ДОДАТОК А
Програмний код

Приклад програмної реалізації виводу графічного зображення:

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Windows.Forms;
using Mathematical.Expressions;

namespace DynamicSystems
{
   public partial class AppForm : Form
   {
      private Calculator calc;
      private Dictionary<string, double> vars;
      private string evolutionVarName;
      private double evolutionStep;

      public AppForm()
      {
```

```
    InitializeComponent();
    calc = new Calculator();
}

private void init()
{
    vars = initVars();
    evolutionVarName = getEvolutionVarName();
    evolutionStep = getEvolutionStep();
}

private void calculateBtn_Click(object sender, EventArgs e)
{
    init();

    List<Pair> results = new List<Pair>();
    for (long i = 0; i < iterCountControl.Value; ++i)
    {
        if (i != 0)
        {
            goNextStep();
        }
        Matrix matrix = initMatrix();
        double result = calc.calculate(matrix);
        results.Add(prepareResult(result));
```

```
        }
        resultGraph.Image = new ResultPrinter(resultGraph.Width,
resultGraph.Height).draw(results, evolutionVarName);
    }

    private Dictionary<string, double> initVars()
    {
        Dictionary<string, double> dictionary = new Dictionary<string,
double>();
        for (int rowIdx = 0; rowIdx < variablesGrd.Rows.Count - 1; ++rowIdx)
        {

dictionary.Add(variablesGrd.Rows[rowIdx].Cells[0].Value.ToString(),
Convert.ToDouble(variablesGrd.Rows[rowIdx].Cells[1].Value.ToString()));
        }
        return dictionary;
    }

    private Matrix initMatrix()
    {
        Matrix matrix = new Matrix();
        matrix.A = evalExpr(matrix11.Text);
        matrix.B = evalExpr(matrix12.Text);
        matrix.C = evalExpr(matrix21.Text);
        matrix.D = evalExpr(matrix22.Text);
```

```
    return matrix;
}

private double evalExpr(string expr)
{
    ExpressionEval eval = new ExpressionEval(expr);

    foreach (KeyValuePair<string, double> var in vars)
    {
        eval.SetVariable(var.Key, var.Value);
    }
    return (double)eval.Evaluate();
}

private int getEvolutionVarId()
{
    for (int rowIdx = 0; rowIdx < variablesGrd.Rows.Count - 1; ++rowIdx)
    {
        if (variablesGrd.Rows[rowIdx].Cells[2].Value != null)
        {
            return rowIdx;
        }
    }
    return -1;
}
```

```
        private string getEvolutionVarName()
        {
            int varId = getEvolutionVarId();
            if (varId != -1)
            {
                return variablesGrd.Rows[varId].Cells[0].Value.ToString();
            }
            return null;
        }

        private double getEvolutionStep()
        {
            int varId = getEvolutionVarId();
            if (varId != -1)
            {
                return
Convert.ToDouble(variablesGrd.Rows[varId].Cells[2].Value.ToString());
            }
            return 0;
        }

        private void goNextStep()
        {
            vars[evolutionVarName] += evolutionStep;
        }
```

```
private Pair prepareResult(double result)
{
   return new Pair
   {
      Var = (evolutionVarName != null ? vars[evolutionVarName] : 0),
      Result = result
   };
}

private class Pair
{
   public double Var { get; set; }
   public double Result { get; set; }
}

private class ResultPrinter
{
   private const int MARGIN = 5;
   private int width;
   private int height;
   private double horizontalFactor;
   private double verticalFactor;
   private double xMinValue;
   private double yMinValue;
```

```
        public ResultPrinter(int width, int height)
        {
            this.width = width;
            this.height = height;
        }

        public Bitmap draw(List<Pair> result, string varName)
        {
            init(result);
            Bitmap bm = new Bitmap(width, height);
            Graphics gr = Graphics.FromImage(bm);
            gr.Clear(Color.White);
            gr.SmoothingMode = SmoothingMode.AntiAlias;
            Pen redPen = new Pen(Color.Red, 1);
            Font mainFont = new Font("Verdana", 14f, FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            Font smallFont = new Font("Verdana", 8f, FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            Brush blackBrush = new SolidBrush(Color.Black);
            gr.DrawString("e", mainFont, blackBrush, new Point(5, 7));
            gr.DrawString("A", smallFont, blackBrush, new Point(16, 5));
            gr.DrawString(varName, mainFont, blackBrush, new Point(width-
20, height - 20));
            for (int i = 1; i < result.Count; ++i)
            {
```

```
                gr.DrawLine(redPen, calculateX(result[i - 1].Var),
calculateY(result[i - 1].Result), calculateX(result[i].Var),
calculateY(result[i].Result));
            }
            return bm;
        }

        private float calculateX(double value)
        {
            return (float)(MARGIN + (value - xMinValue) * horizontalFactor);
        }

        private float calculateY(double value)
        {
            return (float)(height - MARGIN - ((value - yMinValue) *
verticalFactor));
        }

        private void init(List<Pair> result)
        {
            initHorizontalCharacteristics(result);
            initVerticalCharacteristics(result);
        }

        private void initHorizontalCharacteristics(List<Pair> result)
        {
```

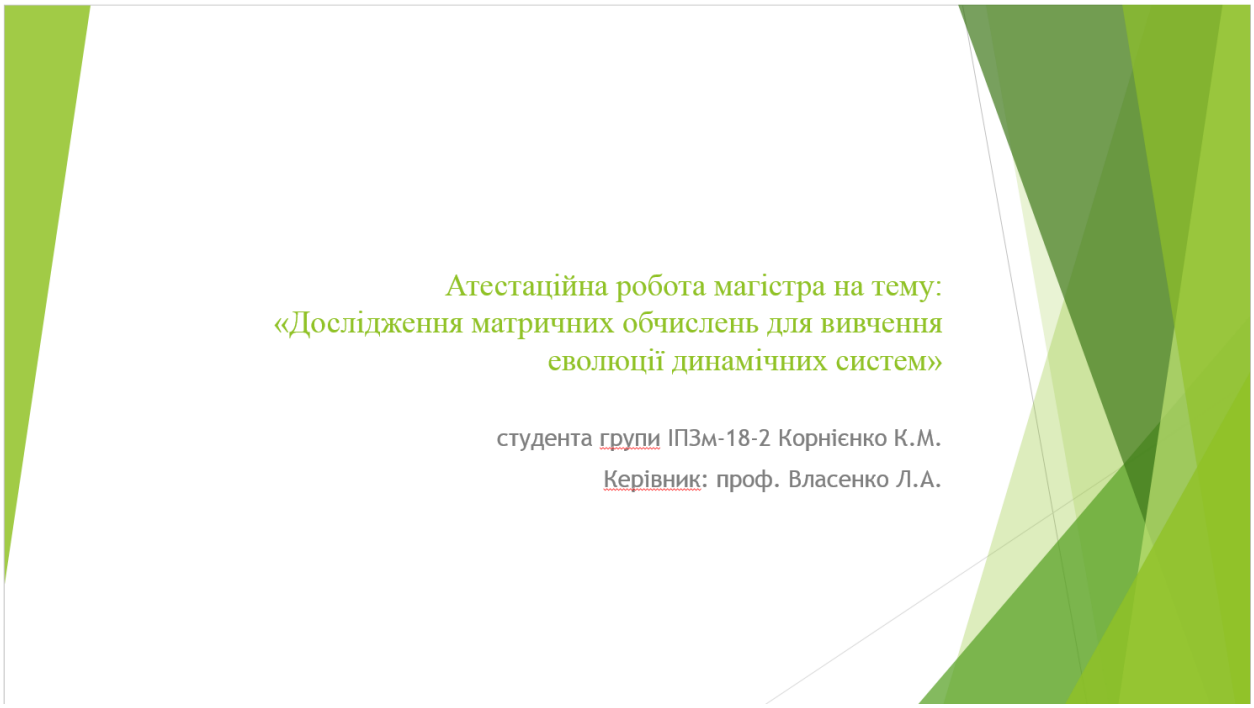
```
            double minValue = result[0].Var;
            double maxValue = result[0].Var;
            foreach (Pair pair in result)
            {
                if (pair.Var > maxValue) { maxValue = pair.Var; }
                if (pair.Var < minValue) { minValue = pair.Var; }
            }
            horizontalFactor = ((double)width - MARGIN * 2) / (maxValue -
minValue);
            xMinValue = minValue;
        }

        private void initVerticalCharacteristics(List<Pair> result)
        {
            double minValue = result[0].Result;
            double maxValue = result[0].Result;
            foreach (Pair pair in result)
            {
                if (pair.Result > maxValue) { maxValue = pair.Result; }
                if (pair.Result < minValue) { minValue = pair.Result; }
            }
            verticalFactor = ((double)height - MARGIN * 2) / (maxValue -
minValue);
            yMinValue = minValue;
        }
```

```
        }
    }
}
```

ДОДАТОК Б

Слайди презентації



Вступ

- В останній час увага до математичних методів, які дозволяють якісно проаналізувати досліджувану динамічну систему дуже підвищилась. Ця увага обумовлена проблемами, які виникають при моделюванні складних систем, які вимагають чисельного дослідження за умовою виявлених особливостей існуючої системи з використанням сучасних обчислювальних засобів
- Експонента матриці - матрична функція від квадратної матриці, аналогічна звичайній експоненційної функції. Матрична експонента встановлює зв'язок між алгеброю Лі матриць і відповідний групою Лі. Саме експоненціала матриці відіграє важливу роль у лінійних системах управління та звичайних диференціальних рівняннях. Однак теоретичний аналіз та чисельні обчислення все ще вивчаються.

Огляд наукової і патентної літератури

- Перший приклад стаття «Цікавий метод експоненти деяких спеціальних матриць» Цзінь Чена та Гонффен Зуу в книзі Systems Science & Control Engineering від компанії Taylor & Francis. В статті було описано центральну роль експоненціальної матиці у теорії лінійної системи та управління. Ними був розроблений метод для обчислення точного рішення для матричної експоненти з припущенням, що матриця, яка була використана має власне значення, на прикладах було показано ефективність запропонованого засобу.
- Другий яскравий приклад – стаття Хунгуо Сюя «Два результати щодо матричної експоненти» в якій розглядається дві проблеми, які виникають внаслідок лінійних систем управління, саме до яких і відносяться деякі матричні експоненціальні задачі (більшість відкритих проблем було сформовано Бернштейном).
- І останньою наведеною науковою працею буде стаття Деніса Бернштейна та Вазіна Со «Деякі явні формули для матричного експоненціалу», мета якої складається в тому, щоб представити явні формули для полегшення використання динамічних систем.

Дослідження експоненти загальної матриці 2 x 2

- Згідно з темою магістерської роботи та аналізу існуючої літератури були проведені наступні теоретичні дослідження існуючої явної формули:
- Нехай є матриця $A = \begin{pmatrix} a & b \\ 0 & d \end{pmatrix} \in C^{2x2}$.
- Розглянемо 2 випадки:
- якщо a = d, тоді $e^A = e^a \begin{pmatrix} 1 & b \\ 0 & 1 \end{pmatrix}$
- 2) якщо a ≠ d, тоді $e^A = e^a \begin{pmatrix} e^a & b(e^a - e^d)/(a-d) \\ 0 & e^d \end{pmatrix}$
- Результати спеціалізуються на тому випадку, коли e^A - реальна матриця. Наступний результат характеризує його з точки зору власних значень.
- Нехай μ та λ позначать власні значення $A \in C^{2x2}$.
- Також розглянемо 2 випадки:
- якщо $\mu = \lambda$, тоді $e^A = e^\lambda[(1-\mu)I + A]$
- якщо $\mu \neq \lambda$, тоді $e^A = \frac{\mu e^\lambda - \lambda e^\mu}{\mu - \lambda} I + \frac{e^\mu - e^\lambda}{\mu - \lambda} A$

Дослідження експоненціальності матриць n * n, що виставляється квадратичним поліноміялом

- Для дослідження матричних обчислень необхідно отримати формулу для матриці n * n, яка задовольняє квадратичний многочлен. Оскільки за теоремою Кейлі-Гамільтона цей клас включає всі матриці 2 * 2, результати досліджень, які були описані в попередньому підрозділі відносяться до особливого випадку. Крім того, ці результати застосовуються до певних матриць n * n, таких як рангових або ідентифікуючих матриць.
- Розглянемо наступну лему:
- Нехай $A \in C^{n*n}$ та припустимо, що $A^2 = pI$, де $p \in C$.
- Розглянемо наступні випадки:
- p = 0, тоді $e^A = I + A$.
- p ≠ 0, тоді $e^A = \cosh(\sqrt{p})I + (\sinh(\sqrt{p})/(\sqrt{p}))A$.
- Доказ:
- Так як $A^2 = pI$, $A^{2k} = p^k I$ та $A^{2k+1} = p^k A$ у випадку k ≥0. Тоді,
- $e^A = \sum_{k=0}^{\infty} \frac{A^k}{k!} = \left[I + \frac{p^2}{2!} + \frac{p^4}{4!} + \cdots\right] + \left[A + \frac{p^3}{3!} + \frac{p^5}{5!} + \cdots\right] = \left[1 + \frac{p}{2!} + \frac{p^2}{4!} + \cdots\right] + \left[I + \frac{p}{3!} + \frac{p^2}{5!} + \cdots\right]^* A = \cosh(\sqrt{p})I + (\sinh(\sqrt{p})/(\sqrt{p}))A$.

Вибір технологій

- Згідно з попередніми дослідженнями та сформульованою задачею необхідно створити програмний продукт, який полегшить процес дослідження матричних обчислень для обчислень еволюцій.
- Програмний продукт має графічно відображати формули, які використовуються для дослідження еволюцій матричних систем. Спочатку потрібно вибрати мову програмування, найбільш популярними на сьогоднішній день є Java, C# та Python. Необхідно проаналізувати переваги та недоліки перелічених мов програмування.
- Java – це об'єктно-орієнтована мова програмування, яка зазвичай використовується для мобільних, мережевих та корпоративних застосунків. Саме ця мова використовується для більшості наукових проектів, особливо в сфері аналізу природньої мови. До переваг можна віднести – простий синтаксис, стандарт для корпоративних обчислювальних систем, безпека, незалежність від платформи та многопоточність. До недоліків можна віднести – низьку продуктивність, відсутність нативного дизайну, складний код.
- C# – це також об'єктно-орієнтована мова програмування, яка в свою чергу використовується для мобільних та десктопних застосунків. До переваг відносяться функціональне програмування, гнучкість, орієнтація на безпеку та підтримку майкрософт. З недоліків можливо виділити орієнтованість здебільшого на .NET, складний синтаксис та низьку продуктивність.

Вибір технологій

- Python – високорівнева мова програмування, стандартна бібліотека якої включає в себе великий об'єм корисних функцій. Python підтримує об'єктно-орієнтоване, функціональне, імперативне та аспектно-орєінтоване програмування. До переваг відносяться широке використання, простий синтаксис, велика кількість якісних бібліотек. До недоліків – низька швидкість виконання програм, незручність використання для мобільних розробок та використання великої кількості пам'яті через гнучкість типів даних.
- Отже, після аналізу переваг та недоліків найбільш популярних мов програмування десктопних застосунків був обраний C# для подальшого програмного продукту.
- Так як програмний продукт повинен графічно відображати формули потрібно використати простір імен C# - Drawing.Drawing2D, який забезпечує розширений функціонал двовимірної та векторної графіки. Також у цьому просторі імен є такий клас як Matrix та метод MatrixOrder, який вказує порядок операцій перетворення матриці.
- Для подібного програмного продукту не потрібно зберігати дані, тож необхідності у бази даних немає, але якщо у майбутньому потрібно буде аналізувати та порівнювати усі обчислення можливо буде її додати.

Етапи розробки

- Розробка програмного продукту – це не тільки створення її коду, а ще багато інших процесів, до яких відносяться – постановка задачі, формулювання задачі, створення алгоритму, створення програми, синтаксична відладка, тестування програми, документування та аналіз отриманих даних.
- Постановка задачі – це етап на якому задача розбивається на логічні частини, частково це було зроблено у розділі з аналізом предметної області.
- До етапу формулювання задачі відноситься також і аналіз задачі. На цьому етапі необхідно детально проаналізувати стан розв'язання проблеми та описати поставлену задачу, цей етап був також детально виконаний у першому розділі.
- На етапі створення алгоритму треба детально описати кроки, які потрібні для вирішення поставленої задачі. Згідно з задачею можливо виділити наступні кроки – користувач вводить необхідні дані до матриці (коефіцієнти та усі числові значення), потім на основі введених даних, кількості кроків та теоретичних досліджень будується графічне зображення.
- На наступному етапі створюється програма, яка потім підлягає відладці, яка дозволяє виявити помилки, які були допущені в коді. Код програми повинен бути легкозрозумілим та відповідати стандартним, також повинні бути функціональні тести, які покриватимуть значну частину програмної реалізації.

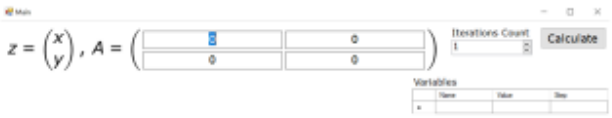
Етапи розробки

- На етапі тестування програма тестується с різними наборами даних, щоб перевірити чи вона виконує поставлену задачу и які помилки потрібно виправити.
- Існує два основних види тестування – функціональне та структурне. При функціональному тестування програма розглядається в якості «чорного яшику та перевіряється згідно з специфікацією. При структурному тестуванні програма розглядається як «білий яшик», перевіряється саме логіка програми та усі можливі дії. Отже, структурне та функціональне тестування не може в повному обсязі перевірити правильність програми.
- Етап документування включає в себе багато описів, які полегшують процес програмування, до цього етапу входять постановка задачі, проектні документи, алгоритми та програми. Також необхідно створити інструкцію користувача, в якій повинне бути детальне описання як користуватись програмним продуктом.
- На останньому етапі аналізу отриманих даних необхідно перевірити на реальних даних чи правильно працює програма. На цьому етапі створення програми закінчено.
- Отже, етапи розробки були визначені, також були обрані технології відповідно до існуючий вимог, тож можливо переходити до етапу програмної реалізації.

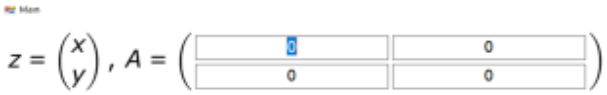
Програмна реалізація

«Рисунок 4.2 – Матриця»

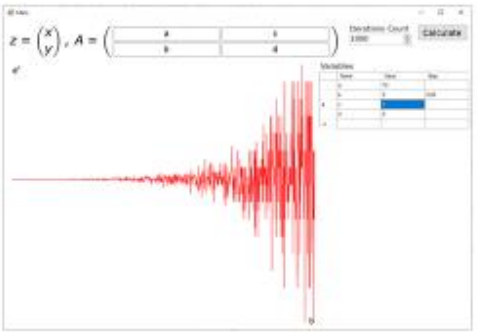
При запуску програми з'являється головне вікно програми:

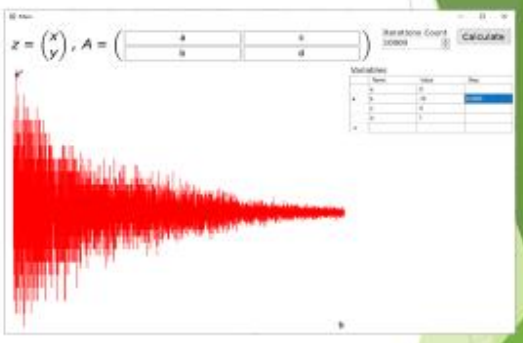


На головному вікні знаходиться матриця, яку необхідно заповнити коефіцієнтами або виразами:

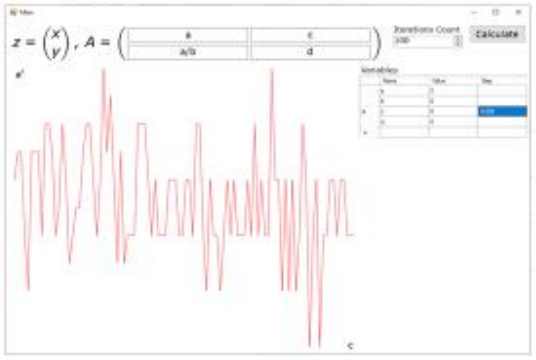


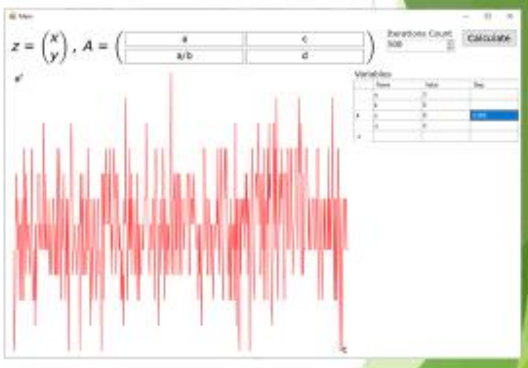
Програмна реалізація





Програмна реалізація





Програмна реалізація