Лабораторная работа № 5

Построение графиков

Старовойтов Егор Сергеевич

Содержание

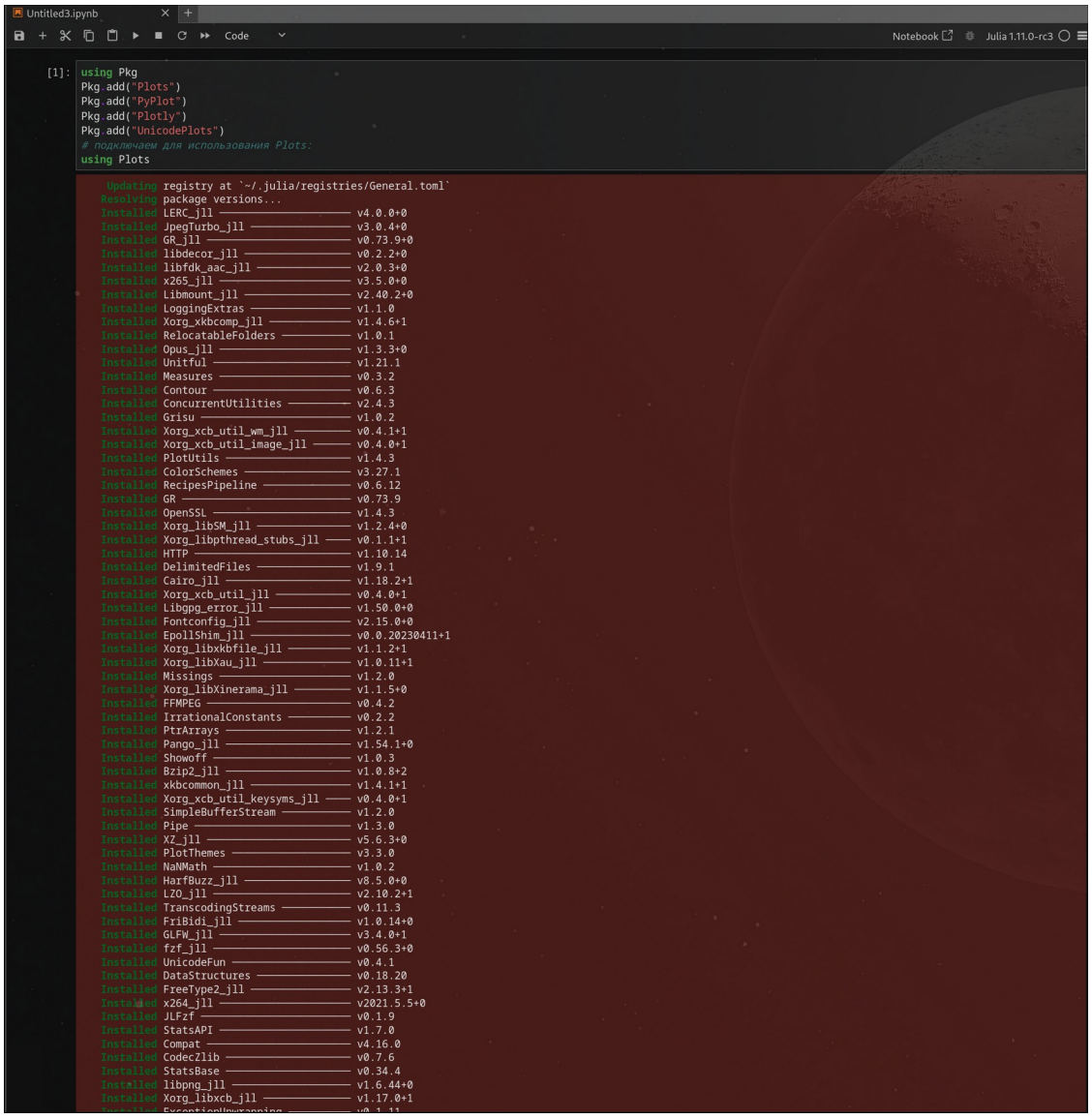
Цель работы

Оновная цель работы — освоить синтаксис языка Julia для построения графиков.

Задание

1. Используя Jupyter Lab, повторите примеры из раздела 5.2. При этом дополните графики обозначениями осей координат, легендой с названиями траекторий, названиями графиков и т.п.
2. Выполните задания для самостоятельной работы (раздел 5.4)

Выполнение лабораторной работы


Untitled3.ipynb
Code
Notebook
Julia 1.11.0-rc3
[1]:
using Pkg
Pkg.add("Plots")
Pkg.add("PyPlot")
Pkg.add("Plotly")
Pkg.add("UnicodePlots")
# подключаем для использования Plots:
using Plots
Updating registry at `~/.julia/registries/General.toml`
Resolving package versions...
Installed LERC_jll ─ v4.0.0+0
Installed JpegTurbo_jll ─ v3.0.4+0
Installed GR_jll ─ v0.73.9+0
Installed libdecor_jll ─ v0.2.2+0
Installed libfdk_aac_jll ─ v2.0.3+0
Installed x265_jll ─ v3.5.0+0
Installed Libmount_jll ─ v2.40.2+0
Installed LoggingExtras ─ v1.1.0
Installed Xorg_xkbcomp_jll ─ v1.4.6+1
Installed RelocatableFolders ─ v1.0.1
Installed Opus_jll ─ v1.3.3+0
Installed Unitful ─ v1.21.1
Installed Measures ─ v0.3.2
Installed Contour ─ v0.6.3
Installed ConcurrentUtilities ─ v2.4.3
Installed Grisu ─ v1.0.2
Installed Xorg_xcb_util_wm_jll ─ v0.4.1+1
Installed Xorg_xcb_util_image_jll ─ v0.4.0+1
Installed PlotUtils ─ v1.4.3
Installed ColorSchemes ─ v3.27.1
Installed RecipesPipeline ─ v0.6.12
Installed GR ─ v0.73.9
Installed OpenSSL ─ v1.4.3
Installed Xorg_libSM_jll ─ v1.2.4+0
Installed Xorg_libpthread_stubs_jll ─ v0.1.1+1
Installed HTTP ─ v1.10.14
Installed DelimitedFiles ─ v1.9.1
Installed Cairo_jll ─ v1.18.2+1
Installed Xorg_xcb_util_jll ─ v0.4.0+1
Installed Libgpg_error_jll ─ v1.50.0+0
Installed Fontconfig_jll ─ v2.15.0+0
Installed EpollShim_jll ─ v0.0.20230411+1
Installed Xorg_libxkbfile_jll ─ v1.1.2+1
Installed Xorg_libXau_jll ─ v1.0.11+1
Installed Missings ─ v1.2.0
Installed Xorg_libXinerama_jll ─ v1.1.5+0
Installed FFMPEG ─ v0.4.2
Installed IrrationalConstants ─ v0.2.2
Installed PtrArrays ─ v1.2.1
Installed Pango_jll ─ v1.54.1+0
Installed Showoff ─ v1.0.3
Installed Bzip2_jll ─ v1.0.8+2
Installed xkbcommon_jll ─ v1.4.1+1
Installed Xorg_xcb_util_keysyms_jll ─ v0.4.0+1
Installed SimpleBufferStream ─ v1.2.0
Installed Pipe ─ v1.3.0
Installed XZ_jll ─ v5.6.3+0
Installed PlotThemes ─ v3.3.0
Installed NaNMath ─ v1.0.2
Installed HarfBuzz_jll ─ v8.5.0+0
Installed LZO_jll ─ v2.10.2+1
Installed TranscodingStreams ─ v0.11.3
Installed FriBidi_jll ─ v1.0.14+0
Installed GLFW_jll ─ v3.4.0+1
Installed fzf_jll ─ v0.56.3+0
Installed UnicodeFun ─ v0.4.1
Installed DataStructures ─ v0.18.20
Installed FreeType2_jll ─ v2.13.3+1
Installed x264_jll ─ v2021.5.5+0
Installed JLFzf ─ v0.1.9
Installed StatsAPI ─ v1.7.0
Installed Compat ─ v4.16.0
Installed CodecZlib ─ v0.7.6
Installed StatsBase ─ v0.34.4
Installed libpng_jll ─ v1.6.44+0
Installed Xorg_libxcb_jll ─ v1.17.0+1

```
This may mean IJulia [7073ff75-c697-5162-941a-fcdaad2a7d2a] does not support precompilation but is imported by a module that does.
Info: Skipping precompilation due to precompilable error. Importing IJuliaExt [2f4121a4-3b3a-5ce6-9c5e-1f2673ce168a].
  exception = Error when precompiling module, potentially caused by a __precompile__(false) declaration in the module.
```

[2]:
```
f(x) = (3x.^2 + 6x .- 9).*exp.(-0.3x)
```

[2]:
```
f (generic function with 1 method)
```

[6]:
```
x = collect(range(-5,10,length=151))
```

[6]:
```
151-element Vector{Float64}:
 -5.0
 -4.9
 -4.8
 -4.7
 -4.6
 -4.5
 -4.4
 -4.3
 -4.2
 -4.1
 -4.0
 -3.9
 -3.8
  ⋮
  8.9
  9.0
  9.1
  9.2
  9.3
  9.4
  9.5
  9.6
  9.7
  9.8
  9.9
 10.0
```

[7]:
```
y = f(x)
```

[7]:
```
151-element Vector{Float64}:
 161.34080653217032
 146.26477779394003
 132.19219298833204
 119.06942359634911
 106.8453557470588
  95.47128188475011
  84.9007968362764
  75.08969810741056
  65.99589024347995
  57.579293095755176
  49.80175384104821
  42.62696260773204
  36.02037156694452
   ⋮
  19.531205103994854
  19.355187669047936
  19.176427741119536
  18.995125520095375
  18.811475185747092
  18.625664977689375
  18.437877278854756
  18.248288702120195
  18.057070179740368
  17.86438705526336
  17.6703991776229
  17.475260997120245
```

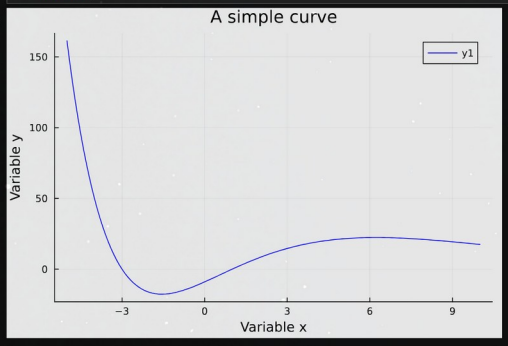
[8]:
```
gr()
```

[8]:
```
Plots.GRBackend()
```

[9]:
```
plot(x,y,
title="A simple curve",
xlabel="Variable_X",
```

```
title="A simple curve",
xlabel="Variable x",
    ylabel="Variable y",
color="blue")
```

[9]:



[11]:

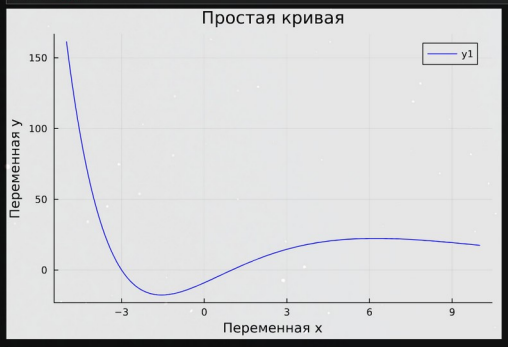
```
pyplot()
```

```
┌ Warning: You are using Matplotlib 0.0.0, which is no longer
│ officially supported by the Plots community. To ensure smooth Plots.jl
│ integration update your Matplotlib library to a version >= 3.4.0
│
│ If you have used Conda.jl to install PyPlot (default installation),
│ upgrade your matplotlib via Conda.jl and rebuild the PyPlot.
│
│ If you are not sure, here are the default instructions:
│
│ In Julia REPL:
│ ```
│ import Pkg;
│ Pkg.add("Conda")
│ import Conda
│ Conda.update()
│ Pkg.build("PyPlot")
│ ```
└ @ PyPlot ~/.julia/packages/PyPlot/rWSdf/src/deprecated.jl:36
┌ Warning: You are using Matplotlib 0.0.0, which is no longer
│ officially supported by the Plots community. To ensure smooth Plots.jl
│ integration update your Matplotlib library to a version >= 3.4.0
│
│ If you have used Conda.jl to install PyPlot (default installation),
│ upgrade your matplotlib via Conda.jl and rebuild the PyPlot.
│
│ If you are not sure, here are the default instructions:
│
│ In Julia REPL:
│ ```
│ import Pkg;
│ Pkg.add("Conda")
│ import Conda
│ Conda.update()
│ Pkg.build("PyPlot")
│ ```
└ @ PyPlot ~/.julia/packages/PyPlot/rWSdf/src/deprecated.jl:36
ArgumentError: hasproperty of NULL PyObject

Stacktrace:
  [1] pyhasproperty(o::PyCall.PyObject, s::String)
    @ PyCall ~/.julia/packages/PyCall/1gn3u/src/PyCall.jl:371
  [2] hasproperty
    @ ~/.julia/packages/PyCall/1gn3u/src/PyCall.jl:377 [inlined]
  [3] ioff(; kws::@Kwargs{})
    @ PyPlot ~/.julia/packages/PyPlot/rWSdf/src/PyPlot.jl:191
```

[12]:
```
plot(x,y,
title="Простая кривая",
xlabel="Переменная x",
ylabel="Переменная y",
color="blue")
```

[12]:



[13]:
```
sin_theor(x) = sin(x)
```

[13]:
```
sin_theor (generic function with 1 method)
```

[14]:
```
plot(sin_theor)
```

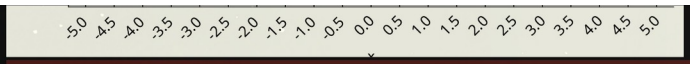
[14]:

[15]:
```
sin_taylor(x) = [(-1)^i*x^(2*i+1)/factorial(2*i+1) for i in 0:4] |> sum
```

[15]:
```
sin_taylor (generic function with 1 method)
```

[16]:
```
plot(sin_taylor)
```

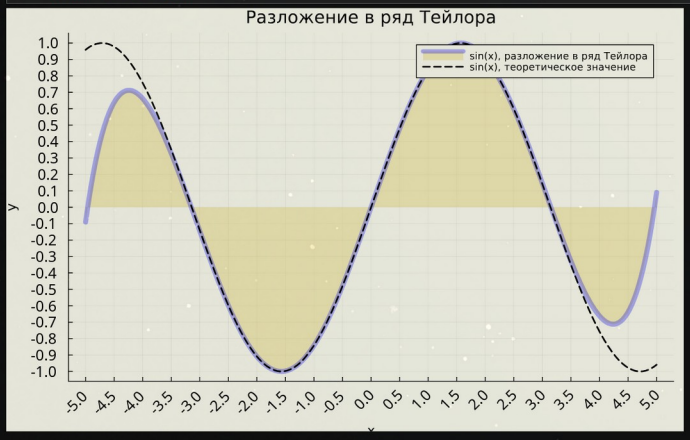
[16]:



```
GKS: could not find font Times New Roman.ttf
```

[19]:
```
plot!(
# функция sin_theor:
sin_theor,
# подпись в легенде, цвет и тип линии:
label = "sin(x), теоретическое значение",
line=(:black, 1.0, 2, :dash))
```

[19]:



[20]:
```
savefig("taylor.pdf")
savefig("taylor.png")
```

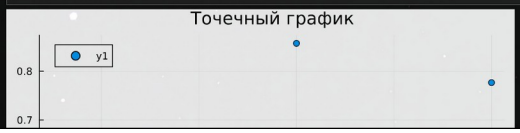
[20]:
```
"/home/starovoid/work/taylor.png"
```

[22]:
```
# параметры распределения точек на плоскости:
x = range(1,10,length=10)
y = rand(10)
```

[22]:
```
10-element Vector{Float64}:
 0.5479726739037795
 0.596813961158129
 0.6224827067815775
 0.666438156498657
 0.3758178564018656
 0.857123409610873
 0.2983541471719863
 0.3675524715298603
 0.6753355101655082
 0.7767483692890431
```

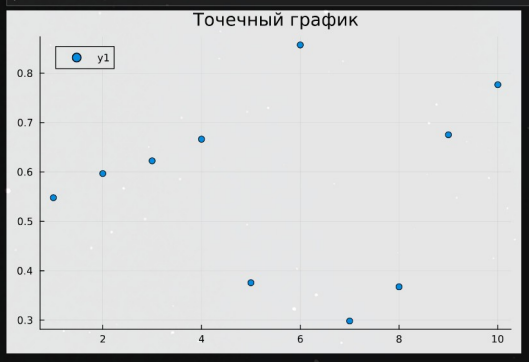
[23]:
```
plot(x, y,
seriestype = :scatter,
title = "Точечный график"
)
```

[23]:



```
title = "Точечный график"
)
```

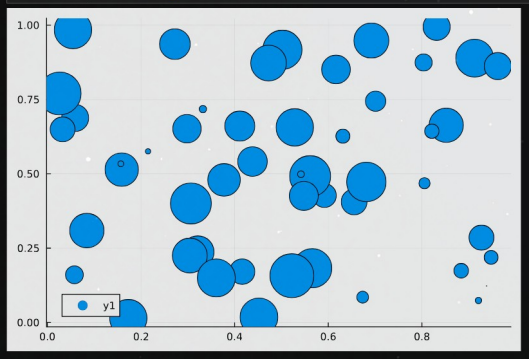
23]:



25]:

```
n = 50
x = rand(n)
y = rand(n)
ms = rand(50) * 30
# параметры построения графика:
scatter(x, y, markersize=ms)
```

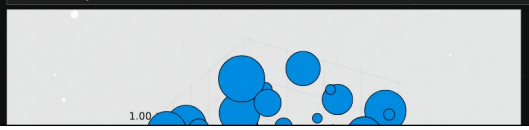
25]:



26]:

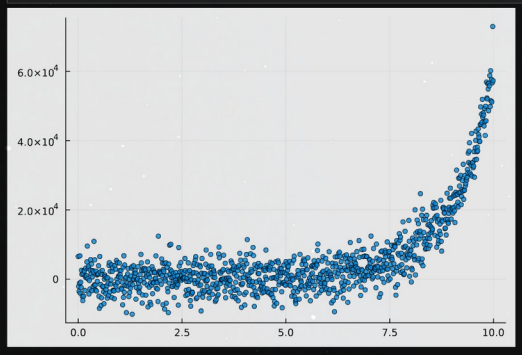
```
n = 50
x = rand(n)
y = rand(n)
z = rand(n)
ms = rand(50) * 30
# параметры построения графика:
scatter(x, y, z, markersize=ms)
```

26]:



[27]:
```
# массив данных от 0 до 10 с шагом 0.01:
x = collect(0:0.01:9.99)
# экспоненциальная функция со случайным сдвигом значений:
y = exp.(ones(1000)+x) + 4000*randn(1000)
# построение графика:
scatter(x,y,markersize=3,alpha=.8,legend=false)
```

[27]:

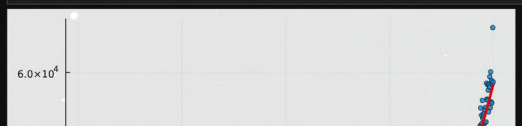


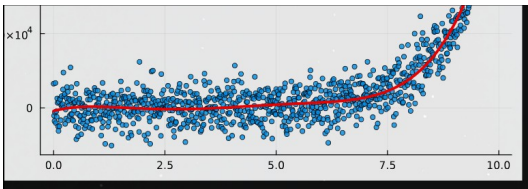
[32]:
```
# определение массива для нахождения коэффициентов полинома:
A = [ones(1000) x x.^2 x.^3 x.^4 x.^5]
# решение матричного уравнения:
c = A\y
# построение полинома:
g = c[1]*ones(1000) + c[2]*x + c[3]*x.^2 + c[4]*x.^3 + c[5]*x.^4 + c[6]*x.^5
```

[32]:
```
1000-element Vector{Float64}:
  -828.172038846246
  -795.5142512340921
  -763.4584919399991
  -731.9985123204036
  -701.1281000171448
  -670.8410788616624
  -641.1313087791918
  -611.9926856929609
  -583.4191414283861
  -555.404643617268
  -527.943195601988
  -501.028836339705
  -474.65564030655054
     ⋮
 51385.44182056596
 51835.821004672674
 52289.4324781223
 52746.29335141322
 53206.420793411555
 53669.83203144767
 54136.5443514135
 54606.57509785588
 55079.941674074274
 55556.661542216316
 56036.75222337188
 56520.23129767226
```

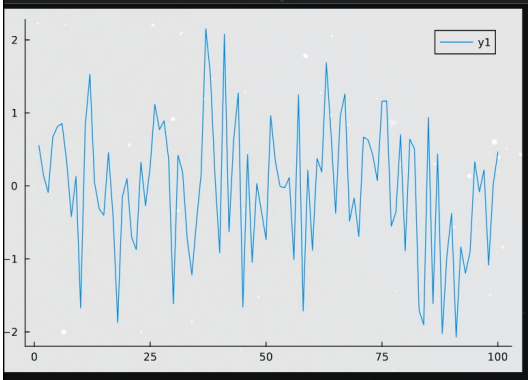
[33]:
```
plot!(x,g,linewidth=3, color=:red)
```

[33]:

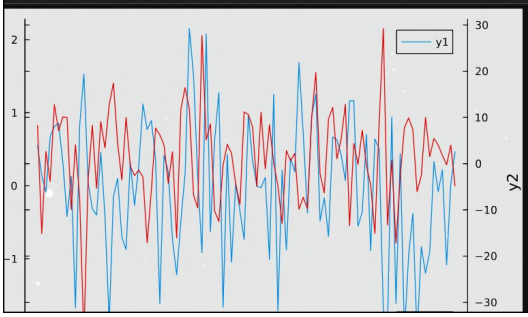




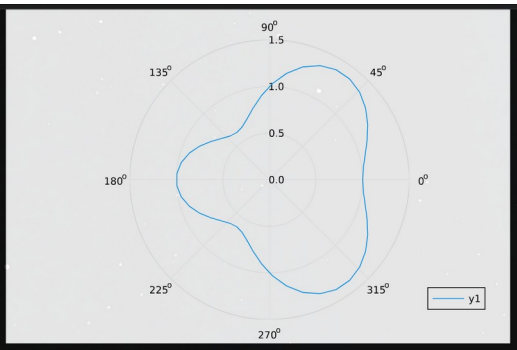

```
ример случайной траектории
заданы обозначение траектории, легенда вверху справа, без сетки)
t(randn(100),
bel="y1",
=:topright,
d = :off,
```




```
t!(twinx(), randn(100)*10,
red,
bel="y2",
=:bottomright,
d = :off,
 = :on)
```



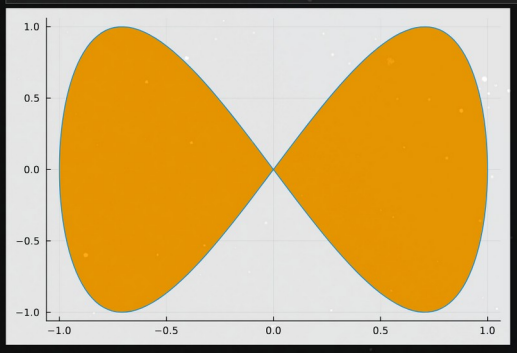
[36]:



[38]:

```
# параметрическое уравнение:
a(t) = sin(t)
b(t) = sin(2t)
# построение графика:
plot(a, b, 0, 2π, leg=false, fill=(0,:orange))
```

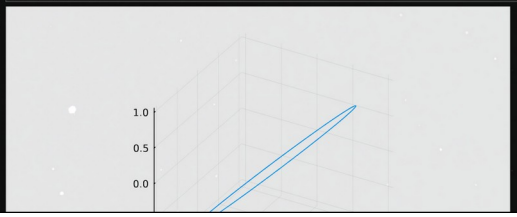
[38]:



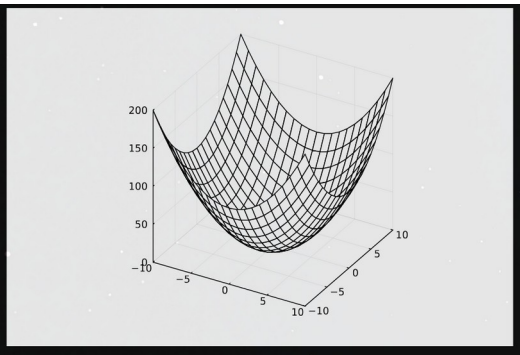
[39]:

```
# параметрическое уравнение
t = range(0, stop=10, length=1000)
x = cos.(t)
y = sin.(t)
z = sin.(5t)
# построение графика:
plot(x, y, x)
```

[39]:



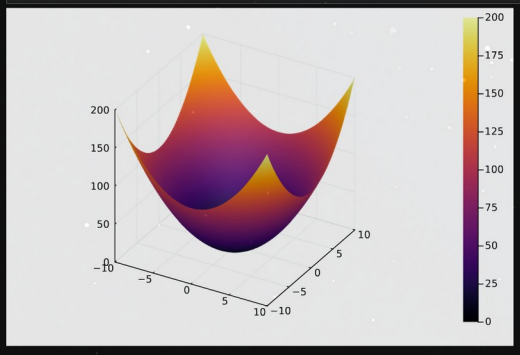
[41]:



[42]:

```
f(x,y) = x^2 + y^2
x = -10:0.1:10
y = x
plot(x, y, f,
linetype = :surface
)
```

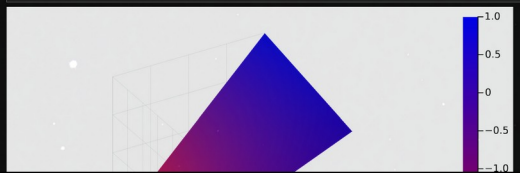
[42]:



[43]:

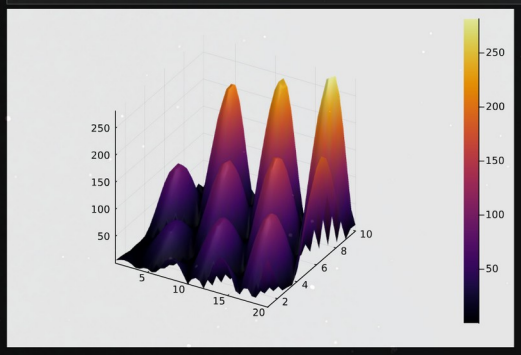
```
x=range(-2,stop=2,length=100)
y=range(sqrt(2),stop=2,length=100)
f(x,y) = x*y-x-y+1
plot(x,y,f,
linetype = :surface,
c=cgrad([:red,:blue]),
camera=(-30,30),
)
```

[43]:



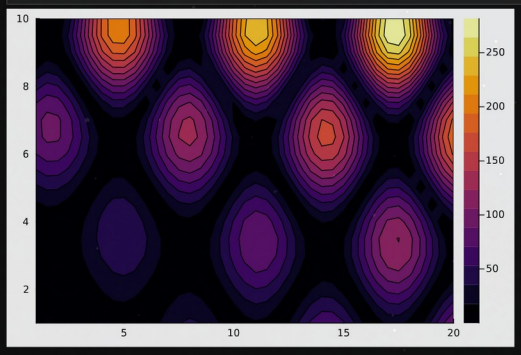
[45]:
```
x = 1:0.5:20
y = 1:0.5:10
h(x, y) = (3x + y ^ 2) * abs(sin(x) + cos(y))
plot(x,y,h,
linetype = :surface,
)
```

[45]:



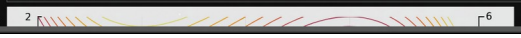
[48]:
```
p = contour(x, y, h,
fill=true)
plot(p)
```

[48]:



[49]:
```
# определение переменных:
X = range(-2, stop=2, length=100)
Y = range(-2, stop=2, length=100)
# определение функции:
h(x, y) = x^3 - 3x + y^2
# построение поверхности:
plot(X,Y,h,
linetype = :surface
)
# построение линий уровня:
contour(X, Y, h)
```

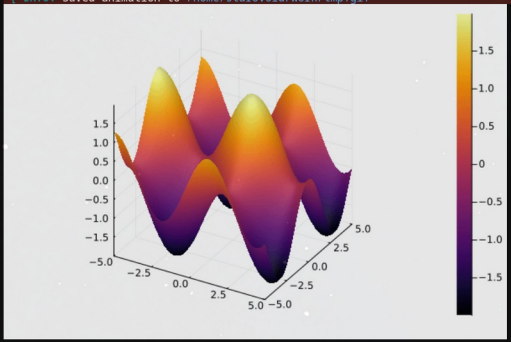
[49]:



```
# анимация:
X = Y = range(-5,stop=5,length=40)
@gif for i in range(0,stop=2π,length=100)
surface(X, Y, (x,y) -> sin(x+10sin(i))*cos(y))
end
```

```
[ Info: Saved animation to /home/starovoid/work/tmp.gif
```

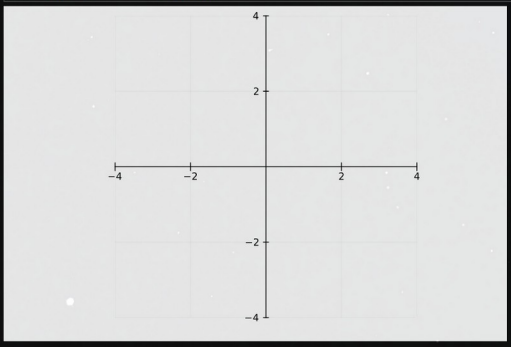
[52]:



[56]:

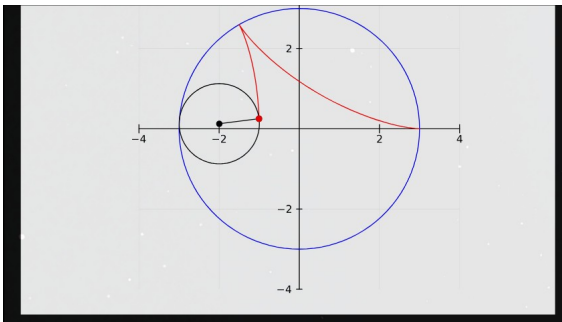
```
# радиус малой окружности:
rr = 1
# коэффициент для построения большой окружности:
k = 3
# число отсчётов:
n = 100
# массив значений угла θ:
# theta from 0 to 2pi ( + a little extra)
θ = collect(0:2*π/100:2*π+2*π/100)
# массивы значений координат:
X = rr*k*cos.(θ)
Y = rr*k*sin.(θ)
# задаём оси координат:
plt=plot(5,xlim=(-4,4),ylim=(-4,4), c=:red, aspect_ratio=1,
legend=false, framestyle=:origin)
```

[56]:



[57]:

```
# большая окружность:
plot!(plt, X,Y, c=:blue, legend=false)
i = 50
```

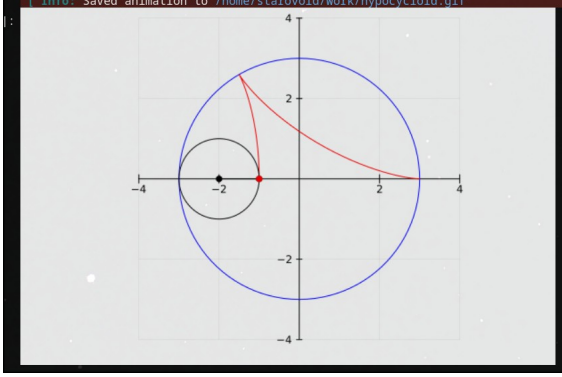


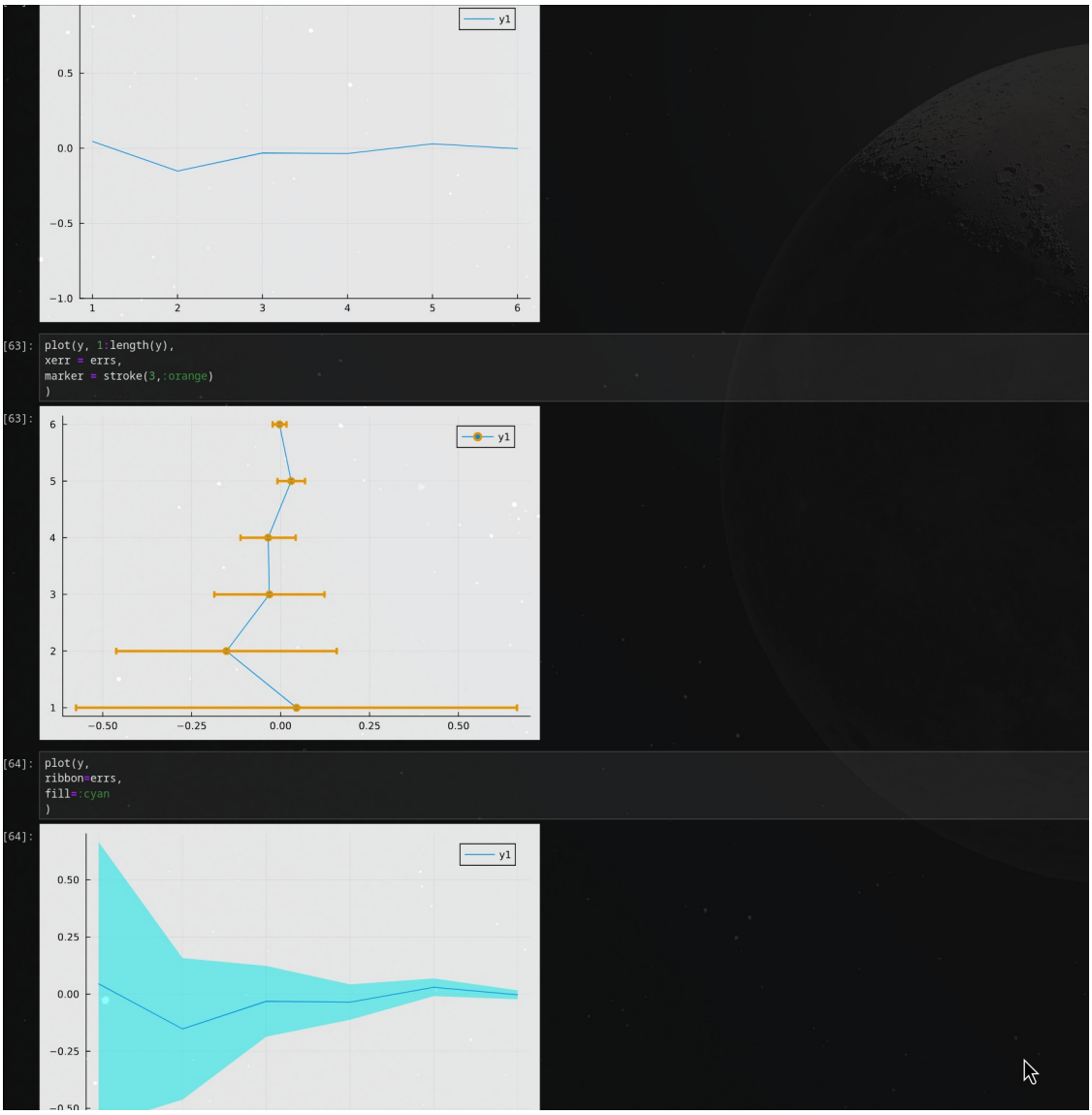

```
anim = @animate for i in 1:n
# задаём оси координат:
plt=plot(5,xlim=(-4,4),ylim=(-4,4), c=:red, aspect_ratio=1,legend=false, framestyle=:origin)
# большая окружность:
plot!(plt, X,Y, c=:blue, legend=false)
t = θ[1:i]
# гипоциклоида:
x = rr*(k-1)*cos.(t) + rr*cos.((k-1)*t)
y = rr*(k-1)*sin.(t) - rr*sin.((k-1)*t)
plot!(x,y, c=:red)
# малая окружность:
xc = rr*(k-1)*cos(t[end]) .+ rr*cos.(θ)
yc = rr*(k-1)*sin(t[end]) .+ rr*sin.(θ)
plot!(xc,yc,c=:black)
# радиус малой окружности:
xl = transpose([rr*(k-1)*cos(t[end]) x[end]])
yl = transpose([rr*(k-1)*sin(t[end]) y[end]])
plot!(xl,yl,markershape=:circle,markersize=4,c=:black)
scatter!([x[end]],[y[end]],c=:red, markerstrokecolor=:red)
end
```

```
Animation("/tmp/jl_EsA6Cz", ["000001.png", "000002.png", "000003.png", "000004.png", "000005.png", "000006.png", "000007.png", "000008.png", "000009.png", "00
 10.png"  …  "000091.png", "000092.png", "000093.png", "000094.png", "000095.png", "000096.png", "000097.png", "000098.png", "000099.png", "000100.png"])
```

```
gif(anim,"hypocycloid.gif")
```

```
[ Info: Saved animation to /home/starovoid/work/hypocycloid.gif
```





Выводы

Я изучил возможностей специализированных пакетов Julia для выполнения и оценки эффективности операций над объектами линейной алгебры.

Я освоил синтаксис языка Julia для построения графиков.

Список литературы