**Algorithm For Optimizations Practical**

**Aim: Apply Random Forest in Surrogate Model.**

Random forest is a supervised learning algorithm that randomly creates and merges multiple decision trees into one forest.

We are going to use a Random forests surrogate to optimize

*f*(*x*)=*sin*(*x*)+*sin*(10/3∗*x*)

**First of all, import Surrogates and Plots:**

**using Pkg**

**Pkg.add("Surrogates")**

**Pkg.add("SurrogatesRandomForest")**

**Pkg.add("Plots")**

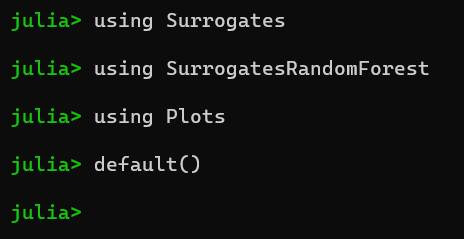
**using Surrogates**

**using SurrogatesRandomForest**

**using Plots**

**default()**

**Output:**

****

**Sampling:**

We choose to sample f in 4 points between 0 and 1 using the sample function. The sampling points are chosen using a Sobol sequence, this can be done by passing SobolSample() to the sample function.

f(x) = sin(x) + sin(10 / 3 \* x)

n\_samples = 5

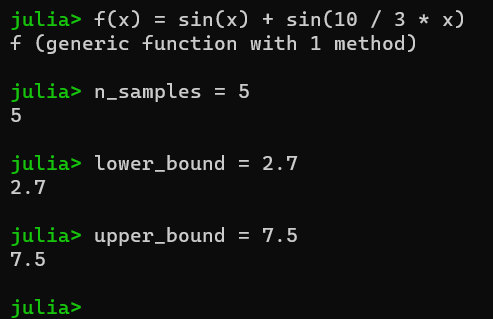
lower\_bound = 2.7

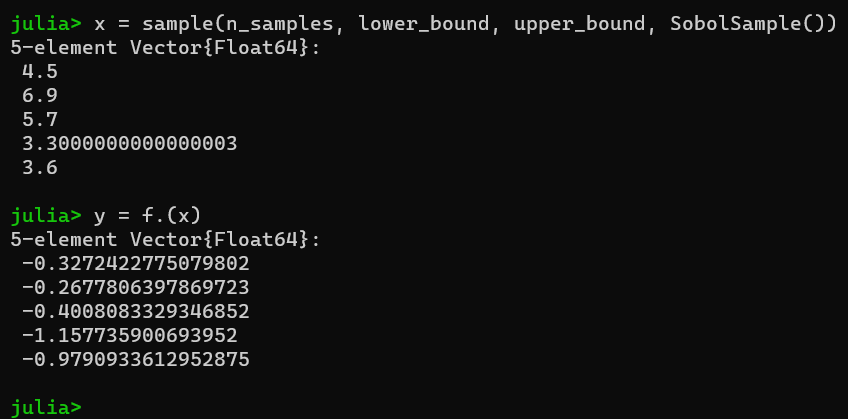
upper\_bound = 7.5

x = sample(n\_samples, lower\_bound, upper\_bound, SobolSample())

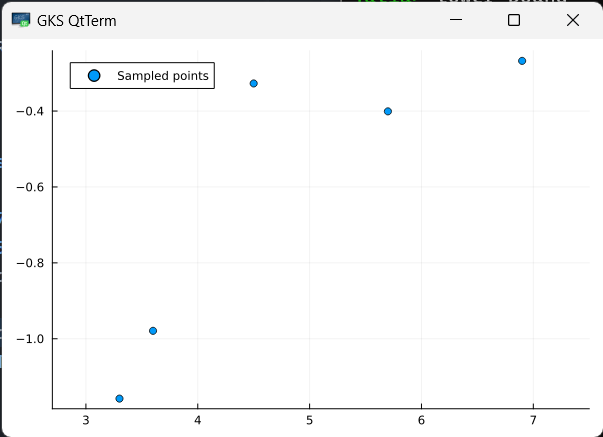
y = f.(x)

**Output:**

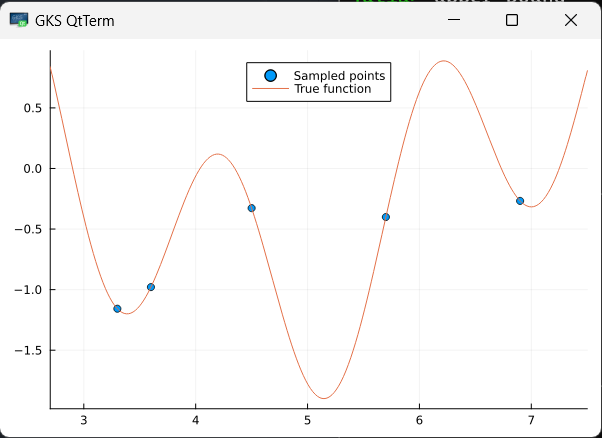
****

****

**scatter(x, y, label="Sampled points", xlims=(lower\_bound, upper\_bound))**

****

**plot!(f, label="True function", xlims=(lower\_bound, upper\_bound), legend=:top)**

****

**Building a surrogate:**

With our sampled points we can build the Random forests surrogate using

the RandomForestSurrogate function.

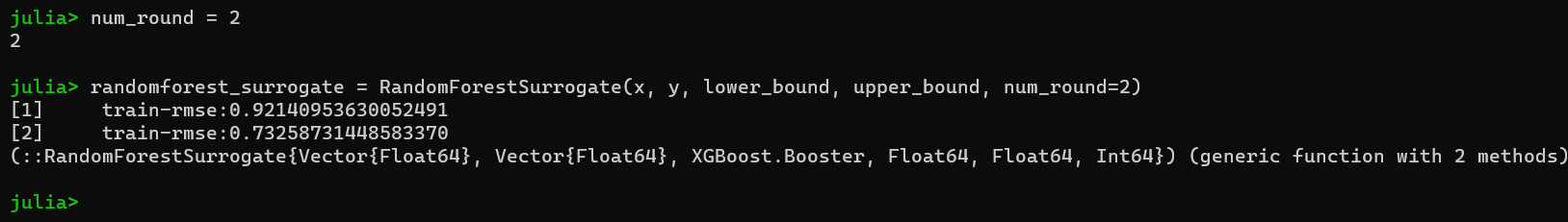
randomforest\_surrogate behaves like an ordinary function which we can simply plot.

Additionally, you can specify the number of trees created using the parameter num\_round

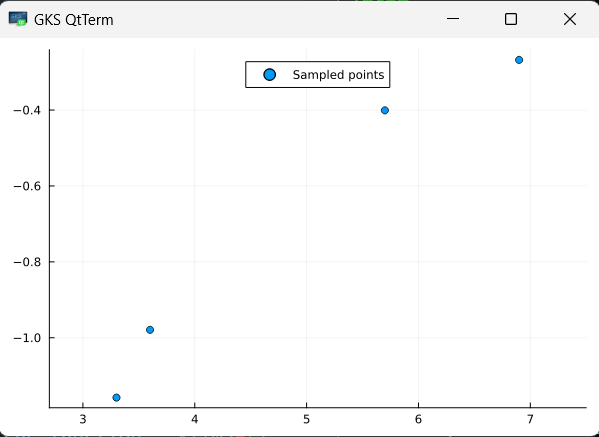
num\_round = 2

randomforest\_surrogate = RandomForestSurrogate(x, y, lower\_bound, upper\_bound, num\_round=2)

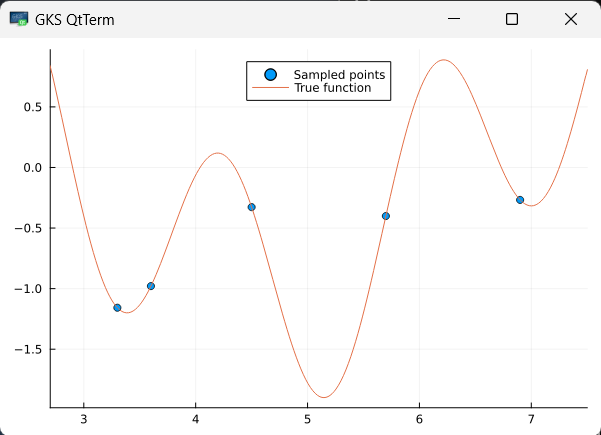
**Output:**

****

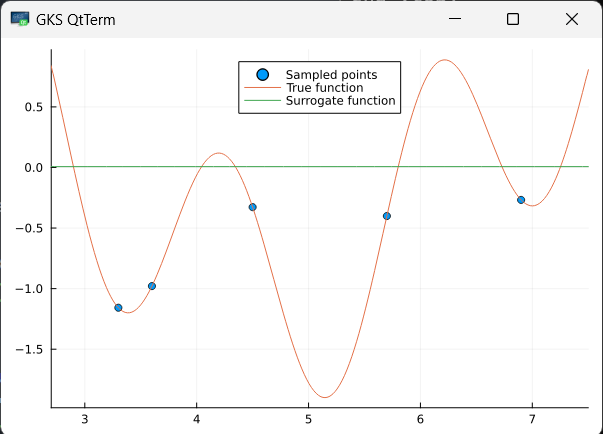
**plot(x, y, seriestype=:scatter, label="Sampled points", xlims=(lower\_bound, upper\_bound), legend=:top)**

****

**plot!(f, label="True function", xlims=(lower\_bound, upper\_bound), legend=:top)**

****

**plot!(randomforest\_surrogate, label="Surrogate function", xlims=(lower\_bound, upper\_bound), legend=:top)**

****

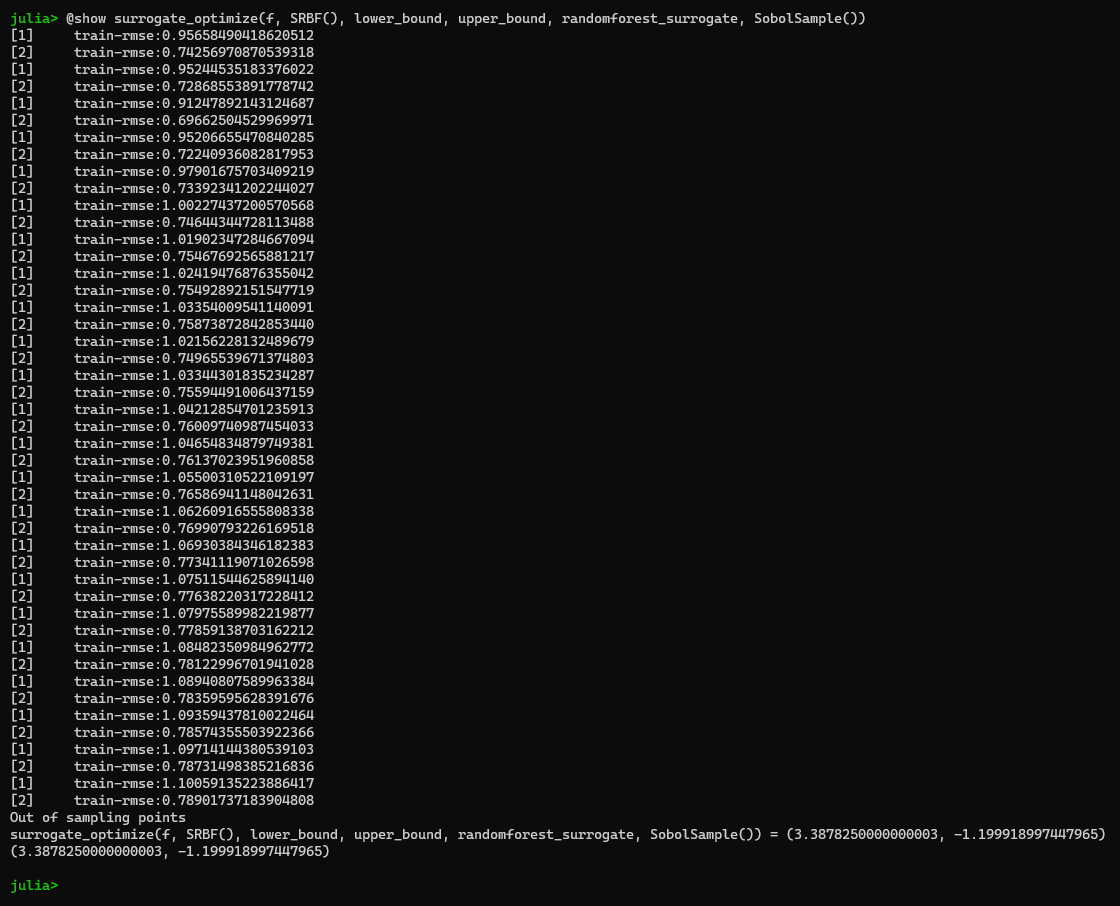
**Optimizing:**

Having built a surrogate, we can now use it to search for minima in our original function f.

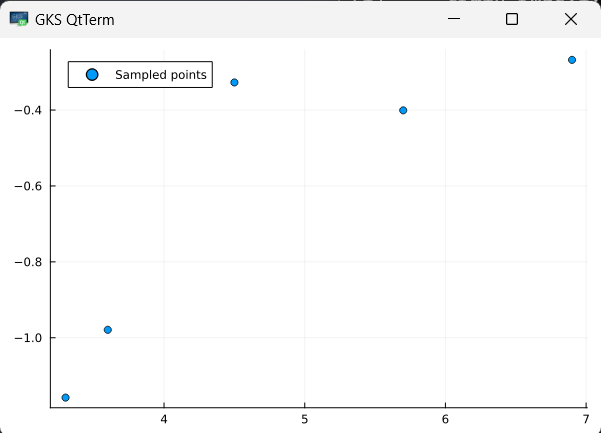
To optimize using our surrogate we call surrogate\_optimize method. We choose to use Stochastic RBF as optimization technique and again Sobol sampling as sampling technique.

@show surrogate\_optimize(f, SRBF(), lower\_bound, upper\_bound,

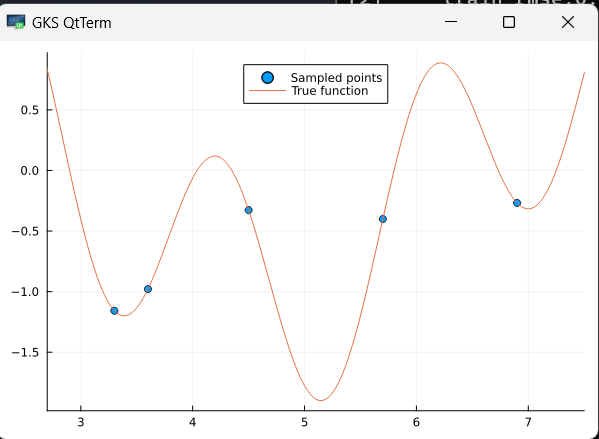
randomforest\_surrogate, SobolSample())

****

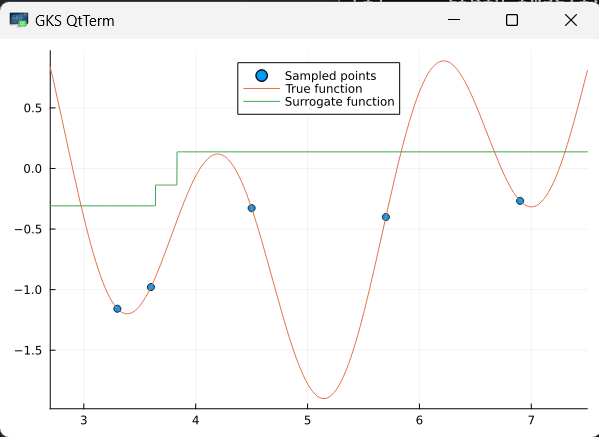
scatter(x, y, label="Sampled points")

****

plot!(f, label="True function", xlims=(lower\_bound, upper\_bound), legend=:top)

****

plot!(randomforest\_surrogate, label="Surrogate function", xlims=(lower\_bound, upper\_bound), legend=:top)

****