数値計算法　演習課題 2　提出日：2024 年 6 月 18 日

202310330　長田悠生

課題 1

(1-1)

$f(x) = \frac{1}{25x^2+5x+2}$ を Julia で書いたものを以下に示す。

(1-1)

```
function f(x::Float64)
  return 1 / (25*x^2 + 5*x + 2)
end
```

[-1, 1] の区間で等間隔に取った値と $[0, \pi]$ の区間で θ_i で等間隔に取ったときの $cos(\theta_i)$ の値をそれぞれ補完点として補完多項式を求め、グラフにプロットした。$cos(\theta_i)$ の方が精度が良いことがわかる。

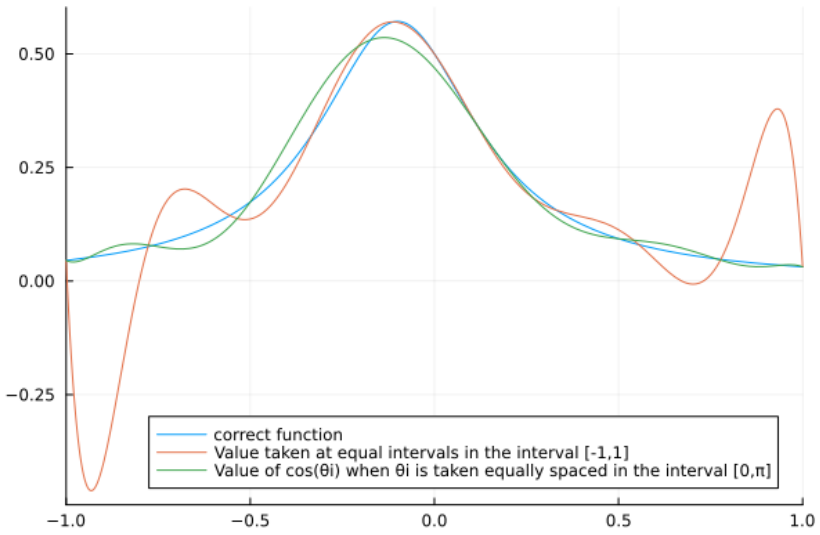

図 1　Runge の現象

以下が (1-1) のプログラムの全体である。

(1-1)

```
module Runge

  using Plots
  using Polynomials

  function f(x::Float64)
```

```
    return 1 / (25*x^2 + 5*x + 2)
  end

  function runge_graph(start_point::Float64, end_point::Float64,
      cos_start_point::Float64, cos_end_point::Float64,
     point_quantity::Int64)
    x_r::Array{Float64} = range(start_point, end_point,
       point_quantity)
    range_r_poly = Polynomials.fit(x_r, f.(x_r))

    x_cos::Array{Float64} = cos.(range(cos_start_point,
       cos_end_point, point_quantity));
    range_cos_poly = Polynomials.fit(x_cos, f.(x_cos));

    correct_plot = Plots.plot(f, xlim=[-1.0, 1.0], label="
       correct function", legend=:bottomright)
    r_poly_plot = Plots.plot!(correct_plot, range_r_poly, xlim
       =[-1.0, 1.0], label="Value taken at equal intervals in
       the interval [-1,1]", legend=:bottomright)
    Plots.plot!(r_poly_plot, range_cos_poly, xlim=[-1.0, 1.0],
       label="Value of cos(θi) when θi is taken equally spaced
        in the interval [0,π]", legend=:bottomright)
    Plots.savefig("runge.png")
  end

end

using .Runge

Runge.runge_graph(-1.0, 1.0, 0.0, Float64(pi), 10)
```

数値計算法　演習課題 2　提出日：2024 年 6 月 18 日

202310330　長田悠生

課題 2

(2-1)

x_0 から x_{m-1} を温度とする。温度のデータの数が 9 個で次関数で近似するので、$A = \begin{pmatrix} x_0^0 & x_1^1 \\ \vdots & \vdots \\ x_8^0 & x_8^1 \end{pmatrix}$ を生成するプログラムをかけば良い。関数は以下の通りである。

(2-1)

```
function make_a(temp::Array{Int64}, n::Int64)::Matrix{Float64}
  linage::Int64 = length(temp)
  a::Matrix{Float64} = [temp[i]^b for i in 1:linage, b in
     Float64(0):Float64(n)]
  return a
end
```

(2.2)

表 1 のデータをグラフにプロットした。 以下が (2.2) のグラフの画像を生成する関数である。

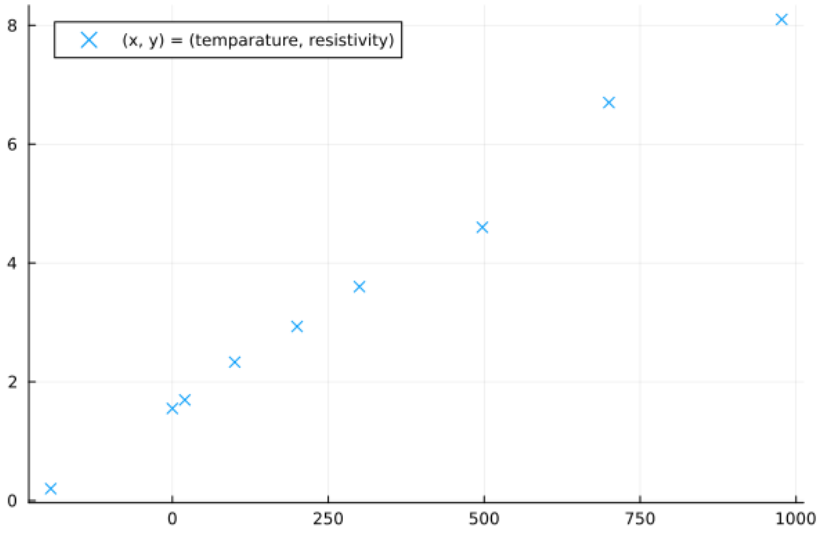

図 2　表 1 のデータ

(2-2)

数値計算法　演習課題 2　提出日：2024 年 6 月 18 日

202310330　長田悠生

```
function complete_true_graph(temp::Array{Int64}, resistivity::
    Array{Float64})
  plot(temp, resistivity, markershape=:x, la=0.0, label="(x, y)
      = (temparature, resistivity)")
  savefig("complete_plot.png")
end
```

数値計算法　演習課題 2　提出日：2024 年 6 月 18 日
202310330　長田悠生

(2.3)

以下が作成したグラフである。 以下がグラフを作成すために書いたプログラムである。

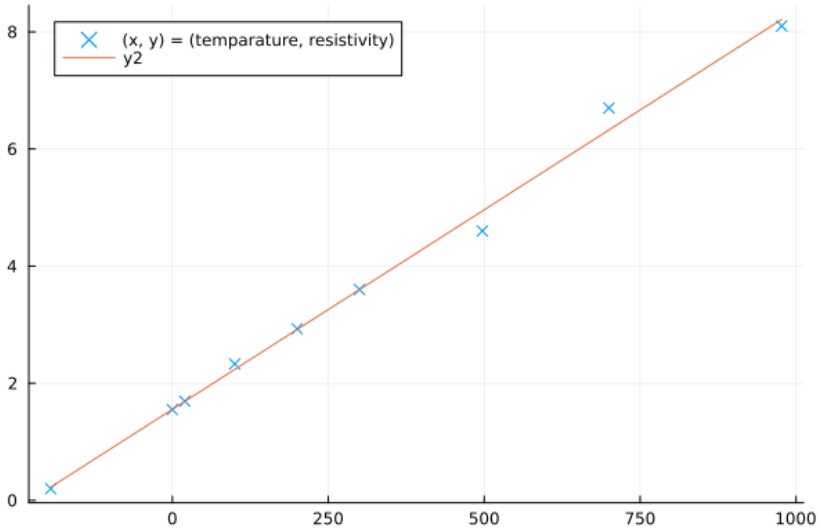

図 3　(2.3)

(2-3)

```
module Complete

  using Plots
  using LinearAlgebra

  function make_a(temp::Array{Int64}, n::Int64)::Matrix{Float64}
    linage::Int64 = length(temp)
    a::Matrix{Float64} = [temp[i]^b for i in 1:linage, b in
        Float64(0):Float64(n)]
    return a
  end

  function gen_a_b(a::Matrix{Float64}, resistivity::Array{
      Float64})::Vector{Float64}
    Q0::LinearAlgebra.QRCompactWYQ{Float64, Matrix{Float64},
        Matrix{Float64}}, R::Matrix{Float64} = LinearAlgebra.qr(a
        )
```

```
    Q::Matrix{Float64} = Matrix(Q0)
    c::Vector{Float64} = inv(R) * Q' * resistivity
    return c
  end

  function complete_graph(temp::Array{Int64}, resistivity::Array
     {Float64}, a::Matrix{Float64})
    c::Vector{Float64} = gen_a_b(a, resistivity)
    function f(x)
      return c[2] * x + c[1]
    end
    true_graph = Plots.plot(temp, resistivity, markershape=:x,
       la=0.0,  label="(x, y) = (temparature, resistivity)")
    Plots.plot!(true_graph, f)
    savefig("complete.png")
  end

end

using .Complete

temp::Array{Int64} = [-195, 0, 20, 100, 200, 300, 497, 700, 977]
resistivity::Array{Float64} = [0.2,     1.55,   1.694, 2.33,
   2.93, 3.6, 4.6, 6.7, 8.1]
a::Matrix{Float64} = Complete.make_a(temp, 1)
Complete.complete_graph(temp, resistivity, a)
```

(2.4-1)

$y = ax + b$ の a と b はわかっているのので、x または y を代入すれば、もう一方の未知数は求めることができる。
400 度は x なので y を計算するプログラムに 400.0 を代入すれば良い。
以下が y を推測するために必要なプログラムである。

(2-3)

```
  module Complete
```

数値計算法　演習課題 2　提出日：2024 年 6 月 18 日

202310330　長田悠生

```
  using Plots
  using LinearAlgebra

  function make_a(temp::Array{Int64}, n::Int64)::Matrix{Float64}
    linage::Int64 = length(temp)
    a::Matrix{Float64} = [temp[i]^b for i in 1:linage, b in
       Float64(0):Float64(n)]
    return a
  end

  function gen_a_b(a::Matrix{Float64}, resistivity::Array{
     Float64})::Vector{Float64}
    Q0::LinearAlgebra.QRCompactWYQ{Float64, Matrix{Float64},
       Matrix{Float64}}, R::Matrix{Float64} = LinearAlgebra.qr(a
       )
    Q::Matrix{Float64} = Matrix(Q0)
    c::Vector{Float64} = inv(R) * Q' * resistivity
    return c
  end

  function guess_y(x::Float64, c::Vector{Float64})
    return c[2] * x + c[1]
  end

  function guess_x(y::Float64, c::Vector{Float64})
    return (y - c[1]) / c[2]
  end

end

using .Complete

temp::Array{Int64} = [-195, 0, 20, 100, 200, 300, 497, 700, 977]
resistivity::Array{Float64} = [0.2,     1.55,   1.694, 2.33,
   2.93, 3.6, 4.6, 6.7, 8.1]
```

```
a::Matrix{Float64} = Complete.make_a(temp, 1)
c::Vector{Float64} = Complete.gen_a_b(a, resistivity)

temp_1 = Complete.guess_x(2.00, c)
println(temp_1)
```

プログラムの実行結果を以下に示す。

(2.4-1 の実行結果)

```
$ julia --project ./src/complete2-41.jl
65.33316000373105
```

2.00 は、y なので x を計算するプログラムに 2.00 を代入すれば良い。以下が x を推測するために必要なプログラムである。

(2.4-2)

```
module Complete

using Plots
using LinearAlgebra

function make_a(temp::Array{Int64}, n::Int64)::Matrix{Float64}
  linage::Int64 = length(temp)
  a::Matrix{Float64} = [temp[i]^b for i in 1:linage, b in
     Float64(0):Float64(n)]
  return a
end

function gen_a_b(a::Matrix{Float64}, resistivity::Array{
    Float64})::Vector{Float64}
  Q0::LinearAlgebra.QRCompactWYQ{Float64, Matrix{Float64},
     Matrix{Float64}}, R::Matrix{Float64} = LinearAlgebra.qr(a
     )
  Q::Matrix{Float64} = Matrix(Q0)
  c::Vector{Float64} = inv(R) * Q' * resistivity
  return c
end

function guess_y(x::Float64, c::Vector{Float64})
```

```
    return c[2] * x + c[1]
  end

  function guess_x(y::Float64, c::Vector{Float64})
    return (y - c[1]) / c[2]
  end

end

using .Complete

temp::Array{Int64} = [-195, 0, 20, 100, 200, 300, 497, 700, 977]
resistivity::Array{Float64} = [0.2,     1.55,   1.694, 2.33,
   2.93, 3.6, 4.6, 6.7, 8.1]

a::Matrix{Float64} = Complete.make_a(temp, 1)
c::Vector{Float64} = Complete.gen_a_b(a, resistivity)

resistivity_1 = Complete.guess_y(400.0, c)
println(resistivity_1)
```

数値計算法　演習課題 2　提出日：2024 年 6 月 18 日
202310330　長田悠生

課題 3

(3.1)

$n = 1, \cdots, 15$ までの mysin 関数の計算結果を $sin\left(\frac{\pi}{4}\right)$ の結果との相対誤差を取ってグラフにプロットしたものを以下に示す。

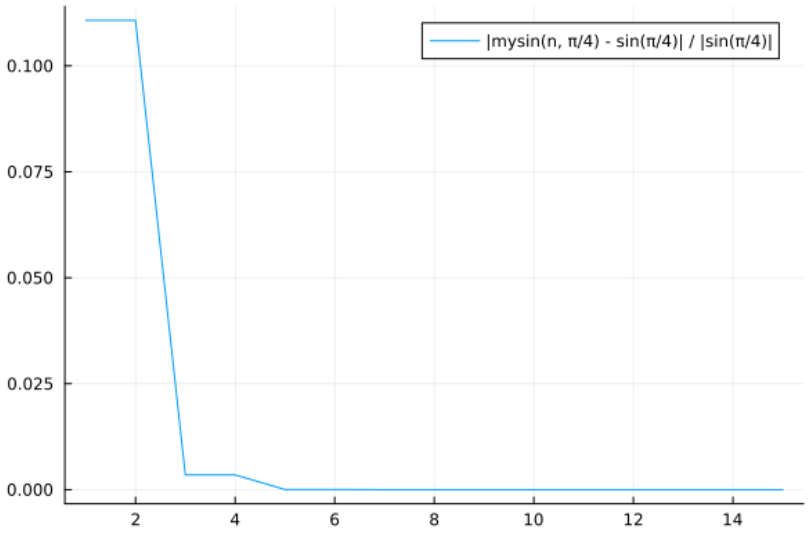

図 4　(2.3)

(3.2)

$n = 1, 3, 5, 7$ のときの区間 $[0, \pi]$ のグラフを以下に示す。
n の値が大きくなればなるほど、精度が良くなっていることがわかる。
課題 3 の全体のプログラムは以下のようになっている。

(2-3)

```
module Sin

using Plots

function mysin(n::Int64, x::Float64)::Float64
  xn = x
  P  = x
  k  = ceil(n / 2) - 1
```

数値計算法　演習課題 2　提出日：2024 年 6 月 18 日

202310330　長田悠生

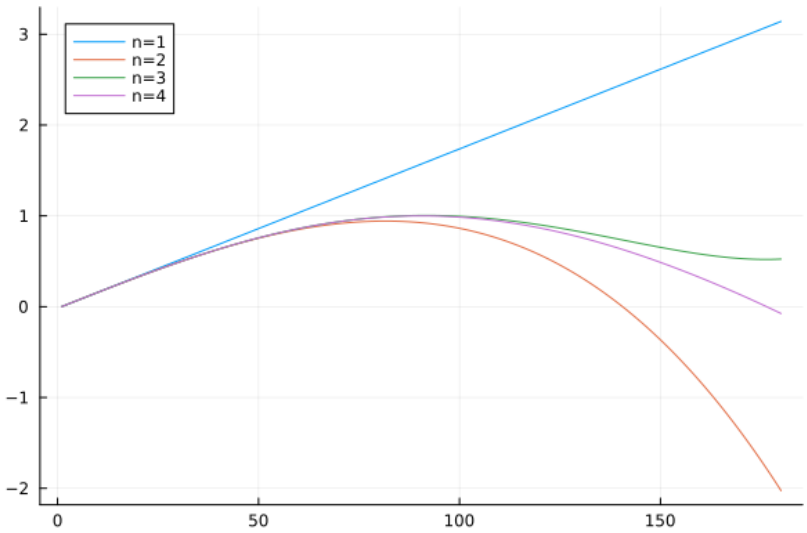

図 5　(2.3)

```
    for i = 1:k
      r  = ((-1) / ( (2*i) * (2*i + 1))) * x^2
      xn = xn * r
      P  = P + xn
    end
    return P
end

function sin_maclaurin(power_start::Int64, power_end::Int64, x
    ::Float64)
    p_array::Array{Float64} = zeros(Float64, 0)
    correct::Float64 = sin(x)
    for n=range(power_start, power_end, (power_end - power_start
        + 1))
      p::Float64 = mysin(Int64(n), x)
      #相対誤差の配列
      push!(p_array, abs(p - correct) / abs(correct))
    end
    Plots.plot(power_start:power_end, p_array, label="|mysin(n,
        π/4) - sin(π/4)| / |sin(π/4)|")
    savefig("mysin.png")
end
```

```
  function sin_maclaurin_data(start_point::Float64, end_point::
      Float64, step_point::Int64, n::Int64)::Array{Float64}
    sin_array::Array{Float64} = zeros(Float64, 0)
    for x=range(start_point, end_point, step_point)
      sin_point::Float64 = mysin(n, x)
      push!(sin_array, sin_point)
    end
    return sin_array
  end

  function sin_maclaurin_graph(start_point::Float64, end_point::
      Float64, step_point::Int64, n_array::Array{Int64})
    for n=1:length(n_array)
      sin_array = sin_maclaurin_data(start_point, end_point,
          step_point, n_array[n])
      if n == 1
        Plots.plot(1:step_point, sin_array, label="n=$n")
      else
        Plots.plot!(1:step_point, sin_array, label="n=$n")
      end
    end
    savefig("sin_maclaurin.png")
  end

end

using .Sin
using Plots

#(3.1)
const x::Float64 = pi / 4
Sin.sin_maclaurin(1, 15, x)

#(3.2)
Sin.sin_maclaurin_graph(0.0, Float64(pi), 180, [1, 3, 5, 7])
```

数値計算法　演習課題 2　提出日：2024 年 6 月 18 日
202310330　長田悠生

(3.3)

マクローリン展開を第 n 項までで止めた場合、$\frac{f^{(n+1)(c)}}{(n+1)!}x^{n+1}$ の誤差が発生する。n を大きくしていくと誤差は小さくなっていくが、x^{n+1} の増加量に対しての $(n+1)!$ の増加量が小さくなってしまう。そのため、次数 n を十分に大きくしても多項式の値の計算に寄与しなくなる。

課題 4

(4.1)

以下が myexp の関数である。

(4.1)

```
function myexp(n::Int64, x::Float64)::Float64
  xn::Float64 = 1.0
  P::Float64 = 1.0
  if n == 0
  else
    for k = 0:(n-1)
      r = x / (k+1)
      xn = xn * r
      P = P + xn
    end
  end
  return P
end
```

(4.2)

以下が myexp2 の関数である。

(4.2)

```
function myexp2(n::Int64, x::Float64)::Float64
  P::Float64 = 0.0
  if x > 0
    P = myexp(n, x)
  else
    P = (1 / myexp(n, -x))
```

```
  end
  return P
end
```

(4.3)

以下が myexp3 の関数である。

(4.2)

```
function myexp3(n::Int64, x::Float64)::Float64
  #整数部
  x_int::Int64 = convert(Int, floor(x))
  #小数部
  x_decimal::Float64 = x - x_int
  #整数部の計算
  #result_int::Float64=exp(x_int)
  result_int::Float64 = 1.0
  if x_int > 0
    for i=1:x_int
      result_int = result_int * ☒
    end
  elseif x_int == 0
  else
    for i=1:-x_int
        result_int = result_int * ☒
    end
    result_int = 1 / result_int
  end
  #小数部の計算
  P::Float64 = myexp(n, x_decimal)
  return (result_int * P)
end
```

数値計算法　演習課題 2　提出日：2024 年 6 月 18 日

202310330　長田悠生

(4.4)

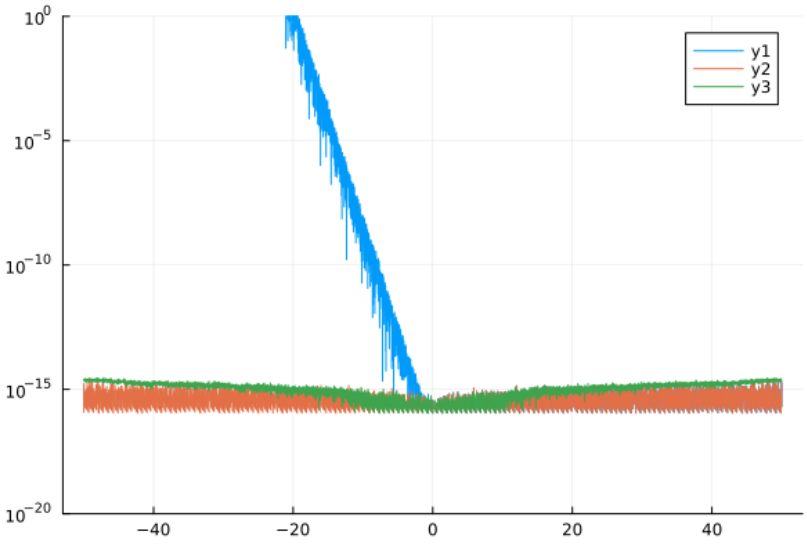

図 6　n=200, 区間 [-50, 50]

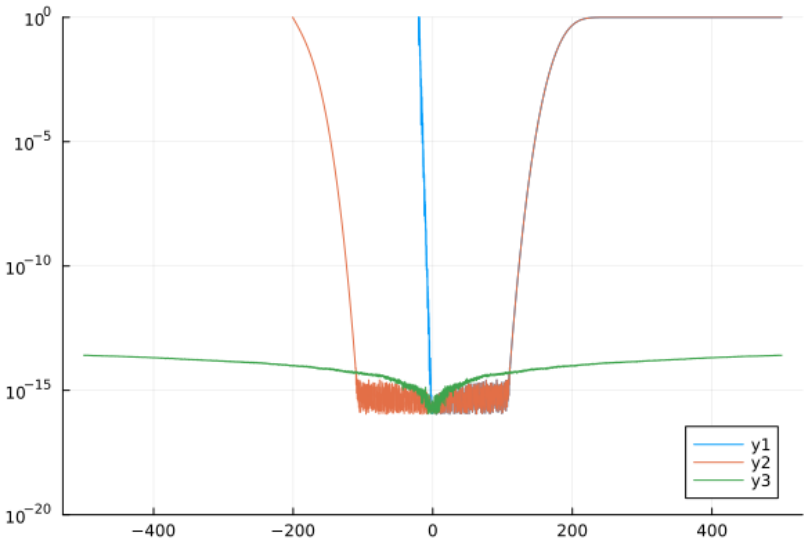

図 7　n=200, 区間 [-500, 500]

数値計算法　演習課題 2　提出日：2024 年 6 月 18 日

202310330　長田悠生

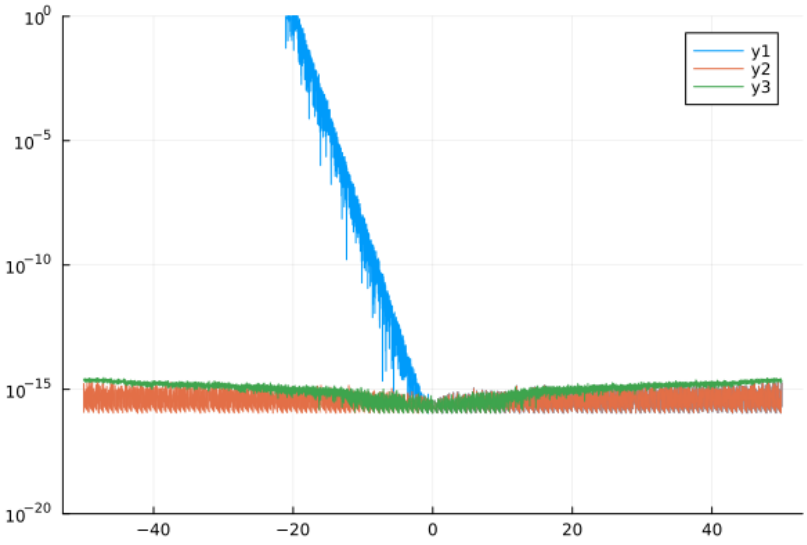

図 8　n=1000, 区間 [-50, 50]

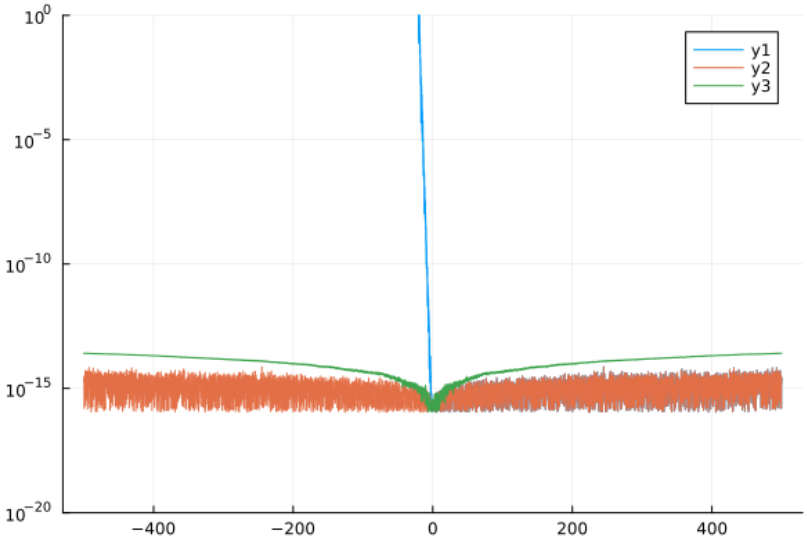

図 9　n=1000, 区間 [-500, 500]

課題 4 の全体のプログラムを以下に示す。

課題 4

```
module Exp

  using Plots

  function myexp(n::Int64, x::Float64)::Float64
```

```
    xn::Float64 = 1.0
    P::Float64 = 1.0
    if n == 0
    else
      for k = 0:(n-1)
        r = x / (k+1)
        xn = xn * r
        P = P + xn
      end
    end
    return P
  end

  function myexp2(n::Int64, x::Float64)::Float64
    P::Float64 = 0.0
    if x > 0
      P = myexp(n, x)
    else
      P = (1 / myexp(n, -x))
    end
    return P
  end

  function myexp3(n::Int64, x::Float64)::Float64
    #整数部
    x_int::Int64 = convert(Int, floor(x))
    #小数部
    x_decimal::Float64 = x - x_int
    #整数部の計算
    #result_int::Float64=exp(x_int)
    result_int::Float64 = 1.0
    if x_int > 0
      for i=1:x_int
        result_int = result_int * ☒
      end
    elseif x_int == 0
    else
```

```
        for i=1:-x_int
            result_int = result_int * ☒
        end
        result_int = 1 / result_int
    end
    #小数部の計算
    P::Float64 = myexp(n, x_decimal)
    return (result_int * P)
end

function exp_relative_error_data(n::Int64, start_point::
    Float64, end_point::Float64, point_quantity::Int64)::Tuple{
    Array{Float64}, Array{Float64}, Array{Float64}}
    myexp_arrary::Array{Float64} = zeros(Float64, 0)
    myexp2_arrary::Array{Float64} = zeros(Float64, 0)
    myexp3_arrary::Array{Float64} = zeros(Float64, 0)
    for x in range(start_point, end_point, point_quantity)
        correct_point::Float64 = exp(x)
        exp_relative_error::Float64 = abs(myexp(n, x) -
            correct_point) / abs(correct_point)
        exp2_relative_error::Float64 = abs(myexp2(n, x) -
            correct_point) / abs(correct_point)
        exp3_relative_error::Float64 = abs(myexp3(n, x) -
            correct_point) / abs(correct_point)

        push!(myexp_arrary, exp_relative_error)
        push!(myexp2_arrary, exp2_relative_error)
        push!(myexp3_arrary, exp3_relative_error)
    end
    return tuple(myexp_arrary, myexp2_arrary, myexp3_arrary)
end

function exp_graph(n::Int64, start_point::Float64, end_point::
    Float64, point_quantity::Int64, png::String)
    myexp_arrary::Array{Float64}, myexp2_arrary::Array{Float64},
        myexp3_arrary::Array{Float64} = exp_relative_error_data(
        n, start_point, end_point, point_quantity)
```

```
    Plots.plot(range(start_point, end_point, point_quantity),
        myexp_arrary, ylim=[10^(-20), 1.0], yaxis=:log)
    Plots.plot!(range(start_point, end_point, point_quantity),
        myexp2_arrary, ylim=[10^(-20), 1.0], yaxis=:log)
    Plots.plot!(range(start_point, end_point, point_quantity),
        myexp3_arrary, ylim=[10^(-20), 1.0], yaxis=:log)
    savefig("$png")
  end

end

import .Exp

hello::Float64 = Exp.myexp(10, -3.14)

println("答え: $(exp(-3.14)), マクローリン: $hello")

hello2::Float64 = Exp.myexp2(10, -3.14)

println("答え: $(exp(-3.14)), マクローリン: $hello2")

hello3::Float64 = Exp.myexp3(10, -3.14)

println("答え: $(exp(-3.14)), マクローリン: $hello3")

a, b, c = Exp.exp_relative_error_data(200, -50.0, 50.0, 101)

println("myexp = $a, myexp2 = $b, myexp3 = $c")

Exp.exp_graph(200, -50.0, 50.0, 10000, "exp200_50.png")
Exp.exp_graph(200, -500.0, 500.0, 10000, "exp200_500.png")
Exp.exp_graph(1000, -50.0, 50.0, 10000, "exp1000_50.png")
Exp.exp_graph(1000, -500.0, 500.0, 10000, "ex1000_500.png")
```