Introduction to Sequential Parameter Optimization

spot as an Optimizer

The `spot` loop consists of the following steps:

1. Init: Build initial design X
2. Evaluate initial design on real objective f: $y = f(X)$
3. Build surrogate: $S = S(X, y)$
4. Optimize on surrogate: $X_0 = \text{optimize}(S)$
5. Evaluate on real objective: $y_0 = f(X_0)$
6. Impute (Infill) new points: $X = X \cup X_0$, $y = y \cup y_0$.
7. Got 3.

- Central Idea:
  - Evaluation of the surrogate model `S` is much cheaper (or / and much faster) than running the real-world experiment f.

We start with a small example.

1 Example: `Spot` and the Sphere Function

```
import numpy as np
from math import inf
from spotPython.fun.objectivefunctions import analytical
from spotPython.spot import spot
from scipy.optimize import shgo
from scipy.optimize import direct
from scipy.optimize import differential_evolution
import matplotlib.pyplot as plt
```

The Objective Function: Sphere

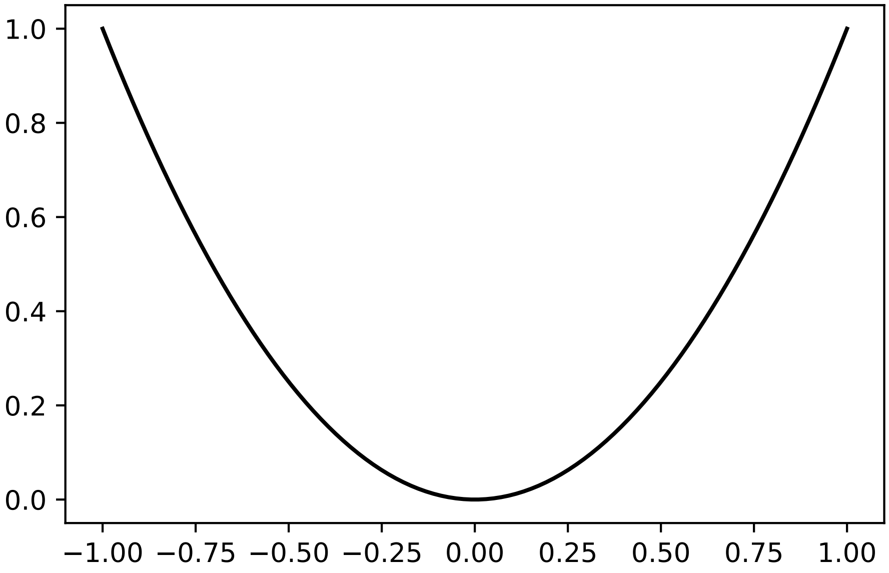
- The `spotPython` package provides several classes of objective functions.
- We will use an analytical objective function, i.e., a function that can be described by a (closed) formula:

$$f(x) = x^2$$

```
fun = analytical().fun_sphere
```

- We can apply the function `fun` to input values and plot the result:

```
x = np.linspace(-1,1,100).reshape(-1,1)
y = fun(x)
plt.figure()
plt.plot(x, y, "k")
plt.show()
```



```
spot_0 = spot.Spot(fun=fun,
                   lower = np.array([-1]),
                   upper = np.array([1]))
```

```
spot_0.run()
```

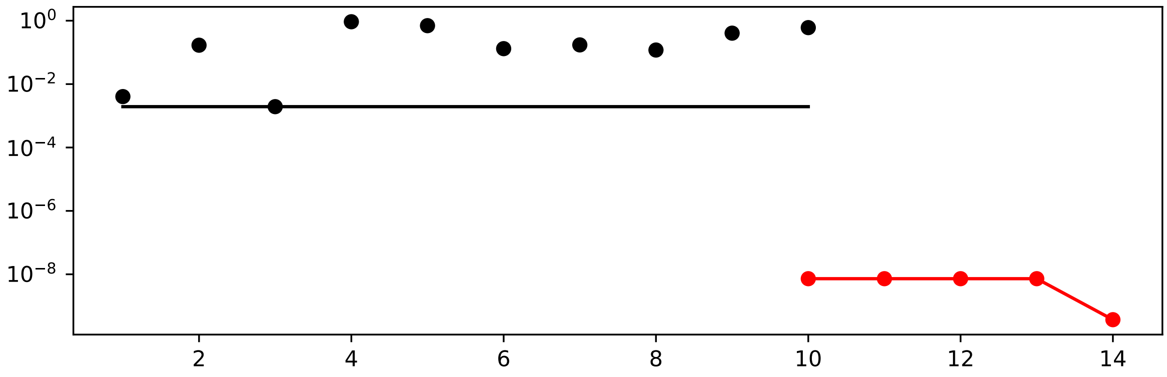
```
<spotPython.spot.spot.Spot at 0x16dc7fd90>
```

```
spot_0.print_results()
```

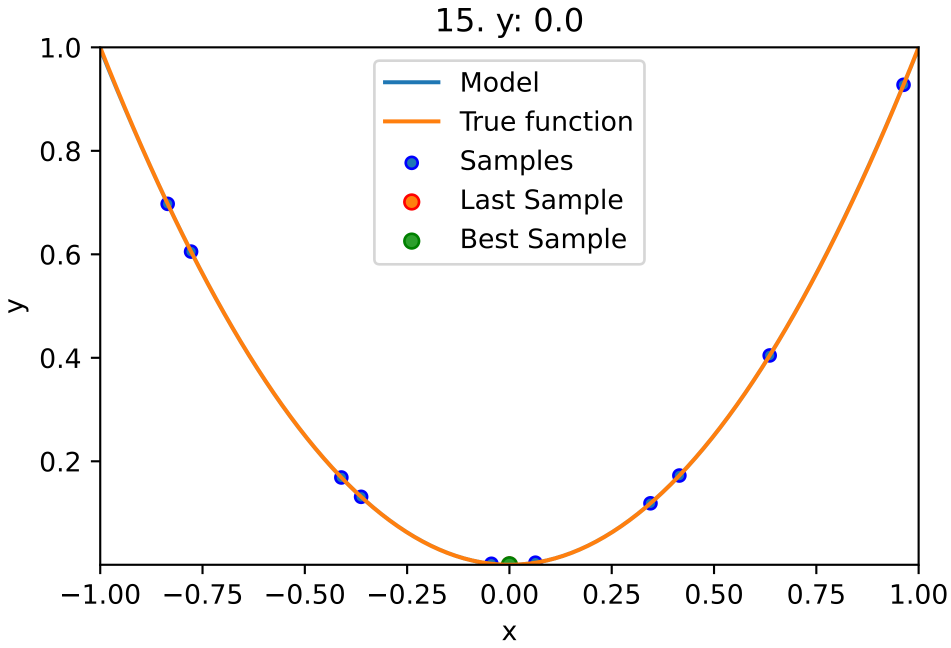
```
min y: 3.696886711914087e-10
x0: 1.922728975158508e-05

[['x0', 1.922728975158508e-05]]
```

```
spot_0.plot_progress(log_y=True)
```



```
spot_0.plot_model()
```



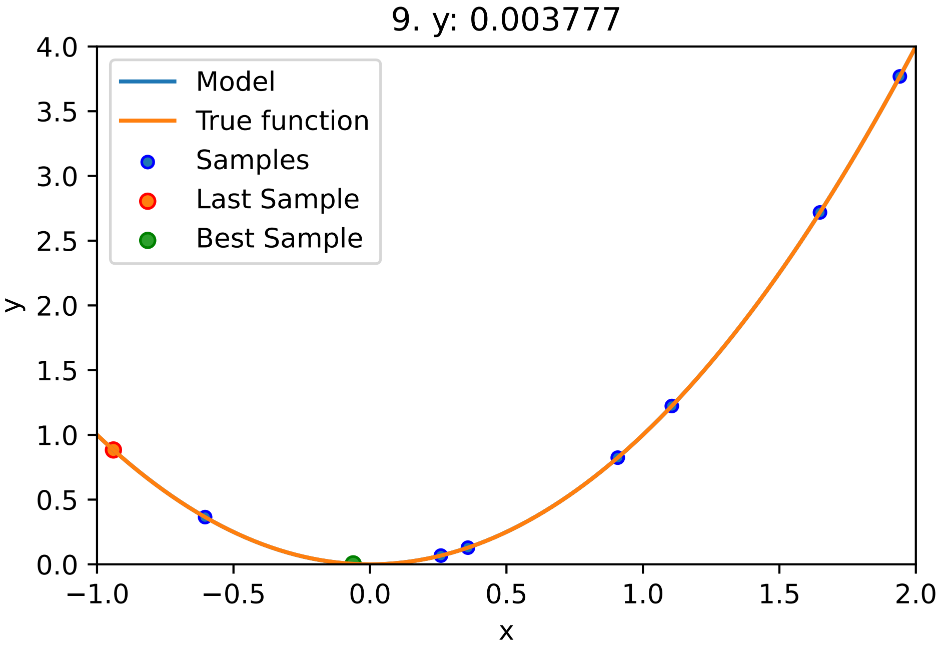
Spot Parameters: `fun_evals`, `init_size` and `show_models`

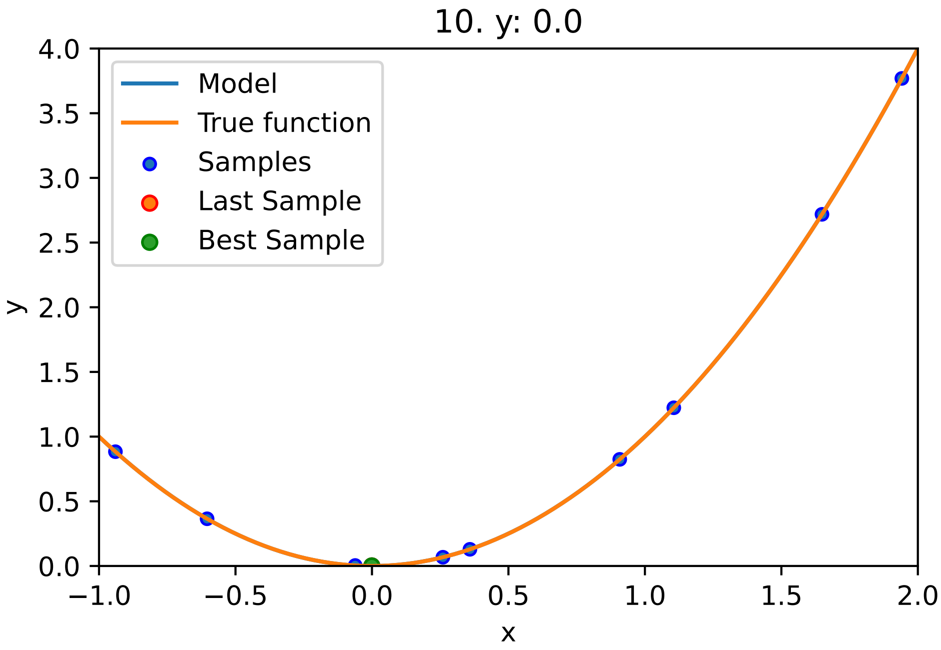
- We will modify three parameters:

1. The number of function evaluations (`fun_evals`)
2. The size of the initial design (`init_size`)
3. The parameter `show_models`, which visualizes the search process for 1-dim functions.

- The full list of the `Spot` parameters is shown in the Help System and in the notebook `spot_doc.ipynb`.

```
spot_1 = spot.Spot(fun=fun,
                   lower = np.array([-1]),
                   upper = np.array([2]),
                   fun_evals= 10,
                   seed=123,
                   show_models=True,
                   design_control={"init_size": 9})
spot_1.run()
```





```
<spotPython.spot.spot.Spot at 0x17fbeb6d0>
```

Print the Results

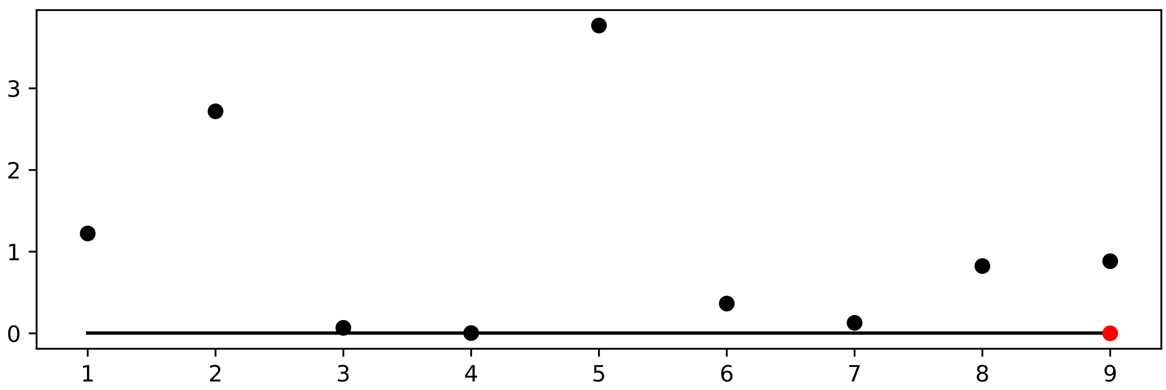
```
spot_1.print_results()
```

```
min y: 3.6779240309761575e-07
x0: -0.0006064589047063418
```

```
[['x0', -0.0006064589047063418]]
```

Show the Progress

```
spot_1.plot_progress()
```


3
2
1
0
1
2
3
4
5
6
7
8
9