Homework 1

Kevin Silberberg

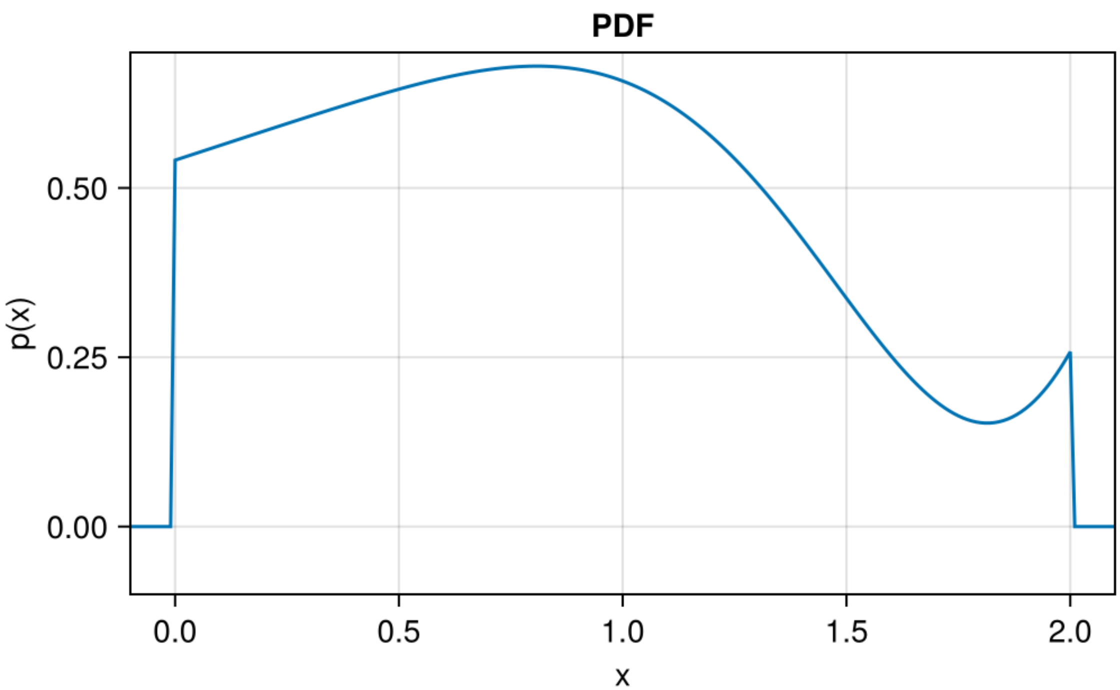
2024-10-02

Problem definition

Let X be a random variable with Probability density function:

$$p(x) = \begin{cases} \frac{2x\cos(x^2)+5}{10+\sin(4)} & x \in [0,2] \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

```
using CairoMakie

function p(x::Real)
    (0   x   2.0) ? (2.0 * x * cos(x^2.0) + 5.0) / (10.0 + sin(4.0)) : 0.0
end

function plotPDF()
    fig = Figure()
    ax = Axis(fig[1,1],
              title = "PDF",
              xlabel = "x",
              ylabel = "p(x)"
              )
    r = -0.1:0.01:2.1
    lines!(ax, r, p.(r))
    xlims!(ax, -0.1, 2.1)
    ylims!(ax, -0.1, 0.7)
    display(fig);
end
plotPDF();
```



Part A

Solve for the mean and standard deviation Numerically

Using the Gauss-Kronrod quadrature we can numerically solve for the mean and the variance by calculating the first moment and second momements of the PDF of X.

```
using QuadGK

function E1(x::Real)
    return x * p(x)
end

function E2(x::Real)
    return x^2 * p(x)
end

function parta_numeric()
    Ex  = quadgk(E1, 0.0, 2.0, rtol=1e-3)[1]
    Ex² = quadgk(E2, 0.0, 2.0, rtol=1e-3)[1]
```

```
     ² = Ex² - (Ex)^2
        = sqrt( ²)
    println("The numeric mean and standard deviation of the PDF of X are:")
    println("  = $Ex")
    println("  = $ ")
    return Ex,
end

 _numeric,  _numeric = parta_numeric();
```

```
The numeric mean and standard deviation of the PDF of X are:
  = 0.8310564083631247
  = 0.4954158522588331
```

Let us solve for the first and second moments of (1) analytically and compare with the numerical findings.

First moment

$$
\begin{aligned}
E[X] &= \int_0^2 x\left(\frac{2x\cos\left(x^2\right)+5}{10+\sin\left(4\right)}\right)dx && (1)\\
&= x\left(\frac{\sin\left(x^2\right)+5x}{10+\sin\left(4\right)}\right)\Bigg|_0^2 - \frac{1}{10+\sin\left(4\right)}\int_0^2\left(\sin\left(x^2\right)+5x\right)dx && (2)\\
&= \frac{-\sqrt{\frac{\pi}{2}}S\left(\sqrt{\frac{2}{\pi}}x\right)+\frac{5x^2}{2}+x\sin\left(x^2\right)}{10+\sin\left(4\right)}\Bigg|_0^2 && (3)\\
& && (4)
\end{aligned}
$$

Where $S(x)$ is the fresnel integral defined as

$$
S(x) = \int_0^x \sin\left(t^2\right)dt
$$

```
using FresnelIntegrals
function first_moment()
    E(x) = (-sqrt( /2.0)*fresnels(sqrt(2.0/ )*x) + (5.0/2.0)*x^2 + x*sin(x^2.0)) / (10.0 + si
      = E(2.0) - E(0.0)
end
 _exact = first_moment();
```

Second moment

$$
\begin{aligned}
E^2[X] &= \int_0^2 x^2 \left(\frac{2x\cos{(x^2)} + 5}{10 + \sin{(4)}} \right) dx && (5)\\
&= \frac{1}{10 + \sin{(4)}} \left[2 \int_0^2 x^3 \cos{(x^2)} dx + 5 \int_0^2 x^2 dx \right] && (6)\\
&= \left. \frac{x^2 \sin{(x^2)} + \cos{(x^2)} + \frac{5x^3}{3}}{10 + \sin{(4)}} \right|_0^2 && (7)
\end{aligned}
$$

```
function stdev()
    E2(x) = (x^2.0*sin(x^2.0) + cos(x^2.0) + (5.0x^3/3.0)) / (10.0 + sin(4.0))
    Ex²   = E2(2.0) - E2(0.0)
     ² = Ex² - _exact^2
    return sqrt( ²)
end
 _exact = stdev()

println("The mean and the standard deviation of the PDF of X by analytical solution:")
println("  = $ _exact")
println("  = $ _exact")
```

```
The mean and the standard deviation of the PDF of X by analytical solution:
  = 0.8310564083631246
  = 0.49541585225883805
```

The numeric and analytic solutions agree.

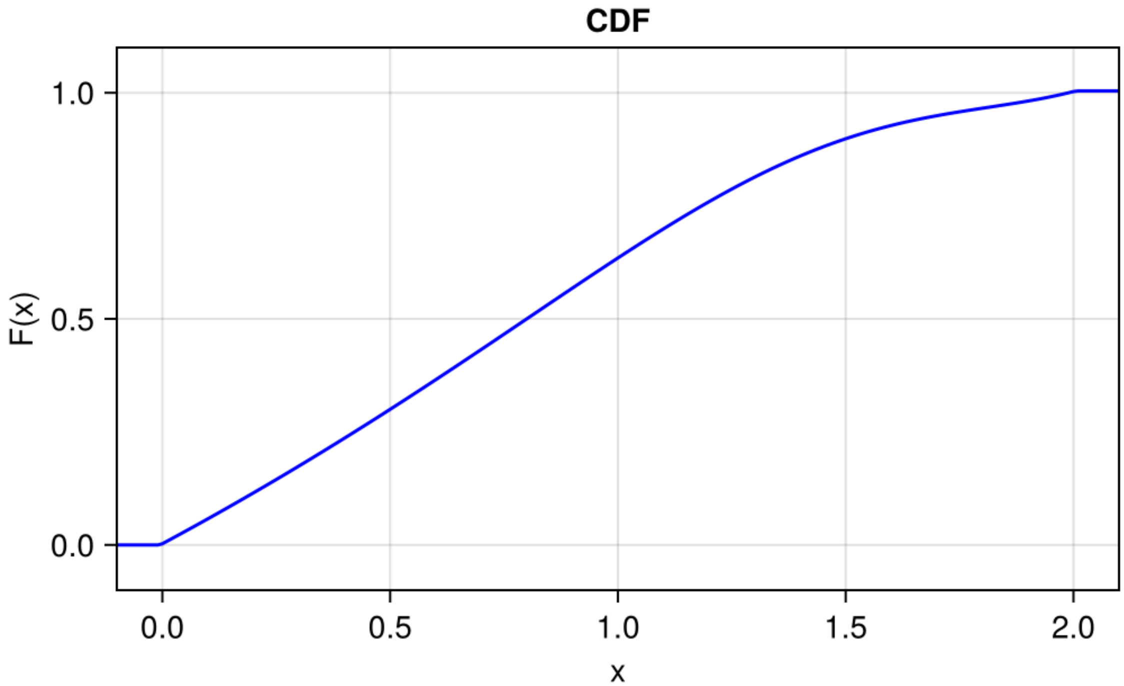
```
err_  = (abs( _numeric -  _exact) /  _exact) * 100
err_  = (abs( _numeric -  _exact) /  _exact) * 100

println("Percent error in the calculated mean = $err_ ")
println("Percent error in the calculated standard deviation = $err_ ")
```

```
Percent error in the calculated mean = 1.3359177709872757e-14
Percent error in the calculated standard deviation = 9.972414966246794e-13
```

Part B

In order to find the CDF numerically, we will need to write a modified version of the trapezoidal rule such that we are populating an array with the cumulative sum of every dx of the PDF of X.

```
function cumsumtrap(f::Function, x)
    y = f.(x)
    N = length(x)
    dx = x[2:N] .- x[1:N-1]
    meanY = (y[2:N] .+ y[1:N-1]) ./ 2
    integral = cumsum(dx .* meanY)

    return [0; integral]
end

function plotCDF()
    fig = Figure();
    ax = Axis(fig[1,1],
              title = "CDF",
              xlabel = "x",
              ylabel = "F(x)"
              )
    r = -0.1:0.01:2.1
    lines!(ax, r, cumsumtrap(p, r), label = "numerical", color = :blue)
    xlims!(ax, -0.1, 2.1)
    ylims!(ax, -0.1, 1.1)

    display(fig)
    return fig, ax, r
end
fig1, ax1, r = plotCDF();
```



Let us calculate the CDF by direct integration. The CDF of a continuous random variable X can be expressed as the integral of its probability density function $p(x)$ as:

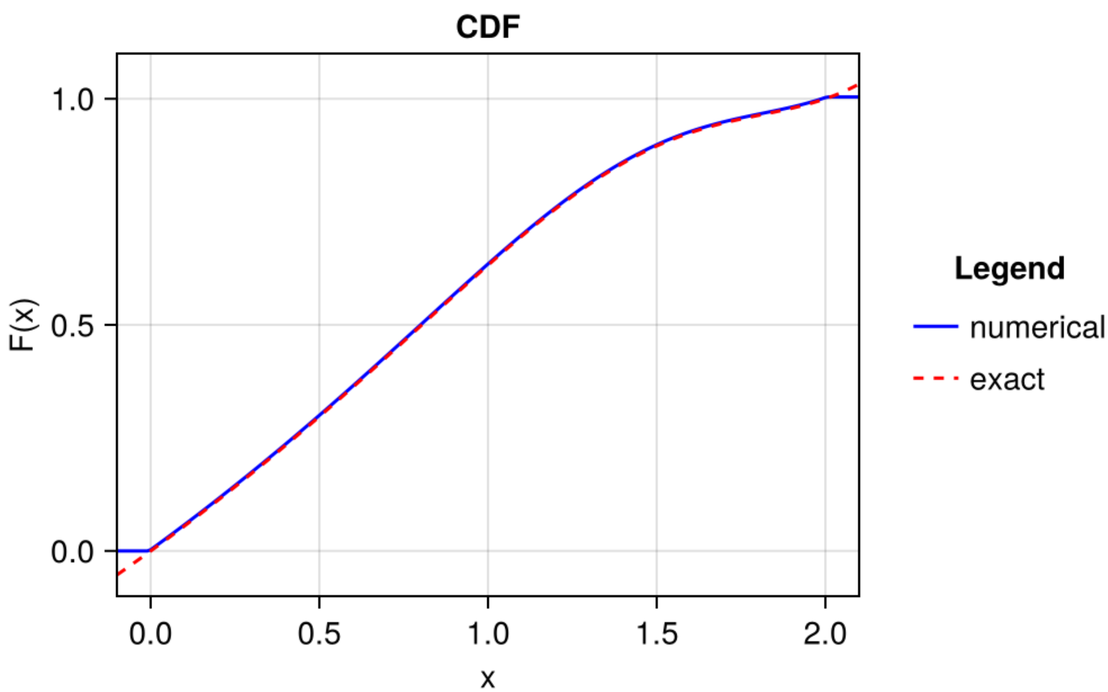
$$F_X(x) = \int_{-\inf}^{x} p_X(t)dt \tag{2}$$

Plugging (1) on the interval [0, 2] into (2) we get:

$$\begin{aligned} F_X(x) &= \int_0^x \left(\frac{2t\cos(t^2)+5}{10+\sin(4)}\right)dt \qquad (8)\\ &= \frac{1}{10+\sin(4)}\left(2\int_0^x t\cos(t^2)dt + 5\int_0^x dt\right) \qquad (9)\\ &= \frac{\sin(x^2)+5x}{10+\sin(4)} \qquad (10) \end{aligned}$$

Let us plot the exact solution of the CDF over the numerical solution.

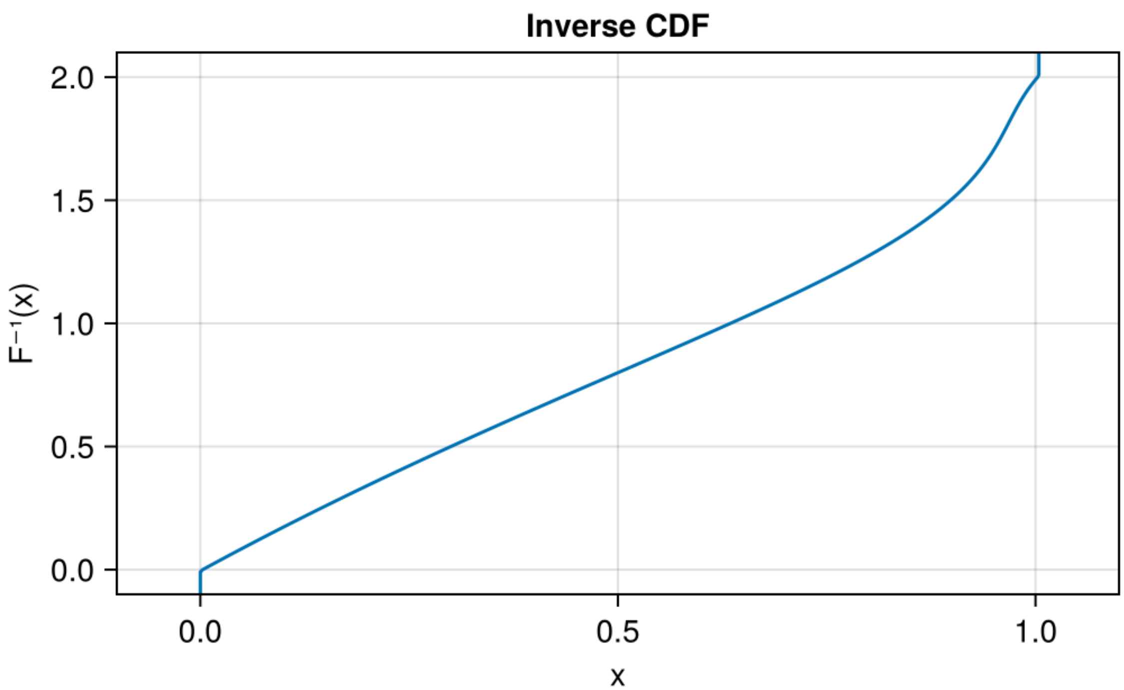
```
F (x) = (sin(x^2) + 5x) / (10 + sin(4))

lines!(ax1, r, F .(r), label = "exact", color = :red, linestyle = :dash);
fig1[1,2] = Legend(fig1, ax1, "Legend", framevisible = false);
display(fig1);
```



The we can plot the inverse CDF by flipping the axis.

```
function plotCDFInverse()
    fig = Figure();
    ax = Axis(fig[1, 1],
              title = "Inverse CDF",
              xlabel = "x",
              ylabel = "F ¹(x)"
              )
    lines!(ax, cumsumtrap(p, r), r)
    xlims!(ax, -0.1, 1.1)
    ylims!(ax, -0.1, 2.1)
    display(fig);
```

```
end
plotCDFInverse();
```



Part c

Let us develop a sampler for the random variable X.

Let the Matrix Σ be a (N,2) matrix such that the rows are (x, y) coordinates, and the vector **x** also be of size N, where each element is a sample from the Uniform normal distribution.

By using linear interpolation from the formula:

$$y = y_1 + \frac{(x - x_1)(y_2 - y_1)}{x_2 - x_1} \tag{3}$$

The first column of Σ comes from the cumulative trapazoidal integration of the pdf function in the range [0, 2], while the second column are equally spaced points from the range [0, 2].

```julia
function liy(x::Float64, p1::Vector{Float64}, p2::Vector{Float64})
    x1, y1 = p1
    x2, y2 = p2
    return y1 + (x - x1)*(y2 - y1)/(x2 - x1)
end

function sampleInverseCDF(x::Vector{Float64}, points::Matrix{Float64})
    output = Vector{Float64}(undef, length(x))
    for (i, x_val) in enumerate(x)
        idx = findfirst(points[:, 1] .> x_val)

        if idx == nothing
            # If x_val is greater than or equal to the last x value in inverse, use the last
            p1 = points[end-1, :]
            p2 = points[end, :]
        elseif idx == 1
            # If x_val is less than or equal to the first x value in inverse, use the first s
            p1 = points[1, :]
            p2 = points[2, :]
        else
            # Otherwise, use the segment between idx-1 and idx
            p1 = points[idx-1, :]
            p2 = points[idx, :]
        end

        # Calculate interpolated y value using the liy function
        output[i] = liy(x_val, p1, p2)
    end
    output
end

function plotsampledist()
    Δr = 1e-3
    r = -0.1:Δr:2.1
    points = [cumsumtrap(p, r) r]
    N = 100000
    x = rand(N)
    fig = Figure()
    ax = Axis(fig[1,1], title = "histogram of $N samples")
    hist!(fig[1,1], sampleInverseCDF(x, points), bins = 80, normalization = :pdf)
    lines!(fig[1,1], r, p.(r), color = :red, label = "p(x)", linestyle = :dash)
    fig[1, 2] = Legend(fig, ax, "Legend", framevisible = false)
```

```
    display(fig)
end
plotsampledist();
```