Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Теоретическая информатика и компьютерные технологии»

Лабораторная работа № 8

по курсу «Численные методы линейной алгебры»

«Изучение эффективности модифицированных методов быстрого умножения матриц.»

Студент группы ИУ9-71Б Окутин Д.А.

Преподаватель Посевин Д. П.

Москва 2024

1 Задание

1. На основе уже реализованного в лабораторной работа No 7 метода Штрассена необходимо выполнить его многопоточную версию.

2. Реализовать классическую версию метода Винограда-Штрассена.

3 Реализовать многопоточную версию метода Винограда-Штрассена.

4 Сравнить скорости умножения матриц всеми реализованными методами: в лабораторной работе No7 и лабораторной работе No8.

5. Построить на одной координатной плоскости разными цветами графики зависимости времени T (мс) умножения двух матриц размера N x N стандартным алгоритмом, алгоритмом Винограда, методом Штрассена, многопоточным методом Штрассена, методом Винограда-Штрассена, многопоточным методом Винограда-Штрассена от размера матрицы N. N изменяется от 2 до 1024. Выбор значения размера матрицы N необходимо осуществлять по формуле N= 2 при этом изменяя пот 1до 10 строго. Во время построения графиков на координатной плоскости на горизонтальной оси необходимо выводить именно значение N, а на вертикальной оси время, которое затрачено на выполнение алгоритма перемножения матриц строго в миллисекундах.

2 Реализация

Исходный код программмы представлен в листинге 1.

Листинг 1: code

```
1
2  using Random
3  using LinearAlgebra
4  using PyPlot
5  using Distributed
6
7  function generate_matrix(l::Int, r::Int, n::Int)
8      return rand(n, n) .* (r - l) .+ l
9  end
10
11 addprocs(7)
12
13 @everywhere function naive_matrix_multiply(A, B)
14     A = copy(A)
15     B = copy(B)
16     n, m = size(A)
17     m2, p = size(B)
18
19     if m != m2
20         print("
                          -                                    ")
21     end
22
23     C = zeros(n, p)
24     for i in 1:n
25         for j in 1:p
26             for k in 1:m
27                 C[i, j] += A[i, k] * B[k, j]
28             end
29         end
30     end
31
32     return C
33 end
34
35 A = [1 2 3 4; 5 6 7 8; 9 10 11 12; 13 14 15 16]
36 B = [1 2 3 4; 5 6 7 8; 9 10 11 12; 13 14 15 16]
37
38 C = naive_matrix_multiply(A, B)
39 println(C)
40
```

```
function vinograd(A, B)
    m, n = size(A)
    n2, p = size(B)

    if n != n2
        print("
                        -                                    ")
    end

    C = zeros(Float64, m, p)
    row_factors = zeros(Float64, m)
    col_factors = zeros(Float64, p)

    for i in 1:m
        row_factors[i] = sum(A[i, 2k-1] * A[i, 2k] for k in 1:div(n, 2))
    end

    for j in 1:p
        col_factors[j] = sum(B[2k-1, j] * B[2k, j] for k in 1:div(n, 2))
    end

    for i in 1:m
        for j in 1:p
            s = -row_factors[i] - col_factors[j]
            C[i, j] = s + sum((A[i, 2k-1] + B[2k, j]) * (A[i, 2k] + B[2k
    -1, j]) for k in 1:div(n, 2))
        end
    end

    return C
end

#                                          :
A = [1 2 3 4; 5 6 7 8; 9 10 11 12; 13 14 15 16]
B = [1 2 3 4; 5 6 7 8; 9 10 11 12; 13 14 15 16]
C = vinograd(A, B)

println(C)

function strassen_vinograd_main(A,B)
    # if size(A, 1) > 64
    #     return strassen(A,B,64)
    # end

    return strassen_vinograd(A,B,64)
end
```

```
85
86 @everywhere function strassen_vinograd(A, B, bound)
87     n = size(A, 1)
88     if n <= bound
89         return naive_matrix_multiply(A,B)  #
90     end
91
92     mid = n     2
93
94     #
95     A11 = A[1:mid, 1:mid]
96     A12 = A[1:mid, mid + 1:end]
97     A21 = A[mid + 1:end, 1:mid]
98     A22 = A[mid + 1:end, mid + 1:end]
99
100    B11 = B[1:mid, 1:mid]
101    B12 = B[1:mid, mid + 1:end]
102    B21 = B[mid + 1:end, 1:mid]
103    B22 = B[mid + 1:end, mid + 1:end]
104
105    S1=A21+A22
106    S2=S1-A11
107    S3=A11-A21
108    S4=A12-S2
109    S5=B12-B11
110    S6=B22-S5
111    S7=B22-B12
112    S8=S6-B21
113    #
114    M1 = strassen_vinograd(S2, S6,bound)
115    M2 = strassen_vinograd(A11, B11,bound)
116    M3 = strassen_vinograd(A12, B21,bound)
117    M4 = strassen_vinograd(S3, S7,bound)
118    M5 = strassen_vinograd(S1, S5,bound)
119    M6 = strassen_vinograd(S4, B22,bound)
120    M7 = strassen_vinograd(A22, S8,bound)
121
122    M1 = fetch(M1)
123    M2 = fetch(M2)
124    M3 = fetch(M3)
125    M4 = fetch(M4)
126    M5 = fetch(M5)
127    M6 = fetch(M6)
128    M7 = fetch(M7)
129
```

```
130     #
131     T1=M1+M2
132     T2=T1+M4
133     C11 = M2 + M3
134     C12 = T1 + M5 + M6
135     C21 = T2-M7
136     C22 = T2+M5
137
138     #

139     C = zeros(n, n)
140     C[1:mid, 1:mid] = C11
141     C[1:mid, mid+1:end] = C12
142     C[mid+1:end, 1:mid] = C21
143     C[mid+1:end, mid+1:end] = C22
144
145     return C
146 end
147
148 A = [1 2 3 4; 5 6 7 8; 9 10 11 12; 13 14 15 16]
149 B = [1 2 3 4; 5 6 7 8; 9 10 11 12; 13 14 15 16]
150
151 C = strassen_vinograd(A, B, 64)
152 println(C)
153
154 function strassen_main(A,B)
155     # if size(A, 1) > 64
156     #     return strassen(A,B,64)
157     # end
158
159     return strassen(A,B,64)
160 end
161
162 @everywhere function strassen(A, B, bound)
163     A = copy(A)
164     B = copy(B)
165     n = size(A, 1)
166
167     if n <= bound
168         return naive_matrix_multiply(A,B)
169     end
170
171     mid = div(n, 2)
172     A11 = A[1:mid, 1:mid]
173     A12 = A[1:mid, mid+1:end]
174     A21 = A[mid+1:end, 1:mid]
```

```
175     A22 = A[mid+1:end, mid+1:end]
176
177     B11 = B[1:mid, 1:mid]
178     B12 = B[1:mid, mid+1:end]
179     B21 = B[mid+1:end, 1:mid]
180     B22 = B[mid+1:end, mid+1:end]
181
182     P1 = strassen(A11 + A22, B11 + B22, bound)
183     P2 = strassen(A21 + A22, B11, bound)
184     P3 = strassen(A11, B12 - B22, bound)
185     P4 = strassen(A22, B21 - B11,bound)
186     P5 = strassen(A11 + A12, B22,bound)
187     P6 = strassen(A21 - A11, B11 + B12,bound)
188     P7 = strassen(A12 - A22, B21 + B22,bound)
189
190     C11 = P1 + P4 - P5 + P7
191     C12 = P3 + P5
192     C21 = P2 + P4
193     C22 = P1 - P2 + P3 + P6
194
195     #

196     C = zeros(n, n)
197     C[1:mid, 1:mid] = C11
198     C[1:mid, mid+1:end] = C12
199     C[mid+1:end, 1:mid] = C21
200     C[mid+1:end, mid+1:end] = C22
201
202     return C
203 end
204
205 A = [1 2 3 4; 5 6 7 8; 9 10 11 12; 13 14 15 16]
206 B = [1 2 3 4; 5 6 7 8; 9 10 11 12; 13 14 15 16]
207
208 C = strassen(A, B, true)
209 println(C)
210
211
212 function strassen_main_multi(A,B)
213     # if size(A, 1) > 64
214     #     return strassen(A,B,64)
215     # end
216
217     return strassen_multi(A,B,64)
218 end
219
```

```
220 @everywhere function strassen_multi(A, B, bound)
221     A = copy(A)
222     B = copy(B)
223     n = size(A, 1)
224 
225     if n <= bound
226         return naive_matrix_multiply(A,B)
227     end
228 
229     mid = div(n, 2)
230     A11 = A[1:mid, 1:mid]
231     A12 = A[1:mid, mid+1:end]
232     A21 = A[mid+1:end, 1:mid]
233     A22 = A[mid+1:end, mid+1:end]
234 
235     B11 = B[1:mid, 1:mid]
236     B12 = B[1:mid, mid+1:end]
237     B21 = B[mid+1:end, 1:mid]
238     B22 = B[mid+1:end, mid+1:end]
239 
240     P1 = @spawn strassen(A11 + A22, B11 + B22, bound)
241     P2 = @spawn strassen(A21 + A22, B11, bound)
242     P3 = @spawn strassen(A11, B12 - B22, bound)
243     P4 = @spawn strassen(A22, B21 - B11,bound)
244     P5 = @spawn strassen(A11 + A12, B22,bound)
245     P6 = @spawn strassen(A21 - A11, B11 + B12,bound)
246     P7 = @spawn strassen(A12 - A22, B21 + B22,bound)
247 
248     P1 = fetch(P1)
249     P2 = fetch(P2)
250     P3 = fetch(P3)
251     P4 = fetch(P4)
252     P5 = fetch(P5)
253     P6 = fetch(P6)
254     P7 = fetch(P7)
255 
256     C11 = P1 + P4 - P5 + P7
257     C12 = P3 + P5
258     C21 = P2 + P4
259     C22 = P1 - P2 + P3 + P6
260 
261     #

262     C = zeros(n, n)
263     C[1:mid, 1:mid] = C11
264     C[1:mid, mid+1:end] = C12
```

```
265     C[mid+1:end, 1:mid] = C21
266     C[mid+1:end, mid+1:end] = C22
267
268     return C
269 end
270
271 A = [1 2 3 4; 5 6 7 8; 9 10 11 12; 13 14 15 16]
272 B = [1 2 3 4; 5 6 7 8; 9 10 11 12; 13 14 15 16]
273
274 C = strassen_main_multi(A, B)
275 println(C)
276
277 function strassen_vinograd_main_multi(A,B)
278     # if size(A, 1) > 64
279     #     return strassen(A,B,64)
280     # end
281
282     return strassen_vinograd_multi(A,B,64)
283 end
284
285 function strassen_vinograd_multi(A, B, bound)
286     n = size(A, 1)
287     if n <= bound
288         return naive_matrix_multiply(A,B)  #
289     end
290
291     mid = n     2
292
293     #
294     A11 = A[1:mid, 1:mid]
295     A12 = A[1:mid, mid + 1:end]
296     A21 = A[mid + 1:end, 1:mid]
297     A22 = A[mid + 1:end, mid + 1:end]
298
299     B11 = B[1:mid, 1:mid]
300     B12 = B[1:mid, mid + 1:end]
301     B21 = B[mid + 1:end, 1:mid]
302     B22 = B[mid + 1:end, mid + 1:end]
303
304     S1=A21+A22
305     S2=S1-A11
306     S3=A11-A21
307     S4=A12-S2
308     S5=B12-B11
309     S6=B22-S5
```

```
310         S7=B22-B12
311         S8=S6-B21
312
313         #
314         M1 = @spawn strassen_vinograd(S2, S6,bound)
315         M2 = @spawn strassen_vinograd(A11, B11,bound)
316         M3 = @spawn strassen_vinograd(A12, B21,bound)
317         M4 = @spawn strassen_vinograd(S3, S7,bound)
318         M5 = @spawn strassen_vinograd(S1, S5,bound)
319         M6 = @spawn strassen_vinograd(S4, B22,bound)
320         M7 = @spawn strassen_vinograd(A22, S8,bound)
321
322         M1 = fetch(M1)
323         M2 = fetch(M2)
324         M3 = fetch(M3)
325         M4 = fetch(M4)
326         M5 = fetch(M5)
327         M6 = fetch(M6)
328         M7 = fetch(M7)
329
330         #
331         T1=M1+M2
332         T2=T1+M4
333         C11 = M2 + M3
334         C12 = T1 + M5 + M6
335         C21 = T2-M7
336         C22 = T2+M5
337
338         #

339         C = zeros(n, n)
340         C[1:mid, 1:mid] = C11
341         C[1:mid, mid+1:end] = C12
342         C[mid+1:end, 1:mid] = C21
343         C[mid+1:end, mid+1:end] = C22
344
345         return C
346 end
347
348 A = [1 2 3 4; 5 6 7 8; 9 10 11 12; 13 14 15 16]
349 B = [1 2 3 4; 5 6 7 8; 9 10 11 12; 13 14 15 16]
350
351 C = strassen_vinograd_main_multi(A, B)
352 println(C)
353
354 n = [2^i for i in 1:10]
```

```
time1 = Float64[]
time2 = Float64[]
time3 = Float64[]
time4 = Float64[]
time5 = Float64[]
time6 = Float64[]

strassen_bound = false

for dim in n
    println(dim)
    A = generate_matrix(-10, 10, dim)
    B = generate_matrix(-10, 10, dim)

    ab = A*B

    t = time()
    C1 = naive_matrix_multiply(A, B)
    push!(time1, time() - t)
    if !isapprox(ab,C1)
        println("error1")
    end

    t = time()
    C2 = strassen_main(A, B)
    push!(time2, time() - t)
    if !isapprox(ab,C2)
        println("error2")
    end

    t = time()
    C3 = vinograd(A, B)
    push!(time3, time() - t)
    if !isapprox(ab,C3)
        println("error3")
    end

    t = time()
    C4 = strassen_main_multi(A, B)
    push!(time4, time() - t)
    if !isapprox(ab,C4)
        println("error4")
    end

    t = time()
    C5 = strassen_vinograd_main(A, B)
```

```
401     if !isapprox(A*B,C5)
402         println("error5")
403     end
404     push!(time5, time() - t)
405
406     t = time()
407     C6 = strassen_vinograd_main_multi(A, B)
408     if !isapprox(A*B,C6)
409         println("error6")
410     end
411     push!(time6, time() - t)
412 end
413
414 PyPlot.figure(figsize=(6, 5))
415 PyPlot.title("
                  ")
416 PyPlot.xlabel("n")
417 PyPlot.ylabel("time")
418 PyPlot.plot(n, time1, label="Ordinary")
419 PyPlot.plot(n, time2, label="Strassen")
420 PyPlot.plot(n, time3, label="Vinograd")
421 PyPlot.plot(n, time4, label="Strassen Multi")
422 PyPlot.plot(n, time5, label="Vinograd-Strassen")
423 PyPlot.plot(n, time6, label="Vinograd-Strassen-Multi")
424 PyPlot.grid()
425 PyPlot.legend()
426 PyPlot.show()
```

3 Результаты

Результаты запуска представлены на рисунках 1.

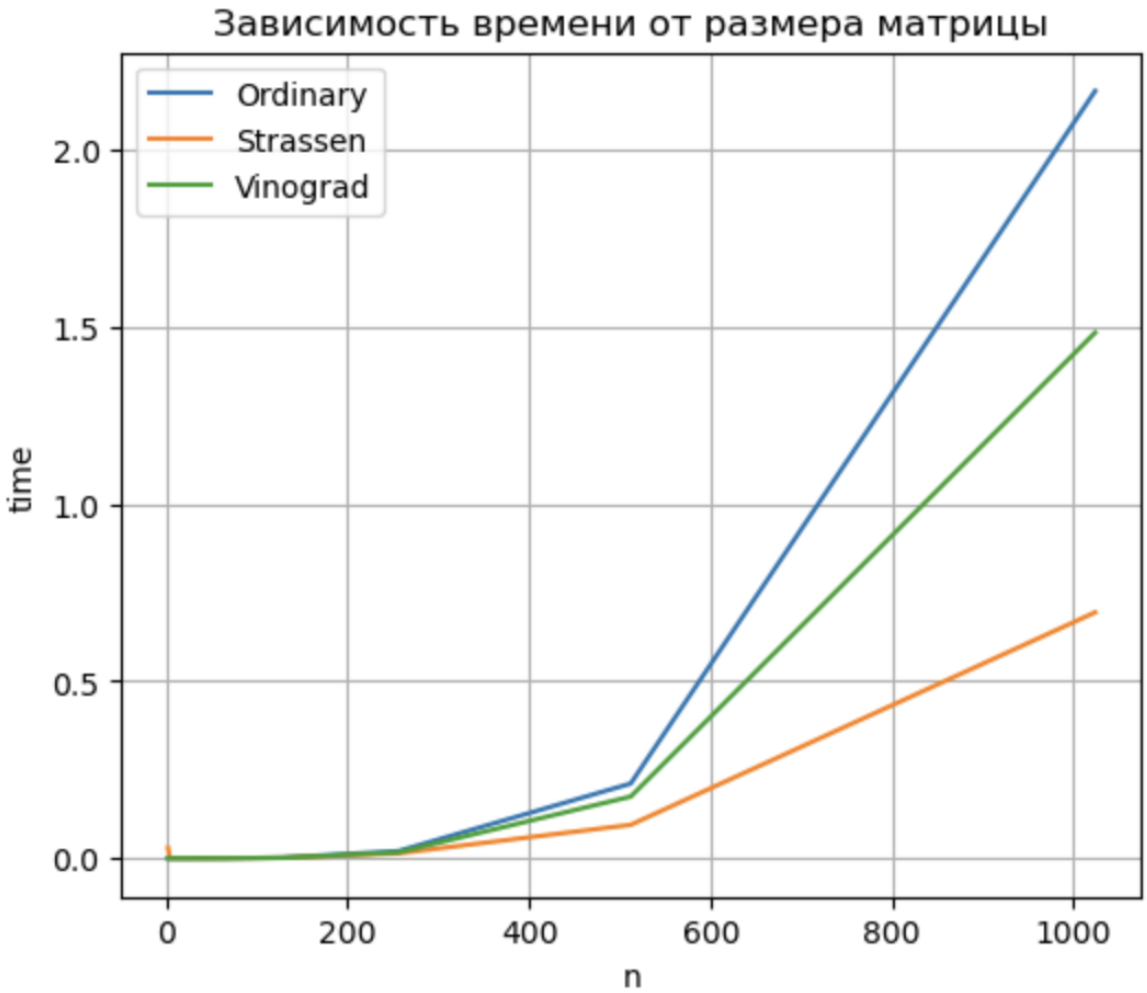

Рис. 1 — Результаты

4 Выводы

В результате выполнения данной лабораторной работы был реализован алгоритм Винограда и метод Штрассена и Винограда-Штрассена. Также реализована многопоточность. Корректность доказана и произведен замер производительности.