщо до них не застосовувати ІТ їхнього «очищення» від динамічних спотворень. А для тих дослідників, які недостатньо володіють програмуванням на мові Python, можемо порекомендувати наші навчальні посібники [5]—[7], в яких є достатньо інформації і про Python і про його застосування для реалізації алгоритмів розв'язання конкретних задач.

Вихідні передумови для створення комп'ютерної програми, яка є базовою в ІТ реалізації ФІМІ в задачі відновлення вхідних сигналів ІВС за їхніми вихідними сигналами, візьмемо з роботи [3].

Оскільки в ІВС використовується лише лінійний відрізок характеристики «вхід–вихід», то їх без будь-яких натяжок можна віднести до класу лінійних динамічних систем, для яких, як це показано в підручниках з теоретичних основ електротехніки та теорії автоматичного керування, справедливим є інтегральне рівняння згортки

$$y(t) = \int_0^{\infty} x(t-\tau)\, g(\tau)\, d\tau \tag{1}$$

або у загальнішому вигляді

$$y = Ax, \tag{2}$$

де $x(t)$ — сигнал який діє на вході лінійної динамічної системи з імпульсною перехідною характеристикою $g(t)$; $y(t)$ — реакція; A — оператор системи.

Якщо моделі сигналів подати у вигляді рядів Фур'є з тим самим спектром частот, то кожну гармонічну складову сигналу $x(t)$ можна однозначно пов'язати із гармонічною складовою тієї ж частоти сигналу $y(t)$ алгебраїчним виразом, який містить у собі тільки коефіцієнти Фур'є сигналів $x(t)$, $y(t)$ та спектральні складові амплітудно-фазової частотної характеристики (АФЧХ) $W(j\omega)$ системи на цій же частоті.

Якщо відрізок часу спостереження вихідного сигналу $y(t)$ дорівнює T, тоді, як відомо з курсу математичного аналізу, цей сигнал можна подати на цьому відрізку у вигляді ряду Фур'є

$$y(t) = \frac{m_0}{2} + \sum_{i=1}^{\infty} m_i \cos i\,\omega_1 t + \sum_{i=1}^{\infty} n_i \sin i\,\omega_1 t\,. \tag{3}$$

Представимо формально невідомий нам вхідний сигнал $x(t)$ теж рядом Фур'є на тому ж відрізку часу, тобто у вигляді

$$x(t) = \frac{a_0}{2} + \sum_{i=1}^{\infty} a_i \cos i\,\omega_1 t + \sum_{i=1}^{\infty} b_i \sin i\,\omega_1 t\,. \tag{4}$$

У виразах (3), (4) $m_i, i = \overline{0,\infty}$; $n_i, i = \overline{1,\infty}$; $a_i, i = \overline{0,\infty}$; $b_i, i = \overline{1,\infty}$ — коефіцієнти Фур'є сигналів $x(t)$, $y(t)$, які знаходяться за відомими з курсу математичного аналізу формулами

$$\begin{Bmatrix} a_i \\ m_i \end{Bmatrix} = \frac{2}{T}\int_0^T \begin{Bmatrix} x(t) \\ y(t) \end{Bmatrix} \cos i\omega_1 t\, dt, \quad i = 0, 1,\ 2, \ldots; \tag{5}$$

$$\begin{Bmatrix} b_i \\ n_i \end{Bmatrix} = \frac{2}{T}\int_0^T \begin{Bmatrix} x(t) \\ y(t) \end{Bmatrix} \sin i\omega_1 t\, dt, \quad i = 1, 2, \ldots, \tag{6}$$

де $\omega_1 = \dfrac{2\pi}{T}$ — частота першої гармоніки.

Підставляючи ряди (3) та (4) в інтеграл згортки (1), після низки перетворень, отримаємо вираз

$$\begin{aligned} \frac{m_0}{2} + \sum_{i=1}^{\infty} m_i \cos i\omega_1 t + \sum_{i=1}^{\infty} n_i \sin i\omega_1 t = \frac{a_0}{2} R(0) + \sum_{i=1}^{\infty} \left(a_i \cdot R(i\omega_1) + b_i \cdot Q(i\omega_1)\right) \cos i\omega_1 t + \\ + \sum_{i=1}^{\infty} \left(b_i \cdot R(i\omega_1) - a_i \cdot Q(i\omega_1)\right) \sin i\omega_1 t, \end{aligned} \tag{7}$$

де $R(i\omega_1)$, $Q(i\omega_1)$ — відповідно значення дійсної $R(\omega)$ та уявної $Q(\omega)$ частотних характеристик ІВС на частотах $i\omega_1$, $i = 0, 1, 2, \ldots$, що пов'язані з її АФЧХ відомим співвідношенням

$$W(j\omega) = R(\omega) + jQ(\omega). \tag{8}$$

Оскільки вираз (7) — це тотожність, то справедливим є, по-перше, рівняння

$$a_0 R(0) = m_0 \tag{9}$$

для постійних складових a_0, m_0 сигналів $x(t)$, $y(t)$, а, по-друге, справедливою є система рівнянь

$$\begin{cases} a_i R(i\omega_1) + b_i Q(i\omega_1) = m_i, \\ b_i R(i\omega_1) - a_i Q(i\omega_1) = n_i, \end{cases} \tag{10}$$

де $i = 1, 2, \ldots$ для всіх інших коефіцієнтів Фур'є a_i, b_i, m_i, n_i сигналів $x(t)$, $y(t)$.

З теорії рядів Фур'є та властивостей уявної частотної характеристики $Q(\omega)$ відомо, що

$$\begin{cases} b_0 = 0, \\ Q(0) = 0, \end{cases} \tag{11}$$

а тому рівняння (9) теж можна отримати з системи (10), коли $i = 0$. Тож у подальшому будемо розглядати лише систему рівнянь (10), припустивши в ній: $i = 0, 1, 2, \ldots$.

Розв'яжемо цю систему відносно a_i, b_i. Отримаємо:

$$a_i = \frac{m_i R(i\omega_1) - n_i Q(i\omega_1)}{R^2(i\omega_1) + Q^2(i\omega_1)}, \quad i = 0, 1, 2, \ldots; \tag{12}$$

$$b_i = \frac{n_i R(i\omega_1) + m_i Q(i\omega_1)}{R^2(i\omega_1) + Q^2(i\omega_1)}, \quad i = 0, 1, 2, \ldots. \tag{13}$$

Далі розв'язання задачі відновлення вхідного сигналу $x(t)$ ІВС будемо здійснювати за таким алгоритмом: спочатку за реалізацією вихідного сигналу $y(t)$, зафіксованою на відрізку часу T, за допомогою формул (5) і (6) визначаємо коефіцієнти Фур'є m_i, n_i цього сигналу для значень $i = \overline{0, N}$. Потім для цих же значень i за заданими частотними характеристиками $R(\omega)$, $Q(\omega)$ ІВС визначаємо їхні значення на частотах $i\omega_1$, тобто $R(i\omega_1)$, $Q(i\omega_1)$. А підставляючи m_i, n_i, $R(i\omega_1)$, $Q(i\omega_1)$ у формули (12), (13), отримаємо для кожного значення i із множини $\overline{0, N}$ пару коефіцієнтів Фур'є a_i, b_i вхідного сигналу $x(t)$.

Підстановкою всіх цих коефіцієнтів в зрізаний ряд (4) і завершуємо розв'язання задачі ідентифікації (відновлення) вхідного сигналу ІВС $x(t)$ за її вихідним сигналом $y(t)$.

Отже ми визначили усі вихідні передумови, які нам будуть потрібні в процесі створення комп'ютерної програми для ІТ, яка відновлюватиме вхідний сигнал ІВС за її вихідним сигналом, тож можемо розпочати процес її створення і реалізації.

Розв'язання поставленої задачі

Для створення можливості оцінити отримані результати розв'язання поставленої задачі конкретизуємо ІВС та математичну модель її вихідного сигналу.

Отже будемо розглядати ІВС для вимірювання температури потоку води чи газу в трубопроводі з терморезисторним первинним перетворювачем, включеним за мостовою схемою, та з підсилювачем сигналу, який виникає в діагоналі мостового перетворювача.

Передаточну функцію такої ІВС можна записати у вигляді

$$W(p)=\frac{K}{T_s p+1}, \tag{14}$$

де T_s — стала часу терморезисторного первинного перетворювача, а K — коефіцієнт підсилення підсилювача сигналу.

Якщо використати терморезистор із захисною діелектричною оболонкою, то його стала часу T_s в залежності від товщини та матеріалу оболонки знаходитиметься в діапазоні значень (1—2) одиниць за секунду. Приймемо, що

$$T_s=1. \tag{15}$$

Для зручного порівнювання вихідного сигналу $y(t)$ з вхідним сигналом $x(t)$ налаштуємо підсилювач так, щоб мати

$$K=1. \tag{16}$$

Підставляючи умови (15) і (16) у вираз (14), отримаємо конкретне значення передаточної функції терморезисторної ІВС у вигляді

$$W(p)=\frac{1}{p+1}. \tag{17}$$

Припустимо, що після включення цієї ІВС в роботу на її виході ми протягом 10 секунд зафіксували сигнал $y(t)$, математична модель якого має вигляд

$$y(t)=5\sqrt{t}\,. \tag{18}$$

З використанням виразів (17) та (18) Python-програма відновлення вхідного сигналу $x(t)$ буде мати такий вигляд:

```
In [1]: import sympy
In [2]: from sympy import*
In [3]: p=symbols('p')
In [4]: r,v= symbols('r v',real=True)
In [5]: p=r+v*I
In [6]: W=Function ('W')(p)
In [7]: W=1/(p+1)
In [8]: W_expr=W
In [9]: W1=Function('W1')(v)
In [10]: W1=W_expr.subs(p,r+v*I)
In [11]: W1
Out[11]: 1/(r + I*v + 1)
In [12]: W1_expr=W1
In [13]: W1=W1_expr.subs(r,0)
In [14]: W1
Out[14]: 1/(I*v + 1)
In [15]: R=Function('R')(v)
In [16]: Q=Function('Q')(v)
In [17]: R=re(W1);R
Out[17]: 1/(v**2 + 1)
In [18]: Q=im(W1);Q
Out[18]: -v/(v**2 + 1)
In [19]: k,T,v0=symbols('k T v0')
In [20]: T=10
In [21]: v0=(2*pi/T).evalf(3);v0
Out[21]: 0.628
In [22]: R_expr=R
In [23]: R_k_expr=R_expr.subs(v,k*\
         v0).evalf(3);R_k_expr
Out[23]: 1/(0.395*k**2 + 1.0)
```

```
In [24]: Q_expr=Q
In [25]: Q_k_expr=Q_expr.subs(v,k*\
         v0).evalf(3);Q_k_expr
Out[25]: -0.628*k/(0.395*k**2 + 1.0)
In [26]: R_0_expr=R_k_expr.subs(k,\
         0);R_0_expr
Out[26]: 1.00
In [27]: Q_0_expr=Q_k_expr.subs(k,\
         0);Q_0_expr
Out[27]: 0
In [28]: R_1_expr=R_k_expr.subs(k,\
         1).evalf(3)
In [29]: Q_1_expr=Q_k_expr.subs(k,\
         1).evalf(3)
In [30]: R_2_expr=R_k_expr.subs(k,\
         2).evalf(3)
In [31]: Q_2_expr=Q_k_expr.subs(k,\
         2).evalf(3)
In [32]: R_3_expr=R_k_expr.subs(k,\
         3).evalf(3)
In [33]: Q_3_expr=Q_k_expr.subs(k,\
         3).evalf(3)
In [34]: R_4_expr=R_k_expr.subs(k,\
         4).evalf(3)
In [35]: Q_4_expr=Q_k_expr.subs(k,\
         4).evalf(3)
In [36]: R_5_expr=R_k_expr.subs(k,\
         5).evalf(3)
In [37]: Q_5_expr=Q_k_expr.subs(k,\
         5).evalf(3)
In [38]: R_6_expr=R_k_expr.subs(k,\
         6).evalf(3)
In [39]: Q_6_expr=Q_k_expr.subs(k,\
         6).evalf(3)
In [40]: R_7_expr=R_k_expr.subs(k,\
         7).evalf(3)
In [41]: Q_7_expr=Q_k_expr.subs(k,\
         7).evalf(3)
In [42]: R_8_expr=R_k_expr.subs(k,\
         8).evalf(3)
In [43]: Q_8_expr=Q_k_expr.subs(k,\
         8).evalf(3)
In [44]: R_9_expr=R_k_expr.subs(k,\
         9).evalf(3)
In [45]: Q_9_expr=Q_k_expr.subs(k,\
         9).evalf(3)
In [46]: t=symbols('t')
In [47]: f=Function('f')(v,t)
In [48]: g=Function('g')(v,t)
In [49]: f=cos(v*t)
In [50]: f_expr=f
In [51]: g=sin(v*t)
In [52]: g_expr=g

In [53]: f_k_t_expr=f_expr.subs(v,k*\
         v0).evalf(3);f_k_t_expr
Out[53]: cos(0.628*k*t)
In [54]: g_k_t_expr=g_expr.subs(v,k*\
         v0).evalf(3);g_k_t_expr
Out[54]: sin(0.628*k*t)
In [55]: f_0_t_expr=f_k_t_expr.subs(k,\
         0).evalf(3);f_0_t_expr
Out[55]: 1.00
In [56]: g_0_t_expr=g_k_t_expr.subs(k,\
         0).evalf(3);g_0_t_expr
Out[56]: 0
In [57]: f_1_t_expr=f_k_t_expr.subs(k,\
         1).evalf(3);f_1_t_expr
Out[57]: cos(0.628*t)
In [58]: g_1_t_expr=g_k_t_expr.subs(k,\
         1).evalf(3);g_1_t_expr
In [59]: f_2_t_expr=f_k_t_expr.subs(k,\
         2).evalf(3);f_2_t_expr
Out[59]: cos(1.26*t)
In [60]: g_2_t_expr=g_k_t_expr.subs(k,\
         2).evalf(3);g_2_t_expr
Out[60]: sin(1.26*t)
In [61]: f_3_t_expr=f_k_t_expr.subs(k,\
         3).evalf(3);f_3_t_expr
Out[61]: cos(1.88*t)

In [62]: g_3_t_expr=g_k_t_expr.subs(k,\
         3).evalf(3);g_3_t_expr
Out[62]: sin(1.88*t)
In [63]: f_4_t_expr=f_k_t_expr.subs(k,\
         4).evalf(3);f_4_t_expr
Out[63]: cos(2.51*t)
In [64]: g_4_t_expr=g_k_t_expr.subs(k,\
         4).evalf(3);g_4_t_expr
Out[64]: sin(2.51*t)
In [65]: f_5_t_expr=f_k_t_expr.subs(k,\
         5).evalf(3);f_5_t_expr
Out[65]: cos(3.14*t)
In [66]: g_5_t_expr=g_k_t_expr.subs(k,\
         5).evalf(3);g_5_t_expr
Out[66]: sin(3.14*t)
In [67]: f_6_t_expr=f_k_t_expr.subs(k,\
         6).evalf(3);f_6_t_expr
Out[67]: cos(3.77*t)
In [68]: g_6_t_expr=g_k_t_expr.subs(k,\
         6).evalf(3);g_6_t_expr
Out[68]: sin(3.77*t)
In [69]: f_7_t_expr=f_k_t_expr.subs(k,\
         7).evalf(3);f_7_t_expr
Out[69]: cos(4.4*t)
In [70]: g_7_t_expr=g_k_t_expr.subs(k,\
         7).evalf(3);g_7_t_expr
Out[70]: sin(4.4*t)

In [71]: f_8_t_expr=f_k_t_expr.subs(k,\
         8).evalf(3);f_8_t_expr
Out[71]: cos(5.03*t)
In [72]: g_8_t_expr=g_k_t_expr.subs(k,\
         8).evalf(3);g_8_t_expr
Out[72]: sin(5.03*t)
```

```
In [73]: f_9_t_expr=f_k_t_expr.subs(k,\
         9).evalf(3);f_9_t_expr
Out[73]: cos(5.65*t)
In [74]: g_9_t_expr=g_k_t_expr.subs(k,\
         9).evalf(3);g_9_t_expr
Out[74]: sin(5.65*t)
Out[58]: sin(0.628*t)
In [75]: y=Function('y')(t)
In [76]: y=5*t**0.5
In [77]: a=symbols('a:10')
In [78]: b=symbols('b:10')
In [79]: m=symbols('m:10')
In [80]: n=symbols('n:10')
In [81]: m0=((2/T)*integrate(y*f_0_t_expr,(t,0,T))).evalf(3)
In [82]: m1=((2/T)*integrate(y*f_1_t_expr,(t,0,T))).evalf(3)
In [83]: n1=((2/T)*integrate(y*g_1_t_expr,(t,0,T))).evalf(3)
In [84]: m2=((2/T)*integrate(y*f_2_t_expr,(t,0,T))).evalf(3)
In [85]: n2=((2/T)*integrate(y*g_2_t_expr,(t,0,T))).evalf(3)
In [86]: m3=((2/T)*integrate(y*f_3_t_expr,(t,0,T))).evalf(3)
In [87]: n3=((2/T)*integrate(y*g_3_t_expr,(t,0,T))).evalf(3)
In [88]: m4=((2/T)*integrate(y*f_4_t_expr,(t,0,T))).evalf(3)
In [89]: n4=((2/T)*integrate(y*g_4_t_expr,(t,0,T))).evalf(3)
In [90]: m5=((2/T)*integrate(y*f_5_t_expr,(t,0,T))).evalf(3)
In [91]: n5=((2/T)*integrate(y*g_5_t_expr,(t,0,T))).evalf(3)
In [92]: m6=((2/T)*integrate(y*f_6_t_expr,(t,0,T))).evalf(3)
In [93]: n6=((2/T)*integrate(y*g_6_t_expr,(t,0,T))).evalf(3)
In [94]: m7=((2/T)*integrate(y*f_7_t_expr,(t,0,T))).evalf(3)
In [95]: n7=((2/T)*integrate(y*g_7_t_expr,(t,0,T))).evalf(3)
In [96]: m8=((2/T)*integrate(y*f_8_t_expr,(t,0,T))).evalf(3)
In [97]: n8=((2/T)*integrate(y*g_8_t_expr,(t,0,T))).evalf(3)
In [98]: m9=((2/T)*integrate(y*f_9_t_expr,(t,0,T))).evalf(3)
In [99]: n9=((2/T)*integrate(y*g_9_t_expr,(t,0,T))).evalf(3)
In[100]: a0=m0/R_0_expr
In [101]: a1=(m1*R_1_expr-n1*Q_1_expr)/(R_1_expr**2+Q_1_expr**2)
In [102]: b1=(n1*R_1_expr+m1*Q_1_expr)/(R_1_expr**2+Q_1_expr**2)
In [103]: a2=(m2*R_2_expr-n2*Q_2_expr)/(R_2_expr**2+Q_2_expr**2)
In [104]: b2=(n2*R_2_expr+m2*Q_2_expr)/(R_2_expr**2+Q_2_expr**2)
In [105]: a3=(m3*R_3_expr-n3*Q_3_expr)/(R_3_expr**2+Q_3_expr**2)
In [106]: b3=(n3*R_3_expr+m3*Q_3_expr)/(R_3_expr**2+Q_3_expr**2)
In [107]: a4=(m4*R_4_expr-n4*Q_4_expr)/(R_4_expr**2+Q_4_expr**2)
In [108]: b4=(n4*R_4_expr+m4*Q_4_expr)/(R_4_expr**2+Q_4_expr**2)
In [109]: a5=(m5*R_5_expr-n5*Q_5_expr)/(R_5_expr**2+Q_5_expr**2)
In [110]: b5=(n5*R_5_expr+m5*Q_5_expr)/(R_5_expr**2+Q_5_expr**2)
In [111]: a6=(m6*R_6_expr-n6*Q_6_expr)/(R_6_expr**2+Q_6_expr**2)
In [112]: b6=(n6*R_6_expr+m6*Q_6_expr)/(R_6_expr**2+Q_6_expr**2)
In [113]: a7=(m7*R_7_expr-n7*Q_7_expr)/(R_7_expr**2+Q_7_expr**2)
In [114]: b7=(n7*R_7_expr+m7*Q_7_expr)/(R_7_expr**2+Q_7_expr**2)
In [115]: a8=(m8*R_8_expr-n8*Q_8_expr)/(R_8_expr**2+Q_8_expr**2)
In [116]: b8=(n8*R_8_expr+m8*Q_8_expr)/(R_8_expr**2+Q_8_expr**2)
In [117]: a9=(m9*R_9_expr-n9*Q_9_expr)/(R_9_expr**2+Q_9_expr**2)
In 118]: b9=(n9*R_9_expr+m9*Q_9_expr)/(R_9_expr**2+Q_9_expr**2)
In [119]: x=Function('x')(t)
In [120]: x=a0/2+a1*f_1_t_expr+b1*g_1_t_expr+a2*f_2_t_expr+\
          b2*g_2_t_expr+ a3*f_3_t_expr+b3*g_3_t_expr+a4*f_4_t_expr+\
          b4*g_4_t_expr+ a5*f_5_t_expr+b5*g_5_t_expr+a6*f_6_t_expr+\
          b6*g_6_t_expr+ a7*f_7_t_expr+b7*g_7_t_expr+a8*f_8_t_expr+\
          b8*g_8_t_expr+ a9*f_9_t_expr+b9*g_9_t_expr;x
Out[120]:
-3.26*sin(0.628*t) - 1.64*sin(1.26*t) - 1.06*sin(1.88*t) - 0.767*sin(2.51*t) - 0.587*sin(3.14*t) -
0.463*sin(3.77*t) - 0.399*sin(4.4*t) - 0.322*sin(5.03*t) - 0.27*sin(5.65*t) - 3.26*cos(0.628*t) -
2.95*cos(1.26*t) - 2.91*cos(1.88*t) - 2.9*cos(2.51*t) - 2.91*cos(3.14*t) - 2.92*cos(3.77*t) -
2.92*cos(4.4*t) - 2.93*cos(5.03*t) - 2.94*cos(5.65*t) + 10.5
```

```
In [121]: from sympy.plotting import plot
In [122]: gr1=plot(y,(t,0,10),show=False,line_color='g')
In [123]: gr2=plot(x,(t,0,10),show=False,line_color='r')
In 124]: gr1.extend(gr2)
In [125]: gr1.show()

In [126]: x_expr=x
In [127]: x_k_expr=x_expr.subs(t,k)
In [128]: x_0_expr=x_k_expr.subs(k,\
          0);x_0_expr
Out[128]: -16.1
In [129]: x_1_expr=x_k_expr.subs(k,\
          1);x_1_expr
Out[129]: 9.30
In [130]: x_2_expr=x_k_expr.subs(k,\
          2);x_2_expr
Out[130]: 10.5
In [131]: x_3_expr=x_k_expr.subs(k,\
          3);x_3_expr
Out[131]: 11.7
In [132]: x_4_expr=x_k_expr.subs(k,\
          4);x_4_expr
Out[132]: 12.8
In [133]: x_5_expr=x_k_expr.subs(k,\
          5);x_5_expr
Out[133]: 13.8
In [134]: x_6_expr=x_k_expr.subs(k,\
          6);x_6_expr
Out[134]: 14.7
In [135]: x_7_expr=x_k_expr.subs(k,\
          7);x_7_expr
Out[135]: 15.6
In [136]: x_8_expr=x_k_expr.subs(k,\
          8);x_8_expr
Out[136]: 16.3
In [137]: x_9_expr=x_k_expr.subs(k,\
          9);x_9_expr
Out[137]: 16.9
In [138]: print(x_0_expr,x_1_expr,x_2_expr,x_3_expr,x_4_expr,x_5_expr,
                x_6_expr,x_7_expr,x_8_expr,x_9_expr)
   -16.1 9.30 10.5 11.7 12.8 13.8 14.7 15.6 16.3 16.9

In [139]: import numpy as np
In [140]: import matplotlib
In [141]: import matplotlib.pyplot\
           as plt
In [142]: import scipy
In [143]: import scipy.linalg as la
In [144]: from scipy.optimize\
           import curve_fit
In [145]: def model(t,A,a):
                return A*(a*t)**0.5

In [146]: t=np.array([0.0,1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0])
In [147]: z=np.array([-16.1,9.30,10.5,11.7,12.8,13.8,14.7,15.6,16.3,16.9])
In [148]: initial_guess=[5.0,1.0]
In [149]: params=curve_fit(model,t,z,p0=initial_guess)
In [150]: A=params[0][0]
In [151]: A=A.round(3);A
Out[151]: 6.09
In [152]: a=params[0][1]
In [153]: a=a.round(3);a
Out[153]: 1.007
In [154]: x1=model(t,A,a);x1
Out[154]:
array([ 0.        ,  6.11127783,  8.64265199, 10.5850437 , 12.22255566,
       13.66523265, 14.96951236, 16.16892133, 17.28530398, 18.33383349])
In [155]: def y(t):
              return 5*t**0.5

In [156]: yvec=np.vectorize(y)
In [157]: y1=yvec(t); y1
Out[157]:
array([ 0.        ,  5.        ,  7.07106781,  8.66025404, 10.        ,
       11.18033989, 12.24744871, 13.22875656, 14.14213562, 15.        ])
In [158]: plt.plot(t,y1,'-g',t,x1,'-r',linewidth=4)
Out[158]:
[<matplotlib.lines.Line2D at 0x1b4511f97c0>,
 <matplotlib.lines.Line2D at 0x1b4511f9880>]
```

Зауважимо, що у приведеній вище Python-програмі, покладеній в основу ІТ відновлення вхідних сигналів ІВС за ФІМІ з використанням вихідних сигналів:

– командою In [7] в програму вводиться передаточна функція $W(p)$;

– командою In [17] обчислюється дійсна частотна характеристика $R(\omega)$, в якій для зручності грецька ω замінена на латинську υ;

– командою In [18] обчислюється уявна частотна характеристика $Q(\omega)$, в якій для зручності грецька ω замінена на латинську υ;

– командою In [20] в програму вводиться відрізок часу T, протягом якого фіксується реалізація вихідного сигналу ІВС (в тих же одиницях часу, що й стала часу T_s ІВС);

– командою In [21] обчислюється частота першої гармонічної складової ω_0, яка в програмі позначена для зручності υ_0;

– командами In [22] — In [45] обчислюються значення дійсної та уявної частотних характеристик ІВС на частотах $k\upsilon_0 (k = 0,1,...,9)$;

– командами In [49] — In [74] створюються базові функції гармонічних складових;

– командою In [76] в програму вводиться вихідний сигнал $y(t)$ ІВС;

– командами In [81] — In [99] обчислюються значення коефіцієнтів Фур'є вихідного сигнал $y(t)$ ІВС;

– командами In [100] — In [118] обчислюються значення коефіцієнтів Фур'є вхідного сигналу $x(t)$ ІВС;

– командою In [120] формується із 9 гармонічних складових відновлений вхідний сигнал $x(t)$ ІВС;

– командами In [121] — In [125] створюються графіки вихідного сигналу $y(t)$ та відновленого дев'ятьма гармонічними складовими вхідного сигналу $x(t)$ ІВС, які показані на рис. 1;

– командами In [144] — In [153] із застосуванням МНК визначається математична модель відновленого вхідного сигналу $x(t)$ ІВС, еквівалентна нескінченній кількості гармонічних складових;

– командами In [154] — In [168] створюються графіки вихідного сигналу $y(t)$ та відновленого еквівалентною моделлю вхідного сигналу $x(t)$ ІВС, які показані на рис. 2.

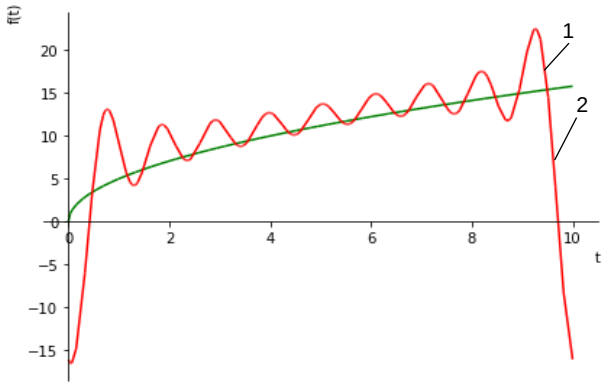

Рис. 1. Графіки сигналів: 1 — на виході ІВС (плавна крива, що починається з нуля) з передаточною функцією, заданою виразом (17); 2 — на вході ІВС (коливальна крива, складена з дев'яти гармонічних складових)

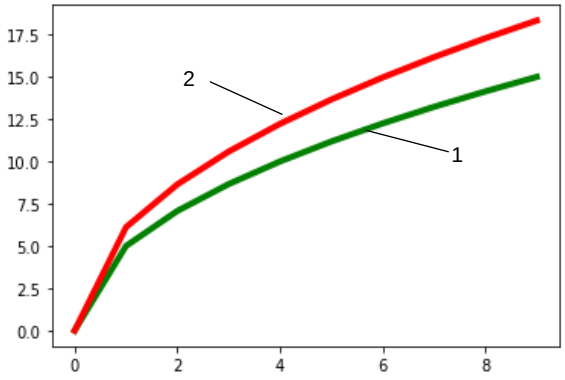

Рис. 2. Графіки сигналів: 1 — на виході ІВС ; 2 — на вході ІВС (крива, складена з усіх гармонічних складових)

Аналіз отриманих результатів

Першим, що ми хочемо пояснити в нашому аналізі отриманих результатів, це те, чому складено Python-програму реалізації ФІМІ в задачі відновлення вхідних сигналів $x(t)$ ІВС з використанням командних рядків, а не у формі файлу, та чому блоки команд In [22] — In [45], In [49] — In [74], In [81] — In [99], In [100] — In [118] не реалізовані у вигляді **for**-циклів, що привело б до зменшення кількості використаних команд у цій програмі.

Пояснення наше буде таким: саме завдяки командним рядкам структура Python-програми є повністю адекватною структурі ФІМІ, і результати кожного обчислення за співвідношеннями цього

методу легко проконтролювати на кожному кроці виконання програми, чого не можна досягти в разі реалізації програми файлом, особливо ж із застосуванням у ній **for**-циклів. А необхідність супроводжувати кожний цикл у файлі програми ще й командою print( ) з пошуком потрібної інформації в загальній роздруківці створює додаткові незручності при файловій реалізації програми з застосуванням **for**-циклів. Але кожен, хто більше звик до роботи з файловою реалізацією комп'ютерних програм і побажає використати нашу Python-програму реалізації ФІМІ в задачі відновлення вхідних сигналів $x(t)$ ІВС, складену з використанням командних рядків, легко зможе її самостійно трансформувати у форму файлу з **for**-циклами.

Другим, що ми хочемо зауважити в нашому аналізі отриманих результатів, це те, що відновлений в процесі застосування ІТ на основі ФІМІ вхідний сигнал $x(t)$ ІВС у вигляді ряду Фур'є у формі (4) навіть з використанням лише дев'яти гармонічних складових суттєво відрізняється від вихідного сигналу $y(t)$, про що однозначно свідчать їхні графіки, показані на рис. 1. І чим більшим буде значення сталої часу первинного вимірювального перетворювача, тим більші динамічні спотворення проявлятимуться у вихідному сигналі ІВС. У цьому легко переконатись, якщо командою In [7] в програму ввести не передаточну функція $W(p)$, задану виразом (17), а задану виразом

$$W(p)=\frac{1}{0,2p+1}, \tag{19}$$

в якому стала часу є у 5 разів меншою.

Реалізуючи нашу ІТ з приведеною вище Python-програмою і використанням передаточної функції (19), отримаємо графік відновленого вхідного сигналу, показаний на рис. 3.

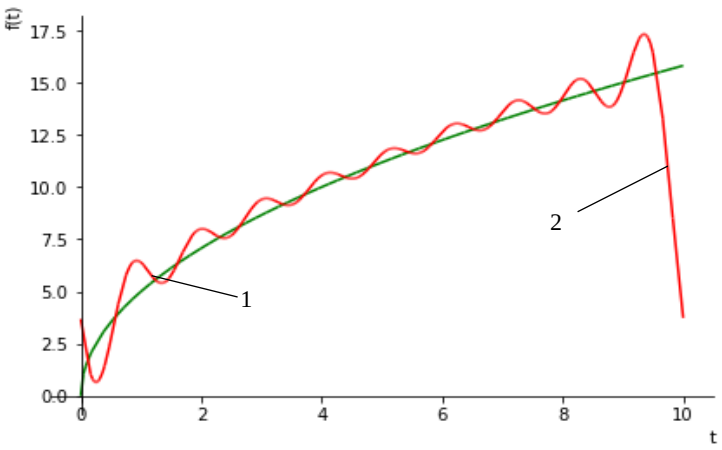

Рис. 3. Графіки сигналів: 1 — на виході ІВС (плавна крива, що починається з нуля) з передаточною функцією, заданою виразом (19); 2 — на вході ІВС (коливальна крива, складена з дев'яти гармонічних складових)

Порівнюючи графіки відновленого вхідного сигналу на рис. 1 та 3, бачимо, що графік відновленого вхідного сигналу ІВС з передаточною функцією (19) менше відхиляється за трендом від вихідного сигналу цієї ІВС у порівнянні з графіком відновленого вхідного сигналу з передаточною функцією (17).

Ну, а третє, що ми хочемо зауважити в нашому аналізі отриманих результатів, це те, що відновлений в процесі застосування ІТ на основі ФІМІ вхідний сигнал $x(t)$ ІВС у вигляді ряду Фур'є у формі (4) з використанням лише дев'яти гармонічних складових за допомогою метода найменших квадратів (МНК) нами трансформовано у ряд Фур'є з нескінченною кількістю гармонічних складових. Це обумовлено тим, що додавання кожної наступної гармонічної складової «згладжує» сумарну криву вхідного сигналу, наближаючи її до тренду, для ідентифікації якого логічно застосувати вираз (18), яким задана модель вихідного сигналу, але з невідомими коефіцієнтами A, a, тобто, задавати тренд вхідного сигналу, який ми будемо трактувати, як його еквівалентну модель, у вигляді

$$x(t)=A\sqrt{at}\ . \tag{20}$$

Ось для визначення коефіцієнтів еквівалентної моделі (20) відновленого вхідного сигналу ми і використали нелінійний варіант МНК, реалізувавши його в нашій Python-програмі командами In [144] — In [153], що дало змогу ідентифікувати вхідний сигнал $x(t)$ ІВС у вигляді еквівалентної моделі

$$x(t) = 6,09\sqrt{1,007t}\,. \tag{21}$$

Висновки

Розроблено інформаційну технологію реалізації Фур'є-інтегрального методу ідентифікації для відновлення вхідних сигналів інформаційно-вимірювальних систем за їхніми вихідними сигналами, в основу якої покладена комп'ютерна програма, створена на мові Python.

Перша частина цієї Python-програми відновлює вхідний сигнал інформаційно-вимірювальної системи за її вихідним сигналом у вигляді зрізаного ряду Фур'є, а друга частина цієї програми трансформує зрізаний ряд Фур'є у ряд Фур'є з нескінченною кількістю гармонічних складових, тобто формує еквівалентну модель вхідного ряду, очищену від перепадів, зумовлених скінченною кількістю відновлених гармонічних складових.

В процесі реалізації Python-програми продемонстровано як впливають динамічні характеристики інформаційно-вимірювальних систем на вихідні сигнали цих систем, що підтверджено і розрахунками і графічно, і є підтвердженням необхідності доповнення інформаційно-вимірювальних систем інформаційними технологіями відновлення їхніх вхідних сигналів за їхніми вихідними сигналами зі структурою, запропонованою в цій статті.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

[1] Jozef Korbicz, and Borys Mokin. *Metody matematyczne w zagadnieniach kontroli I sterowania w energetyci*; monografie. Zielona Gora: Wyzsza Szkola Inzynierska, 1990, 158 p.

[2] B. I. Mokin, O. B. Mokin, "Renewal of input signals of nonlinear measuring converters by fourier-integral method," Dubrovnik (CROATIA): IMEKO: Proceedings XVII World Gongress 3rd Millennium, 2003, pp. 210-216.

[3] Б. І. Мокін, В. Б. Мокін, і О. Б. Мокін, *Математичні методи ідентифікації динамічних систем,* навч. посіб. МОН України, ВНТУ, Вінниця, Україна: ВНТУ, 2010, 260 с.

[4] *Python*. [Електронний ресурс]. Режим доступу: https://www.python.org//downloads/ .

[5] Б. І. Мокін, В. Б. Мокін, і О. Б. Мокін, *Навчальний посібник для опанування студентами способів розв'язання задач з функціонального аналізу мовою Python. Частина 1*. Вінниця, Україна: ВНТУ, 2022, 124 с.

[6] Б. І. Мокін, В. Б. Мокін, і О. Б. Мокін, *Навчальний посібник для опанування студентами способів розв'язання задач з функціонального аналізу мовою Python. Частина 2*. Вінниця, Україна: ВНТУ, 2023. 144 с.

[7] Б. І. Мокін, В. Б. Мокін, і О. Б. Мокін, *Методи та засоби комп'ютерних обчислень,* навч. посіб. Вінниця, Україна: ВНТУ, 2024, 154 с.

Рекомендована кафедрою системного аналізу та інформаційних технологій ВНТУ

Стаття надійшла до редакції 16.04.2025

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

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

Мокін Олександр Борисович — д-р техн. наук, професор, професор кафедри системного аналізу та інформаційних технологій, e-mail: abmokin@gmail.com .

Вінницький національний технічний університет, Вінниця

O. O. Voitsekhovska[1]
B. I. Mokin[1]
O. B. Mokin[1]

Information Technology of the Fourier-Integral Identification Method Implementation for Recovery of Input Signals of Information and Measurement Systems

[1]Vinnytsia National Technical University

Information technology for implementing the Fourier integral identification method for restoring input signals of information and measuring systems from their output signals, created in the 1980s by B. I. Mokin and generalized by O. B. Mokin, has been developed. The developed information technology is based on a computer program created in the Python language. The first part of this Python program restores the input signal of the information and measuring system from its output signal in the form of a truncated Fourier series, which causes the appearance of drops on the graph of this restored input signal, due to the finite number of restored harmonic components. The second part of the Python program transforms the truncated Fourier series into a Fourier series with an infinite number of harmonic components, i.e., it forms an equivalent model of the input series, cleaned of the differences caused by the finite number of restored harmonic components. With appropriate justification, the transformation of the truncated Fourier series into a Fourier series with an infinite number of harmonic components was carried out using a nonlinear variant of the least squares method. In the process of implementing the Python program, it was demonstrated how the dynamic characteristics of information and measuring systems affect the output signals of these systems, which was confirmed both by calculations and graphically, and is a confirmation of the need to supplement information and measuring systems with information technologies for restoring their input signals from their output signals with the structure proposed in this article. The results obtained were analyzed and solutions were proposed, according to which the structure of the proposed information technology and the structure of the Python program, which is the basis of this information technology, can be brought into line with other conditions for the functioning of information and measuring system, the input signal of which is restored from its output signal.

Keywords: information technology, restoration of input signals of information and measuring systems from output signals, Fourier integral identification method, computer program, Python.

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

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

Mokin Oleksandr B. — Dr. Sc. (Eng.), Professor, Professor of the Chair of System Analysis and Information Technologies, e-mail: abmokin@gmail.com