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
«Изучение эффективности модифицированных методов быстрого умножения матриц»

по курсу
«Численные методы линейной алгебры»

Выполнил:
студент группы ИУ9-72Б
Караник А.А.

Проверено:
Посевин Д.П.

Москва, 2024

Цель работы

Реализация и исследование эффективности модификаций методов Штрассена и Винограда-Штрассена быстрого умножения матриц.

Реализация

Исходный код программы:

```
using LinearAlgebra
using Random
using Statistics
using Plots
using Base.Threads

function standard_multiply(A, B)
    m, n = size(A)
    nB, p = size(B)

    if n != nB
        error("Матрицы несовместимы для умножения")
    end

    C = zeros(Float64, m, p)

    for i in 1:m
        for j in 1:p
            for k in 1:n
                C[i, j] += A[i, k] * B[k, j]
            end
        end
    end

    return C
end

function winograd_multiply(A, B)
    m, n = size(A)
    nB, p = size(B)
    C = zeros(Float64, m, p)
    if n != nB
        error("Матрицы должны быть совместимы для умножения")
    end

    rowFactor = [sum(A[i, 1:2:n-1] .* A[i, 2:2:n]) for i in 1:m]
    colFactor = [sum(B[1:2:n-1, j] .* B[2:2:n, j]) for j in 1:p]

    for i in 1:m
        for j in 1:p
            C[i, j] = -rowFactor[i] - colFactor[j]
            for k in 1:2:n-1
                C[i, j] += (A[i, k] + B[k + 1, j]) * (A[i, k + 1] + B[k, j])
            end
        end
    end

    if isodd(n)
        for i in 1:m
            for j in 1:p
                C[i, j] += A[i, n] * B[n, j]
            end
        end
    end
```

```
    return C
end

function strassen_multiply(A, B)
    n = size(A, 1)

    if n <= 64
        return standard_multiply(A, B)
    end

    m = div(n, 2)
    A11 = A[1:m, 1:m]
    A12 = A[1:m, m+1:end]
    A21 = A[m+1:end, 1:m]
    A22 = A[m+1:end, m+1:end]
    B11 = B[1:m, 1:m]
    B12 = B[1:m, m+1:end]
    B21 = B[m+1:end, 1:m]
    B22 = B[m+1:end, m+1:end]

    M1 = strassen_multiply(A11 + A22, B11 + B22)
    M2 = strassen_multiply(A21 + A22, B11)
    M3 = strassen_multiply(A11, B12 - B22)
    M4 = strassen_multiply(A22, B21 - B11)
    M5 = strassen_multiply(A11 + A12, B22)
    M6 = strassen_multiply(A21 - A11, B11 + B12)
    M7 = strassen_multiply(A12 - A22, B21 + B22)

    C11 = M1 + M4 - M5 + M7
    C12 = M3 + M5
    C21 = M2 + M4
    C22 = M1 - M2 + M3 + M6

    C = [C11 C12; C21 C22]

    return C
end

function strassen_multiply2(A, B)
    n = size(A, 1)

    if n <= 64
        return standard_multiply(A, B)
    end

    m = div(n, 2)

    A11 = A[1:m, 1:m]
    A12 = A[1:m, m+1:end]
    A21 = A[m+1:end, 1:m]
    A22 = A[m+1:end, m+1:end]
    B11 = B[1:m, 1:m]
    B12 = B[1:m, m+1:end]
    B21 = B[m+1:end, 1:m]
    B22 = B[m+1:end, m+1:end]

    M = Vector{Any}(undef, 7)

    @threads for i in 1:7
        if i == 1
            M[1] = strassen_multiply2(A11 + A22, B11 + B22)
        elseif i == 2
            M[2] = strassen_multiply2(A21 + A22, B11)
        elseif i == 3
```

```
            M[3] = strassen_multiply2(A11, B12 - B22)
        elseif i == 4
            M[4] = strassen_multiply2(A22, B21 - B11)
        elseif i == 5
            M[5] = strassen_multiply2(A11 + A12, B22)
        elseif i == 6
            M[6] = strassen_multiply2(A21 - A11, B11 + B12)
        elseif i == 7
            M[7] = strassen_multiply2(A12 - A22, B21 + B22)
        end
    end

    M1, M2, M3, M4, M5, M6, M7 = M

    C11 = M1 + M4 - M5 + M7
    C12 = M3 + M5
    C21 = M2 + M4
    C22 = M1 - M2 + M3 + M6

    C = [C11 C12; C21 C22]

    return C
end

function strassen_modified(A, B)
    n = size(A, 1)

    if n <= 64
        return standard_multiply(A, B)
    end

    m = div(n, 2)

    A11 = A[1:m, 1:m]
    A12 = A[1:m, m+1:end]
    A21 = A[m+1:end, 1:m]
    A22 = A[m+1:end, m+1:end]
    B11 = B[1:m, 1:m]
    B12 = B[1:m, m+1:end]
    B21 = B[m+1:end, 1:m]
    B22 = B[m+1:end, m+1:end]

    S1 = A11 + A22
    S2 = B11 + B22
    P1 = strassen_modified(S1, S2)
    S3 = A21 + A22
    P2 = strassen_modified(S3, B11)
    S4 = A11
    P3 = strassen_modified(S4, B12 - B22)
    S5 = A22
    P4 = strassen_modified(S5, B21 - B11)
    S6 = A11 + A12
    P5 = strassen_modified(S6, B22)
    S7 = A21 - A11
    P6 = strassen_modified(S7, B11 + B12)
    S8 = A12 - A22
    P7 = strassen_modified(S8, B21 + B22)

    C11 = P1 + P4 - P5 + P7
    C12 = P3 + P5
    C21 = P2 + P4
    C22 = P1 + P3 - P2 + P6

    C = [C11 C12; C21 C22]
```

```
    return C
end

function strassen_modified2(A, B)
    n = size(A, 1)

    if n <= 64
        return standard_multiply(A, B)
    end

    m = div(n, 2)

    A11 = A[1:m, 1:m]
    A12 = A[1:m, m+1:end]
    A21 = A[m+1:end, 1:m]
    A22 = A[m+1:end, m+1:end]
    B11 = B[1:m, 1:m]
    B12 = B[1:m, m+1:end]
    B21 = B[m+1:end, 1:m]
    B22 = B[m+1:end, m+1:end]

    S1 = A11 + A22
    S2 = B11 + B22
    S3 = A21 + A22
    S4 = A11
    S5 = A22
    S6 = A11 + A12
    S7 = A21 - A11
    S8 = A12 - A22

    P = Vector{Any}(undef, 7)

    @threads for i in 1:7
        if i == 1
            P[1] = strassen_modified2(S1, S2)
        elseif i == 2
            P[2] = strassen_modified2(S3, B11)
        elseif i == 3
            P[3] = strassen_modified2(S4, B12 - B22)
        elseif i == 4
            P[4] = strassen_modified2(S5, B21 - B11)
        elseif i == 5
            P[5] = strassen_modified2(S6, B22)
        elseif i == 6
            P[6] = strassen_modified2(S7, B11 + B12)
        elseif i == 7
            P[7] = strassen_modified2(S8, B21 + B22)
        end
    end

    P1, P2, P3, P4, P5, P6, P7 = P

    C11 = P1 + P4 - P5 + P7
    C12 = P3 + P5
    C21 = P2 + P4
    C22 = P1 + P3 - P2 + P6

    C = [C11 C12; C21 C22]

    return C
end

function benchmark_algorithms()
    ns = 1:1:10
    Ns = []
```

```
    standard_times = []
    winograd_times = []
    strassen_times = []
    strassen_times2 = []
    strassen_modified_times = []
    strassen_modified_times2 = []

    for n in ns
        N = 2^n
        push!(Ns, N)
        A = rand(N, N)
        B = rand(N, N)

        t = @elapsed standard_multiply(A, B) * 1000
        push!(standard_times, t)

        t = @elapsed winograd_multiply(A, B) * 1000
        push!(winograd_times, t)

        t = @elapsed strassen_multiply(A, B) * 1000
        push!(strassen_times, t)

        t = @elapsed strassen_multiply2(A, B) * 1000
        push!(strassen_times2, t)

        t = @elapsed strassen_modified(A, B) * 1000
        push!(strassen_modified_times, t)

        t = @elapsed strassen_modified2(A, B) * 1000
        push!(strassen_modified_times2, t)
    end

    plot(Ns, standard_times, label="Стандартное умножение", lw=2)
    plot!(Ns, winograd_times, label="Алгоритм Винограда", lw=2)
    plot!(Ns, strassen_times, label="Метод Штрассена", lw=2)
    plot!(Ns, strassen_times2, label="Метод Штрассена (многопоточный)", lw=2)
    plot!(Ns, strassen_modified_times, label="Метод Винограда-Штрассена", lw=2)
    plot!(Ns, strassen_modified_times2, label="Метод Винограда-Штрассена
    ↪  (многопоточный)", lw=2)
    xlabel!("Размер матрицы (N)")
    ylabel!("Время (ms)")
    title!("Сравнение времени выполнения алгоритмов умножения")
end

benchmark_algorithms()
```

Результаты

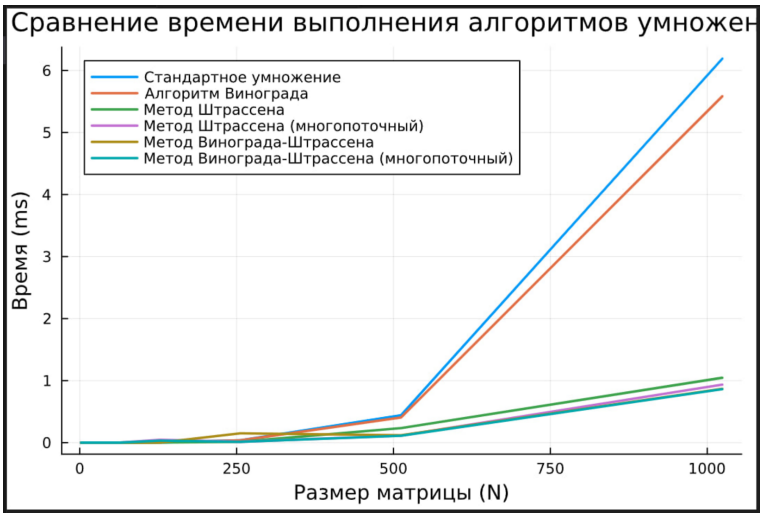

Рис. 1: результаты

Вывод

В ходе выполенения лабораторной работы были успешно реализованы и исследованы эффективности модификаций методов Штрассена и Винограда-Штрассена быстрого умножения матриц.