

Лабораторна робота №2. Використання функціональної декомпозиції для розв'язку обчислювальних задач

Варіант № 8.

Мета роботи: вивчити методи декомпозицій задач. Набути навичок розв'язування задач з використанням функціональної декомпозиції.

Завдання.

8	$x=(K_1y_1'Y_3y_1y_2y_2'+K_2Y_3^2+y_1y_2')K_1(y_2'Y_3y_2y_1+y_1)$		
	$b_i=8/i$ $i=1,2,...n$	$A_1(2b_1+3c_1)$	$A_2(B_2-C_2)$ $C_{ij}=1/(i+j+2)$

Схема декомпозиції задачі:

Для кожного окремого елемента, який потрібно буде обчислити - я створив getter, setter, а також lock object. Наприклад:

```
private Object bl = new Object();
Vector<double> _b = null;
public Vector<double> b {
    get {
        lock(bl) {
            if (_b == null) {
                _b = Vector<double>.Build.Dense(n, i => 8.0 / (1+i));
                if (debug) Console.WriteLine("Generated b: {0}", _b);
            }
        }
        return _b;
    }
    set { this._b = value; }
}
```

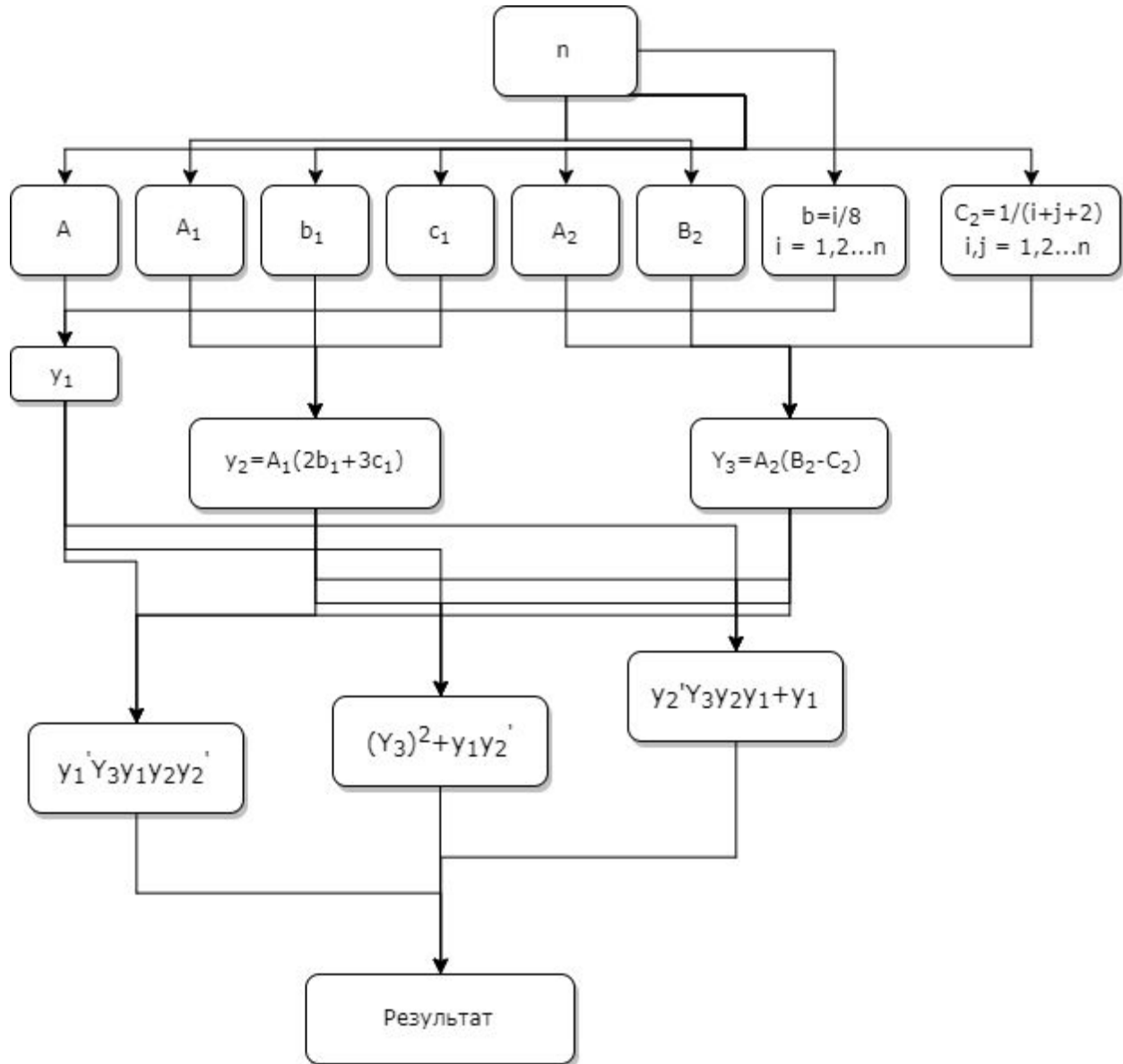
Тут реалізовано lazy calculation - тобто данні не обраховуються до тих пір, поки вони явно не знадобляться.

bl - це lock object для вектора b, він забезпечить відсутність зайвих перерахунків, якщо наприклад до цього вектора спробують одночасно доступитись два різних потоки.

_b - це прихована змінна, яка й містить сам вектор, зроблено це для того щоб до вектора можна було мати доступ лише через відповідно визначені функції.

b - це реалізація функцій для доступу та запису, як бачимо, тут використовується блокування, для того щоб не допустити зайвих обчислень у разі використання декількох потоків.

Декомпозиція прикладу



Як ми всі знаємо - створення потоку це не безкоштовна операція, іноді при занадто малих об'ємах обчислення створення ще одного потоку не тільки не дає пришвидшення, а й сповільнює роботу програми. Отже, я вирішив розбити обчислення прикладу на 3 великих частини, які містять достатню кількість обчислень, щоб виправдати створення для них окремого потоку.

Текст програми

```
using System;
using System.Diagnostics;
using System.Threading;
using MathNet.Numerics;
using MathNet.Numerics.LinearAlgebra;

namespace lab2
{
    class ExpressionNo8 {

        static void Main(string[] args) {
            ExpressionNo8 expr = new ExpressionNo8();
            int k1, k2;
            Stopwatch sw = new Stopwatch();
            while(true) {
                k1 = CLI.GetInt("Enter K1: ", 1);
                k2 = CLI.GetInt("Enter k2: ", 1);
                Console.WriteLine("Calculating results...");
                sw.Start();
                double result = expr.Calculate(k1, k2);
                sw.Stop();
                Console.WriteLine("Done!\nResult: {0}\nTook: {1} ms", result,
sw.ElapsedMilliseconds);
            }
        }

        bool debug;
        int n;

        Random _r = null;
        public Random r {
            get { if (_r == null) _r = new Random(); return _r; }
            set { _r = value; }
        }

        private Object bl = new Object();
        Vector<double> _b = null;
        public Vector<double> b {
            get {
                lock(bl) {
                    if (_b == null) {
                        _b = Vector<double>.Build.Dense(n, i => 8.0 / (1+i));

                        if (debug) Console.WriteLine("Generated b: {0}", _b);
                    }
                }
            }
        }
    }
}
```

```

        return _b;
    }
    set { this._b = value; }
}

Vector<double> b1;
Vector<double> c1;

private Object y1l = new Object();
Vector<double> _y1 = null;
Vector<double> y1 {
    get {
        lock(y1l) {
            if (_y1 == null) {
                _y1 = A * b;

                if (debug) Console.WriteLine("Generated y1: {0}", _y1);
            }
        }
        return _y1;
    }
    set { _y1 = value; }
}

private Object y2l = new Object();
Vector<double> _y2 = null;
Vector<double> y2 {
    get {
        lock(y2l) {
            if (_y2 == null) {
                _y2 = A1 * (b1 + c1);

                if (debug) Console.WriteLine("Generated y2: {0}", _y2);
            }
        }

        return _y2;
    }
    set { _y2 = value; }
}

Matrix<double> A;
Matrix<double> A1;
Matrix<double> A2;
Matrix<double> B2;

private Object C2l = new Object();
Matrix<double> _C2 = null;
Matrix<double> C2 {

```

```

        get {
            lock(C21) {
                if (_C2 == null) {
                    _C2 = Matrix<double>.Build.Dense(n, n, (i,j)=> 1.0 /
((i+1)+(j+1)+2));

                    if (debug) Console.WriteLine("Generated C2: {0}", _C2);
                }
            }
            return _C2;
        }
        set { _C2 = value; }
    }
}

```

```

private Object Y31 = new Object();
Matrix<double> _Y3 = null;
Matrix<double> Y3 {
    get {
        lock(Y31) {
            if (_Y3 == null) {
                _Y3 = A2 * (B2 - C2);

                if (debug) Console.WriteLine("Generated Y3: {0}", _Y3);
            }
        }

        return _Y3;
    }
    set { _Y3 = value; }
}

```

```

public ExpressionNo8(int n, Matrix<double> A, Matrix<double> A1,
    Vector<double> b1, Vector<double> c1,
    Matrix<double> A2, Matrix<double> B2) {

    this.n = n;
    this.A = A;

    this.A1 = A1;
    this.b1 = b1;
    this.c1 = c1;

    this.A2 = A2;
    this.B2 = B2;
}

```

```

public ExpressionNo8() {

```

```

int n = CLI.GetInt("Please, enter n", 8);
bool debug = CLI.GetBool("Debug mode?", false);

Matrix<double> A = null;
bool ans = CLI.GetBool("Random matrix A?", true);
if (ans)
    A = Matrix<double>.Build.Dense(n, n, (i, j) => 1+r.Next(n));
else
    A = CLI.GetMatrix("Enter A matrix.", n, n);
Console.WriteLine(A);

Matrix<double> A1 = null;
ans = CLI.GetBool("Random matrix A1?", true);
if (ans)
    A1 = Matrix<double>.Build.Dense(n, n, (i, j) => 1+r.Next(n));
else
    A1 = CLI.GetMatrix("Enter A1 matrix.", n, n);
Console.WriteLine(A1);

Vector<double> b1 = null;
ans = CLI.GetBool("Random vector b1?", true);
if (ans)
    b1 = Vector<double>.Build.Dense(n, i => 1+r.Next(n));
else
    b1 = CLI.GetVector("Enter b1 vector.", n);
Console.WriteLine(b1);

Vector<double> c1 = null;
ans = CLI.GetBool("Random vector c1?", true);
if (ans)
    c1 = Vector<double>.Build.Dense(n, i => 1+r.Next(n));
else
    c1 = CLI.GetVector("Enter c1 vector.", n);
Console.WriteLine(c1);

Matrix<double> A2 = null;
ans = CLI.GetBool("Random matrix A2?", true);
if (ans)
    A2 = Matrix<double>.Build.Dense(n, n, (i, j) => 1+r.Next(n));
else
    A2 = CLI.GetMatrix("Enter A2 matrix.", n, n);
Console.WriteLine(A2);

Matrix<double> B2 = null;
ans = CLI.GetBool("Random matrix B2?", true);
if (ans)
    B2 = Matrix<double>.Build.Dense(n, n, (i, j) => 1+r.Next(n));
else
    B2 = CLI.GetMatrix("Enter B2 matrix.", n, n);
Console.WriteLine(B2);

```

```

        calculate(n, A, A1, b1, c1, A2, B2, debug);
    }

private void calculate(int n, Matrix<double> A, Matrix<double> A1,
    Vector<double> b1, Vector<double> c1,
    Matrix<double> A2, Matrix<double> B2, bool debug=false) {

    this.debug = debug;
    //
    this.n = n;
    this.A = A;

    this.A1 = A1;
    this.b1 = b1;
    this.c1 = c1;

    this.A2 = A2;
    this.B2 = B2;

}

public int K1;
public int K2;

private double firstPart = 0;
private void calculateFirstPart() {
    firstPart = K1 * y1 * Y3 * y1 * y2 * y2;
}

private Matrix<double> secondPart = null;
private void calculateSecondPart() {
    secondPart = K2 * Y3 * Y3 + y1 * y2;
}

private Vector<double> thirdPart = null;
private void calculateThirdPart() {
    thirdPart = y2 * Y3 * y2 * y1 + y1;
}

public double Calculate(int K1, int K2) {
    reset();
    this.K1 = K1;
    this.K2 = K2;

    Thread firstPartThread = new Thread(calculateFirstPart);
    Thread secondPartThread = new Thread(calculateSecondPart);
    Thread thirdPartThread = new Thread(calculateThirdPart);
    firstPartThread.Start();
    secondPartThread.Start();
}

```

```

        thirdPartThread.Start();

        firstPartThread.Join();
        secondPartThread.Join();
        thirdPartThread.Join();

        return ((firstPart + secondPart) * K1 * (thirdPart))[0];
    }

    private void reset() {
        b = null;
        y1 = null;
        y2 = null;
        Y3 = null;
        C2 = null;
    }
}

class CLI {

    static void showPrompt(string prompt, string current) {
        Console.WriteLine(prompt);
        Console.Write("[{0}]> ", current);
    }

    static string[] BOOL_YES = {"yes", "y", "true", "da", "sure"};
    static string[] BOOL_NO = {"no", "n", "false", "net", "not sure"};

    public static bool GetBool(string prompt, bool d) {
        showPrompt(prompt, d.ToString());
        string s = Console.ReadLine().Trim().ToLower();
        if (Array.Exists(BOOL_YES, e => s.Equals(e))) {
            return true;
        }
        else if (Array.Exists(BOOL_NO, e => s.Equals(e))) {
            return false;
        }
        return d;
    }

    public static int GetInt(string prompt, int d) {
        showPrompt(prompt, d.ToString());
        return ParseInt(d);
    }

    public static int ParseInt(int d) {
        int result = -1;
        bool success = Int32.TryParse(Console.ReadLine(), out result);
        if (success && result > 0)

```



```

        return result;
    return d;
}

public static Vector<double> GetVector(string prompt, int w) {
    int d = 1;
    Vector<double> result = Vector<double>.Build.Dense(w, d);
    Console.WriteLine(prompt);

    for(int j = 0; j < w; j++) {
        showPrompt("Enter element.", (j+1).ToString());
        result[j] = ParseInt(d);
    }
    Console.WriteLine(result);

    return result;
}

public static Matrix<double> GetMatrix(string prompt, int w, int h) {
    int d = 1;
    Matrix<double> result = Matrix<double>.Build.Dense(w, h, d);
    Console.WriteLine(prompt);

    for(int i = 0; i < h; i++) {
        for(int j = 0; j < w; j++) {
            showPrompt("Enter element.", String.Format("{0}, {1}", (j+1),
(i+1)));

            result[i,j] = ParseInt(d);
        }
        Console.WriteLine(result);
    }
    return result;
}

}
}

```

Приклад роботи програми

C:\Windows\System32\cmd.exe - dotnet run lab2

```
Please, enter n
[8]>
Debug mode?
[False]> True
Random matrix A?
[True]>
DenseMatrix 8x8-Double
3 3 6 2 5 8 8 2
4 6 3 3 1 4 6 1
2 7 8 5 7 4 8 7
1 5 6 5 5 4 8 3
5 3 4 7 6 1 5 8
5 2 8 4 2 5 6 1
5 4 3 6 8 5 4 5
3 7 2 6 8 8 3 7
```

```
Random matrix A1?
[True]>
DenseMatrix 8x8-Double
8 3 7 5 7 5 7 1
2 8 6 5 5 4 2 2
1 7 6 4 4 5 5 6
3 1 1 6 8 5 2 7
3 5 5 3 4 8 3 4
7 8 7 6 3 4 4 4
1 2 1 1 5 6 3 2
4 6 7 8 1 3 2 6
```

```
Random vector b1?
[True]>
DenseVector 8-Double
8
7
3
1
4
3
2
4
```

C:\Windows\System32\cmd.exe - dotnet run lab2

```
Random vector c1?
[True]>
DenseVector 8-Double
1
5
5
1
2
5
8
2
```

```
Random matrix A2?
[True]>
DenseMatrix 8x8-Double
7 2 2 1 3 2 8 2
1 3 8 1 5 8 5 3
7 7 5 2 8 7 7 1
1 3 1 7 3 1 2 2
2 5 1 8 6 4 8 5
7 7 6 2 2 8 7 7
5 5 8 1 5 8 1 4
3 1 5 4 2 3 2 1
```

```
Random matrix B2?
[True]>
DenseMatrix 8x8-Double
3 3 1 4 4 1 2 8
4 4 8 6 7 6 8 4
8 7 7 1 7 7 1 6
1 2 4 4 1 5 6 1
7 7 3 1 8 8 1 2
3 7 7 4 7 8 1 1
4 1 4 5 4 4 6 1
1 1 4 4 6 4 4 8
```

```
Enter K1:
[1]>
Enter k2:
[1]>
Calculating results...
```

```

C:\Windows\System32\cmd.exe - dotnet run lab2
Generated b: DenseVector 8-Double
8
4
2.66667
2
1.6
1.33333
1.14286
1

Generated C2: DenseMatrix 8x8-Double
0.25 0.2 0.166667 0.142857 0.125 .. 0.1 0.0909091
0.2 0.166667 0.142857 0.125 0.111111 .. 0.0909091 0.0833333
0.166667 0.142857 0.125 0.111111 0.1 .. 0.0833333 0.0769231
0.142857 0.125 0.111111 0.1 0.0909091 .. 0.0769231 0.0714286
0.125 0.111111 0.1 0.0909091 0.0833333 .. 0.0714286 0.0666667
0.111111 0.1 0.0909091 0.0833333 0.0769231 .. 0.0666667 0.0625
0.1 0.0909091 0.0833333 0.0769231 0.0714286 .. 0.0625 0.0588235
0.0909091 0.0833333 0.0769231 0.0714286 0.0666667 .. 0.0588235 0.0555556

Generated y1: DenseVector 8-Double
85.0895
84.7985
100.01
79.4762
101.314
95.0571
105.038
103.229

Generated Y3: DenseMatrix 8x8-Double
102.795 86.4287 100.884 102.23 136.503 115.725 96.9093 107.065
157.387 167.972 184.419 104.773 213.062 213.301 92.5042 113.678
190.162 195.16 205.889 153.449 256.894 242.256 144.559 150.815
61.1154 65.5003 89.7906 74.0183 88.2022 107.354 91.4019 56.5909
127.741 127.413 174.924 149.327 192.655 207.927 168.157 111.355
165.089 173.084 225.812 176.371 258.817 233.181 164.484 193.742
161.296 184.119 191.723 116.188 232.558 216.061 96.1136 157.328
85.771 91.2317 98.5714 64.8338 107.043 114.215 62.3581 77.4796

C:\Windows\System32\cmd.exe - dotnet run lab2

Generated y2: DenseVector 8-Double
332
266
299
209
275
341
163
266

Done!
Result: 2.6971809003857E+25
Took: 64 ms

```

Висновок

В роботі був використаний паралелізм на рівні підзадач які виникли в результаті функціональної декомпозиції. Вирішення задачі представляє з себе консольну утиліту. В коді програми використано lock та Thread. Процес обчислення результатів продумано так - щоб не виникало колізій в пам'яті. Також є можливість вираховувати час виконання програми.