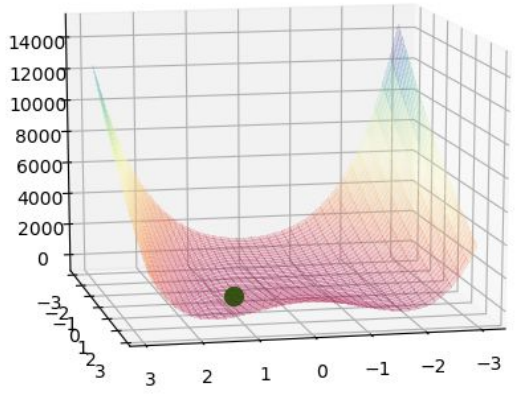
1. Searching for the global min of Rosenbrock function:

   $$f(x,y) = (1-x)^2 + 100(y-x^2)^2$$

   This function has the global min in the point (1,1):



2. We will use Gaussian processes for Global optimization.
   Bayesian optimization is an heuristic for global optimization of black-box functions. Our goal is to solve the global optimization problem of finding
   $x_{min} = argmin\, f(x) \;\; x \in X$

   Every time a new data point is collected the model is updated and the acquisition function optimized again.

   To use BO to solve this problem, we need to create a GPyOpt object in which we need to specify the following elements:

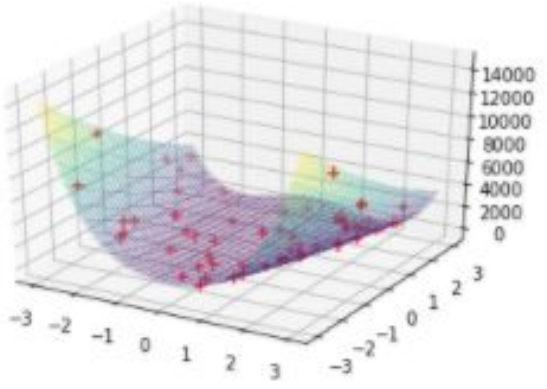
   - The function to optimize: Rosenbrock function.
   - The box constraints of the problem: $(-3, 3) \times (-3, 3)$
   - The model, by default we use a GP with a Mattern32 kernel.

     (not sure is it really good var)

     3 restarts for each iteration of optimization
   - Initial_design_numdata - number of initial data (use **initial_design_type = 'random'**)

- normalize_Y (whether to normalize the outputs before performing any optimization)
- The acquisition function (and its parameters)**

  (we use MPI)



initial data points on the full curve

results:

125 iterations

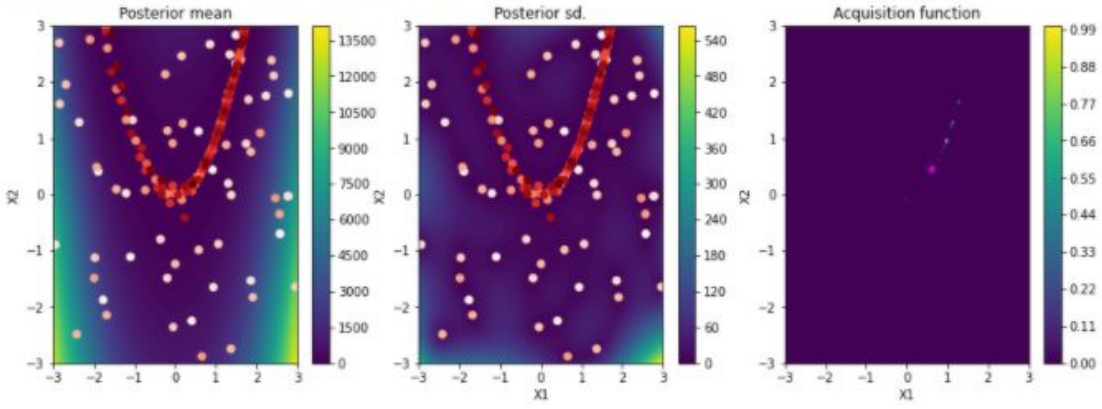
- min ([1.01,1.02])
- model:

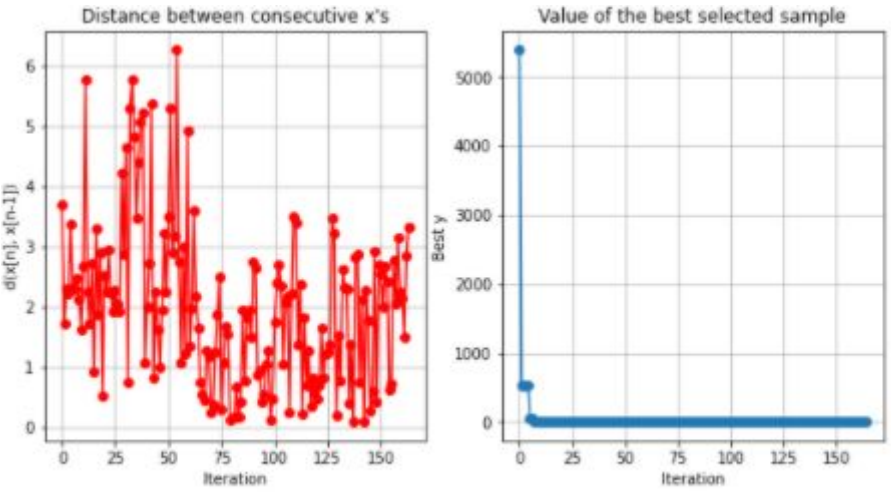
```
Model: GP regression
Objective: 530.8547817220352
Number of Parameters: 3
Number of Optimization Parameters: 2
Updates: True
```

GP_regression.	value	constraints
Mat52.variance	1522568344.3420238	+ve
Mat52.lengthscale	12.845993563572646	+ve
Gaussian_noise.variance	1e-06	+ve fixed

```
[8] myBopt.plot_acquisition()

Optimization restart 1/3, f = 530.8505689846415
Optimization restart 2/3, f = 14743.005322308718
Optimization restart 3/3, f = 1467.6988811008325
```



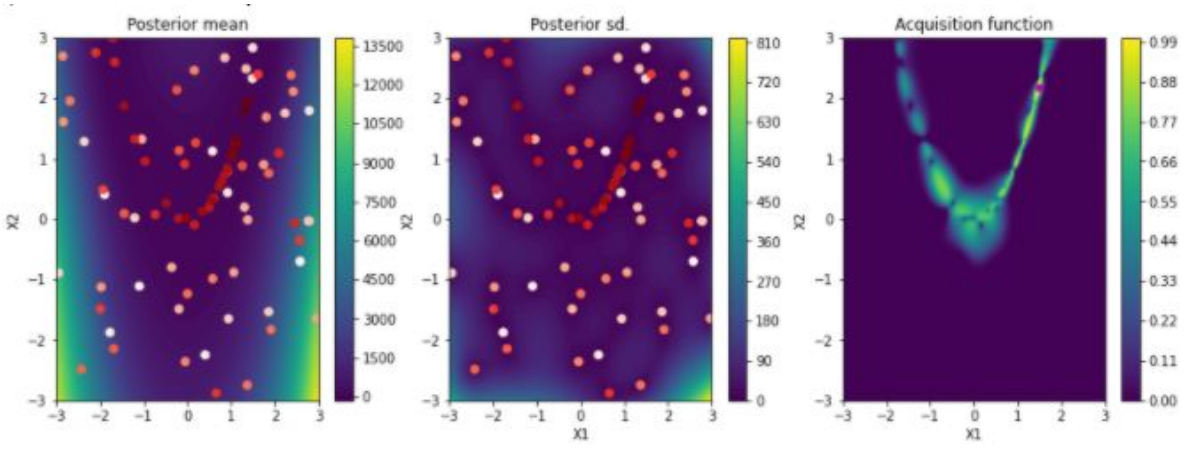


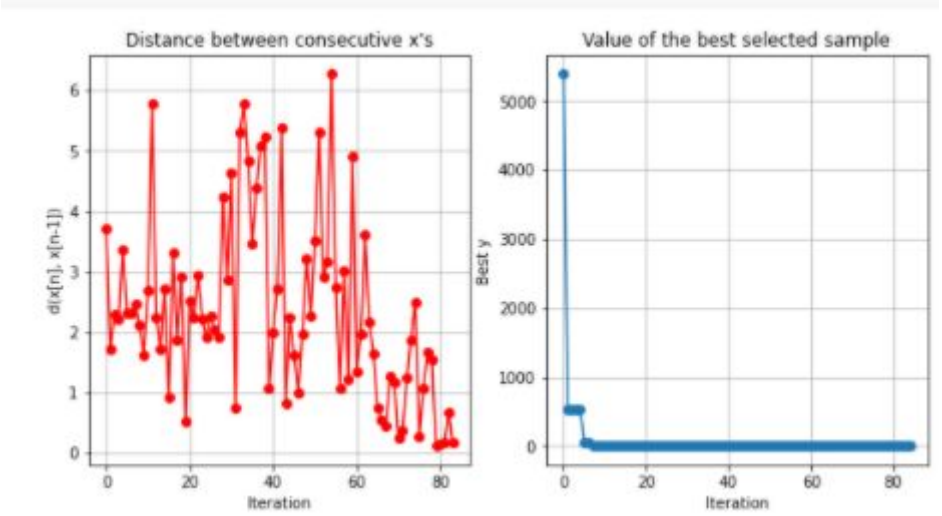
The first plot shows the distance between the last two collected observations at each iteration. This plot is useful to evaluate the convergence of the method. The second plot shows the best found value at each iteration. It is useful to compare different methods. The fastest the curve decreases the best is the method.

25 iterations

- min ([1.01,1.02])
- model

GP_regression.	value	constraints
Mat52.variance	918132876.3270549	+ve
Mat52.lengthscale	9.75057723949825	+ve
Gaussian_noise.variance	1e-06	+ve fixed



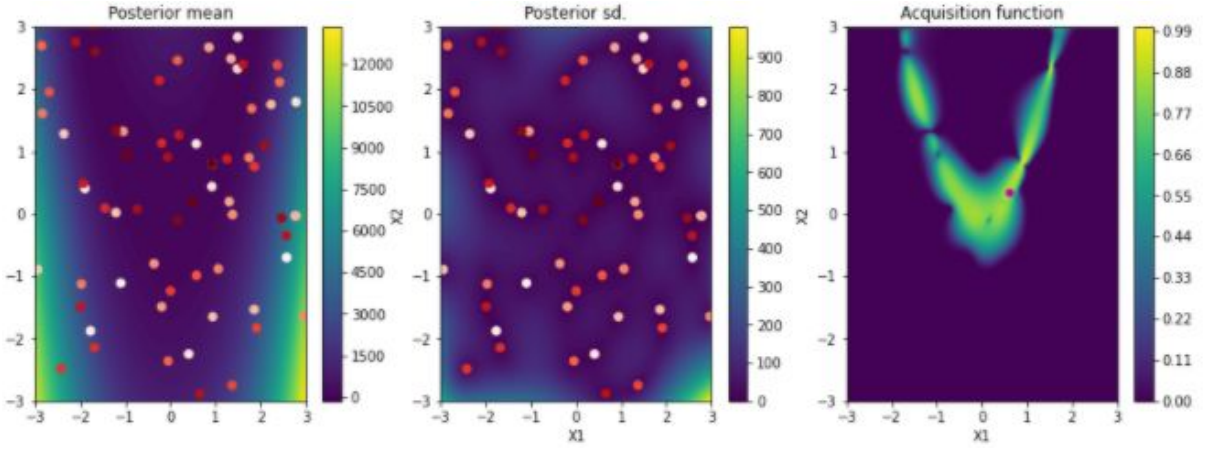


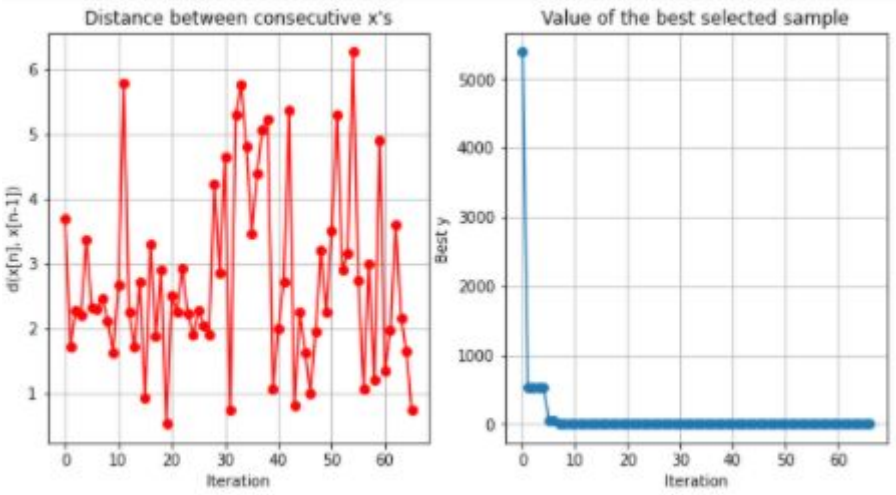
7 iterations

- min ([0.92,0.82])
- model

```
Model: GP regression
Objective: 448.55007609913304
Number of Parameters: 3
Number of Optimization Parameters: 2
Updates: True
```

GP_regression.	value	constraints
Mat52.variance	410529079.87638533	+ve
Mat52.lengthscale	7.506957194763747	+ve
Gaussian_noise.variance	1e-06	+ve fixed




Distance between consecutive x's
d(x[n], x[n-1])
Iteration
Value of the best selected sample
Best y
Iteration

** acquisition function

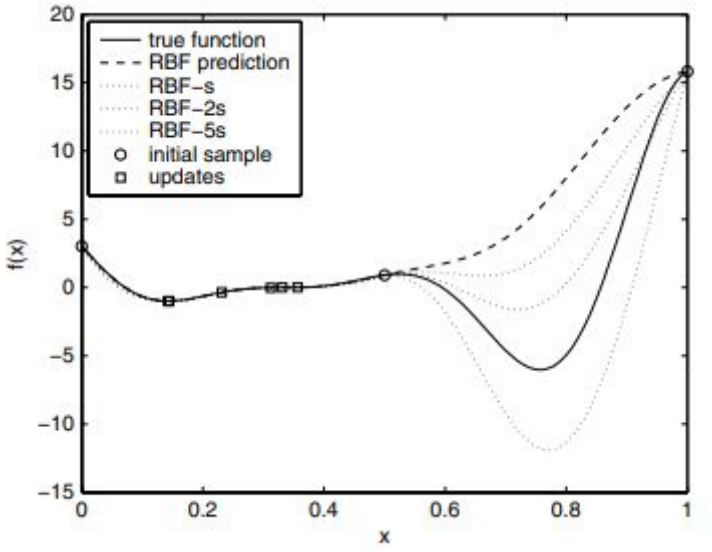
We have to understand how we can add new data points. Searching for the min on each step can give us only local min, but not global optimum. So there are some options.

1) **Statistical Lower Bound (SLB)**

   In such case we want to minimize a statistical lower bound

$$\mathrm{LB}(\mathbf{x}) = \widehat{y}(\mathbf{x}) - A\widehat{s}(\mathbf{x}),$$

   where $\widehat{y}$ - mean value (the most 'likely' prediction found as an MLE) and variance and A is a constant that controls the exploitation/exploration balance. (page 86)



   but it is unclear how one should choose the user defined parameter A to obtain a good balance between exploitation and exploration (still i don't clearly understand how GPopt provides this acquisition function.)

2) **Expected Improvement (EI)**
3) **Maximum probability of improvement (MPI)**