Построение графиков

Лабораторная работа № 5

Шулуужук А. В.

25 октября 2025

Российский университет дружбы народов, Москва, Россия

Цели и задачи

Основной целью работы является изучение возможностей специализированных пакетов Julia для выполнения и оценки эффективности операций над объектами линейной алгебры.

Выполнение лабораторной работы

```
x = 0:0.01:2pi #диапазон
y = sin.(x)
plot(layout=(2, 2), title="y = sin(x)")
plot!(x, y, seriestype=:line, label="Line", subplot=1)
plot!(x, y, seriestype=:scatter, label="Scatter", subplot=2)
histogram!(y, bins=30, label="Histogram", subplot=3)
plot!(x, y, seriestype=:step, label="Step", subplot=4)
```

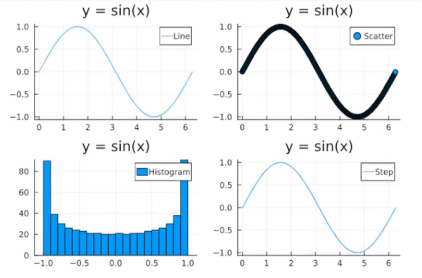

Рис. 1: Построение графиков

```
x = 0:0.01:2pi
r = sin.(x)
line_styles = [:solid, :dash, :dot, :dashdot]

plt = plot()

for ls in line_styles
    plot!(plt, x, r, linestyle=ls, label=string(ls))
end

plot!(plt, xlabel="x", ylabel="y", title="Графики функции с разными стилями")
display(plt)
```

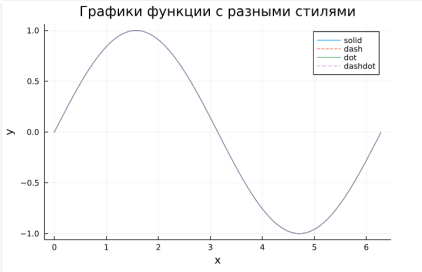

Рис. 2: различные типы

```julia
using LaTeXStrings

y_func(x) = pi * x^2 * log(x)
x = 0.1:0.01:10
plot(x, y_func.(x),
    color = :red,
    label = L"y(x) = \pi x^2 \ln(x)",
    xlabel = "x",
    ylabel = L"y(x)",
    framestyle = :box,
    grid = false,
    xticks = 0:2:10,
    yticks = -50:50:200,
    bordercolor = :green)
```

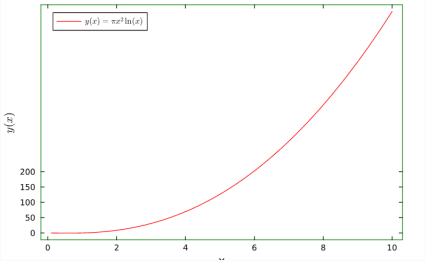

Рис. 3: построение графиокв

```
x = [-2, -1, 0, 1, 2]
y_f(x) = x.^3 .- 3 .* x
y_values = y_f.(x)
plt = plot(layout = (2, 2))
scatter!(plt[1], x, y_values, label = "Точки")
plot!(plt[2], x, y_values, label = "Линии")
plot!(plt[3], x, y_values, label = "Линии и точки")
scatter!(plt[3], x, y_values, label = "")
plot!(plt[4], x, y_values, smooth = true, label = "Кривая")
savefig(plt, "figure_shuluuzhuk.png")
```

```
"C:\\Users\\airan\\figure_shuluuzhuk.png"
```

Рис. 4: построение графиков

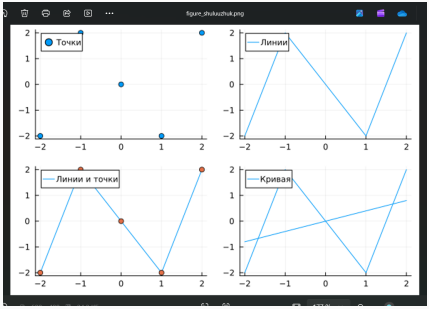

Рис. 5: Вычисление нормы векторов и матриц, повороты, вращения

```
x_1 = 3:0.1:6
y_1(x) = pi * x
y_2(x) = exp(x) * cos(x)

# Сначала создаем основной график
plot(x_1, y_1.(x_1), label="y1(x) = π * x", color=:blue, grid=true, legend=:topright

# Затем добавляем вторую функцию
plot!(x_1, y_2.(x_1), label="y2(x) = exp(x) * cos(x)", color=:red)

# Добавляем подписи осей и заголовок
xlabel!("x")
ylabel!("y")
title!("Графики функций y1(x) и y2(x)")
```

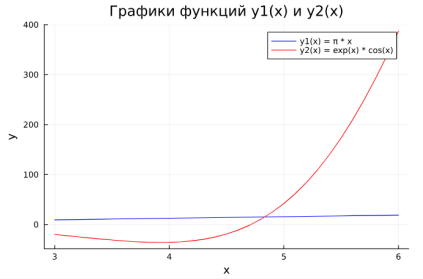

Рис. 6: 5 задание

```julia
# Построение графиков с двумя осями ординат
p = plot(x, y1(x), label="y1(x) = πx", color=:blue, linewidth=2, grid=true, legend=:to
plot!(p, x, y2(x), label="y2(x) = exp(x) * cos(x)", color=:red, linewidth=2)
# Добавляем вторую ось ординат для y2
plot!(p, secondary=true)
# Заголовок и сетка
xlabel!("x")
ylabel!("y1 (primary)", fontsize=10)
ylabel!(p, "y2 (secondary)", fontsize=10)
title!("Графики с двумя осями ординат")
```

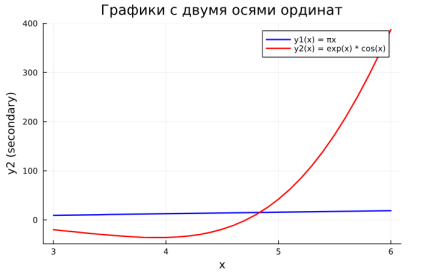

Рис. 7: 5 задание

```julia
using Plots

# Данные по кинетике реакции (концентрация от времени)
time = 0:1:30  # время в минутах
concentration = [1.00, 0.85, 0.72, 0.61, 0.52, 0.44, 0.38, 0.32, 0.27, 0.23, 0.20, 0.17, 0.14,

# Относительные ошибки измерения (5%)
relative_errors = 0.05 .* concentration
absolute_errors = relative_errors

# Построение графика
plot(time, concentration,
    ribbon=absolute_errors,
    fillalpha=0.3,
    fillcolor=:lightblue,
    label="Концентрация реагента",
    color=:blue,
    linewidth=2,
    xlabel="Время, мин",
    ylabel="Концентрация, моль/л",
    title="Кинетика химической реакции",
    legend=:topright,
    grid=true)

scatter!(time, concentration,
    label="Экспериментальные точки",
    color=:blue,
    markersize=4)

# Добавляем теоретическую кривую (экспоненциальный распад)
theoretical = exp.(-0.1 .* time)
plot!(time, theoretical,
    label="Теоретическая модель",
    color=:red,
    linestyle=:dash,
    linewidth=2)

annotate!(15, 0.7, text("Погрешность: ±5%", 10, :left))
```

Рис. 8: 6 задание

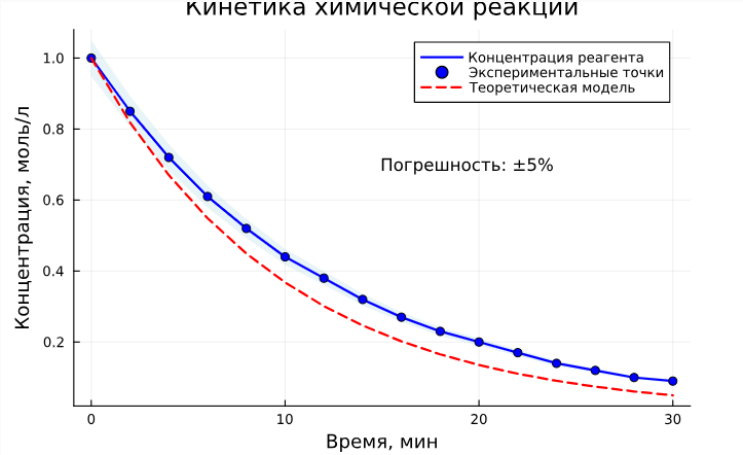

Рис. 9: LU-факторизация

```julia
x = randn(100)  # 100 случайных точек с нормальным распределением
y = randn(100)

# Строим точечный график
scatter(x, y,
    label="Случайные точки",
    color=:blue,
    markersize=6,
    alpha=0.7,
    xlabel="X координата",
    ylabel="Y координата",
    title="Точечный график случайных данных",
    legend=:topright,
    grid=true,
    framestyle=:box)

# Добавляем линию тренда (опционально)
using Statistics
if length(x) > 1
    A = [ones(length(x)) x]
    coeff = A \ y
    trend = A * coeff
    plot!(x, trend, label="Линия тренда", color=:red, linewidth=2)
end
```

Рис. 10: Различные части факторизации

```
y = randn(n)
z = randn(n)

# Строим 3D точечный график
scatter(x, y, z,
    label="Случайные точки",
    color=:blue,
    markersize=4,
    alpha=0.7,
    xlabel="Ось X",
    ylabel="Ось Y",
    zlabel="Ось Z",
    title="3D точечный график случайных данных",
    legend=:topright,
    camera=(45, 30),  # Угол обзора
    marker=:circle)

# Добавляем центр распределения
scatter!([mean(x)], [mean(y)], [mean(z)],
    label="Центр распределения",
    color=:red,
    markersize=8,
    marker=:diamond)
```

Рис. 11: Решение СЛАУ через объект факторизации

```julia
# Создаем анимацию
anim = @animate for n in 1:100
    x_range = range(0, stop=2π * n/100, length=1000)
    y = sin.(x_range)

    plot(x_range, y,
         linewidth=3,
         color=:blue,
         xlabel="x",
         ylabel="sin(x)",
         title="Построение синусоиды: шаг $n/100",
         xlims=(0, 2π),
         ylims=(-1.2, 1.2),
         legend=false,
         grid=true,
         framestyle=:box)

    vline!([2π * n/100], color=:red, linestyle=:dash, linewidth=2)
end

gif(anim, "sine_wave_animation.gif", fps=15)
```

```
[ Info: Saved animation to C:\Users\airan\sine_wave_animation.gif
```

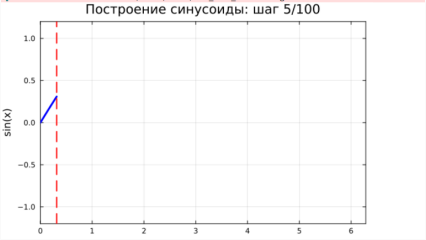

Рис. 12: QR-факторизация

```julia
# Функция для вычисления координат гипоциклоиды
function hypocycloid(R, r, θ)
    x = (R - r) * cos.(θ) + r * cos.((R - r) / r * θ)
    y = (R - r) * sin.(θ) - r * sin.((R - r) / r * θ)
    return x, y
end
# Функция для создания анимации гипоциклоиды для целого значения k
function create_hypocycloid_animation(k)
    R = 10  # Радиус большой окружности
    r = R / k  # Радиус маленькой окружности
    θ = 0:0.05:2 * π  # Углы от 0 до 2π, шаг 0.05
    x, y = hypocycloid(R, r, θ)
    anim = @animate for i in 1:length(θ)  # Итерация по всем углам
        plot(x[1:i], y[1:i], label="Гипоциклоид (k=$k)", color=:blue, linewidth=2,
             xlabel="x", ylabel="y", title="Анимация гипоциклоиды для k=$k", legend=:topright)
    end
    # Сохранение анимации в файл
    gif(anim, "hypocycloid_k_$k.gif", fps=10)
end
# Создание анимации для целого значения k = 2
create_hypocycloid_animation(2)
```

```
[ Info: Saved animation to C:\Users\airan\hypocycloid_k_2.gif
```

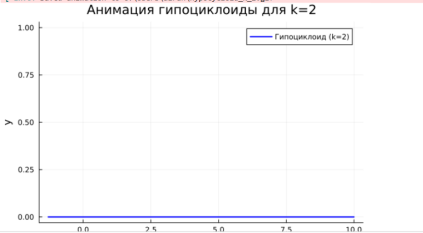

Рис. 13: QR-факторизация

Выводы

Выводы

В результате выполнения лабораторной работы были изучены синтаксис языка Julia для построения графиков