МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное
учреждение высшего образования
«Национальный исследовательский университет «МЭИ»

Институт ИВТИ
Кафедра УИТ

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
(БАКАЛАВРСКУЮ РАБОТУ)

Направление 27.03.04 Управление в технических системах
(код и наименование)

Образовательная программа **Управление и информатика в технических системах**

Форма обучения **очная**
(очная/очно-заочная/заочная)

Тема: **Разработка программного средства для решения задачи декомпозиции математических моделей линейных разнотемповых динамических систем**

Студент	А-02-21			Ермакова П.А.
	группа		подпись	фамилия и инициалы
Руководитель ВКР		ст. преподаватель		Сидорова Е.Ю.
	уч. степень	должность	подпись	фамилия и инициалы
Консультант				
	уч. степень	должность	подпись	фамилия и инициалы
Внешний консультант				
	уч. степень	должность	подпись	фамилия и инициалы
		организация		
Заведующий кафедрой	д.т.н.	доцент		Бобряков А.В.
	уч. степень	звание	подпись	фамилия и инициалы

Место выполнения работы ФГБОУ ВО «НИУ «МЭИ»

Оглавление

Глава 1. Теоретическая справка

1.1. Введение

Когда в системе присутствуют переменные, меняющиеся с разной скоростью, их называют разнотемповыми или многотемповыми системами. Эти системы образуют отдельный класс в классификации динамических систем и широко применяются на практике. Их отличительная черта – возможность описания моделями различных порядков на разных временных интервалах наблюдения.

Декомпозиция модели предполагает замену исходной системы (1.1) на систему дифференциально-алгебраических уравнений.

Упрощенная модель определяется подсистемой дифференциальных уравнений (1.2) полученной из исходной модели путем исключения переменных и учета их алгебраических связей. Решение системы (1.2) может отличаться от решения исходной модели (1.1), что означает, что модель (1) завышенного порядка может быть неадекватной. Поэтому при декомпозиции указывают допустимую погрешность решения системы (1.2) по сравнению с исходной моделью (1.1), чтобы определить временной интервал применимости упрощенной модели (1.2).

$$\vec{\dot{x}} = \vec{\phi}(\vec{x}, t) \tag{1.1}$$

$$\begin{cases} \vec{\tilde{x}}' = \vec{\tilde{\phi}}(\vec{\tilde{x}}) \\ \vec{\Phi}(\vec{x}, t) = 0 \end{cases} \tag{1.2}$$

Временной интервал справедливости упрощенной модели (1.2) определяется как:

$$\Gamma_v = \frac{-1}{\alpha_v} \ln(\delta^0)$$

Где δ^0 - допустимое значение погрешности, α_v - собственное значение системы (1.1).

1.2. Пример декомпозиции системы 4 порядка.

Имеется система дифференциальных уравнений:

$$\vec{\dot{x}} = A\vec{x} \tag{1}$$

Также имеются собственные числа:

$$\vec{\lambda}=(-10,-5,-1,-0.5)$$

Найдем решение в виде:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}=\begin{pmatrix} C_1 & 0 & 0 & 0 \\ 0 & C_2 & 0 & 0 \\ 0 & 0 & C_3 & 0 \\ 0 & 0 & 0 & C_4 \end{pmatrix}(\vec{\alpha}_1\vec{\alpha}_2\vec{\alpha}_3\vec{\alpha}_4)*\begin{pmatrix} e^{\lambda_1} \\ e^{\lambda_2} \\ e^{\lambda_3} \\ e^{\lambda_4} \end{pmatrix}=C_1*\overrightarrow{\alpha^1}*e^{\lambda_1}+C_2*\overrightarrow{\alpha^2}*e^{\lambda_2}+C_3*\overrightarrow{\alpha^3}*e^{\lambda_3}+C_4*\overrightarrow{\alpha^4}*e^{\lambda_4}(*)$$

Для получения матрицы A воспользуемся формулой:

$$A=T^{-1}*B*T \qquad \text{(v)}$$

Получим случайную матрицу T:

$$T=\begin{pmatrix} -2 & -10 & -5 & 14 \\ 12 & -6 & 10 & -2 \\ -6 & 11 & 3 & -11 \\ 14 & 2 & -7 & -11 \end{pmatrix}$$

Проверим матрицу на вырожденность:

$$\det T \neq 0$$
$$\det T=-3968$$

⇒ Матрица невырожденнная, значит дальнейшие расчеты имеют место быть.

Запишем диагональную матрицу B, состоящую из собственных чисел:

$$B=\begin{pmatrix} -10.0000 & 0 & 0 & 0 \\ 0 & -5.0000 & 0 & 0 \\ 0 & 0 & -1.0000 & 0 \\ 0 & 0 & 0 & -0.5000 \end{pmatrix}$$

Составим матрицу A по правилу (v):

$$A=\begin{pmatrix} -3.0504 & -44.6371 & -14.7046 & 56.9748 \\ -1.4677 & -89.0323 & -25.8790 & 110.2661 \\ -3.4798 & -10.1452 & -6.7681 & 15.2601 \\ -1.2984 & -66.4516 & -19.4315 & 82.3508 \end{pmatrix}$$

Система (1) имеет вид:

$$\dot{\vec{x}} = \begin{pmatrix} -3.0504 & -44.6371 & -14.7046 & 56.9748 \\ -1.4677 & -89.0323 & -25.8790 & 110.2661 \\ -3.4798 & -10.1452 & -6.7681 & 15.2601 \\ -1.2984 & -66.4516 & -19.4315 & 82.3508 \end{pmatrix} \vec{x}$$

$$\begin{aligned} \dot{x}_1 &= -3.0504\,x_1 - 44.6371\,x_2 - 14.7046\,x_3 + 56.9748\,x_4 \\ \dot{x}_2 &= -1.4677\,x_1 - 89.0323\,x_2 - 25.8790\,x_3 + 110.2661\,x_4 \\ \dot{x}_3 &= -3.4798\,x_1 - 10.1452\,x_2 - 6.7681\,x_3 + 15.2601\,x_4 \\ \dot{x}_4 &= -1.2984\,x_1 - 66.4516\,x_2 - 19.4315\,x_3 + 82.3508\,x_4 \end{aligned} \qquad (2)$$

Рассчитаем собственные вектора и числа:

$$\alpha = \begin{pmatrix} 0.379009 & 0.318930 & -0.393187 & 0.263195 \\ 0.734644 & 0.755812 & -0.705996 & 0.701854 \\ 0.096839 & -0.042371 & -0.068777 & 0.229452 \\ 0.554322 & 0.570295 & -0.585016 & 0.620871 \end{pmatrix}$$

$$\lambda = \begin{pmatrix} -10.0000 & 0 & 0 & 0 \\ 0 & -5.0000 & 0 & 0 \\ 0 & 0 & -1.0000 & 0 \\ 0 & 0 & 0 & -0.5000 \end{pmatrix}$$

Проверим, что собственные вектора и числа рассчитаны правильно:

$$A\vec{\alpha}_i = \lambda_i \vec{\alpha}_i,$$

где $\vec{\alpha}_i$ - вектор-столбец матрицы α

Для $A\vec{\alpha}_1 = \lambda_1 \vec{\alpha}_1$:

$$\begin{pmatrix} -3.7901 \\ -7.3464 \\ -0.9684 \\ -5.5432 \end{pmatrix} = \begin{pmatrix} -3.7901 \\ -7.3464 \\ -0.9684 \\ -5.5432 \end{pmatrix}$$

Для $A\vec{\alpha}_2 = \lambda_2 \vec{\alpha}_2$:

$$\begin{pmatrix} -1.5946 \\ -3.7791 \\ 0.2119 \\ -2.8515 \end{pmatrix} = \begin{pmatrix} -1.5946 \\ -3.7791 \\ 0.2119 \\ -2.8515 \end{pmatrix}$$

Для $A\vec{\alpha}_3 = \lambda_3 \vec{\alpha}_3$:

$$\begin{pmatrix}0.393187\\0.705996\\0.068777\\0.585016\end{pmatrix}=\begin{pmatrix}0.393187\\0.705996\\0.068777\\0.585016\end{pmatrix}$$

Для $A\vec{\alpha}_4=\lambda_4\vec{\alpha}_4$:

$$\begin{pmatrix}-0.1316\\-0.3509\\-0.1147\\-0.3104\end{pmatrix}=\begin{pmatrix}-0.1316\\-0.3509\\-0.1147\\-0.3104\end{pmatrix}$$

Зададим вектор начальных условий:

$$\vec{x}_0^T=(2.0000-1.5000\,1.0000-2.5000)$$

Найдем коэффициенты C_i. Из уравнения (*) можно сказать, что:

$$C=\alpha^{-1}\vec{x}_0$$

Тогда:

$$C=\begin{pmatrix}35.0181\\-21.1266\\2.4331\\-13.5930\end{pmatrix}$$

Запишем уравнения (*):

$$\begin{pmatrix}x_1\\x_2\\x_3\\x_4\end{pmatrix}=\begin{pmatrix}C_1&0&0&0\\0&C_2&0&0\\0&0&C_3&0\\0&0&0&C_4\end{pmatrix}(\vec{\alpha}_1\vec{\alpha}_2\vec{\alpha}_3\vec{\alpha}_4)*\begin{pmatrix}e^{\lambda_1}\\e^{\lambda_2}\\e^{\lambda_3}\\e^{\lambda_4}\end{pmatrix}=\begin{pmatrix}13.2722&-6.7379&-0.9567&-3.5776\\25.7258&-15.9677&-1.7177&-9.5403\\3.3911&0.8952&-0.1673&-3.1190\\19.4113&-12.0484&-1.4234&-8.4395\end{pmatrix}\begin{pmatrix}e^{\lambda_1}\\e^{\lambda_2}\\e^{\lambda_3}\\e^{\lambda_4}\end{pmatrix}=$$

$$=\begin{pmatrix}13.2722e^{\lambda_1 t}-6.7379e^{\lambda_2 t}-0.9567e^{\lambda_3 t}-3.5776e^{\lambda_4 t}\\25.7258e^{\lambda_1 t}-15.9677e^{\lambda_2 t}-1.7177e^{\lambda_3 t}-9.5403e^{\lambda_4 t}\\3.3911e^{\lambda_1 t}+0.8952e^{\lambda_2 t}-0.1673e^{\lambda_3 t}-3.1190e^{\lambda_4 t}\\19.4113e^{\lambda_1 t}-12.0484e^{\lambda_2 t}-1.4234e^{\lambda_3 t}-8.4395e^{\lambda_4 t}\end{pmatrix}$$

Пусть имеется матрица H:

$$H=\left(\vec{\alpha}_1\vec{\alpha}_2\vec{\alpha}_3\vec{\alpha}_4\right)\begin{pmatrix} C_1 & 0 & 0 & 0 \\ 0 & C_2 & 0 & 0 \\ 0 & 0 & C_3 & 0 \\ 0 & 0 & 0 & C_4 \end{pmatrix}=\begin{pmatrix} 13.2722 & -6.7379 & -0.9567 & -3.5776 \\ 25.7258 & -15.9677 & -1.7177 & -9.5403 \\ 3.3911 & 0.8952 & -0.1673 & -3.1190 \\ 19.4113 & -12.0484 & -1.4234 & -8.4395 \end{pmatrix}$$

Тогда :

$$\begin{pmatrix} e^{\lambda_1 t} \\ e^{\lambda_2 t} \\ e^{\lambda_3 t} \\ e^{\lambda_4 t} \end{pmatrix}=H^{-1}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \quad (**)$$

Найдем H^{-1}:

$$H^{-1}=\begin{pmatrix} 0.068966 & 0.344828 & 0.172414 & -0.482759 \\ 0.250000 & -0.125000 & 0.208333 & -0.041667 \\ -3.000000 & 5.500000 & 1.500000 & -5.500000 \\ 0.307692 & 0.043956 & -0.153846 & -0.241758 \end{pmatrix}$$

Запишем уравнение (**):

$$\begin{pmatrix} e^{-10t} \\ e^{-5t} \\ e^{-t} \\ e^{-0.5t} \end{pmatrix}=\begin{pmatrix} 0.068966 & 0.344828 & 0.172414 & -0.482759 \\ 0.250000 & -0.125000 & 0.208333 & -0.041667 \\ -3.000000 & 5.500000 & 1.500000 & -5.500000 \\ 0.307692 & 0.043956 & -0.153846 & -0.241758 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}=$$

$$\begin{pmatrix} 0.068966\,x_1+0.344828\,x_2+0.172414\,x_3-0.482759\,x_4 \\ 0.250000\,x_1-0.125000\,x_2+0.208333\,x_3-0.041667\,x_4 \\ -3.000000\,x_1+5.500000\,x_2+1.500000\,x_3-5.500000\,x_4 \\ 0.307692\,x_1+0.043956\,x_2-0.153846\,x_3-0.241758\,x_4 \end{pmatrix} \qquad (***)$$

Пусть:

$$\begin{pmatrix} e^{-10t} \\ e^{-5t} \\ e^{-t} \\ e^{-0.5t} \end{pmatrix}=\begin{pmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{pmatrix}$$

Тогда уравнение (***) примет вид :

$$\begin{pmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{pmatrix}=\begin{pmatrix} 0.068966\,x_1 & 0.344828\,x_2 & 0.172414\,x_3 & -0.482759\,x_4 \\ 0.250000\,x_1 & -0.125000\,x_2 & 0.208333\,x_3 & -0.041667\,x_4 \\ -3.000000\,x_1 & 5.500000\,x_2 & 1.500000\,x_3 & -5.500000\,x_4 \\ 0.307692\,x_1 & 0.043956\,x_2 & -0.153846\,x_3 & -0.241758\,x_4 \end{pmatrix}$$

Запишем это уравнение в следующем виде:

$$
\begin{aligned}
e_1 &= 0.068966\,x_1 + 0.344828\,x_2 + 0.172414\,x_3 - 0.482759\,x_4 \\
e_2 &= 0.250000\,x_1 - 0.125000\,x_2 + 0.208333\,x_3 - 0.041667\,x_4 \\
e_3 &= -3.000000\,x_1 + 5.500000\,x_2 + 1.500000\,x_3 - 5.500000\,x_4 \\
e_4 &= 0.307692\,x_1 + 0.043956\,x_2 - 0.153846\,x_3 - 0.241758\,x_4
\end{aligned} \qquad (3)
$$

Введем голономную связь $e_1 = 0$, тогда система (3) примет вид:

$$
\begin{aligned}
0 &= 0.068966\,x_1 + 0.344828\,x_2 + 0.172414\,x_3 - 0.482759\,x_4 \\
e_2 &= 0.250000\,x_1 - 0.125000\,x_2 + 0.208333\,x_3 - 0.041667\,x_4 \\
e_3 &= -3.000000\,x_1 + 5.500000\,x_2 + 1.500000\,x_3 - 5.500000\,x_4 \\
e_4 &= 0.307692\,x_1 + 0.043956\,x_2 - 0.153846\,x_3 - 0.241758\,x_4
\end{aligned} \qquad (4)
$$

Выразим x_1 из (4):

$$x_1 = -\frac{0.344828}{0.068966}x_2 - \frac{0.172414}{0.068966}x_3 + \frac{0.482759}{0.068966}x_4$$

$$-4.999971000203 \cdot x_2 - 2.4999855001015 \cdot x_3 + 6.9999565003045 \cdot x_4$$

После подстановки в исходную систему (2), получаем:

$$
\begin{aligned}
\dot{x}_2 &= -1.4677 * (-4.999971000203 \cdot x_2 - 2.4999855001015 \cdot x_3 + 6.9999565003045 \cdot x_4) - \\
&- 89.0323\,x_2 - 25.8790\,x_3 + 110.2661\,x_4 = \\
&= -81.6938425630021 \cdot x_2 - 22.209771281501 \cdot x_3 + 99.9922638445031 \cdot x_4
\end{aligned}
$$

Для остальных $\dot{x}_i$ аналогичным образом подставляем x_1. В результате получаем следующую систему пониженного порядка:

$$
\begin{aligned}
\dot{x}_2 &= -81.6938425630021 \cdot x_2 - 22.209771281501 \cdot x_3 + 99.9922638445031 \cdot x_4 \\
\dot{x}_3 &= 7.2536990865064 \cdot x_2 + 1.9313495432532 \cdot x_3 - 9.09834862975959 \cdot x_4 \\
\dot{x}_4 &= -59.9596376533364 \cdot x_2 - 16.1855188266682 \cdot x_3 + 73.2620564800046 \cdot x_4
\end{aligned} \qquad (5)
$$

Рассчитаем Γ_1 и Γ_4. Пусть $\delta^0 = 0.05$, тогда:

$$\Gamma_1 = \frac{1}{-10} * \ln(0.1) = 0.299573$$

$$\Gamma_4 = \frac{1}{-0.5} * \ln(0.05) = 5.99146$$

Для упрощенной системы общий интервал справедливости $t \in [\Gamma_1; \Gamma_n]$

1.3. Моделирование в SimInTech

Проведем моделирование систем по схеме, изображенной на Рисунок 1. Рассмотрим графики, полученные после моделирования. На Рисунок 2 изображен график исходной системы и системы с пониженным порядком. Если рассмотреть график представленный на Рисунок 3, то можно увидеть что между вырожденнными переменными и переменными исходной модели максимальная ошибка около 5%.

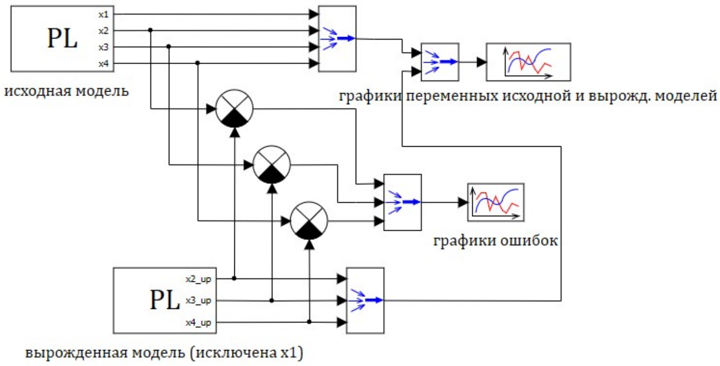

Рисунок 1.Схема моделирования SimInTech

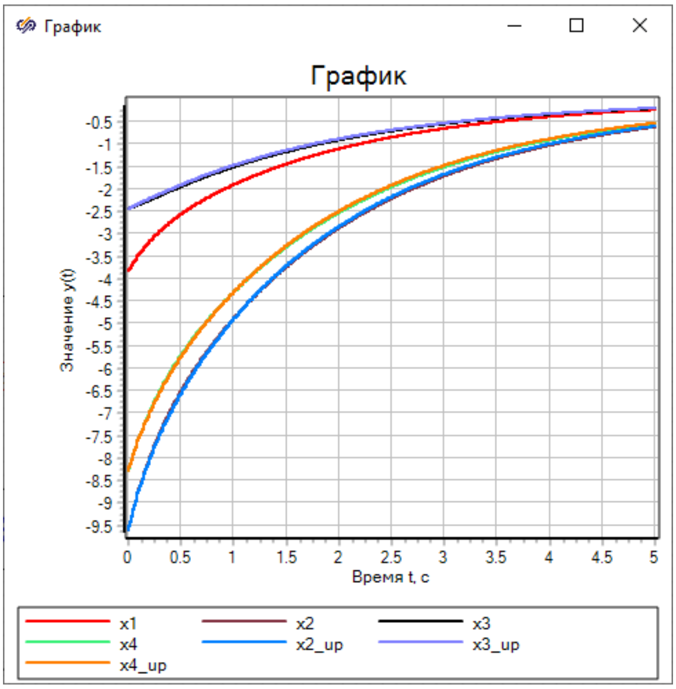

Рисунок 2.График исходной системы и системы с пониженным порядком

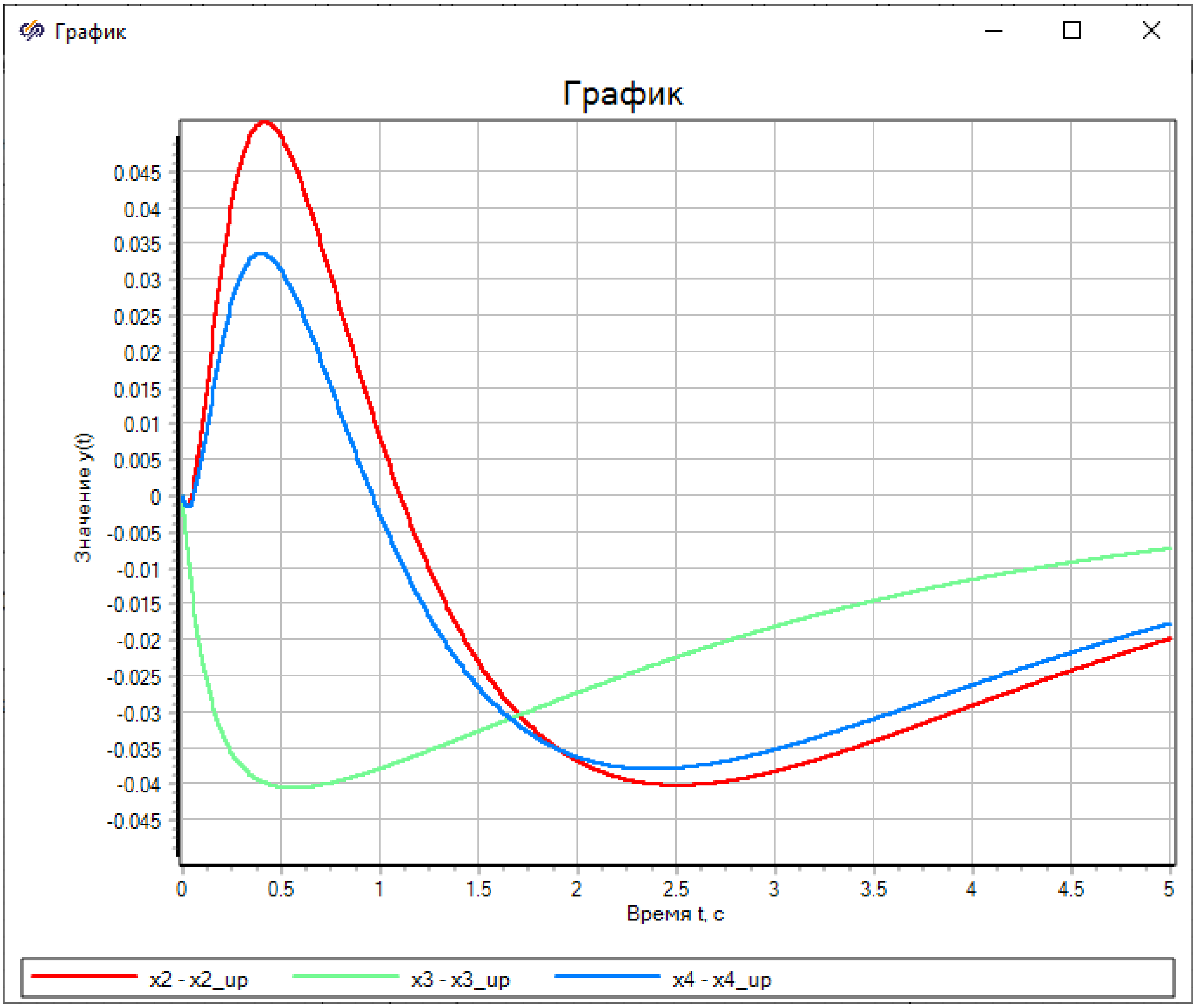

Рисунок 3.График ошибок

Глава 2. ПО для исследования декомпозиции многотемповых линейных систем.

2.1. Введение

Разработка программного обеспечения для исследования декомпозиции многотемповых линейных систем будет проводится на базе языка программирования Python 3. Выбор Python 3 для разработки программного обеспечения имеет множество достоинств. Рассмотрим несколько из них:

1. **Читаемость кода**: Python 3 имеет простую и читаемую синтаксическую структуру, что делает код легким для понимания и поддержки.

2. **Стандартные библиотеки**: Python 3 поставляется с обширным набором стандартных библиотек, которые охватывают множество задач, таких как работа с файлами, сетевое программирование, обработка текстов и многого другого.

3. **Обширное сообщество**: Python имеет огромное сообщество разработчиков, которые активно делятся своими наработками и помогают друг другу решать проблемы.

4. **Кроссплатформенность**: Программы, написанные на Python 3, могут работать на различных операционных системах без необходимости вносить изменения в код.

5. **Активное развитие**: Python 3 продолжает активно развиваться и совершенствоваться, получая новые функции и улучшения производительности. Это гарантирует, что язык остается актуальным и конкурентоспособным.

Таким образом, Python 3 сочетает в себе удобство использования, широкие возможности, поддержку сообщества и постоянное развитие, что делает его идеальным выбором для разработки программного обеспечения, включая исследования в области декомпозиции многотемповых линейных систем.

2.2. Программное обеспечение

Рассмотрим интерфейс полученной программы. На Рисунок 4, Рисунок 5 представлены окна для ввода размерности матрицы, самой матрицы, а так же для ввода начальных условий и ошибки.

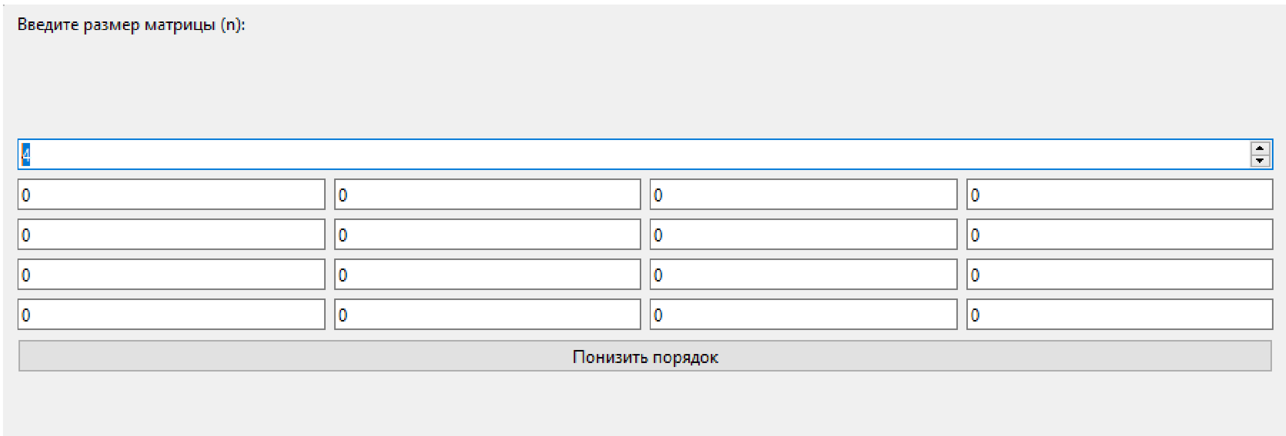

Рисунок 4.Ввод матрицы

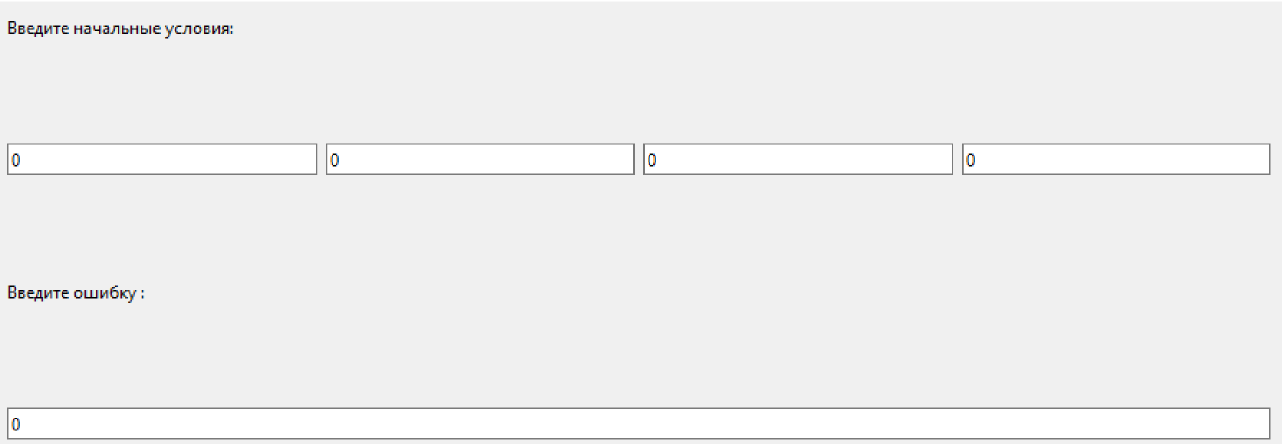

Рисунок 5.Ввод начальных условий и ошибки

Проверим работоспособность программы, введем матрицу системы (2), полученной раннее и сравним полученные результаты. Также введем начальные условия и ошибку, которую использовали ранее (см. Рисунок 6, Рисунок 7). После нажатия на кнопку «Понизить порядок» выводятся матрица пониженной модели, а также границы справедливости модели. Можно увидеть, что границы справедливости совпадают с ранее рассчитанными. Также справа от ввода данных появляются графики переходных процессов и ошибки между системами. Они представлены на Рисунок 9, Рисунок 10. Если сравнить их с Рисунок 2, Рисунок 3, то можно сказать, что программа работает правильно. На Рисунок 11 представлен полный вид программы.

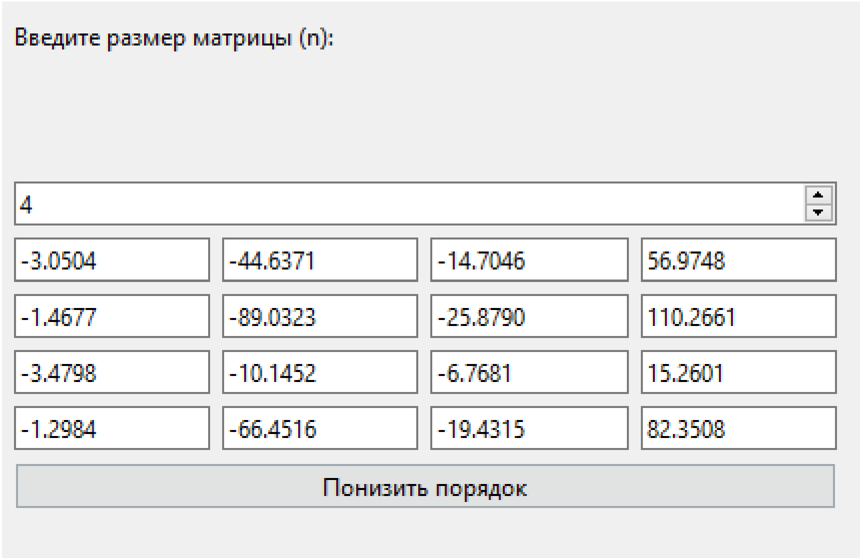

Рисунок 6.Ввод матрицы системы

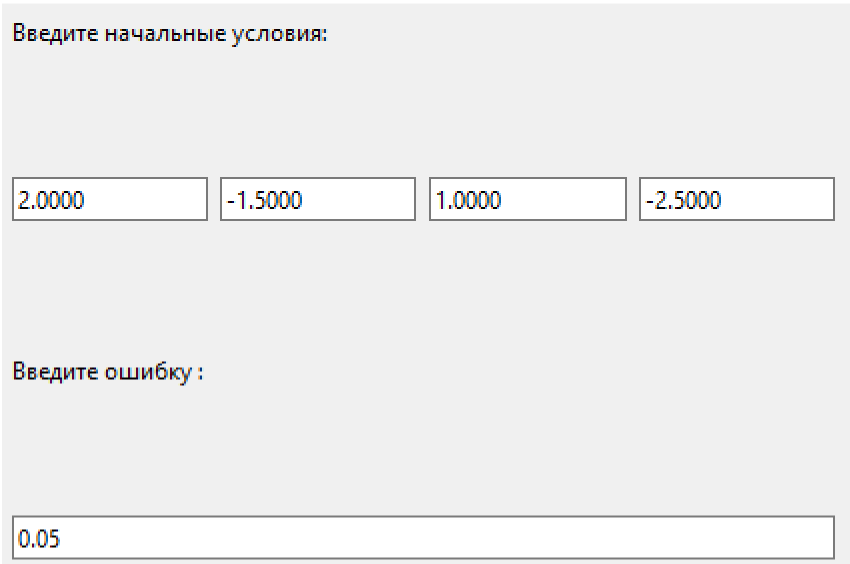

Рисунок 7.Ввод начальных условий и ошибки

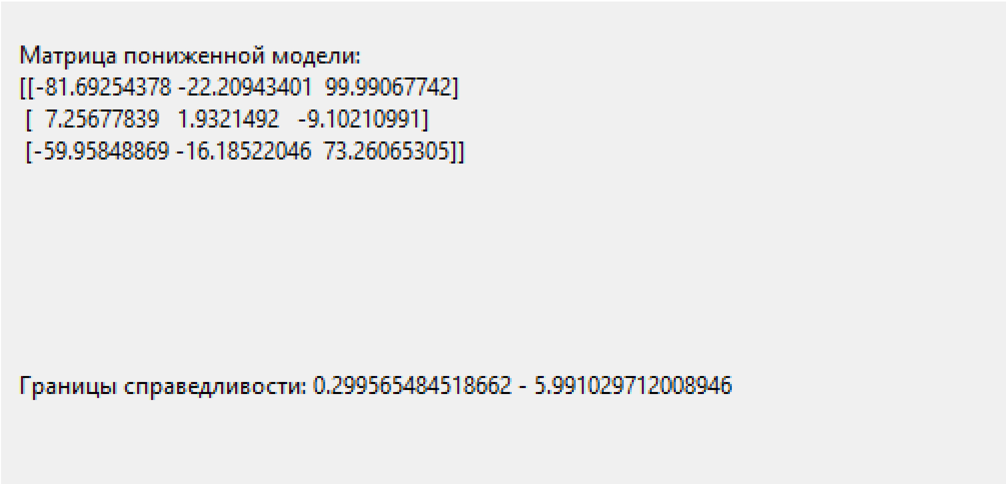

Рисунок 8.матрица пониженной модели и границы справедливости.

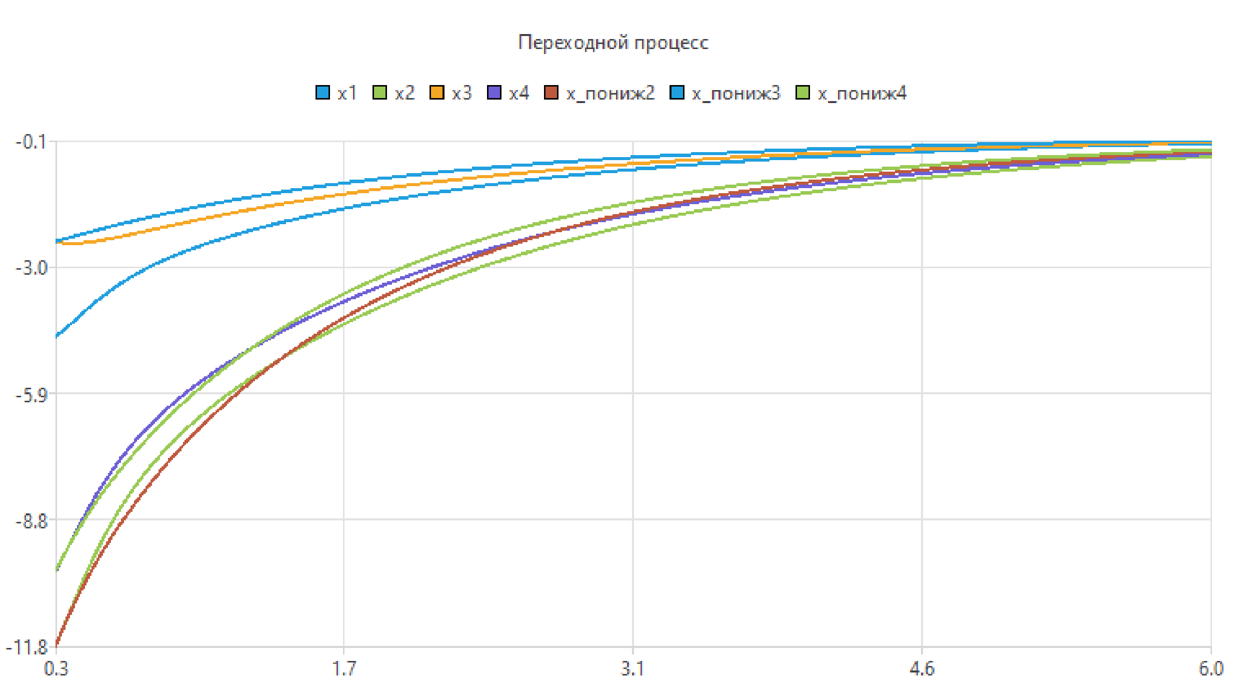

Рисунок 9.Выведенный график

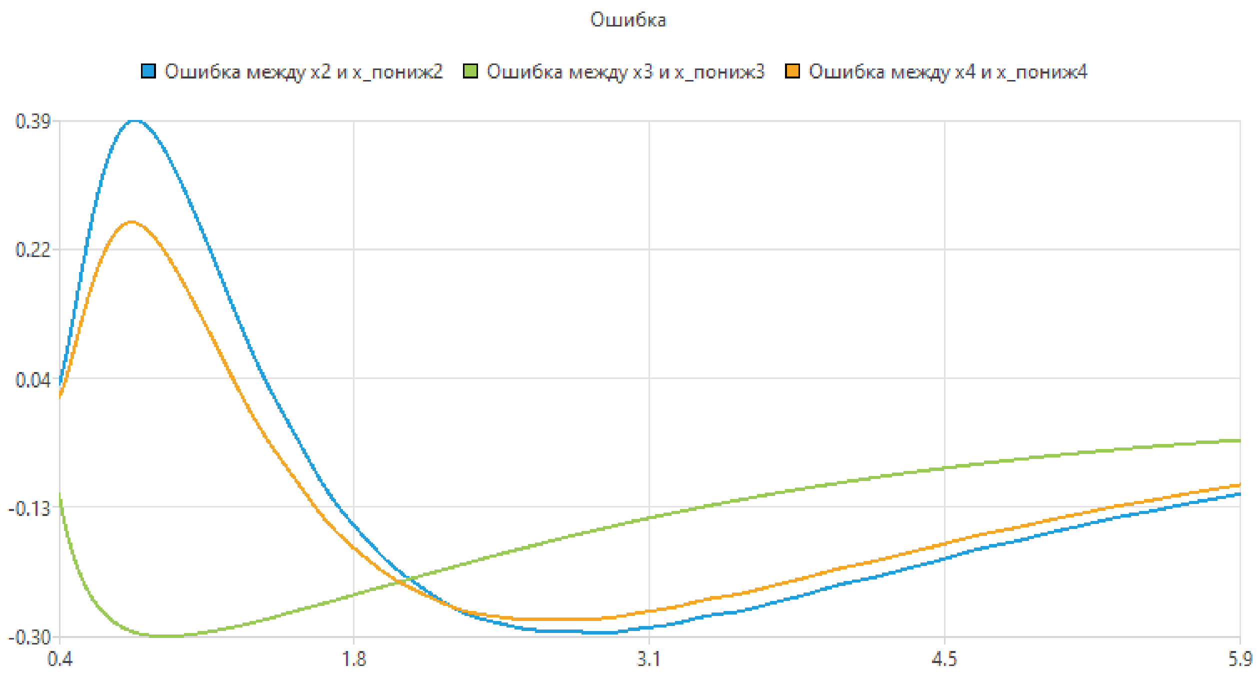

Рисунок 10.Выведенный график ошибки

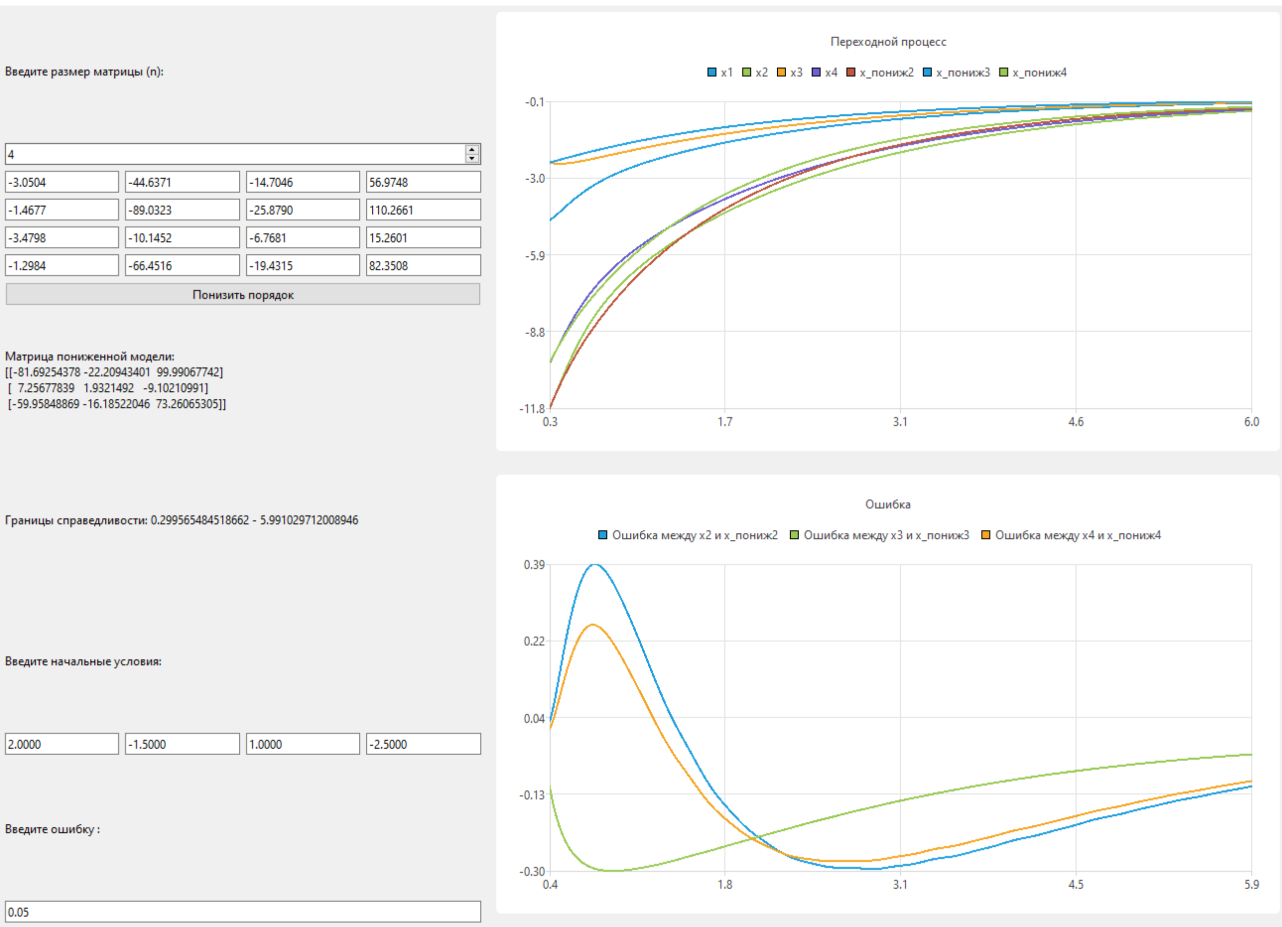

Рисунок 11.Полный вид программы

2.3. Листинг программы

```python
import sys
import numpy as np
from PyQt6.QtWidgets import QApplication, QWidget, QVBoxLayout, QGridLayout,
QLabel, QLineEdit, QPushButton, QMainWindow, QSpinBox, QHBoxLayout
from PyQt6.QtGui import QPalette, QColor
from PyQt6.QtCharts import QChart, QChartView, QLineSeries
import sympy as sp
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp
import math as m

class Matrix:
  def __init__(self, n, values, initial_conditions, err):
    self.size = n
    self.err = err
    self.values = values
    self.determinant = None
    self.eigenvalues = np.zeros(n)
    self.eigenvectors = np.zeros((n, n))
    self.initial_conditions = initial_conditions
    self.coefficients = np.zeros(n)
    self.inverse_eig = None
    self.inverse = None
    self.index_max_eigval = None
    self.matr_with_x = []
    self.matr_resh = np.zeros((n-1, n-1))
    self.gran = []

  def calc_gran(self):
    for i in range(self.size):
      gran_value = (1 / self.eigenvalues[i]) * m.log(self.err)
      print(gran_value)
      self.gran.append(gran_value)

  def is_determinant_negative(self):
    self.determinant = np.linalg.det(self.values)
    return self.determinant < 0

  def calculate_eigenvalues_and_eigenvectors(self):
    self.eigenvalues, self.eigenvectors = np.linalg.eig(self.values)

  def calculate_inverse(self):
    self.inverse_eig = np.linalg.inv(self.eigenvectors)

  def calculate_coefficients(self):
```

```
    self.calculate_inverse()
    if self.inverse_eig is not None:
      self.coefficients = self.inverse_eig @ self.initial_conditions.T

  def calculate_inverse_from_eigenvectors(self):
    diagonal_matrix = np.diag(self.coefficients.flatten())
    self.inverse = np.linalg.inv(self.eigenvectors @ diagonal_matrix)

  def find_index_of_max_eigenvector(self):
    self.index_max_eigval = np.argmax(abs(self.eigenvalues))

  def expr_max_exp(self):
    self.find_index_of_max_eigenvector()
    x_resh = sp.symbols(f'x{self.index_max_eigval + 1}')
    xi = sp.solve(sum(self.inverse[self.index_max_eigval][i] * sp.symbols(f'x{i +
1}') for i in range(self.size)), x_resh)[0]
    for i in range(self.size):
      temp_expr = sum(self.values[i][j] * sp.symbols(f'x{j + 1}') for j in
range(self.size))
      self.matr_with_x.append(temp_expr)
    substitutedequations = [eq.subs(x_resh, xi) for eq in self.matr_with_x]
    simplifiedequations = [sp.simplify(eq) for eq in substitutedequations]
    print("\nПолученные выражения:")
    for i, eq in enumerate(simplifiedequations[1:]):
      coefficients = [float(eq.coeff(sp.symbols(f'x{j+1}'))) for j in range(1,
self.size)]
      if i < self.matr_resh.shape[0]:
        self.matr_resh[i, :len(coefficients)] = coefficients

  def construct_equations_matrix(self):
    equations_matrix = []
    for i in range(self.size):
      equation = sum(self.coefficients[j] * self.eigenvectors[j, i] *
sp.exp(self.eigenvalues[j]) for j in range(self.size))
      equations_matrix.append(equation)
    return equations_matrix

  def calc(self):
    self.calculate_eigenvalues_and_eigenvectors()
    self.calculate_coefficients()
    self.calculate_inverse_from_eigenvectors()
    self.expr_max_exp()
    self.calc_gran()
    equations_matrix = self.construct_equations_matrix()

def system_of_equations(t, y, values):
  x = y
```

```
  dx_dt = values @ x
  return dx_dt

def plot_transient_response2(temp_sol, temp_sol2, size_matr_1, size_matr_2,
chart_view):
  chart = QChart()
  for i in range(size_matr_1):
    series = QLineSeries()
    for j in range(len(temp_sol.t)):
      series.append(temp_sol.t[j], temp_sol.y[i][j])
    series.setName(f'x{i+1}')
    chart.addSeries(series)
  for i in range(size_matr_2):
    series = QLineSeries()
    for j in range(len(temp_sol2.t)):
      series.append(temp_sol2.t[j], temp_sol2.y[i][j])
    series.setName(f'x_пониж{i+2}')
    chart.addSeries(series)
  chart.createDefaultAxes()
  chart.setTitle('Переходной процесс')
  chart_view.setChart(chart)
  chart_view.setRubberBand(QChartView.RubberBand.RectangleRubberBand)

def moving_average(data, window_size):
  return np.convolve(data, np.ones(window_size)/window_size, mode='valid')

def plot_error_response(solution1_new, solution2_new, size_matr_2, chart_view,
window_size=10):
  errors = solution1_new.y[1:size_matr_2+1, :] - solution2_new.y
  chart = QChart()
  for i in range(size_matr_2):
    smoothed_errors = moving_average(errors[i], window_size)
    smoothed_t = moving_average(solution1_new.t, window_size)
    series = QLineSeries()
    for j in range(len(smoothed_t)):
      series.append(smoothed_t[j], smoothed_errors[j])
    series.setName(f'Ошибка между x{i+2} и x_пониж{i+2} ')
    chart.addSeries(series)
  chart.createDefaultAxes()
  chart.setTitle('Ошибка')
  chart_view.setChart(chart)
  chart_view.setRubberBand(QChartView.RubberBand.RectangleRubberBand)
# Приложение для матриц

class MatrixApp(QMainWindow):
  def __init__(self):
    super().__init__()
```

```
    self.setWindowTitle("Matrix Exponent App")
    self.setGeometry(100, 100, 1200, 600)
    self.flag = True
    self.flag2 = True
    self.central_widget = QWidget()
    self.setCentralWidget(self.central_widget)

    self.layout = QHBoxLayout()

    self.input_layout = QVBoxLayout()
    self.grid_layout = QGridLayout()

    self.size_input = QSpinBox()
    self.size_input.setRange(2, 10)
    self.size_input.setValue(3)

    self.input_layout.addWidget(QLabel("Введите размер матрицы (n):"))
    self.input_layout.addWidget(self.size_input)
    self.entries = []
    self.initial_conditions_entries = []
    self.error_entries = []
    self.values_matrix = np.zeros((int(self.size_input.text()),
int(self.size_input.text())))
    self.values_matrix_initial_cond = np.zeros((1, int(self.size_input.text())))
    self.error_value = None

    self.size_input.valueChanged.connect(self.create_matrix_inputs)
    self.plot_button = QPushButton("Понизить порядок")
    self.chart_view1 = QChartView()
    self.chart_view2 = QChartView()

    self.chart_layout = QVBoxLayout()
    self.plot_button.clicked.connect(self.plot_exponents)

    self.input_layout.addLayout(self.grid_layout)
    self.input_layout.addWidget(self.plot_button)

    self.chart_layout.addWidget(self.chart_view1)
    self.chart_layout.addWidget(self.chart_view2)

    self.layout.addLayout(self.input_layout)
    self.layout.addLayout(self.chart_layout)

    # Добавление QLabel для пониженной матрицы и границ
    self.reduced_matrix_label = QLabel()
    self.bounds_label = QLabel()
    self.input_layout.addWidget(self.reduced_matrix_label)
```

```
        self.input_layout.addWidget(self.bounds_label)

        self.central_widget.setLayout(self.layout)

    def create_matrix_inputs(self):
        for i in reversed(range(len(self.entries))):
            for j in range(len(self.entries[i])):
                self.grid_layout.itemAtPosition(i, j).widget().deleteLater()
            self.entries.pop()

        for i in range(self.size_input.value()):
            row_entries = []
            for j in range(self.size_input.value()):
                entry = QLineEdit()
                entry.setPlaceholderText(f"Элемент ({i+1},{j+1})")
                entry.setText("0")
                row_entries.append(entry)
                self.grid_layout.addWidget(entry, i, j)
            self.entries.append(row_entries)

        self.create_initial_conditions_inputs()
        self.create_error_input()

    def create_initial_conditions_inputs(self):
        if (self.flag):
            self.initial_conditions_label = QLabel("Введите начальные условия:")
            self.input_layout.addWidget(self.initial_conditions_label)
            self.flag = False
        else:
            self.initial_conditions_label.deleteLater()
            self.initial_conditions_label = QLabel("Введите начальные условия:")
            self.input_layout.addWidget(self.initial_conditions_label)
        for entry in self.initial_conditions_entries:
            entry.deleteLater()
        self.initial_conditions_entries.clear()

        n = self.size_input.value()
        self.initial_conditions_grid = QGridLayout()
        for j in range(n):
            entry = QLineEdit()
            entry.setPlaceholderText(f"Начальное условие {j+1}")
            entry.setText("0")
            self.initial_conditions_entries.append(entry)
            self.initial_conditions_grid.addWidget(entry, 0, j)
        self.input_layout.addLayout(self.initial_conditions_grid)

    def read_matrix_data(self):
```

```
    n = self.size_input.value()
    self.values_matrix = np.zeros((n, n))
    self.values_matrix_initial_cond = np.array([np.zeros(n)])
    self.error_value = 0.0

    for i in range(n):
      for j in range(n):
        try:
          self.values_matrix[i][j] = float(self.entries[i][j].text())
        except ValueError:
          self.values_matrix[i][j] = 0.0

    initial_conditions = []
    for j in range(n):
      try:
        initial_conditions.append(float(self.initial_conditions_entries[j].text()))
      except ValueError:
        initial_conditions.append(0.0)
    self.values_matrix_initial_cond = np.array([initial_conditions])

    try:
      self.error_value = float(self.error_entry.text())
    except ValueError:
      self.error_value = 0.0

  def create_error_input(self):
    if (self.flag2):
      self.error_label = QLabel("Введите ошибку :")
      self.input_layout.addWidget(self.error_label)
      self.flag2 = False
    else:
      self.error_label.deleteLater()
      self.error_label = QLabel("Введите ошибку :")
      self.input_layout.addWidget(self.error_label)
    if self.error_entries:
      self.error_entries[0].deleteLater()
    self.error_entries.clear()
    self.error_entry = QLineEdit()
    self.error_entry.setPlaceholderText("Значение ошибки")
    self.error_entry.setText("0")
    self.error_entries.append(self.error_entry)
    self.input_layout.addWidget(self.error_entry)

  def plot_exponents(self):
    self.read_matrix_data()
    matrix_1 = Matrix(int(self.size_input.text()), self.values_matrix,
self.values_matrix_initial_cond, self.error_value)
```

```
    matrix_1.calc()
    new_values = matrix_1.matr_resh
    matrix_2 = Matrix(int(self.size_input.text())-1, new_values,
self.values_matrix_initial_cond[:, 1:], self.error_value)
    matrix_2.calc()
    print(matrix_1.values)
    t_span = (0, 10)
    t_eval = np.linspace(t_span[0], t_span[1], 400)
    solution1 = solve_ivp(system_of_equations, t_span,
matrix_1.initial_conditions.flatten(), t_eval=t_eval, args=(matrix_1.values,))
    solution2 = solve_ivp(system_of_equations, t_span,
matrix_2.initial_conditions.flatten(), t_eval=t_eval, args=(matrix_2.values,))

    initial_conditions_new = None
    print("Solution 1:")
    min_gran = min(matrix_1.gran)
    print(min_gran)
    for i, t in enumerate(solution1.t):
      if abs(t - min_gran) <= 0.02:
        initial_conditions_new = np.array([solution1.y[:, i]])
        print(f"t = {t:.2f}, y = {solution1.y[:, i]}")
        break

    a_new = Matrix(int(self.size_input.text()), self.values_matrix,
initial_conditions_new, self.error_value)
    a_new.calc()

    new_values1 = a_new.matr_resh
    b_new = Matrix(int(self.size_input.text())-1, new_values1,
initial_conditions_new[:, 1:], self.error_value)
    b_new.calc()
    t_span = (min(matrix_1.gran), max(matrix_1.gran))
    t_eval = np.linspace(t_span[0], t_span[1], 400)
    solution1_new = solve_ivp(system_of_equations, t_span,
a_new.initial_conditions.flatten(), t_eval=t_eval, args=(a_new.values,))
    solution2_new = solve_ivp(system_of_equations, t_span,
b_new.initial_conditions.flatten(), t_eval=t_eval, args=(b_new.values,))

    plot_transient_response2(solution1_new, solution2_new,
int(self.size_input.text()), int(self.size_input.text())-1, self.chart_view1)
    plot_error_response(solution1_new, solution2_new,
int(self.size_input.text())-1, self.chart_view2)

    self.reduced_matrix_label.setText(f"Матрица пониженной модели:\n{new_values}")
    self.bounds_label.setText(f"Границы справедливости: {min_gran} -
{max(matrix_1.gran)}")
```

```python
if __name__ == "__main__":
  app = QApplication(sys.argv)
  window = MatrixApp()
  window.show()
  sys.exit(app.exec())
```