Міністерствоосвіти і науки України

Національний університет „Львівська політехніка”

Кафедра ЕОМ



**Лабораторна робота №1**

З дисципліни:”Паралельні та розподілені обчислення”

На тему:” ВИКОРИСТАННЯ ФУНКЦІОНАЛЬНОЇ ДЕКОМПОЗИЦІЇ ДЛЯ РОЗВ’ЯЗКУ ОБЧИСЛЮВАЛЬНИХ ЗАДАЧ”

Виконав: ст.гр.КІ-33

Фещенко З.-А.С.

Прийняв:

Козак Н.Б.

Львів 2020

**Мета:**

Вивчити методи декомпозицій задач. Набути навиків розв’язування задач з використанням функціональної декомпозиції.

**Варіант 19**

|  |  |  |  |
| --- | --- | --- | --- |
| 19 | число | | |
| bi=19/(i2+1) для парних і  bi=19 для непарних і | A1(b1+19c1) | A2(B2+C2)  Cij=19/(i+2j)3 |

*Правила знаходження елементів виразу.*

1).Задати\* квадратну матрицю А порядку n. Отримати вектор(стовпець) , де b – вектор-стовпець, елементи якого обраховуються за формулою, згідно варіанту.

2).Задати квадратну матрицю А1 порядку n та вектори-стовпці b1 та c1з nелементами кожен. Отримати вектор  згідно формули, що задається варіантом.

3).Задати квадратні матриці А2 та B2порядку n. Отримати матрицю , яка залежить від А2, B2 та додатково визначеної матриці С2, елементи якої знаходяться за формулою, вказаною варіантом.

A close up of text on a white background

Description automatically generated

Виконання в консолі:

A screenshot of a cell phone

Description automatically generated

Результат з файлу:

Vector B1  
5  
3  
5  
5  
  
Vector C1  
5  
1  
3  
4  
  
Matrix A  
1   
2   
2   
2   
  
3   
1   
4   
4   
  
2   
5   
5   
3   
  
5   
2   
1   
4   
  
  
Matrix A1  
4   
2   
1   
2   
  
4   
2   
2   
3   
  
4   
5   
3   
1   
  
4   
2   
5   
4   
  
  
Matrix A2  
4   
5   
4   
5   
  
4   
3   
3   
1   
  
1   
2   
1   
2   
  
3   
3   
5   
3   
  
  
Matrix B2  
1   
3   
1   
2   
  
3   
3   
5   
5   
  
4   
1   
2   
5   
  
4   
2   
5   
1   
  
  
Vector B  
19  
95  
19  
323  
  
Matrix C2  
57   
95   
133   
171   
  
76   
114   
152   
190   
  
95   
133   
171   
209   
  
114   
152   
190   
228   
  
  
B2 + C2  
58   
98   
134   
173   
  
79   
117   
157   
195   
  
99   
134   
173   
214   
  
118   
154   
195   
229   
  
  
b1 + 19c1  
100   
22   
62   
81   
  
A2(B2+C2)  
1277   
1371   
1525   
1175   
  
1526   
1645   
1799   
1411   
  
1747   
1885   
2041   
1617   
  
1970   
2129   
2274   
1821   
  
  
A1(b1 + 19c1)  
668   
811   
777   
1078   
  
A\*b  
893   
1520   
1577   
1596   
  
Y3^2  
8.7018e+06   
9.382262e+06   
1.0198329e+07   
8.040556e+06   
  
1.0381495e+07   
1.1193305e+07   
1.2166878e+07   
9.592559e+06   
  
1.1858546e+07   
1.278584e+07   
1.3898029e+07   
1.0957314e+07   
  
1.3324592e+07   
1.4366474e+07   
1.5616509e+07   
1.2311868e+07   
  
  
y'2\*y1  
4.775061e+06  
y2\*y'1  
596524   
1.01536e+06   
1.053436e+06   
1.066128e+06   
  
724223   
1.23272e+06   
1.278947e+06   
1.294356e+06   
  
693861   
1.18104e+06   
1.225329e+06   
1.240092e+06   
  
962654   
1.63856e+06   
1.700006e+06   
1.720488e+06   
  
  
y'2\*y1\*Y3 number \* matrix  
0   
0   
0   
0   
  
0   
0   
0   
0   
  
0   
0   
0   
0   
  
0   
0   
0   
0   
  
  
y2\*y'1+Y3^2 matrix\*matrix  
9.298324e+06   
1.0397622e+07   
1.1251765e+07   
9.106684e+06   
  
1.1105718e+07   
1.2426025e+07   
1.3445825e+07   
1.0886915e+07   
  
1.2552407e+07   
1.396688e+07   
1.5123358e+07   
1.2197406e+07   
  
1.4287246e+07   
1.6005034e+07   
1.7316515e+07   
1.4032356e+07   
  
  
y2\*y'1+Y3^2+y'2\*y1\*Y3  
9.298324e+06   
1.0397622e+07   
1.1251765e+07   
9.106684e+06   
  
1.1105718e+07   
1.2426025e+07   
1.3445825e+07   
1.0886915e+07   
  
1.2552407e+07   
1.396688e+07   
1.5123358e+07   
1.2197406e+07   
  
1.4287246e+07   
1.6005034e+07   
1.7316515e+07   
1.4032356e+07   
  
  
y'1(y2\*y'1+Y3^2+y'2\*y1\*Y3)  
5.6386089841e+10  
6.7384546539e+10  
7.5755552593e+10  
8.6789946689e+10  
  
Result(y'1\*(y2\*y'1+Y3^2+y'2\*y1\*Y3)\*y2): 2.4473640215242e+14

**Висновок:**

Виконуючи дану лабораторну роботувивчив метод декомпозицій задач. Набув навиків розв’язування задач з використанням функціональної декомпозиції.

**Лістинги програми:**

**Модуль генерації/вводу даних:**

package main  
  
import (  
 "fmt"  
 log2 "log"  
 "math/rand"  
 "os"  
 "sync"  
)  
  
type startData struct {  
 \*inputData  
 matrixC2 [][]float64  
 vectorB []float64  
}  
  
type inputData struct {  
 size float64  
 matrixA [][]float64  
 matrixA1 [][]float64  
 matrixA2 [][]float64  
 matrixB2 [][]float64  
 vectorB1 []float64  
 vectorC1 []float64  
 printer Printer  
}  
  
func getStartData() \*startData {  
 data := &startData{}  
  
 data.inputData = processInput()  
  
 wg := &sync.WaitGroup{}  
 wg.Add(2)  
  
 go func() {  
 data.matrixC2 = generateC2(data.size)  
 wg.Done()  
 }()  
  
 go func() {  
 data.vectorB = generateB(data.size)  
 wg.Done()  
 }()  
  
 wg.Wait()  
  
 data.printer.Println("Vector B")  
 data.printer.showVectorHorizontal(data.vectorB)  
  
 data.printer.Println("Matrix C2")  
 data.printer.showMatrix(data.matrixC2)  
  
 return data  
}  
  
func generateB(size float64) []float64 {  
 vector := make([]float64, int(size))  
  
 var i float64 = 1  
 for ; i <= size; i++ {  
 if int(i)%2 == 0 {  
 vector[int(i-1)] = 19 \* (i\*i + 1)  
 continue  
 }  
  
 vector[int(i-1)] = 19  
 }  
  
 return vector  
}  
  
func generateC2(size float64) [][]float64 {  
 matrix := make([][]float64, int(size))  
  
 var i float64 = 0  
 for ; i < size; i++ {  
 matrix[int(i)] = make([]float64, int(size))  
  
 var j float64 = 0  
 for ; j < size; j++ {  
 var temp float64 = i + 1 + 2\*(j+1)  
 matrix[int(i)][int(j)] = 19 / temp \* temp \* temp  
 }  
 }  
  
 return matrix  
}  
  
func processInput() \*inputData {  
 var size float64  
 fmt.Print("Enter size: ")  
 fmt.Scan(&size)  
  
 var choice int  
  
 var log \*log2.Logger  
  
 fmt.Print("Do you want output into file or into console(0 - file, 1 - console): ")  
 fmt.Scan(&choice)  
  
 if choice == 1 {  
 log = log2.New(os.Stdout, "", 0)  
 } else {  
 file, err := os.Create("/Users/mmtretiak/go/src/PRO/output")  
 if err != nil {  
 panic(err)  
 }  
  
 log = log2.New(file, "", 0)  
 }  
  
 printer := &printer{log: log}  
  
 fmt.Print("Do you want to input data or generate(0 - input, 1 - generate): ")  
 fmt.Scan(&choice)  
  
 if choice == 1 {  
 return generateInput(size, printer)  
 }  
  
 return readInput(size, printer)  
}  
  
func readInput(size float64, printer Printer) \*inputData {  
 inputData := &inputData{  
 size: size,  
 printer: printer,  
 }  
  
 fmt.Println("Enter Matrix A")  
 inputData.matrixA = inputMatrix(inputData.size)  
  
 fmt.Println("Enter Matrix A1")  
 inputData.matrixA1 = inputMatrix(inputData.size)  
  
 fmt.Println("Enter Matrix A2")  
 inputData.matrixA2 = inputMatrix(inputData.size)  
  
 fmt.Println("Enter Matrix B2")  
 inputData.matrixB2 = inputMatrix(inputData.size)  
  
 fmt.Println("Enter Vector B1")  
 inputData.vectorB1 = inputVector(inputData.size)  
  
 fmt.Println("Enter Vector C1")  
 inputData.vectorC1 = inputVector(inputData.size)  
  
 return inputData  
}  
  
func inputMatrix(size float64) [][]float64 {  
 matrix := make([][]float64, int(size))  
  
 var i float64 = 0  
 for ; i < size; i++ {  
 matrix[int(i)] = make([]float64, int(size))  
  
 var j float64 = 0  
 for ; j < size; j++ {  
 fmt.Print("Enter element", i, j)  
  
 var temp float64  
  
 fmt.Scan(&temp)  
  
 matrix[int(i)][int(j)] = temp  
 }  
 }  
  
 return matrix  
}  
  
func inputVector(size float64) []float64 {  
 vector := make([]float64, int(size))  
  
  
 var j float64 = 0  
 for ; j < size; j++ {  
 fmt.Print("Enter element", j)  
  
 var temp float64  
  
 fmt.Scan(&temp)  
  
 vector[int(j)] = temp  
 }  
  
 return vector  
}  
  
func generateInput(size float64, printer Printer) \*inputData {  
 inputData := &inputData{  
 size: size,  
 printer: printer,  
 }  
  
 inputData.matrixA = genMatrix(inputData.size)  
 inputData.matrixA1 = genMatrix(inputData.size)  
 inputData.matrixA2 = genMatrix(inputData.size)  
 inputData.matrixB2 = genMatrix(inputData.size)  
  
 inputData.vectorB1 = genVector(inputData.size)  
 inputData.vectorC1 = genVector(inputData.size)  
  
 printer.Println("Vector B1")  
 printer.showVectorHorizontal(inputData.vectorB1)  
 printer.Println("Vector C1")  
 printer.showVectorHorizontal(inputData.vectorC1)  
 printer.Println("Matrix A")  
 printer.showMatrix(inputData.matrixA)  
 printer.Println("Matrix A1")  
 printer.showMatrix(inputData.matrixA1)  
 printer.Println("Matrix A2")  
 printer.showMatrix(inputData.matrixA2)  
 printer.Println("Matrix B2")  
 printer.showMatrix(inputData.matrixB2)  
  
 return inputData  
}  
  
func genMatrix(size float64) [][]float64 {  
 matrix := make([][]float64, int(size))  
  
 for i := 0; i < int(size); i++ {  
 matrix[i] = make([]float64, int(size))  
 for j := 0; j < int(size); j++ {  
 matrix[i][j] = rand.Float64()  
 }  
 }  
  
 return matrix  
}  
  
func genVector(size float64) []float64 {  
 vector := make([]float64, int(size))  
  
 for i := 0; i < int(size); i++ {  
 vector[i] = rand.Float64()  
 }  
  
 return vector  
}

**Модуль обчислень:**

package main  
  
import (  
 log2 "log"  
)  
  
func addMatrices(first, second [][]float64) [][]float64 {  
 size := len(first)  
  
 result := make([][]float64, size)  
  
 for i := 0; i < size; i++ {  
 result[i] = make([]float64, size)  
 for j := 0; j < size; j++ {  
 result[i][j] = first[i][j] + second[i][j]  
 }  
 }  
  
 return result  
}  
  
func addVectors(first, second []float64) []float64 {  
 size := len(first)  
  
 vector := make([]float64, size)  
  
 for i := 0; i < size; i ++ {  
 vector[i] = first[i] + second[i]  
 }  
  
 return vector  
}  
  
func multipleVectorByNumber(src []float64, multiplier float64) []float64 {  
 size := len(src)  
  
 vector := make([]float64, size)  
  
 for i := 0; i < size; i ++ {  
 vector[i] = src[i] \* multiplier  
 }  
  
 return vector  
}  
  
func multipleMatrices(first, second [][]float64) [][]float64{  
 size := len(first)  
  
 result := make([][]float64, size)  
 var total float64  
  
 for i := 0; i < size; i++ {  
 result[i] = make([]float64, size)  
 for j := 0; j < size; j++ {  
 for k := 0; k < size; k++ {  
 total = total + first[i][k]\*second[k][j]  
 }  
 result[i][j] = total  
 total = 0  
 }  
 }  
  
 return result  
}  
  
func multipleMatrixByNumber(matrix [][]float64, number float64) [][]float64{  
 size := len(matrix)  
  
 result := make([][]float64, size)  
  
 for i := 0; i < size; i++ {  
 result[i] = make([]float64, size)  
 for j := 0; j < size; j++ {  
 result[i][j] \*= number  
 }  
 }  
  
 return result  
}  
  
func multipleMatrixByVector(matrix [][]float64, vector []float64) []float64 {  
 size := len(vector)  
  
 result := make([]float64, size)  
 temp := make([][]float64, size)  
  
 for i := 0; i < size; i++ {  
 temp[i] = make([]float64, size)  
 for j := 0; j < size; j++ {  
 temp[i][j] = matrix[i][j]\*vector[j]  
 }  
 }  
  
 for i := 0; i < size; i++ {  
 result[i] = 0  
 for j := 0; j < size; j++ {  
 result[i] += temp[i][j]  
 }  
 }  
  
 return result  
}  
  
// scalar multiple of vector line with vector column  
func dot(first, second []float64) float64 {  
 var result float64 = 0  
 size := len(first)  
  
 for i := 0; i < size; i++ {  
 result += first[i] \* second[i]  
 }  
  
 return result  
}  
  
// matrix multiple of vector column with vector line  
func cross(first, second []float64) [][]float64 {  
 size := len(first)  
  
 result := make([][]float64, size)  
  
 for i := 0; i < size; i++ {  
 result[i] = make([]float64, size)  
 for j := 0; j < size; j++ {  
 result[i][j] = first[i]\*second[j]  
 }  
 }  
  
 return result  
}  
  
type Printer interface {  
 showMatrix(matrix [][]float64)  
 showVectorHorizontal(vector []float64)  
 showVectorVertical(vector []float64)  
 Println(v ...interface{})  
}  
  
type printer struct {  
 log \*log2.Logger  
}  
  
func (c \*printer) Println(v ...interface{}) {  
 c.log.Println(v...)  
}  
  
func (c \*printer) showMatrix(matrix [][]float64) {  
 size := len(matrix)  
  
 for i := 0; i < size; i++ {  
 for j := 0; j < size; j++ {  
 c.log.Print(matrix[i][j], " ")  
 }  
 c.log.Println()  
 }  
 c.log.Println()  
}  
  
func (c \*printer) showVectorHorizontal(vector []float64) {  
 size := len(vector)  
  
 for i := 0; i < size; i++ {  
 c.log.Println(vector[i])  
 }  
  
 c.log.Println()  
}  
  
func (c \*printer) showVectorVertical(vector []float64) {  
 size := len(vector)  
  
 for i := 0; i < size; i++ {  
 c.log.Print(vector[i], " ")  
 }  
  
 c.log.Println()  
}

**Модуль послідовного обчислення виразу:**

package main  
  
import "sync"  
  
func main() {  
 data := getStartData()  
  
 data.process()  
}  
  
func (s \*startData) process() {  
 firstStepResult := s.firstStep()  
  
 secondStepResult := s.secondStep(firstStepResult)  
 thirdStepResult := s.thirdStep(secondStepResult)  
 fourthStepResult := s.fourthStep(thirdStepResult)  
 fifthStepResult := s.fifthStep(fourthStepResult)  
 sixthStepResult := s.sixthStep(fifthStepResult)  
 seventhStepResult := s.seventhStep(sixthStepResult)  
  
 s.printer.Println("Result(y'1\*(y2\*y'1+Y3^2+y'2\*y1\*Y3)\*y2): ", seventhStepResult)  
}  
  
type firstStepResult struct {  
 // B2 + C2  
 B2PlusC2 [][]float64  
 // b1 + 19c1  
 b1PlusC1 []float64  
}  
  
func (s \*startData) firstStep() \*firstStepResult {  
 result := &firstStepResult{}  
  
 wg := &sync.WaitGroup{}  
  
 wg.Add(2)  
  
 go func() {  
 result.B2PlusC2 = addMatrices(s.matrixB2, s.matrixC2)  
 wg.Done()  
 }()  
  
 go func() {  
 multipliedC1 := multipleVectorByNumber(s.vectorC1, 19)  
 result.b1PlusC1 = addVectors(s.vectorB1, multipliedC1)  
 wg.Done()  
 }()  
  
 wg.Wait()  
  
 s.printer.Println("B2 + C2")  
 s.printer.showMatrix(result.B2PlusC2)  
  
 s.printer.Println("b1 + 19c1")  
 s.printer.showVectorVertical(result.b1PlusC1)  
  
 return result  
}  
  
type secondStepResult struct {  
 // A2(B2+C2)  
 matrixY3 [][]float64  
 // A1(b1 + 19c1)  
 vectorY2 []float64  
 // A\*b  
 vectorY []float64  
}  
  
func (s \*startData) secondStep(firstStepResult \*firstStepResult) \*secondStepResult {  
 result := &secondStepResult{}  
  
 wg := &sync.WaitGroup{}  
  
 wg.Add(3)  
  
 go func() {  
 result.matrixY3 = multipleMatrices(firstStepResult.B2PlusC2, s.matrixA2)  
 wg.Done()  
 }()  
  
 go func() {  
 result.vectorY2 = multipleMatrixByVector(s.matrixA1, firstStepResult.b1PlusC1)  
 wg.Done()  
 }()  
  
 go func() {  
 result.vectorY = multipleMatrixByVector(s.matrixA, s.vectorB)  
 wg.Done()  
 }()  
  
 wg.Wait()  
  
 s.printer.Println("A2(B2+C2)")  
 s.printer.showMatrix(result.matrixY3)  
  
 s.printer.Println("A1(b1 + 19c1)")  
 s.printer.showVectorVertical(result.vectorY2)  
  
 s.printer.Println("A\*b")  
 s.printer.showVectorVertical(result.vectorY)  
  
 return result  
}  
  
type thirdStepResult struct {  
 // Y3^2  
 matrixY3Square [][]float64  
 // y'2\*y1  
 transpY2MultipleY1 float64  
 // y2\*y'1  
 Y2MultipleTranspY1 [][]float64  
  
 // A2(B2+C2)  
 matrixY3 [][]float64  
 // A\*b  
 vectorY []float64  
 // A1(b1 + 19c1)  
 vectorY2 []float64  
}  
  
func (s \*startData) thirdStep(secondStepResult \*secondStepResult) \*thirdStepResult {  
 result := &thirdStepResult{}  
  
 wg := &sync.WaitGroup{}  
  
 wg.Add(3)  
  
 go func() {  
 result.matrixY3Square = multipleMatrices(secondStepResult.matrixY3, secondStepResult.matrixY3)  
 wg.Done()  
 }()  
  
 go func() {  
 result.transpY2MultipleY1 = dot(secondStepResult.vectorY2, secondStepResult.vectorY)  
 wg.Done()  
 }()  
  
 go func() {  
 result.Y2MultipleTranspY1 = cross(secondStepResult.vectorY2, secondStepResult.vectorY)  
 wg.Done()  
 }()  
  
 wg.Wait()  
  
 result.matrixY3 = secondStepResult.matrixY3  
 result.vectorY = secondStepResult.vectorY  
 result.vectorY2 = secondStepResult.vectorY2  
  
 s.printer.Println("Y3^2")  
 s.printer.showMatrix(result.matrixY3Square)  
  
 s.printer.Println("y'2\*y1")  
 s.printer.Println(result.transpY2MultipleY1)  
  
 s.printer.Println("y2\*y'1")  
 s.printer.showMatrix(result.Y2MultipleTranspY1)  
  
 return result  
}  
  
type fourthStepResult struct {  
 // y'2\*y1\*Y3 number \* matrix  
 res1 [][]float64  
 // y2\*y'1+Y3^2 matrix\*matrix  
 res2 [][]float64  
 // A\*b  
 vectorY []float64  
 // A1(b1 + 19c1)  
 vectorY2 []float64  
}  
  
func (s \*startData) fourthStep(thirdStepResult \*thirdStepResult) \*fourthStepResult {  
 result := &fourthStepResult{}  
  
 wg := &sync.WaitGroup{}  
  
 wg.Add(2)  
  
 go func() {  
 result.res1 = multipleMatrixByNumber(thirdStepResult.matrixY3, thirdStepResult.transpY2MultipleY1)  
 wg.Done()  
 }()  
  
 go func() {  
 result.res2 = addMatrices(thirdStepResult.matrixY3Square, thirdStepResult.Y2MultipleTranspY1)  
 wg.Done()  
 }()  
  
 wg.Wait()  
  
 result.vectorY = thirdStepResult.vectorY  
 result.vectorY2 = thirdStepResult.vectorY2  
  
 s.printer.Println("y'2\*y1\*Y3 number \* matrix")  
 s.printer.showMatrix(result.res1)  
  
 s.printer.Println("y2\*y'1+Y3^2 matrix\*matrix")  
 s.printer.showMatrix(result.res2)  
  
 return result  
}  
  
type fifthStepResult struct {  
 // A\*b  
 vectorY []float64  
 // y2\*y'1+Y3^2+y'2\*y1\*Y3  
 res1 [][]float64  
 // A1(b1 + 19c1)  
 vectorY2 []float64  
}  
  
func (s \*startData) fifthStep(fourthStepResult \*fourthStepResult) \*fifthStepResult {  
 result := &fifthStepResult{}  
  
 result.vectorY = fourthStepResult.vectorY  
 result.res1 = addMatrices(fourthStepResult.res2, fourthStepResult.res1)  
 result.vectorY2 = fourthStepResult.vectorY2  
  
 s.printer.Println("y2\*y'1+Y3^2+y'2\*y1\*Y3")  
 s.printer.showMatrix(result.res1)  
  
 return result  
}  
// 1 2 3  
type sixthStepResult struct {  
 // A1(b1 + 19c1)  
 vectorY2 []float64  
 // y'1(y2\*y'1+Y3^2+y'2\*y1\*Y3)  
 res1 []float64  
}  
  
func (s \*startData) sixthStep(fifthStepResult \*fifthStepResult) \*sixthStepResult {  
 result := &sixthStepResult{}  
  
 result.vectorY2 = fifthStepResult.vectorY2  
 result.res1 = multipleMatrixByVector(fifthStepResult.res1, fifthStepResult.vectorY)  
  
 s.printer.Println("y'1(y2\*y'1+Y3^2+y'2\*y1\*Y3)")  
 s.printer.showVectorHorizontal(result.res1)  
  
 return result  
}  
  
func (s \*startData) seventhStep(result \*sixthStepResult) float64 {  
 // y'1\*(y2\*y'1+Y3^2+y'2\*y1\*Y3)\*y2  
 return dot(result.res1, result.vectorY2)  
}