Branin:

GP EI: derivation of exact partial-order GP EI derivatives wrt **x1**, **x2**

```
#pip install pyGPGO

```

```
### Import:

import numpy as np
import scipy as sp
import pandas as pd
import matplotlib.pyplot as plt
import warnings

from pyGPGO.GPGO import GPGO
from pyGPGO.surrogates.GaussianProcess import GaussianProcess
from pyGPGO.acquisition import Acquisition
from pyGPGO.covfunc import squaredExponential

from joblib import Parallel, delayed
from numpy.linalg import solve
from scipy.optimize import minimize, approx_fprime
from scipy.optimize._numdiff import _dense_difference, _compute_absolute_step, approx_d
from scipy.spatial.distance import cdist
from scipy.stats import norm
import time

warnings.filterwarnings("ignore", category=RuntimeWarning)

```

```
n_start_AcqFunc = 100 #multi-start iterations to avoid local optima in AcqFunc optimiza

```

```
### Inputs:

n_test = 500
eps = 1e-08

util_grad_exact = 'dEI_GP'
util_grad_approx = 'ExpectedImprovement'

n_init = 5 # random initialisations
iters = 20
opt = True
```

```
### Objective Function - Branin(x) 2-D:

def objfunc(x1_training, x2_training, a = 1, b = (5.1 / (4 * (np.pi) ** 2)), c = (5 / (
        return operator * ((a * (x2_training - b * x1_training ** 2 + c * x1_training -
                 s * (1 - t) * np.cos(x1_training) + s)

# Constraints:
lb_x1 = -5
ub_x1 = +10
lb_x2 = +0
ub_x2 = +15

# Input array dimension(s):
dim = 2

# 2-D inputs' parameter bounds:
param = {'x1_training': ('cont', [lb_x1, ub_x1]),
             'x2_training': ('cont', [lb_x2, ub_x2])}

# True y bounds:
operator = -1
y_lb = 0.397887 # targets global minimum
y_global_orig = y_lb * operator # targets global minimum


# Test data:
x1_test = np.linspace(lb_x1, ub_x1, n_test)
x2_test = np.linspace(lb_x2, ub_x2, n_test)

x_test = np.column_stack((x1_test,x2_test))

```

```
### Cumulative Regret Calculator:

def min_max_array(x):
    new_list = []
    for i, num in enumerate(x):
            new_list.append(np.min(x[0:i+1]))
    return new_list

```

```
### Surrogate derivatives:

cov_func = squaredExponential()

class dGaussianProcess(GaussianProcess):
    l = GaussianProcess(cov_func, optimize=opt).getcovparams()['l']
    sigmaf = GaussianProcess(cov_func, optimize=opt).getcovparams()['sigmaf']
    sigman = GaussianProcess(cov_func, optimize=opt).getcovparams()['sigman']

    def AcqGrad(self, Xstar):
        Xstar = np.atleast_2d(Xstar)
        Kstar = squaredExponential.K(self, self.X, Xstar).T
        dKstar = Kstar * cdist(self.X, Xstar).T * -1

        v = solve(self.L, Kstar.T)
        dv = solve(self.L, dKstar.T)

        ds = -2 * np.diag(np.dot(dv.T, v))
        dm = np.dot(dKstar, self.alpha)
        return ds, dm

```

```
class Acquisition_new(Acquisition):
    def __init__(self, mode, eps=1e-08, **params):

        self.params = params
        self.eps = eps

        mode_dict = {
            'dEI_GP': self.dEI_GP
        }

        self.f = mode_dict[mode]

    def dEI_GP(self, tau, mean, std, ds, dm):
        gamma = (mean - tau - self.eps) / (std + self.eps)
        gamma_h = (mean - tau) / (std + self.eps)
        dsdx = ds / (2 * (std + self.eps))
        dmdx = (dm - gamma * dsdx) / (std + self.eps)

        f = (std + self.eps) * (gamma * norm.cdf(gamma) + norm.pdf(gamma))
        df1 = f / (std + self.eps) * dsdx
        df2 = (std + self.eps) * norm.cdf(gamma) * dmdx
        df = df1 + df2

        df_arr = []

        for j in range(0, dim):
          df_arr.append([df])
        return f, np.asarray(df_arr).transpose()

    def d_eval(self, tau, mean, std, ds, dm):

        return self.f(tau, mean, std, ds, dm, **self.params)

```

```
## dGPGO:

class dGPGO(GPGO):
    n_start = n_start_AcqFunc
    eps = 1e-08

    def d_optimizeAcq(self, method='L-BFGS-B', n_start=n_start_AcqFunc):
        start_points_dict = [self._sampleParam() for i in range(n_start)]
        start_points_arr = np.array([list(s.values())
                                     for s in start_points_dict])
        x_best = np.empty((n_start, len(self.parameter_key)))
        f_best = np.empty((n_start,))
        opt = Parallel(n_jobs=self.n_jobs)(delayed(minimize)(self.acqfunc,
                                                                 x0=start_point,
                                                                 method=method,
                                                                 method=method,
                                                                 jac = True,
                                                                 bounds=self.parameter_
                                               start_points_arr)
        x_best = np.array([res.x for res in opt])
        f_best = np.array([np.atleast_1d(res.fun)[0] for res in opt])

        self.x_best = x_best
        self.f_best = f_best
        self.best = x_best[np.argmin(f_best)]
        self.start_points_arr = start_points_arr

        return x_best, f_best

    def run(self, max_iter=10, init_evals=3, resume=False):

        if not resume:
            self.init_evals = init_evals
            self._firstRun(self.init_evals)
            self.logger._printInit(self)
        for iteration in range(max_iter):
            self.d_optimizeAcq()
            self.updateGP()
            self.logger._printCurrent(self)

    def acqfunc(self, xnew, n_start=n_start_AcqFunc):
        new_mean, new_var = self.GP.predict(xnew, return_std=True)
        new_std = np.sqrt(new_var + eps)
        ds, dm = self.GP.AcqGrad(xnew)
        f, df = self.A.d_eval(-self.tau, new_mean, new_std, ds=ds, dm=dm)

        return -f, df

    def acqfunc_h(self, xnew, n_start=n_start_AcqFunc, eps=eps):
        f = self.acqfunc(xnew)[0]

        new_mean_h, new_var_h = self.GP.predict(xnew + eps, return_std=True)
        new_std_h = np.sqrt(new_var_h + eps)
        ds_h, dm_h = self.GP.AcqGrad(xnew + eps)
        f_h = self.A.d_eval(-self.tau, new_mean_h, new_std_h, ds=ds_h, dm=dm_h)[0]

        approx_grad = (-f_h - f)/eps
        return approx_grad

```

```
###Reproducible set-seeds:

run_num_1 = 1
run_num_2 = 2
run_num_3 = 3
run_num_4 = 4
run_num_5 = 5
run_num_6 = 6
run_num_7 = 7
run_num_8 = 8
```

```
run_num_9 = 9
run_num_10 = 10
run_num_11 = 11
run_num_12 = 12
run_num_13 = 13
run_num_14 = 14
run_num_15 = 15
run_num_16 = 16
run_num_17 = 17
run_num_18 = 18
run_num_19 = 19
run_num_20 = 20

```

```
start_approx = time.time()
start_approx

```

```
1623405554.8673966
```

```
### ESTIMATED GP EI GRADIENTS

np.random.seed(run_num_1)
surrogate_approx_1 = GaussianProcess(cov_func, optimize=opt)

approx_1 = GPGO(surrogate_approx_1, Acquisition(util_grad_approx), objfunc, param)
approx_1.run(init_evals=n_init, max_iter=iters)

```

```
Evaluation       Proposed point          Current eval.          Best eval.
init     [ 1.25533007 10.8048674 ].       -56.52874004692151     -27.68166689936483
init     [-4.99828438  4.53498859].       -172.66531086516164    -27.68166689936483
init     [-2.79866164  1.38507892].       -102.58290348816024    -27.68166689936483
init     [-2.20609683  5.18341091].       -28.868064601155666    -27.68166689936483
init     [0.95151211 8.08225101].         -27.68166689936483     -27.68166689936483
1        [8.41909995 1.27566317].         -5.089934878453152     -5.089934878453152
2        [10. 15.].       -145.87219087939556    -5.089934878453152
3        [-5. 15.].       -17.508299515778166    -5.089934878453152
4        [7.90355496 8.38355717].         -57.03477656850546     -5.089934878453152
5        [3.09545804 0.        ].         -5.749996691813506     -5.089934878453152
6        [ 4.44885833 15.        ].       -190.38958142014138    -5.089934878453152
7        [-5.         10.00149973].       -64.36034239837272     -5.089934878453152
8        [4.54379235 4.43734747].         -17.39998559668396     -5.089934878453152
9        [-0.62125597 15.        ].       -81.19151841324849     -5.089934878453152
10       [10.          4.60523104].       -4.510424031234777     -4.510424031234777
11       [1.1309625  3.00014358].         -15.952001765586363    -4.510424031234777
12       [ 5.12326709 10.88820044].       -106.98311627713647    -4.510424031234777
13       [5.96196546 0.        ].         -20.327814275678143    -4.510424031234777
14       [10.         11.16954914].       -68.63637436648813     -4.510424031234777
15       [7.55017529 4.08438927].         -20.361956399962384    -4.510424031234777
16       [-2.12839138 11.78945695].       -8.219849210160788     -4.510424031234777
17       [0.48977701 0.        ].         -46.05135814049239     -4.510424031234777
18       [4.20880835 7.37751993].         -38.86340555048443     -4.510424031234777
19       [10.  0.].       -10.960889035651505    -4.510424031234777
20       [-2.08692957  8.42967828].       -7.376458591176743     -4.510424031234777
```

```
### ESTIMATED GP EI GRADIENTS

np.random.seed(run_num_2)
surrogate_approx_2 = GaussianProcess(cov_func, optimize=opt)

approx_2 = GPGO(surrogate_approx_2, Acquisition(util_grad_approx), objfunc, param)
approx_2.run(init_evals=n_init, max_iter=iters)

```

```
Evaluation       Proposed point           Current eval.          Best eval.
init     [1.53992353 0.38889348].        -22.31361308916297     -6.692051508754487
init     [3.24493717 6.52983589].        -19.233109967858276    -6.692051508754487
init     [1.30551703 4.95502232].        -13.177851563387286    -6.692051508754487
init     [-1.93027049  9.2890645 ].      -6.692051508754487     -6.692051508754487
init     [-0.50517989  4.00240913].      -26.437522758780958    -6.692051508754487
1        [ 8.8924193 14.1167381].        -147.03001228842345    -6.692051508754487
2        [9.06869679 2.0164014 ].        -1.0307174678975652    -1.0307174678975652
3        [ 1.36060871 15.        ].      -131.3879388015716     -1.0307174678975652
4        [10.          7.54055547].      -22.532944128421136    -1.0307174678975652
5        [-5. 15.].       -17.508299515778166    -1.0307174678975652
6        [-4.45110035  0.        ].      -252.2413653810009     -1.0307174678975652
7        [-5.          5.78397276].      -142.76099421684668    -1.0307174678975652
8        [ 4.7508558  11.00172144].      -103.43711085858219    -1.0307174678975652
9        [6.02402314 0.        ].        -20.492390609875855    -1.0307174678975652
10       [6.85243076 4.50856244].        -29.300935515364387    -1.0307174678975652
11       [-5.         11.01027283].      -50.88016069509787     -1.0307174678975652
12       [10.  0.].       -10.960889035651505    -1.0307174678975652
13       [ 0.74703876 10.84482256].      -52.586715430202965    -1.0307174678975652
14       [ 5.17152762 15.        ].      -204.02647166124498    -1.0307174678975652
15       [-2.13059297 13.31864092].      -16.06525222576589     -1.0307174678975652
16       [3.84996892 2.82826672].        -3.791393592509783     -1.0307174678975652
17       [10.          3.74491743].      -2.4936465294415733    -1.0307174678975652
18       [ 8.50100837 10.31811021].      -76.66735855426205     -1.0307174678975652
19       [6.23279115 7.60172407].        -61.878932823740676    -1.0307174678975652
20       [-0.27817652  7.36168258].      -20.059183938001933    -1.0307174678975652
```

```
### ESTIMATED GP EI GRADIENTS

np.random.seed(run_num_3)
surrogate_approx_3 = GaussianProcess(cov_func, optimize=opt)

approx_3 = GPGO(surrogate_approx_3, Acquisition(util_grad_approx), objfunc, param)
approx_3.run(init_evals=n_init, max_iter=iters)

```

```
Evaluation       Proposed point           Current eval.          Best eval.
init     [ 3.26196854 10.62221734].      -71.68783452098575     -18.07886746449891
init     [-0.63642892  7.66241408].      -18.07886746449891     -18.07886746449891
init     [ 8.39420432 13.44439633].      -141.98653529865882    -18.07886746449891
init     [-3.11622034  3.10864317].      -83.31045707266176     -18.07886746449891
init     [-4.22799195  6.61214765].      -76.5294170688398      -18.07886746449891
1        [6.70472147 4.59545299].        -30.726791812855225    -18.07886746449891
2        [-4.7134253 15.        ].        -11.891405374053779    -11.891405374053779
3        [3.21559981 0.        ].        -5.3436612636626375    -5.3436612636626375
4        [10.  0.].       -10.960889035651505    -5.3436612636626375
5        [ 0.31730532 15.        ].      -109.22084446808685    -5.3436612636626375
6        [10.          8.34436145].      -30.47374632713747     -5.3436612636626375
7        [1.98438537 4.03074364].        -6.603721636751023     -5.3436612636626375
```

```
8        [-3.05610153 11.22529   ].       -1.1473161488487023    -1.1473161488487023
9        [-0.3179314  0.        ].       -61.61905762543009     -1.1473161488487023
10       [ 4.48556335 15.        ].       -191.1659266565978     -1.1473161488487023
11       [6.48971286 0.        ].       -20.63478820707109     -1.1473161488487023
12       [10.          3.50658284].       -2.196780044092054     -1.1473161488487023
13       [3.66130787 6.78259717].       -25.460712390739396    -1.1473161488487023
14       [6.58550354 8.83561403].       -78.6752797100621      -1.1473161488487023
15       [-5.         10.53541952].       -56.972066793440256    -1.1473161488487023
16       [-2.0252941  11.54423571].       -8.992207004607074     -1.1473161488487023
17       [-5.  0.].       -308.12909601160663    -1.1473161488487023
18       [4.2061207  2.46396377].       -6.105467036056245     -1.1473161488487023
19       [10.          5.29631355].       -7.202626721273782     -1.1473161488487023
20       [-0.20602399  4.81346771].       -21.709182908108197    -1.1473161488487023
```

```
### ESTIMATED GP EI GRADIENTS

np.random.seed(run_num_4)
surrogate_approx_4 = GaussianProcess(cov_func, optimize=opt)

approx_4 = GPGO(surrogate_approx_4, Acquisition(util_grad_approx), objfunc, param)
approx_4.run(init_evals=n_init, max_iter=iters)

```

```
Evaluation       Proposed point          Current eval.          Best eval.
init     [9.50544759 8.20848374].       -32.51676744781521     -7.247126865776948
init     [ 9.5902654  10.72223991].       -66.20566674364716     -7.247126865776948
init     [5.46593237 3.24134243].       -20.90089401680587     -7.247126865776948
init     [9.64411682 0.09345383].       -7.247126865776948     -7.247126865776948
init     [-1.20526456  6.52187299].       -15.941376982363263    -7.247126865776948
1        [ 0.5775056  14.31328634].       -102.48870217898984    -7.247126865776948
2        [-0.13367617  0.17881992].       -55.952651178085794    -7.247126865776948
3        [-5.         10.64820829].       -55.48425984802846     -7.247126865776948
4        [3.72336998 9.18654362].       -55.582264756322004    -7.247126865776948
5        [-5.          2.74886772].       -221.19381209897196    -7.247126865776948
6        [ 6.17850605 15.        ].       -212.8128335272435     -7.247126865776948
7        [-4.33551193 15.        ].       -6.574104226712752     -6.574104226712752
8        [10.          3.97109164].       -2.880426110882599     -2.880426110882599
9        [1.6937807  4.08595093].       -8.991043450037363     -2.880426110882599
10       [-0.65217279 10.24672532].       -27.57798553460698     -2.880426110882599
11       [3.95655257 0.        ].       -6.390405094788827     -2.880426110882599
12       [-5.          6.88872878].       -118.78555885693459    -2.880426110882599
13       [10. 15.].       -145.87219087939556    -2.880426110882599
14       [6.38841813 6.50802932].       -48.74409560082792     -2.880426110882599
15       [6.8103764 0.       ].       -19.62705637605663     -2.880426110882599
16       [-1.20049951  3.47913076].       -34.798120813534354    -2.880426110882599
17       [ 6.24419694 11.24491928].       -122.53539981091201    -2.880426110882599
18       [-2.66661975 13.01160242].       -4.879342343159136     -2.880426110882599
19       [1.71815229 6.68545222].       -17.8233842272911      -2.880426110882599
20       [8.49368638 2.70521121].       -5.084451999242475     -2.880426110882599
```

```
### ESTIMATED GP EI GRADIENTS

np.random.seed(run_num_5)
surrogate_approx_5 = GaussianProcess(cov_func, optimize=opt)

approx_5 = GPGO(surrogate_approx_5, Acquisition(util_grad_approx), objfunc, param)
approx_5.run(init_evals=n_init, max_iter=iters)

```

```
Evaluation       Proposed point          Current eval.          Best eval.
init     [-1.67010243 13.06098459].       -25.390690545664548    -25.311309190989615
init     [-1.89921267 13.77916362].       -25.311309190989615    -25.311309190989615
init     [2.32616783 9.17615794].         -41.60147975414021     -25.311309190989615
init     [6.48861785 7.77626982].         -63.81281921768236     -25.311309190989615
init     [-0.54799248  2.81581843].       -34.966203644931994    -25.311309190989615
1        [10.  0.].       -10.960889035651505    -10.960889035651505
2        [ 8.90352463 15.        ].       -168.84733860933238    -10.960889035651505
3        [-5.         7.1442263].         -113.58828844509826    -10.960889035651505
4        [4.40256958 0.        ].         -9.313533992579627     -9.313533992579627
5        [-5.  0.].       -308.12909601160663    -9.313533992579627
6        [ 3.28455811 15.        ].       -165.20362005987207    -9.313533992579627
7        [10.         4.49481459].        -4.1687809039062325    -4.1687809039062325
8        [3.73428711 4.15207739].         -7.2976825774345775    -4.1687809039062325
9        [10.         10.40600568].       -56.74827627602043     -4.1687809039062325
10       [7.10300969 2.63656064].         -18.578743850140377    -4.1687809039062325
11       [-5.         11.41757667].       -46.01415453016331     -4.1687809039062325
12       [-0.78455773  6.78778412].       -17.087449984111828    -4.1687809039062325
13       [ 5.9448145  11.72116086].       -131.7813154964645     -4.1687809039062325
14       [-5. 15.].       -17.508299515778166    -4.1687809039062325
15       [1.28674293 0.        ].         -30.046344055982637    -4.1687809039062325
16       [-1.56108233  9.83208067].       -11.159786412525692    -4.1687809039062325
17       [-3.80619215  3.75121164].       -106.03406334088206    -4.1687809039062325
18       [10.         6.91897176].        -17.278315330649434    -4.1687809039062325
19       [1.9440385  5.79489555].         -12.26209978145576     -4.1687809039062325
20       [ 1.12003286 12.06966608].       -73.3222899478788      -4.1687809039062325
```

```
### ESTIMATED GP EI GRADIENTS

np.random.seed(run_num_6)
surrogate_approx_6 = GaussianProcess(cov_func, optimize=opt)

approx_6 = GPGO(surrogate_approx_6, Acquisition(util_grad_approx), objfunc, param)
approx_6.run(init_evals=n_init, max_iter=iters)

```

```
Evaluation       Proposed point          Current eval.          Best eval.
init     [8.39290227 4.97969708].         -15.553860636684597    -15.31731051317483
init     [7.31843685 0.62544939].         -15.31731051317483     -15.31731051317483
init     [-3.3851498   8.92578096].       -16.222288919361837    -15.31731051317483
init     [2.94726043 6.28211143].         -15.406373581304088    -15.31731051317483
init     [0.03111774 9.33779148].         -31.07053245856209     -15.31731051317483
1        [ 9.72522083 14.5299912 ].       -139.8297584868663     -15.31731051317483
2        [-4.42013634  0.37777   ].       -237.69785665817545    -15.31731051317483
3        [-3.99671519 14.86986649].       -3.898129686703599     -3.898129686703599
4        [ 3.01682823 14.83374236].       -155.70998935074863    -3.898129686703599
5        [1.76387394 0.        ].         -21.078930203265006    -3.898129686703599
6        [ 6.51112433 10.00143544].       -98.34083324613164     -3.898129686703599
7        [-1.36588047  4.53364094].       -27.017874835590963    -3.898129686703599
8        [-1.11716557 13.18710515].       -41.747893399911376    -3.898129686703599
9        [-5.         5.34411144].        -152.98628997678327    -3.898129686703599
10       [4.41768041 2.92039548].         -9.25643352742401      -3.898129686703599
11       [10.         8.19251001].        -28.874605179549697    -3.898129686703599
12       [-5.         11.99773964].       -39.65591448874086     -3.898129686703599
13       [10.         2.12150829].        -2.720091790618171     -2.720091790618171
14       [ 2.9295331  10.93621466].       -72.69356723271218     -2.720091790618171
15       [1.53683993 3.24064477].         -10.708553016980726    -2.720091790618171
16       [5.84836947 6.3538518 ].         -46.20047447530566     -2.720091790618171
```

```
17       [10.  0.].       -10.960889035651505    -2.720091790618171
18       [ 6.33172196 13.56051827].       -174.8092369131531     -2.720091790618171
19       [-0.84739394  1.32557388].       -53.75975213277636     -2.720091790618171
20       [4.6523125 0.       ].       -11.36027487594677     -2.720091790618171
```

```
### ESTIMATED GP EI GRADIENTS

np.random.seed(run_num_7)
surrogate_approx_7 = GaussianProcess(cov_func, optimize=opt)

approx_7 = GPGO(surrogate_approx_7, Acquisition(util_grad_approx), objfunc, param)
approx_7.run(init_evals=n_init, max_iter=iters)

```

```
Evaluation       Proposed point          Current eval.          Best eval.
init     [-3.85537566 11.69878188].       -8.299317850233932     -5.211335324193094
init     [ 1.57613847 10.85197767].       -59.50408380701259     -5.211335324193094
init     [9.66984268 8.07743806].       -29.714921414987273    -5.211335324193094
init     [2.51680695 1.080767  ].       -5.211335324193094     -5.211335324193094
init     [-0.9734153   7.49823751].       -15.431057204948033    -5.211335324193094
1        [-4.97859679  1.38393519].       -260.3293952774437     -5.211335324193094
2        [ 9.25668108 14.87713055].       -157.79168966390583    -5.211335324193094
3        [9.84403446 2.23618575].       -1.607946967220208     -1.607946967220208
4        [4.55900388 5.89375631].       -28.465589739949205    -1.607946967220208
5        [ 4.07549503 15.        ].       -182.26231569631855    -1.607946967220208
6        [6.63656991 0.        ].       -20.27975546004532     -1.607946967220208
7        [-1.10321767 15.        ].       -64.55271949626872     -1.607946967220208
8        [-5.          6.49338833].       -127.0847837975716     -1.607946967220208
9        [-1.00615862  3.53548229].       -32.75005148348958     -1.607946967220208
10       [ 6.15564868 10.11747234].       -100.87425641230558    -1.607946967220208
11       [-5. 15.].       -17.508299515778166    -1.607946967220208
12       [8.07676988 4.42951178].       -16.03995360443043     -1.607946967220208
13       [-0.66114141  0.        ].       -68.11258227581459     -1.607946967220208
14       [10.  0.].       -10.960889035651505    -1.607946967220208
15       [4.83706984 2.64288996].       -12.933239102013518    -1.607946967220208
16       [10.         11.254153].       -70.025382571958       -1.607946967220208
17       [1.47317618 5.68982396].       -14.012718146993524    -1.607946967220208
18       [-1.79688796 10.29049125].       -8.874761496408627     -1.607946967220208
19       [-5.          9.86428721].       -66.35114962135708     -1.607946967220208
20       [2.93515195 8.16198683].       -33.3257388312568      -1.607946967220208
```

```
### ESTIMATED GP EI GRADIENTS

np.random.seed(run_num_8)
surrogate_approx_8 = GaussianProcess(cov_func, optimize=opt)

approx_8 = GPGO(surrogate_approx_8, Acquisition(util_grad_approx), objfunc, param)
approx_8.run(init_evals=n_init, max_iter=iters)

```

```
Evaluation       Proposed point          Current eval.          Best eval.
init     [ 8.10144104 14.52810994].       -175.17297136347514    -15.416245468470875
init     [8.0379181  7.96283537].       -49.32173799887473     -15.416245468470875
init     [-1.50907508  0.17098206].       -83.26753461673219     -15.416245468470875
init     [1.45703227 6.0352704 ].       -15.416245468470875    -15.416245468470875
init     [2.84012007 7.17587694].       -22.49093277852639     -15.416245468470875
1        [-4.98884166 14.83408411].       -18.008434758014253    -15.416245468470875
```

```
    2        [10.  0.].       -10.960889035651505    -10.960889035651505
    3        [-4.8401206   6.42823123].       -117.34273274148853    -10.960889035651505
    4        [ 1.46752997 14.78454975].       -128.53831736599975    -10.960889035651505
    5        [4.44479756 0.89726458].       -7.798462032411121     -7.798462032411121
    6        [-1.53733241 10.48501503].       -13.324418197268233    -7.798462032411121
    7        [ 4.59101393 11.15592345].       -103.70255460707435    -7.798462032411121
    8        [7.68054207 3.49376241].       -16.054612311234727    -7.798462032411121
    9        [-5.          2.14980151].       -238.85191822262627    -7.798462032411121
    10       [-5.         10.92816854].       -51.90123254600216     -7.798462032411121
    11       [1.64304625 2.61760521].       -10.55264701517783     -7.798462032411121
    12       [10.         11.01204094].       -66.08857250704284     -7.798462032411121
    13       [-1.38500482  4.34595593].       -28.63424903579045     -7.798462032411121
    14       [4.56603278 4.24557199].       -16.548223116171087    -7.798462032411121
    15       [7.05867001 0.        ].       -18.302394626226477    -7.798462032411121
    16       [-2.10201367 13.64598882].       -19.046641323885666    -7.798462032411121
    17       [10.          5.19673156].       -6.755789217465072     -6.755789217465072
    18       [-1.14446026  7.66355057].       -14.077841644959545    -6.755789217465072
    19       [ 1.22722397 10.1289271 ].       -47.89777521583031     -6.755789217465072
    20       [2.03523621 0.        ].       -16.56213832971314     -6.755789217465072
```

```python
### ESTIMATED GP EI GRADIENTS

np.random.seed(run_num_9)
surrogate_approx_9 = GaussianProcess(cov_func, optimize=opt)

approx_9 = GPGO(surrogate_approx_9, Acquisition(util_grad_approx), objfunc, param)
approx_9.run(init_evals=n_init, max_iter=iters)

```

```
    Evaluation       Proposed point          Current eval.          Best eval.
    init     [-4.84438769  7.52811888].       -96.15564306553264     -3.4640248583909496
    init     [2.4365994  2.00744293].       -3.4640248583909496    -3.4640248583909496
    init     [-2.86833372  3.27838013].       -70.46930174488207     -3.4640248583909496
    init     [1.27762271 3.72151753].       -12.982825924833865    -3.4640248583909496
    init     [-3.73910523  5.1824796 ].       -75.58547400560856     -3.4640248583909496
    1        [ 7.05609655 14.75038872].       -200.4388688685465     -3.4640248583909496
    2        [9.79053515 5.9968998 ].       -11.24805159216164     -3.4640248583909496
    3        [-1.16788554 15.        ].       -62.276893026499415    -3.4640248583909496
    4        [7.94468781 0.        ].       -11.40885179920395     -3.4640248583909496
    5        [2.83905495 9.81057089].       -53.94615567523124     -3.4640248583909496
    6        [10.         10.54613846].       -58.8427331976896      -3.4640248583909496
    7        [5.40021154 5.51823191].       -34.980375915287354    -3.4640248583909496
    8        [-5.         12.38435749].       -35.79258827471595     -3.4640248583909496
    9        [-1.33336996 10.31600592].       -16.116547978109825    -3.4640248583909496
    10       [-0.29866745  0.        ].       -61.25647245562254     -3.4640248583909496
    11       [-5.  0.].       -308.12909601160663    -3.4640248583909496
    12       [ 2.76587243 14.06021978].       -132.71978661440204    -3.4640248583909496
    13       [4.46689221 0.        ].       -9.82238601092727      -3.4640248583909496
    14       [10.          2.60845129].       -2.098775109155979     -2.098775109155979
    15       [0.00880137 6.9572859 ].       -20.545132778190705    -2.098775109155979
    16       [6.55675899 9.09280976].       -82.8366086031989      -2.098775109155979
    17       [7.57104949 3.00824675].       -15.413014515803981    -2.098775109155979
    18       [4.49125841 2.59442006].       -9.185908487061909     -2.098775109155979
    19       [-5. 15.].       -17.508299515778166    -2.098775109155979
    20       [2.66723923 6.41723987].       -15.469941833671024    -2.098775109155979
```

```python
### ESTIMATED GP EI GRADIENTS

np.random.seed(run_num_10)
np.random.seed(run_num_10)
surrogate_approx_10 = GaussianProcess(cov_func, optimize=opt)

approx_10 = GPGO(surrogate_approx_10, Acquisition(util_grad_approx), objfunc, param)
approx_10.run(init_evals=n_init, max_iter=iters)

```

```
Evaluation       Proposed point           Current eval.         Best eval.
init     [6.56980965 0.31127924].         -19.863985024602144    -2.710610964564512
init     [ 4.50472352 11.23205824].       -103.66999147252169    -2.710610964564512
init     [2.47760518 3.37194968].         -2.710610964564512     -2.710610964564512
init     [-2.02905703 11.40796068].       -8.463930835255017     -2.710610964564512
init     [-2.46333745  1.32509721].       -90.49481266026973     -2.710610964564512
1        [8.70372597 6.86774268].         -27.130126017218842    -2.710610964564512
2        [ 9.45036259 14.63089076].       -147.64044674485945    -2.710610964564512
3        [-5.          6.13548754].       -134.86763853640946    -2.710610964564512
4        [-5. 15.].        -17.508299515778166    -2.710610964564512
5        [0.53535976 7.40116606].         -23.17014149092612     -2.710610964564512
6        [ 0.92476669 15.        ].       -123.13791799284002    -2.710610964564512
7        [7.30917046 4.63031131].         -26.277167965113733    -2.710610964564512
8        [2.04065446 0.        ].         -16.477681249661387    -2.710610964564512
9        [10.          1.71596188].       -3.5994960877383804    -2.710610964564512
10       [10.         10.41835981].       -56.931345305911215    -2.710610964564512
11       [4.29173593 6.92088674].         -34.93673328169899     -2.710610964564512
12       [-5.         10.40160815].       -58.770182824044745    -2.710610964564512
13       [ 5.26787321 15.        ].       -205.48018946265273    -2.710610964564512
14       [-0.14217217  4.1899552 ].       -23.662468280046603    -2.710610964564512
15       [ 0.80211348 10.84796118].       -53.17438859113578     -2.710610964564512
16       [4.54649196 2.76245714].         -10.178194925001506    -2.710610964564512
17       [-2.19319068  8.47435323].       -7.083894788963509     -2.710610964564512
18       [10.  0.].        -10.960889035651505    -2.710610964564512
19       [10.          3.72658725].       -2.4667819700235754    -2.4667819700235754
20       [-2.42863439 14.00519658].       -14.147153055354629    -2.4667819700235754
```

```
### ESTIMATED GP EI GRADIENTS

np.random.seed(run_num_11)
surrogate_approx_11 = GaussianProcess(cov_func, optimize=opt)

approx_11 = GPGO(surrogate_approx_11, Acquisition(util_grad_approx), objfunc, param)
approx_11.run(init_evals=n_init, max_iter=iters)

```

```
Evaluation       Proposed point           Current eval.         Best eval.
init     [-2.29595467  0.29212862].       -104.49282729548965    -22.37435843952312
init     [ 1.9482779  10.87400894].       -62.47758067812407     -22.37435843952312
init     [1.30305407 7.28140647].         -22.37435843952312     -22.37435843952312
init     [-4.80828778  7.31057411].       -97.94503054415523     -22.37435843952312
init     [ 9.12709979 12.76192634].       -111.62900800771942    -22.37435843952312
1        [10.  0.].        -10.960889035651505    -10.960889035651505
2        [-3.99917487 15.        ].       -4.041425554956509     -4.041425554956509
3        [8.0701298   6.37438228].        -31.02861060087873     -4.041425554956509
4        [3.91721456 1.58212528].         -3.171689933951897     -3.171689933951897
5        [ 4.54037959 15.        ].       -192.31399373181276    -3.171689933951897
6        [-2.60740522 11.41248237].       -1.883391194202435     -1.883391194202435
7        [-0.2309989  15.        ].        -93.74563162359628     -1.883391194202435
8        [0.85044034 3.33404553].         -18.31052349905193     -1.883391194202435
9        [5.9388425   9.75317472].        -93.840494574843       -1.883391194202435
10       [7.16333845 2.43331853].         -17.569435925180144    -1.883391194202435
```

```
11       [4.37872351 5.09898827].        -19.7509396429289       -1.883391194202435
12       [-5.          3.26329121].      -206.60344467183972     -1.883391194202435
13       [-5.        11.8792143].        -40.90016582675334      -1.883391194202435
14       [1.7116418 0.       ].          -22.006046182138995     -1.883391194202435
15       [-1.5583204   9.38545838].      -10.469791676121407     -1.883391194202435
16       [10.          9.08131724].      -38.88960866070064      -1.883391194202435
17       [10.          3.55876887].      -2.252067935644445      -1.883391194202435
18       [-1.55235116  5.48038341].      -21.07745556846134      -1.883391194202435
19       [5.55369622 0.        ].        -18.470748386474238     -1.883391194202435
20       [-1.59819722 11.98733714].      -19.432426261227548     -1.883391194202435
```

```
### ESTIMATED GP EI GRADIENTS

np.random.seed(run_num_12)
surrogate_approx_12 = GaussianProcess(cov_func, optimize=opt)

approx_12 = GPGO(surrogate_approx_12, Acquisition(util_grad_approx), objfunc, param)
approx_12.run(init_evals=n_init, max_iter=iters)

```

```
Evaluation       Proposed point          Current eval.           Best eval.
init     [-2.68755736 11.10074545].      -1.3827702760021356     -0.5499315281120278
init     [-1.05027477  8.0060909 ].      -14.81231853080056      -0.5499315281120278
init     [-4.78137556 13.78120512].      -18.401131000214548     -0.5499315281120278
init     [8.51072281 0.50132141].        -5.855172364344769      -0.5499315281120278
init     [9.35424004 2.05813982].        -0.5499315281120278     -0.5499315281120278
1        [10.          4.42222996].      -3.9574775283256747     -0.5499315281120278
2        [ 8.68497867 14.25940608].      -155.1276557842789      -0.5499315281120278
3        [-5.  0.].       -308.12909601160663    -0.5499315281120278
4        [5.30078686 8.4325947 ].        -67.7353026392114       -0.5499315281120278
5        [1.81449881 0.93833083].        -14.438525235006493     -0.5499315281120278
6        [ 2.69349751 13.83339951].      -126.40556738566146     -0.5499315281120278
7        [-5.          5.45174749].      -150.44835450431262     -0.5499315281120278
8        [10.          9.18338707].      -40.140861437849495     -0.5499315281120278
9        [5.48959113 3.59983853].        -22.705708490302452     -0.5499315281120278
10       [1.55403075 5.02943763].        -11.578916130929688     -0.5499315281120278
11       [-1.46045028  2.84645165].      -44.15978588070091      -0.5499315281120278
12       [-1.32998724 15.        ].      -56.57568341451988      -0.5499315281120278
13       [-5.          9.73229132].      -68.30180347015676      -0.5499315281120278
14       [5.24225669 0.        ].        -16.309559190040787     -0.5499315281120278
15       [1.67646168 9.9449355 ].        -48.05016383270482      -0.5499315281120278
16       [8.02096249 6.04308816].        -28.63295089854334      -0.5499315281120278
17       [ 5.87835294 11.83413301].      -133.86975855122358     -0.5499315281120278
18       [-1.37390968 11.50921611].      -21.35684592378965      -0.5499315281120278
19       [10.          2.29060641].      -2.4505834659722137     -0.5499315281120278
20       [-0.8916461  0.        ].        -72.60894139200812      -0.5499315281120278
```

```
### ESTIMATED GP EI GRADIENTS

np.random.seed(run_num_13)
surrogate_approx_13 = GaussianProcess(cov_func, optimize=opt)

approx_13 = GPGO(surrogate_approx_13, Acquisition(util_grad_approx), objfunc, param)
approx_13.run(init_evals=n_init, max_iter=iters)

```

```
Evaluation       Proposed point          Current eval.           Best eval.
```

```
init     [6.66553616 3.5631183 ].         -24.823670200298267    -18.038943061558626
init     [ 7.36417799 14.48623797].       -188.7803334745195     -18.038943061558626
init     [9.58901671 6.80173871].         -18.038943061558626    -18.038943061558626
init     [ 4.13563694 11.63289772].       -104.87319703906111    -18.038943061558626
init     [ 4.62420017 10.83027344].       -98.03271225790056     -18.038943061558626
1        [-5.         10.55852118].       -56.66525878591628     -18.038943061558626
2        [-4.14720717  0.94877071].       -197.3342113761513     -18.038943061558626
3        [-0.12335059  6.35840654].       -19.554794556716274    -18.038943061558626
4        [-0.99272771 15.        ].       -68.43031894475294     -18.038943061558626
5        [2.20276677 0.        ].         -14.068419772396153    -14.068419772396153
6        [10.  0.].       -10.960889035651505    -10.960889035651505
7        [-5.          5.84821499].       -141.29996322506628    -10.960889035651505
8        [-0.41220542 10.52481489].       -33.595854385590286    -10.960889035651505
9        [10.         10.78931988].       -62.57059363401944     -10.960889035651505
10       [3.90491025 6.50962489].         -25.668584447722168    -10.960889035651505
11       [-5. 15.].       -17.508299515778166    -10.960889035651505
12       [6.00936985 0.        ].         -20.45653172757895     -10.960889035651505
13       [-0.14059623  2.69846735].       -31.95310561715899     -10.960889035651505
14       [ 2.81601883 15.        ].       -156.08924891414276    -10.960889035651505
15       [10.          3.30737489].       -2.0358111549603777    -2.0358111549603777
16       [3.32760876 3.07412193].         -1.4466053975072377    -1.4466053975072377
17       [6.86999407 8.0765211 ].         -65.79031832440131     -1.4466053975072377
18       [-2.45258807  8.26924644].       -8.402396403233654     -1.4466053975072377
19       [1.88664671 8.60481495].         -33.51597964124437     -1.4466053975072377
20       [-0.6700678  0.        ].         -68.2837087849351      -1.4466053975072377
```

```
### ESTIMATED GP EI GRADIENTS

np.random.seed(run_num_14)
surrogate_approx_14 = GaussianProcess(cov_func, optimize=opt)

approx_14 = GPGO(surrogate_approx_14, Acquisition(util_grad_approx), objfunc, param)
approx_14.run(init_evals=n_init, max_iter=iters)

```

```
Evaluation       Proposed point           Current eval.          Best eval.
init     [ 2.70915016 11.59747578].       -81.58264375064097     -1.4149920024014744
init     [8.05641529 0.12070423].         -10.148649076616518    -1.4149920024014744
init     [-0.35396112 14.36405609].       -79.6056658956529      -1.4149920024014744
init     [2.69675068 4.77426637].         -5.855607664288365     -1.4149920024014744
init     [3.08799906 3.31882414].         -1.4149920024014744    -1.4149920024014744
1        [-3.9690762   0.55943317].       -193.7395503314444     -1.4149920024014744
2        [9.91083167 7.9391897 ].         -26.74726853005628     -1.4149920024014744
3        [10. 15.].       -145.87219087939556    -1.4149920024014744
4        [-3.75022473  8.67485872].       -28.241223279942684    -1.4149920024014744
5        [1.31384966 0.        ].         -29.513114536508205    -1.4149920024014744
6        [-5.         13.37062657].       -27.291209415000157    -1.4149920024014744
7        [6.99460041 4.39917956].         -27.584513770007984    -1.4149920024014744
8        [-1.42817233  4.8023786 ].       -25.30852675190059     -1.4149920024014744
9        [ 7.19418072 11.22622498].       -115.68611123966527    -1.4149920024014744
10       [ 5.16524605 15.        ].       -203.92798169924106    -1.4149920024014744
11       [4.60182334 7.71580065].         -48.6824054442007      -1.4149920024014744
12       [0.28095521 8.25502638].         -26.472397154229505    -1.4149920024014744
13       [-5.          4.91912178].       -163.23342236713881    -1.4149920024014744
14       [4.82272532 1.15777313].         -11.08665119630106     -1.4149920024014744
15       [10.          2.74833398].       -2.007973346672351     -1.4149920024014744
16       [-1.68775573 11.12722116].       -13.177203060888006    -1.4149920024014744
17       [10.          4.80310908].       -5.1836895987670975    -1.4149920024014744
18       [0.96825453 2.98230028].         -17.994811333359575    -1.4149920024014744
```

```
19       [10.         1.2353938].         -5.067418935910082     -1.4149920024014744
20       [4.19520379 3.51081173].         -8.916050132306102     -1.4149920024014744
```

```
### ESTIMATED GP EI GRADIENTS

np.random.seed(run_num_15)
surrogate_approx_15 = GaussianProcess(cov_func, optimize=opt)

approx_15 = GPGO(surrogate_approx_15, Acquisition(util_grad_approx), objfunc, param)
approx_15.run(init_evals=n_init, max_iter=iters)

```

```
Evaluation       Proposed point           Current eval.          Best eval.
init     [7.73226546 2.68343887].         -12.768724532005583    -12.768724532005583
init     [-4.18455179  5.42307669].       -95.39334176626551     -12.768724532005583
init     [-0.86898607  7.95000337].       -16.419487782111716    -12.768724532005583
init     [-0.41121626  4.56711539].       -23.250362743740954    -12.768724532005583
init     [-3.32388086  3.74848521].       -80.99796277938532     -12.768724532005583
1        [ 6.44369262 14.11318321].       -188.60210814630312    -12.768724532005583
2        [-4.79844257 15.        ].       -13.422010816783201    -12.768724532005583
3        [10.         7.77588884].        -24.72402279647976     -12.768724532005583
4        [2.82282728 0.        ].         -7.316554287678703     -7.316554287678703
5        [ 0.46549494 13.40537109].       -84.4862131471065      -7.316554287678703
6        [4.50952792 8.16052408].         -53.09738757367675     -7.316554287678703
7        [-5.         10.41329741].       -58.611678607638595    -7.316554287678703
8        [-1.11782388  0.        ].       -77.25368174404295     -7.316554287678703
9        [3.79188383 3.80740748].         -6.297488441485162     -6.297488441485162
10       [10.  0.].       -10.960889035651505    -6.297488441485162
11       [10.        11.6983054].         -77.55223128279921     -6.297488441485162
12       [-5.  0.].       -308.12909601160663    -6.297488441485162
13       [6.00093923 0.        ].         -20.43505628790286     -6.297488441485162
14       [7.18646382 5.74494392].         -36.29128990520782     -6.297488441485162
15       [ 3.52411836 11.472272  ].       -90.89944330246465     -6.297488441485162
16       [10.         4.19696432].         -3.368795097118179     -3.368795097118179
17       [10. 15.].       -145.87219087939556    -3.368795097118179
18       [-1.68371737 10.77559849].       -11.909723049756376    -3.368795097118179
19       [7.18412072 9.92007307].         -91.41767859293532     -3.368795097118179
20       [2.03815398 6.135088  ].         -13.752464609443821    -3.368795097118179
```

```
### ESTIMATED GP EI GRADIENTS

np.random.seed(run_num_16)
surrogate_approx_16 = GaussianProcess(cov_func, optimize=opt)

approx_16 = GPGO(surrogate_approx_16, Acquisition(util_grad_approx), objfunc, param)
approx_16.run(init_evals=n_init, max_iter=iters)

```

```
Evaluation       Proposed point           Current eval.          Best eval.
init     [-1.65063381  7.84745012].       -10.514702126319445    -2.715864006988424
init     [3.26052185 0.68402925].         -2.715864006988424     -2.715864006988424
init     [0.41093253 3.34621413].         -22.889515127492515    -2.715864006988424
init     [5.33089243 2.45597138].         -17.178157611778595    -2.715864006988424
init     [-3.945127   14.1151629].        -3.3649224341694195    -2.715864006988424
1        [ 5.63879248 14.58068289].       -198.51319639462758    -2.715864006988424
2        [9.91844219 8.81695252].         -36.28422213736768     -2.715864006988424
```

```
    3       [-4.79859594  0.        ].      -286.78105858747693    -2.715864006988424
    4       [3.81408659 8.61177775].        -48.766704850031       -2.715864006988424
    5       [10.  0.].      -10.960889035651505    -2.715864006988424
    6       [ 0.44391888 12.69865484].      -73.13164220805213     -2.715864006988424
    7       [-5.          5.13899229].      -157.88691797579548    -2.715864006988424
    8       [10.         4.2682766].        -3.54417535675414      -2.715864006988424
    9       [-5.         10.32671612].      -59.792189151288454    -2.715864006988424
    10      [10.         13.26461567].      -107.2447873223146     -2.715864006988424
    11      [6.96068526 5.95677145].        -40.29072338303703     -2.715864006988424
    12      [-0.32961053  0.        ].      -61.83885216164565     -2.715864006988424
    13      [ 6.82622134 10.65931328].      -108.5455303017713     -2.715864006988424
    14      [3.31979007 5.07480389].        -9.162230281425373     -2.715864006988424
    15      [6.99195503 0.        ].        -18.699625704219592    -2.715864006988424
    16      [0.87219172 6.69048846].        -20.09738956200713     -2.715864006988424
    17      [-1.9866179 15.        ].        -34.51265554160604     -2.715864006988424
    18      [8.52927322 2.67929763].        -4.730349637572919     -2.715864006988424
    19      [-5. 15.].      -17.508299515778166    -2.715864006988424
    20      [-1.76824828 10.39782081].      -9.50790168258634      -2.715864006988424
```

```
### ESTIMATED GP EI GRADIENTS

np.random.seed(run_num_17)
surrogate_approx_17 = GaussianProcess(cov_func, optimize=opt)

approx_17 = GPGO(surrogate_approx_17, Acquisition(util_grad_approx), objfunc, param)
approx_17.run(init_evals=n_init, max_iter=iters)

```

```
    Evaluation       Proposed point          Current eval.          Best eval.
    init    [-0.58002496  7.95880133].      -19.016141117164256    -19.016141117164256
    init    [-2.1271882   1.01850537].      -85.05949339404415     -19.016141117164256
    init    [6.8047819  9.84500283].        -93.89792695202516     -19.016141117164256
    init    [4.56281344 8.63404341].        -60.50217141122736     -19.016141117164256
    init    [-4.41405626  5.36720407].      -110.70834086593878    -19.016141117164256
    1       [10.  0.].      -10.960889035651505    -10.960889035651505
    2       [-4.6254625 15.        ].       -10.433218359909354    -10.433218359909354
    3       [ 2.13775952 15.        ].      -144.36571223765986    -10.433218359909354
    4       [10. 15.].      -145.87219087939556    -10.433218359909354
    5       [4.33245489 0.        ].        -8.77833321061879      -8.77833321061879
    6       [10.          5.23799378].      -6.9385318265124925    -6.9385318265124925
    7       [-5.         10.43936201].      -58.25923149903159     -6.9385318265124925
    8       [2.22930361 4.05972038].        -5.056779526589618     -5.056779526589618
    9       [6.25561109 3.78338802].        -26.80321767574614     -5.056779526589618
    10      [-1.10952057 12.04965534].      -31.28752175381043     -5.056779526589618
    11      [ 6.05336035 13.8979508 ].      -183.14977791252898    -5.056779526589618
    12      [1.4024169 0.       ].          -27.786142455688946    -5.056779526589618
    13      [10.          8.21799432].      -29.139759047775705    -5.056779526589618
    14      [-0.55709851  4.5360587 ].      -23.86557697822957     -5.056779526589618
    15      [ 2.10433196 11.07507459].      -66.77306758542363     -5.056779526589618
    16      [7.12211142 0.        ].        -17.89933501696511     -5.056779526589618
    17      [-1.76053939 15.        ].      -41.80122854389079     -5.056779526589618
    18      [10.          2.72526762].      -2.0202518349282093    -2.0202518349282093
    19      [10.         11.40692499].      -72.5698252646211 3    -2.0202518349282093
    20      [2.12160756 6.45693569].        -15.5505568974713      -2.0202518349282093
```

```
### ESTIMATED GP EI GRADIENTS

np.random.seed(run_num_18)
```

```
surrogate_approx_18 = GaussianProcess(cov_func, optimize=opt)

approx_18 = GPGO(surrogate_approx_18, Acquisition(util_grad_approx), objfunc, param)
approx_18.run(init_evals=n_init, max_iter=iters)

```

```
Evaluation 	 Proposed point 	  Current eval. 	 Best eval.
init   	 [4.75561363 7.58180061]. 	 -49.215059064668324 	 -8.150075223157177
init   	 [8.17902206 2.72760338]. 	 -8.150075223157177 	 -8.150075223157177
init   	 [ 7.78349603 11.25204429]. 	 -106.98107856533271 	 -8.150075223157177
init   	 [ 4.99152501 14.81843172]. 	 -196.08601750690153 	 -8.150075223157177
init   	 [-1.14547366  0.42458888]. 	 -71.23649635555944 	 -8.150075223157177
1      	 [-4.83474958 11.77354131]. 	 -35.58412906267954 	 -8.150075223157177
2      	 [-4.3294464   5.61430449]. 	 -100.45699054630283 	 -8.150075223157177
3      	 [ 0.51275923 11.10460609]. 	 -53.020724598536006 	 -8.150075223157177
4      	 [4.16816792 0.        ]. 	 -7.622384903229538 	 -7.622384903229538
5      	 [10.          7.20264219]. 	 -19.580499656524992 	 -7.622384903229538
6      	 [0.68815102 5.04168539]. 	 -17.42261558796049 	 -7.622384903229538
7      	 [-1.72921987 15.        ]. 	 -42.84319715187649 	 -7.622384903229538
8      	 [10. 15.]. 	 -145.87219087939556 	 -7.622384903229538
9      	 [-5.          1.24200756]. 	 -266.9780170194058 	 -7.622384903229538
10     	 [4.41908142 3.51927413]. 	 -11.343556041356456 	 -7.622384903229538
11     	 [10.  0.]. 	 -10.960889035651505 	 -7.622384903229538
12     	 [-2.37002081  8.88176695]. 	 -5.72808405213056 	 -5.72808405213056
13     	 [6.95600895 0.        ]. 	 -18.90163505117335 	 -5.72808405213056
14     	 [ 3.9798879  11.01130369]. 	 -90.05524484920635 	 -5.72808405213056
15     	 [-5. 15.]. 	 -17.508299515778166 	 -5.72808405213056
16     	 [7.36750134 5.36647591]. 	 -31.136154737852188 	 -5.72808405213056
17     	 [1.97124129 1.95002408]. 	 -8.25801035753311 	 -5.72808405213056
18     	 [1.17366787 7.88013606]. 	 -26.459693239294232 	 -5.72808405213056
19     	 [ 1.38740769 15.        ]. 	 -131.86084219290987 	 -5.72808405213056
20     	 [-5.          8.84332938]. 	 -82.34660136328705 	 -5.72808405213056
```

```
### ESTIMATED GP EI GRADIENTS

np.random.seed(run_num_19)
surrogate_approx_19 = GaussianProcess(cov_func, optimize=opt)

approx_19 = GPGO(surrogate_approx_19, Acquisition(util_grad_approx), objfunc, param)
approx_19.run(init_evals=n_init, max_iter=iters)

```

```
Evaluation 	 Proposed point 	  Current eval. 	 Best eval.
init   	 [-3.53699597 11.41874575]. 	 -4.475608269502271 	 -4.475608269502271
init   	 [-1.2959304   2.07197531]. 	 -51.13947793770689 	 -4.475608269502271
init   	 [-0.02830155  1.24499348]. 	 -42.63973921760034 	 -4.475608269502271
init   	 [ 5.07965622 12.09890697]. 	 -131.17235664506808 	 -4.475608269502271
init   	 [9.74112872 9.53491102]. 	 -46.84458913925112 	 -4.475608269502271
1      	 [8.6773951  0.80876842]. 	 -4.184707152540867 	 -4.184707152540867
2      	 [4.45041553 5.47999855]. 	 -23.54840201771013 	 -4.184707152540867
3      	 [-5.          6.34985382]. 	 -130.17529401788659 	 -4.184707152540867
4      	 [ 0.0873144 15.        ]. 	 -103.06821773257954 	 -4.184707152540867
5      	 [10. 15.]. 	 -145.87219087939556 	 -4.184707152540867
6      	 [0.38615144 8.6436678 ]. 	 -29.386080997722466 	 -4.184707152540867
7      	 [4.52927623 0.        ]. 	 -10.329659446180516 	 -4.184707152540867
8      	 [10.          4.77775304]. 	 -5.093043061641548 	 -4.184707152540867
9      	 [-5. 15.]. 	 -17.508299515778166 	 -4.184707152540867
10     	 [-5.  0.]. 	 -308.12909601160663 	 -4.184707152540867
11     	 [0.87690023 5.09927128]. 	 -16.297379330348566 	 -4.184707152540867
```

```
12       [7.03894792 3.18274589].        -20.927761063267077    -4.184707152540867
13       [6.31498789 8.27590644].        -71.07448820748486     -4.184707152540867
14       [3.14942604 2.54652035].        -0.47525558225881426   -0.475255582258142
15       [ 1.34143153 11.64567908].      -69.15805867652578     -0.475255582258142
16       [-2.72757155  9.18806926].      -5.678510435253411     -0.475255582258142
17       [ 3.55278946 15.        ].      -170.81850897929945    -0.475255582258142
18       [10.          2.3524637].       -2.3662816781093357    -0.475255582258142
19       [-5.         10.02508797].      -64.0218952296447      -0.475255582258142
20       [-2.54379641 12.57903117].      -4.934481076732693     -0.475255582258142
```

```
### ESTIMATED GP EI GRADIENTS

np.random.seed(run_num_20)
surrogate_approx_20 = GaussianProcess(cov_func, optimize=opt)

approx_20 = GPGO(surrogate_approx_20, Acquisition(util_grad_approx), objfunc, param)
approx_20.run(init_evals=n_init, max_iter=iters)

```

```
Evaluation       Proposed point          Current eval.          Best eval.
init     [ 3.82196202 13.46570592].      -138.5264349938869     -14.042667401507376
init     [ 8.37296094 12.23756216].      -115.63178540512689    -14.042667401507376
init     [-4.46165622 10.37636373].      -35.66708529307584     -14.042667401507376
init     [0.68021413 7.77766418].        -25.307769914281764    -14.042667401507376
init     [4.86927198 2.90775327].        -14.042667401507376    -14.042667401507376
1        [-4.88337277  1.15554411].      -258.03874991509736    -14.042667401507376
2        [10.          5.67308311].      -9.072716233411997     -9.072716233411997
3        [9.84832611 0.06591567].        -9.027847655844058     -9.027847655844058
4        [-1.70866498 15.        ].      -43.53068468679724     -9.027847655844058
5        [0.11401177 1.25905543].        -40.34403021450308     -9.027847655844058
6        [6.34167483 7.90009461].        -65.79565375919643     -9.027847655844058
7        [-3.52279418  5.91051374].      -54.3680517740781      -9.027847655844058
8        [-0.1848059  11.40919125].      -45.5573656393724      -9.027847655844058
9        [6.44407639 0.        ].        -20.706796158679435    -9.027847655844058
10       [1.92173517 4.53655254].        -7.948918546291848     -7.948918546291848
11       [-5.        14.08925918].      -22.321984249993857    -7.948918546291848
12       [8.3426774  3.06196412].        -7.310425890385614     -7.310425890385614
13       [10.          8.78276435].      -35.349318268169796    -7.310425890385614
14       [3.22054292 0.        ].        -5.330634712932251     -5.330634712932251
15       [3.29040332 9.72739009].        -57.74202036956133     -5.330634712932251
16       [10. 15.].       -145.87219087939556     -5.330634712932251
17       [-0.67741045  4.42648342].      -24.831090235816156    -5.330634712932251
18       [3.81886761 5.90554444].        -19.322858995875393    -5.330634712932251
19       [-2.00719238  8.71701415].      -6.937407242150097     -5.330634712932251
20       [6.97723007 4.81293023].        -30.54762029665723     -5.330634712932251
```

```
end_approx = time.time()
end_approx

time_approx = end_approx - start_approx
time_approx

start_exact = time.time()
start_exact
```

```
1623406360.7274048
```

```
### EXACT GP EI GRADIENTS

np.random.seed(run_num_1)
surrogate_exact_1 = dGaussianProcess(cov_func, optimize=opt)

exact_1 = dGPGO(surrogate_exact_1, Acquisition_new(util_grad_exact), objfunc, param)
exact_1.run(init_evals=n_init, max_iter=iters)

```

```
Evaluation       Proposed point           Current eval.          Best eval.
init     [ 1.25533007 10.8048674 ].       -56.52874004692151     -27.68166689936483
init     [-4.99828438  4.53498859].       -172.66531086516164    -27.68166689936483
init     [-2.79866164  1.38507892].       -102.58290348816024    -27.68166689936483
init     [-2.20609683  5.18341091].       -28.868064601155666    -27.68166689936483
init     [0.95151211 8.08225101].         -27.68166689936483     -27.68166689936483
1        [9.47260071 9.95162247].         -55.70277575038468     -27.68166689936483
2        [6.77944042 0.33495642].         -19.104233620056625    -19.104233620056625
3        [-3.40124185 14.78563236].       -4.246346957669259     -4.246346957669259
4        [ 5.75393503 14.80989262].       -205.71967217110094    -4.246346957669259
5        [9.61605203 4.67554377].         -4.712086132001213     -4.246346957669259
6        [-5.         10.68552092].       -54.99766615417242     -4.246346957669259
7        [5.08598148 4.73310463].         -25.65697663075452     -4.246346957669259
8        [2.01024959 0.        ].         -16.954744203759482    -4.246346957669259
9        [5.16240709 9.27255048].         -78.91408248056042     -4.246346957669259
10       [1.27150924 3.77985512].         -12.99537209775962     -4.246346957669259
11       [ 1.62068846 14.58208631].       -126.6405369153871     -4.246346957669259
12       [ 9.55928048 13.94325565].       -129.36302497139116    -4.246346957669259
13       [9.99723239 1.79434474].         -3.382829474803896     -3.382829474803896
14       [-1.95651649 12.39047307].       -14.127331272125602    -3.382829474803896
15       [7.91601241 6.67694683].         -36.242857432606996    -3.382829474803896
16       [-2.60061393  8.19802829].       -9.691427385043374     -3.382829474803896
17       [4.23359578 1.67747107].         -5.586232071608859     -3.382829474803896
18       [7.90798322 3.02389547].         -11.8260925458786      -3.382829474803896
19       [2.83117972 6.34624688].         -15.42415351716031     -3.382829474803896
20       [9.14003975 0.50455077].         -3.8146909252306624    -3.382829474803896
```

```
### EXACT GP EI GRADIENTS

np.random.seed(run_num_2)
surrogate_exact_2 = dGaussianProcess(cov_func, optimize=opt)

exact_2 = dGPGO(surrogate_exact_2, Acquisition_new(util_grad_exact), objfunc, param)
exact_2.run(init_evals=n_init, max_iter=iters)

```

```
Evaluation       Proposed point           Current eval.          Best eval.
init     [1.53992353 0.38889348].         -22.31361308916297     -6.692051508754487
init     [3.24493717 6.52983589].         -19.233109967858276    -6.692051508754487
init     [1.30551703 4.95502232].         -13.177851563387286    -6.692051508754487
init     [-1.93027049  9.2890645 ].       -6.692051508754487     -6.692051508754487
init     [-0.50517989  4.00240913].       -26.437522758780958    -6.692051508754487
1        [ 9.5587047  12.00387527].       -89.09951357262268     -6.692051508754487
2        [9.06869679 2.0164014 ].         -1.0307174678975652    -1.0307174678975652
3        [-0.16427306 14.97419295].       -95.32402138742069     -1.0307174678975652
4        [-5.  0.].       -308.12909601160663    -1.0307174678975652
5        [8.87401661 6.61039509].         -22.618568623676445    -1.0307174678975652
6        [-5.         12.97618816].       -30.45772350488552     -1.0307174678975652
```

```
    7        [ 4.18301482 11.56379641].       -104.36954494369935     -1.0307174678975652
    8        [-5.          5.7954092].        -142.5002965706689      -1.0307174678975652
    9        [6.02212844 0.        ].         -20.48785907865212      -1.0307174678975652
    10       [6.39374973 3.54158379].         -25.479942045701176     -1.0307174678975652
    11       [-5.          9.53932812].       -71.21614604777105      -1.0307174678975652
    12       [ 7.5999292  14.98756105].       -197.963062952534       -1.0307174678975652
    13       [ 0.13788539 11.1489037 ].       -48.303849917898134     -1.0307174678975652
    14       [9.33551326 0.        ].         -6.199635174137813      -1.0307174678975652
    15       [6.12693335 8.37865806].         -72.49049044538489      -1.0307174678975652
    16       [0.23097845 7.5788441 ].         -23.10902271562412      -1.0307174678975652
    17       [-1.2753278  0.       ].         -80.6913269029646       -1.0307174678975652
    18       [3.30023938 2.83747178].         -0.9848813016418614     -0.9848813016418614
    19       [9.12246647 4.0165424 ].         -4.01863936350104       -0.9848813016418614
    20       [-3.16189021  3.09706011].       -85.53320518671964      -0.9848813016418614
```

```python
### EXACT GP EI GRADIENTS

np.random.seed(run_num_3)
surrogate_exact_3 = dGaussianProcess(cov_func, optimize=opt)

exact_3 = dGPGO(surrogate_exact_3, Acquisition_new(util_grad_exact), objfunc, param)
exact_3.run(init_evals=n_init, max_iter=iters)

```

```
    Evaluation       Proposed point           Current eval.           Best eval.
    init     [ 3.26196854 10.62221734].       -71.68783452098575      -18.07886746449891
    init     [-0.63642892  7.66241408].       -18.07886746449891      -18.07886746449891
    init     [ 8.39420432 13.44439633].       -141.98653529865882     -18.07886746449891
    init     [-3.11622034  3.10864317].       -83.31045707266176      -18.07886746449891
    init     [-4.22799195  6.61214765].       -76.5294170688398       -18.07886746449891
    1        [8.38589781 9.60264925].         -66.98095812668986      -18.07886746449891
    2        [5.36445556 1.34242487].         -15.853300865113308     -15.853300865113308
    3        [-3.08864377 14.70503451].       -6.949190296303928      -6.949190296303928
    4        [8.69056724 4.31708052].         -8.592312272503058      -6.949190296303928
    5        [0.56047254 0.52748086].         -39.4874086937192       -6.949190296303928
    6        [3.29041148 5.41159541].         -11.065169073156763     -6.949190296303928
    7        [ 2.13483631 14.96181456].       -143.41722782959573     -6.949190296303928
    8        [9.12628913 1.01046409].         -2.3212961739849582     -2.3212961739849582
    9        [-5.         11.17586988].       -48.86176834548933      -2.3212961739849582
    10       [-1.17009389 11.92075001].       -28.81241382788842      -2.3212961739849582
    11       [-5.  0.].       -308.12909601160663     -2.3212961739849582
    12       [0.48421884 4.20033201].         -19.620358051938585     -2.3212961739849582
    13       [5.89785926 6.89304443].         -52.37733461429231      -2.3212961739849582
    14       [ 5.64106616 14.42648623].       -194.41105549770649     -2.3212961739849582
    15       [7.69190727 0.        ].         -13.51283395340081      -2.3212961739849582
    16       [2.79584682 2.89585474].         -1.0788595154055631     -1.0788595154055631
    17       [5.94218381 3.80060928].         -26.32002367627955      -1.0788595154055631
    18       [9.26719579 6.31925914].         -16.3093395449881       -1.0788595154055631
    19       [-5.         14.36659717].       -20.680458551884495     -1.0788595154055631
    20       [3.3795623 0.       ].           -5.064776999696175      -1.0788595154055631
```

```python
### EXACT GP EI GRADIENTS

np.random.seed(run_num_4)
surrogate_exact_4 = dGaussianProcess(cov_func, optimize=opt)

exact_4 = dGPGO(surrogate_exact_4, Acquisition_new(util_grad_exact), objfunc, param)
exact_4.run(init_evals=n_init, max_iter=iters)
```

```
Evaluation       Proposed point           Current eval.          Best eval.
init     [9.50544759 8.20848374].         -32.51676744781521     -7.247126865776948
init     [ 9.5902654  10.72223991].       -66.20566674364716     -7.247126865776948
init     [5.46593237 3.24134243].         -20.90089401680587     -7.247126865776948
init     [9.64411682 0.09345383].         -7.247126865776948     -7.247126865776948
init     [-1.20526456  6.52187299].       -15.941376982363263    -7.247126865776948
1        [ 0.5775056  14.31328634].       -102.48870217898984    -7.247126865776948
2        [-0.13367617  0.17881992].       -55.952651178085794    -7.247126865776948
3        [-4.98639995 14.6489551 ].       -18.844355450809267    -7.247126865776948
4        [4.32819126 9.17004107].         -64.7480751694855 3    -7.247126865776948
5        [-4.93569217  3.54331379].       -193.27527353634537    -7.247126865776948
6        [ 5.76797903 14.82190962].       -206.15580235617168    -7.247126865776948
7        [-4.95336009  9.46492024].       -69.87288098121975     -7.247126865776948
8        [-0.16914819 10.36255978].       -36.190362240300715    -7.247126865776948
9        [9.16862347 3.07805919].         -1.3683511749105275    -1.3683511749105275
10       [1.77097701 3.87702624].         -8.175018294001896     -1.3683511749105275
11       [3.8933132 0.       ].           -6.089324660639614     -1.3683511749105275
12       [-3.88638993  0.        ].       -202.7834023208659     -1.3683511749105275
13       [6.1638444  6.43571539].         -48.024582292979304    -1.3683511749105275
14       [6.41902483 0.37792916].         -20.044775995583656    -1.3683511749105275
15       [2.01735199 6.29703812].         -14.745601666768007    -1.3683511749105275
16       [ 9.61508667 14.10805784].       -132.08282128659621    -1.3683511749105275
17       [-2.6738185  12.28179721].       -2.6453574456380977    -1.3683511749105275
18       [9.57964912 4.82463969].         -5.4230419977061635    -1.3683511749105275
19       [-1.10360376  3.42302935].       -34.4914468369297      -1.3683511749105275
20       [ 2.71837567 12.16338735].       -92.16508024778177     -1.3683511749105275
```

```
### EXACT GP EI GRADIENTS

np.random.seed(run_num_5)
surrogate_exact_5 = dGaussianProcess(cov_func, optimize=opt)

exact_5 = dGPGO(surrogate_exact_5, Acquisition_new(util_grad_exact), objfunc, param)
exact_5.run(init_evals=n_init, max_iter=iters)

```

```
Evaluation       Proposed point           Current eval.          Best eval.
init     [-1.67010243 13.06098459].       -25.390690545664548    -25.311309190989615
init     [-1.89921267 13.77916362].       -25.311309190989615    -25.311309190989615
init     [2.32616783 9.17615794].         -41.60147975414021     -25.311309190989615
init     [6.48861785 7.77626982].         -63.81281921768236     -25.311309190989615
init     [-0.54799248  2.81581843].       -34.966203644931994    -25.311309190989615
1        [8.84124421 0.03321319].         -5.96108856337545      -5.96108856337545
2        [ 9.0736314  14.50090003].       -152.42565438241107    -5.96108856337545
3        [-4.81236982  8.00518018].       -85.70683082507513     -5.96108856337545
4        [-5.  0.].       -308.12909601160663    -5.96108856337545
5        [3.91824516 0.        ].         -6.204019264406566     -5.96108856337545
6        [ 2.622107   14.91945666].       -150.61370255044642    -5.96108856337545
7        [9.51881542 3.9966536 ].         -2.517336699008217     -2.517336699008217
8        [3.88320041 3.95002052].         -7.682085372222798     -2.517336699008217
9        [ 5.83337313 11.69578969].       -130.6673919213544     -2.517336699008217
10       [-1.12997151  7.15285708].       -14.753986420091532    -2.517336699008217
11       [-5.         12.03506732].       -39.26987488525771     -2.517336699008217
12       [-5.          4.13974479].        -182.96401575387907    -2.517336699008217
13       [6.8031699  2.88861857].         -21.350617802677874    -2.517336699008217
14       [1.05053693 0.        ].         -34.75943692279499     -2.517336699008217
15       [ 9.44363707 10.08398027].       -58.05364275224202     -2.517336699008217
```

```
16        [-1.40220613 10.07633611].       -14.141349306607253     -2.517336699008217
17        [1.30497636 5.81307932].         -15.311437560928843     -2.517336699008217
18        [9.05272596 5.306356  ].         -10.834885094395876     -2.517336699008217
19        [5.9380293 0.        ].          -20.255512741690204     -2.517336699008217
20        [9.93148416 2.20711877].         -2.1350786473167114     -2.1350786473167114
```

```
### EXACT GP EI GRADIENTS

np.random.seed(run_num_6)
surrogate_exact_6 = dGaussianProcess(cov_func, optimize=opt)

exact_6 = dGPGO(surrogate_exact_6, Acquisition_new(util_grad_exact), objfunc, param)
exact_6.run(init_evals=n_init, max_iter=iters)

```

```
Evaluation       Proposed point           Current eval.          Best eval.
init      [8.39290227 4.97969708].         -15.553860636684597     -15.31731051317483
init      [7.31843685 0.62544939].         -15.31731051317483      -15.31731051317483
init      [-3.3851498   8.92578096].       -16.222288919361837     -15.31731051317483
init      [2.94726043 6.28211143].         -15.406373581304088     -15.31731051317483
init      [0.03111774 9.33779148].         -31.07053245856209      -15.31731051317483
1         [ 9.72522083 14.5299912 ].       -139.8297584868663      -15.31731051317483
2         [-4.42013634  0.37777   ].       -237.69785665817545     -15.31731051317483
3         [ 3.61480227 14.3758058 ].       -156.22942206205929     -15.31731051317483
4         [1.33236176 0.68568901].         -23.985581932967577     -15.31731051317483
5         [-4.19302139 14.61872658].       -5.3403971656550695     -5.3403971656550695
6         [ 6.40979378 10.06666115].       -99.81689660275181      -5.3403971656550695
7         [-5.          5.12364844].       -158.25688995372795     -5.3403971656550695
8         [-0.59824397  4.17707408].       -25.894181651083727     -5.3403971656550695
9         [-0.81914508 13.50481785].       -53.942961683213866     -5.3403971656550695
10        [4.52645109 2.90779502].         -10.37120197751454      -5.3403971656550695
11        [-5.         11.78842408].       -41.87226403856418      -5.3403971656550695
12        [9.93332299 8.15272864].         -28.812880358314132     -5.3403971656550695
13        [4.36369847 0.        ].         -9.014108437934913      -5.3403971656550695
14        [5.72619803 6.57459356].         -47.87774929312015      -5.3403971656550695
15        [ 2.97185193 10.21145323].       -61.38145738681465      -5.3403971656550695
16        [9.35985297 1.95156086].         -0.6382835313791233     -0.6382835313791233
17        [ 9.62688676 11.13421189].       -72.56229037536801      -0.6382835313791233
18        [9.95754335 3.12079698].         -1.7541985493835206     -0.6382835313791233
19        [-1.49901661  6.73859765].       -14.442340452186656     -0.6382835313791233
20        [2.26160142 3.46007602].         -4.0409541925142065     -0.6382835313791233
```

```
### EXACT GP EI GRADIENTS

np.random.seed(run_num_7)
surrogate_exact_7 = dGaussianProcess(cov_func, optimize=opt)

exact_7 = dGPGO(surrogate_exact_7, Acquisition_new(util_grad_exact), objfunc, param)
exact_7.run(init_evals=n_init, max_iter=iters)

```

```
Evaluation       Proposed point           Current eval.          Best eval.
init      [-3.85537566 11.69878188].       -8.299317850233932      -5.211335324193094
init      [ 1.57613847 10.85197767].       -59.50408380701259      -5.211335324193094
init      [9.66984268 8.07743806].         -29.714921414987273     -5.211335324193094
init      [2.51680695 1.080767  ].         -5.211335324193094      -5.211335324193094
init      [-0.9734153   7.49823751].       -15.431057204948033     -5.211335324193094
```

```
1        [-3.15644272  0.        ].       -151.95269311203583    -5.211335324193094
2        [ 9.25668108 14.87713055].       -157.79168966390583    -5.211335324193094
3        [7.83339381 0.01037029].         -12.29850161167343     -5.211335324193094
4        [4.4556015  6.00369035].         -28.08564757730271     -5.211335324193094
5        [-5.          5.19297584].       -156.58900479915567    -5.211335324193094
6        [-0.66517009 14.88752362].       -77.9546048777314      -5.211335324193094
7        [ 4.59415787 14.23709363].       -173.27920595767617    -5.211335324193094
8        [9.53046987 3.95926056].         -2.39376862049631      -2.39376862049631
9        [-0.84004858  3.75712261].       -29.88508899821236     -2.39376862049631
10       [ 6.15564868 10.11747234].       -100.87425641228145    -2.39376862049631
11       [5.11067786 2.34886485].         -14.953066053244225    -2.39376862049631
12       [-5.          9.05365463].       -78.88091737733349     -2.39376862049631
13       [7.56840459 5.12548076].         -26.92686223753419     -2.39376862049631
14       [0.48297955 0.        ].         -46.18662095403514     -2.39376862049631
15       [-4.12505239 14.16441748].       -5.037851363850817     -2.39376862049631
16       [ 9.62070745 11.05602901].       -71.32314558447405     -2.39376862049631
17       [1.49156341 5.84639153].         -14.496047062587       -2.39376862049631
18       [9.45734956 2.27605218].         -0.4543096953905579    -0.4543096953905579
19       [4.4547232 0.       ].           -9.724934487844362     -0.4543096953905579
20       [-1.69425977 11.22620026].       -13.478244704038248    -0.4543096953905579
```

```
### EXACT GP EI GRADIENTS

np.random.seed(run_num_8)
surrogate_exact_8 = dGaussianProcess(cov_func, optimize=opt)

exact_8 = dGPGO(surrogate_exact_8, Acquisition_new(util_grad_exact), objfunc, param)
exact_8.run(init_evals=n_init, max_iter=iters)

```

```
Evaluation       Proposed point            Current eval.          Best eval.
init     [ 8.10144104 14.52810994].       -175.17297136347514    -15.416245468470875
init     [8.0379181  7.96283537].         -49.32173799887473     -15.416245468470875
init     [-1.50907508  0.17098206].       -83.26753461673219     -15.416245468470875
init     [1.45703227 6.0352704 ].         -15.416245468470875    -15.416245468470875
init     [2.84012007 7.17587694].         -22.49093277852639     -15.416245468470875
1        [-0.47864575 14.6326007 ].       -80.00768934620143     -15.416245468470875
2        [6.2584639  0.79690936].         -19.69061392229066     -15.416245468470875
3        [-4.8401206   6.42823123].       -117.34273274148853    -15.416245468470875
4        [-3.93434094 11.15335008].       -12.920033320986416    -12.920033320986416
5        [ 3.37643278 12.34569511].       -105.65657433447679    -12.920033320986416
6        [2.39907019 2.26982123].         -3.355132209626336     -3.355132209626336
7        [8.96015034 3.6583582 ].         -3.8102476776448864    -3.355132209626336
8        [-0.72764928  8.94814568].       -20.13438107467893     -3.355132209626336
9        [-4.84490371 14.8271652 ].       -14.940133796228146    -3.355132209626336
10       [-5.          2.13992428].       -239.14907437782054    -3.355132209626336
11       [5.74588614 4.4168561 ].         -29.11711586406138     -3.355132209626336
12       [-1.37625533  4.1811101 ].       -29.95236792745573     -3.355132209626336
13       [9.99008953 0.12991645].         -10.089797543288475    -3.355132209626336
14       [1.81641076 0.        ].         -20.163690937652753    -3.355132209626336
15       [5.09727359 9.28523236].         -78.26760358414715     -3.355132209626336
16       [ 9.32441646 10.92305278].       -73.2311382417214      -3.355132209626336
17       [-0.34413652 11.51006361].       -43.51245650020666     -3.355132209626336
18       [2.26077277 3.59385919].         -4.1707953776312285    -3.355132209626336
19       [9.92583404 5.63487006].         -8.894086683040968     -3.355132209626336
20       [4.13588038 0.        ].         -7.414019350324661     -3.355132209626336
```

```
### EXACT GP EI GRADIENTS

```

```

np.random.seed(run_num_9)
surrogate_exact_9 = dGaussianProcess(cov_func, optimize=opt)

exact_9 = dGPGO(surrogate_exact_9, Acquisition_new(util_grad_exact), objfunc, param)
exact_9.run(init_evals=n_init, max_iter=iters)

```

```
Evaluation       Proposed point           Current eval.          Best eval.
init     [-4.84438769  7.52811888].       -96.15564306553264     -3.4640248583909496
init     [2.4365994  2.00744293].         -3.4640248583909496    -3.4640248583909496
init     [-2.86833372  3.27838013].       -70.46930174488207     -3.4640248583909496
init     [1.27762271 3.72151753].         -12.982825924833865    -3.4640248583909496
init     [-3.73910523  5.1824796 ].       -75.58547400560856     -3.4640248583909496
1        [ 7.05609655 14.75038872].       -200.4388688685465     -3.4640248583909496
2        [9.79053515 5.9968998 ].         -11.24805159216164     -3.4640248583909496
3        [-0.91490984 13.25685032].       -48.26155495184477     -3.4640248583909496
4        [4.42329559 9.16761691].         -66.24415745540334     -3.4640248583909496
5        [8.09381179 0.36546333].         -9.19702700265266      -3.4640248583909496
6        [-0.11304091  8.3679709 ].       -24.321217375314745    -3.4640248583909496
7        [ 9.43884454 10.2183652 ].       -60.17453038358803     -3.4640248583909496
8        [5.57235414 4.63725127].         -29.48892327426774     -3.4640248583909496
9        [-5.        12.0047373].         -39.58333314384184     -3.4640248583909496
10       [-0.29887753  0.        ].       -61.26042699608233     -3.4640248583909496
11       [-5.  0.].        -308.12909601160663    -3.4640248583909496
12       [4.60501306 0.86161159].         -9.27211266246828      -3.4640248583909496
13       [ 2.57498773 13.51903583].       -117.69088059470447    -3.4640248583909496
14       [8.46361221 3.26183489].         -6.687401338525152     -3.4640248583909496
15       [-3.94672    14.81957836].       -3.6221444768682645    -3.4640248583909496
16       [2.07361513 6.40526612].         -15.295577538921345    -3.4640248583909496
17       [7.27870001 7.42081772].         -53.18340068847093     -3.4640248583909496
18       [9.91313931 1.85072915].         -2.658887237261661     -2.658887237261661
19       [-2.22894912 10.02200043].       -4.154793912149543     -2.658887237261661
20       [ 9.88660101 13.1221454 ].       -106.05741144993789    -2.658887237261661
```

```
### EXACT GP EI GRADIENTS

np.random.seed(run_num_10)
surrogate_exact_10 = dGaussianProcess(cov_func, optimize=opt)

exact_10 = dGPGO(surrogate_exact_10, Acquisition_new(util_grad_exact), objfunc, param)
exact_10.run(init_evals=n_init, max_iter=iters)

```

```
Evaluation       Proposed point           Current eval.          Best eval.
init     [6.56980965 0.31127924].         -19.863985024602144    -2.710610964564512
init     [ 4.50472352 11.23205824].       -103.66999147252169    -2.710610964564512
init     [2.47760518 3.37194968].         -2.710610964564512     -2.710610964564512
init     [-2.02905703 11.40796068].       -8.463930835255017     -2.710610964564512
init     [-2.46333745  1.32509721].       -90.49481266026973     -2.710610964564512
1        [8.70372597 6.86774268].         -27.130126017218842    -2.710610964564512
2        [ 9.45036259 14.63089076].       -147.64044674485945    -2.710610964564512
3        [-5.         7.4952192].         -106.66134726369775    -2.710610964564512
4        [ 1.08105942 14.97116497].       -125.62413988622455    -2.710610964564512
5        [0.03910533 7.52803182].         -22.123101688582683    -2.710610964564512
6        [4.29852499 6.53236012].         -31.0056064602294      -2.710610964564512
7        [-4.31113374 14.62074842].       -6.661365851958085     -2.710610964564512
8        [2.04084071 0.        ].         -16.474782473739975    -2.710610964564512
9        [9.44963174 2.45886016].         -0.40223554223319447   -0.4022355422331944
```

```
 10       [ 9.90623718 10.80791959].       -63.84974428406751      -0.4022355422331944
 11       [6.92221404 3.72277849].         -24.208268689326776     -0.4022355422331944
 12       [-5.         11.40803482].       -46.12435444234146      -0.4022355422331944
 13       [-0.36707105  4.28232863].       -24.34154796738504      -0.4022355422331944
 14       [ 5.32830902 14.94021784].       -204.68738881850703     -0.4022355422331944
 15       [-4.98760559  3.74188908].       -192.43089231074592     -0.4022355422331944
 16       [9.90545853 1.84875862].         -2.612891840338106      -0.4022355422331944
 17       [ 0.67873078 10.8234573 ].       -51.6284233232048       -0.4022355422331944
 18       [3.88743533 2.26060726].         -3.192509256132417      -0.4022355422331944
 19       [9.90384587 3.76388583].         -2.210098482264814      -0.4022355422331944
 20       [-2.13717298 13.84580476].       -19.70363121337588      -0.4022355422331944
```

```
### EXACT GP EI GRADIENTS

np.random.seed(run_num_11)
surrogate_exact_11 = dGaussianProcess(cov_func, optimize=opt)

exact_11 = dGPGO(surrogate_exact_11, Acquisition_new(util_grad_exact), objfunc, param)
exact_11.run(init_evals=n_init, max_iter=iters)

```

```
Evaluation       Proposed point            Current eval.           Best eval.
init      [-2.29595467  0.29212862].       -104.49282729548965     -22.37435843952312
init      [ 1.9482779  10.87400894].       -62.47758067812407      -22.37435843952312
init      [1.30305407 7.28140647].         -22.37435843952312      -22.37435843952312
init      [-4.80828778  7.31057411].       -97.94503054415523      -22.37435843952312
init      [ 9.12709979 12.76192634].       -111.62900800771942     -22.37435843952312
1         [6.94653631 0.95529646].         -17.615403152614682     -17.615403152614682
2         [-3.32392335 14.51197433].       -3.777267680584016      -3.777267680584016
3         [9.72363233 6.87677997].         -17.947808548975658     -3.777267680584016
4         [2.47196947 2.06083336].         -3.1023288619513876     -3.1023288619513876
5         [ 3.58353585 14.94720983].       -170.10293330444745     -3.1023288619513876
6         [5.39808201 5.61457141].         -35.807435550549954     -3.1023288619513876
7         [-5.         11.61011793].       -43.82938406177341      -3.1023288619513876
8         [-1.233849    4.37974435].       -27.467888955719456     -3.1023288619513876
9         [6.40338978 9.39336155].         -88.21827786975473      -3.1023288619513876
10        [-5.          3.22438031].       -207.68855472534503     -3.1023288619513876
11        [-1.61206974  9.59321134].       -10.082390570475399     -3.1023288619513876
12        [-0.49043072 14.0558337 ].       -70.94899545871193      -3.1023288619513876
13        [9.83750142 2.61125041].         -1.2588665383981255     -1.2588665383981255
14        [3.72886005 0.        ].         -5.47207969132192       -1.2588665383981255
15        [1.10383114 0.        ].         -33.68796375749133      -1.2588665383981255
16        [8.23326782 4.05065076].         -12.191850927163275     -1.2588665383981255
17        [4.31426076 2.41757474].         -7.05074132025608       -1.2588665383981255
18        [9.81708139 0.83789111].         -5.079125374695652      -1.2588665383981255
19        [2.68515675 3.72552065].         -2.520744035550546      -1.2588665383981255
20        [1.00909168 3.14156426].         -17.029717350521512     -1.2588665383981255
```

```
### EXACT GP EI GRADIENTS

np.random.seed(run_num_12)
surrogate_exact_12 = dGaussianProcess(cov_func, optimize=opt)

exact_12 = dGPGO(surrogate_exact_12, Acquisition_new(util_grad_exact), objfunc, param)
exact_12.run(init_evals=n_init, max_iter=iters)

```

```
Evaluation       Proposed point          Current eval.          Best eval.
init     [-2.68755736 11.10074545].      -1.3827702760021356    -0.5499315281120278
init     [-1.05027477  8.0060909 ].      -14.81231853080056     -0.5499315281120278
init     [-4.78137556 13.78120512].      -18.401131000214548    -0.5499315281120278
init     [8.51072281 0.50132141].        -5.855172364344769     -0.5499315281120278
init     [9.35424004 2.05813982].        -0.5499315281120278    -0.5499315281120278
1        [9.82941798 5.1689241 ].        -6.608972192183428     -0.5499315281120278
2        [ 8.68497867 14.25940608].      -155.1276557842789     -0.5499315281120278
3        [-5.  0.].       -308.12909601160663    -0.5499315281120278
4        [2.64482858 0.        ].        -8.8176752663584       -0.5499315281120278
5        [ 2.33380414 14.34125566].      -132.23186597332315    -0.5499315281120278
6        [5.62360485 9.20963842].        -82.78452011516237     -0.5499315281120278
7        [3.09998873 4.65591628].        -5.9204624447012675    -0.5499315281120278
8        [-5.          5.45104449].      -150.4648553184957     -0.5499315281120278
9        [-0.62223179  2.58482126].      -37.65403378567214     -0.5499315281120278
10       [6.4409178  3.32313442].        -24.38869218110994     -0.5499315281120278
11       [-5.          9.72826959].      -68.36178431846182     -0.5499315281120278
12       [ 9.56058806 10.18663569].      -58.16568478001416     -0.5499315281120278
13       [ 1.70684142 10.16634919].      -51.03247510339132     -0.5499315281120278
14       [-1.6783647  12.69610094].      -22.37202562158393     -0.5499315281120278
15       [5.63023566 0.        ].        -18.913532859363595    -0.5499315281120278
16       [7.55807437 6.28323147].        -37.131397397403916    -0.5499315281120278
17       [0.76616574 5.80881136].        -17.82604551570398     -0.5499315281120278
18       [ 5.32380935 12.84687022].      -151.43268246126414    -0.5499315281120278
19       [2.3766028  2.70097342].        -3.1337577489183115    -0.5499315281120278
20       [9.48262579 2.819244  ].        -0.5009833369059624    -0.5009833369059624
```

```
### EXACT GP EI GRADIENTS

np.random.seed(run_num_13)
surrogate_exact_13 = dGaussianProcess(cov_func, optimize=opt)

exact_13 = dGPGO(surrogate_exact_13, Acquisition_new(util_grad_exact), objfunc, param)
exact_13.run(init_evals=n_init, max_iter=iters)

```

```
Evaluation       Proposed point          Current eval.          Best eval.
init     [6.66553616 3.5631183 ].        -24.823670200298267    -18.038943061558626
init     [ 7.36417799 14.48623797].      -188.7803334745195     -18.038943061558626
init     [9.58901671 6.80173871].        -18.038943061558626    -18.038943061558626
init     [ 4.13563694 11.63289772].      -104.87319703906111    -18.038943061558626
init     [ 4.62420017 10.83027344].      -98.03271225790056     -18.038943061558626
1        [-1.62789089  4.1918933 ].      -31.93216745479765     -18.038943061558626
2        [-4.35385058 13.32256485].      -10.85616402870354     -10.85616402870354
3        [2.9523568  0.05149534].        -6.213290736446655     -6.213290736446655
4        [-5.  0.].       -308.12909601160663    -6.213290736446655
5        [-5.          8.73987037].      -84.0838352604296      -6.213290736446655
6        [-0.45911352  9.66427263].      -27.054588141146777    -6.213290736446655
7        [ 0.19688744 14.21576331].      -92.07708032463351     -6.213290736446655
8        [2.9171967  5.50508571].        -9.93248653798666      -6.213290736446655
9        [9.74243132 0.28918417].        -6.963365868843974     -6.213290736446655
10       [-0.50165264  0.        ].      -65.08042616156222     -6.213290736446655
11       [ 9.71084004 10.83857417].      -66.58782483256539     -6.213290736446655
12       [5.14103601 7.03439101].        -47.65684193387447     -6.213290736446655
13       [6.29582412 0.        ].        -20.81227638434767     -6.213290736446655
14       [1.57425972 2.81839768].        -10.959252839706751    -6.213290736446655
15       [-5.          4.61746228].      -170.72608210829023    -6.213290736446655
16       [0.2369298  6.62713374].        -20.32780417718545     -6.213290736446655
```

```
17       [3.95235416 2.72884215].        -4.3870669358895125     -4.3870669358895125
18       [-2.91432663 11.38522627].      -0.7674857269668536     -0.7674857269668536
19       [9.8299885  3.82020006].        -2.140159086292873      -0.7674857269668536
20       [ 3.78085656 14.4265263 ].      -160.985174324529       -0.7674857269668536
```

```
### EXACT GP EI GRADIENTS

np.random.seed(run_num_14)
surrogate_exact_14 = dGaussianProcess(cov_func, optimize=opt)

exact_14 = dGPGO(surrogate_exact_14, Acquisition_new(util_grad_exact), objfunc, param)
exact_14.run(init_evals=n_init, max_iter=iters)

```

```
Evaluation       Proposed point          Current eval.           Best eval.
init     [ 2.70915016 11.59747578].      -81.58264375064097      -1.4149920024014744
init     [8.05641529 0.12070423].        -10.148649076616518     -1.4149920024014744
init     [-0.35396112 14.36405609].      -79.6056658956529       -1.4149920024014744
init     [2.69675068 4.77426637].        -5.855607664288365      -1.4149920024014744
init     [3.08799906 3.31882414].        -1.4149920024014744     -1.4149920024014744
1        [-5.          1.71244664].      -252.1966961125983      -1.4149920024014744
2        [9.91083167 7.9391897 ].        -26.74726853005628      -1.4149920024014744
3        [-3.66670736  8.25640817].      -29.95349403696578      -1.4149920024014744
4        [ 9.14372498 14.98887041].      -163.10109673985522     -1.4149920024014744
5        [0.3921633 0.       ].          -47.98696017190056      -1.4149920024014744
6        [-5.         12.95513221].      -30.63550731378922      -1.4149920024014744
7        [7.03280169 4.32120341].        -26.792334386451934     -1.4149920024014744
8        [5.26695842 7.93629954].        -60.41965862881821      -1.4149920024014744
9        [-1.13962388  4.24944697].      -27.941627651336383     -1.4149920024014744
10       [4.40009097 0.        ].        -9.294249499050936      -1.4149920024014744
11       [1.01714263 7.64375223].        -24.83900161320126      -1.4149920024014744
12       [ 7.60116152 11.07880939].      -106.73362003157524     -1.4149920024014744
13       [ 4.9762777  14.22648165].      -180.140622096974       -1.4149920024014744
14       [-1.65868501 10.94126534].      -12.943964751892187     -1.4149920024014744
15       [9.83162899 2.85709599].        -1.1820007042698428     -1.1820007042698428
16       [-4.05411965  5.14748752].      -93.01510588278751      -1.1820007042698428
17       [2.77870657 2.16721846].        -1.1895085643088663     -1.1820007042698428
18       [4.24076527 2.28029862].        -6.136469752575828      -1.1820007042698428
19       [1.46056    2.97645243].        -12.006155733257126     -1.1820007042698428
20       [9.56548939 1.49088682].        -1.7146198322776307     -1.1820007042698428
```

```
### EXACT GP EI GRADIENTS

np.random.seed(run_num_15)
surrogate_exact_15 = dGaussianProcess(cov_func, optimize=opt)

exact_15 = dGPGO(surrogate_exact_15, Acquisition_new(util_grad_exact), objfunc, param)
exact_15.run(init_evals=n_init, max_iter=iters)

```

```
Evaluation       Proposed point          Current eval.           Best eval.
init     [7.73226546 2.68343887].        -12.768724532005583     -12.768724532005583
init     [-4.18455179  5.42307669].      -95.39334176626551      -12.768724532005583
init     [-0.86898607  7.95000337].      -16.419487782111716     -12.768724532005583
init     [-0.41121626  4.56711539].      -23.250362743740954     -12.768724532005583
init     [-3.32388086  3.74848521].      -80.99796277938532      -12.768724532005583
1        [ 6.5655417  11.87699252].      -134.95011349993544     -12.768724532005583
```

```
2        [-3.60562328 14.8673558 ].       -3.5139061901593145    -3.5139061901593145
3        [3.70191311 0.         ].       -5.395287153486009     -3.5139061901593145
4        [ 1.1492482  14.87856364].       -124.95784214201342    -3.5139061901593145
5        [4.48125008 6.99235223].         -38.383967237383466    -3.5139061901593145
6        [-5.         10.62235039].       -55.823105861698835    -3.5139061901593145
7        [9.59540512 6.67459399].         -16.955261629397306    -3.5139061901593145
8        [-5.01648060e-01 -5.55111512e-17].       -65.0803396750184       -3.51390619
9        [-5.  0.].       -308.12909601160663     -3.5139061901593145
10       [ 2.16263768 10.35244426].       -56.34194944059825     -3.5139061901593145
11       [3.06960752 3.37294058].         -1.5067127234240214    -1.5067127234240214
12       [ 9.57760795 14.48621149].       -141.6270720718067     -1.5067127234240214
13       [-1.36225854 11.52038799].       -21.675902983916835    -1.5067127234240214
14       [9.94404917 9.91027059].         -50.138929140564244    -1.5067127234240214
15       [4.85270452 2.49791699].         -12.733194037409271    -1.5067127234240214
16       [6.99530507 0.         ].        -18.680361156939213    -1.5067127234240214
17       [1.89296949 2.20732254].         -8.504332566535222     -1.5067127234240214
18       [9.71179631 0.72341832].         -4.808026773401662     -1.5067127234240214
19       [ 5.0254296  14.85771607].       -197.73720446173766    -1.5067127234240214
20       [6.96242292 5.47214431].         -35.88295729004187     -1.5067127234240214
```

```
### EXACT GP EI GRADIENTS

np.random.seed(run_num_16)
surrogate_exact_16 = dGaussianProcess(cov_func, optimize=opt)

exact_16 = dGPGO(surrogate_exact_16, Acquisition_new(util_grad_exact), objfunc, param)
exact_16.run(init_evals=n_init, max_iter=iters)

```

```
Evaluation       Proposed point           Current eval.          Best eval.
init     [-1.65063381  7.84745012].       -10.514702126319445    -2.715864006988424
init     [3.26052185 0.68402925].         -2.715864006988424     -2.715864006988424
init     [0.41093253 3.34621413].         -22.889515127492515    -2.715864006988424
init     [5.33089243 2.45597138].         -17.178157611778595    -2.715864006988424
init     [-3.945127   14.1151629].        -3.3649224341694195    -2.715864006988424
1        [ 5.63879248 14.58068289].       -198.51319639462758    -2.715864006988424
2        [9.91844219 8.81695252].         -36.28422213736768     -2.715864006988424
3        [-5.  0.].       -308.12909601160663     -2.715864006988424
4        [3.85654417 9.13706995].         -56.82460928884615     -2.715864006988424
5        [9.99139112 1.55175532].         -3.9797647677295895    -2.715864006988424
6        [ 0.40948096 12.76888861].       -73.55254909962426     -2.715864006988424
7        [-5.          5.14765042].       -157.678360254256      -2.715864006988424
8        [-5.         10.25993479].       -60.712974646155395    -2.715864006988424
9        [8.59339051 4.84654713].         -12.431157376265867    -2.715864006988424
10       [ 9.97855902 14.65271178].       -138.04449874167767    -2.715864006988424
11       [-0.50779219  0.        ].       -65.19646384909731     -2.715864006988424
12       [3.13335423 5.5180022 ].         -10.87358963697566     -2.715864006988424
13       [7.42632566 0.         ].        -15.685922322919057    -2.715864006988424
14       [6.18039596 6.62907409].         -50.143182479314746    -2.715864006988424
15       [ 7.10301331 10.45283618].       -101.9279648753849     -2.715864006988424
16       [0.74686818 7.06946503].         -21.825177154408294    -2.715864006988424
17       [-2.29061893  2.65829196].       -62.42453445451797     -2.715864006988424
18       [-2.91201402 12.80602327].       -1.807485731494669     -1.807485731494669
19       [-1.85489977 10.49462588].       -8.51412186644138      -1.807485731494669
20       [2.77713907 2.30922964].         -1.0999371186934965    -1.0999371186934965
```

```
### EXACT GP EI GRADIENTS
```

```
np.random.seed(run_num_17)
surrogate_exact_17 = dGaussianProcess(cov_func, optimize=opt)

exact_17 = dGPGO(surrogate_exact_17, Acquisition_new(util_grad_exact), objfunc, param)
exact_17.run(init_evals=n_init, max_iter=iters)

```

```
  Evaluation       Proposed point            Current eval.          Best eval.
  init     [-0.58002496  7.95880133].       -19.016141117164256    -19.016141117164256
  init     [-2.1271882   1.01850537].       -85.05949339404415     -19.016141117164256
  init     [6.8047819  9.84500283].         -93.89792695202516     -19.016141117164256
  init     [4.56281344 8.63404341].         -60.50217141122736     -19.016141117164256
  init     [-4.41405626  5.36720407].       -110.70834086593878    -19.016141117164256
  1        [9.76575296 0.98553483].         -4.1623278326263815    -4.1623278326263815
  2        [-4.0692092  14.64043819].       -4.241825729253317     -4.1623278326263815
  3        [4.06607537 0.        ].         -6.987522719297042     -4.1623278326263815
  4        [ 3.03819562 14.9592515 ].       -159.26550321070087    -4.1623278326263815
  5        [-5.         10.41147783].       -58.63633387366506     -4.1623278326263815
  6        [ 9.7165504  14.35949467].       -135.9996997459606     -4.1623278326263815
  7        [8.19532719 4.95303034].         -17.806908799512883    -4.1623278326263815
  8        [1.11582031 4.1832658 ].         -14.260239804067487    -4.1623278326263815
  9        [-0.34010601 11.89060981].       -47.50761351775903     -4.1623278326263815
  10       [4.64303553 3.99239564].         -16.079445109534554    -4.1623278326263815
  11       [6.63454958 0.79833416].         -19.123570939816663    -4.1623278326263815
  12       [1.32968955 0.        ].         -29.2024737149897      -4.1623278326263815
  13       [ 6.38692789 13.15818715].       -164.8372187830587     -4.1623278326263815
  14       [ 9.99695878 10.33071554].       -55.66755430261044     -4.1623278326263815
  15       [-5.  0.].       -308.12909601160663    -4.1623278326263815
  16       [2.64367617 6.18622385].         -13.75010405502346     -4.1623278326263815
  17       [ 2.66790796 11.37512455].       -77.17526687618609     -4.1623278326263815
  18       [-1.66653152 14.52358219].       -39.46899750437008     -4.1623278326263815
  19       [9.81961649 7.06573159].         -19.093409543666745    -4.1623278326263815
  20       [-5.         13.46984178].       -26.54369713901015     -4.1623278326263815
```

```
### EXACT GP EI GRADIENTS

np.random.seed(run_num_18)
surrogate_exact_18 = dGaussianProcess(cov_func, optimize=opt)

exact_18 = dGPGO(surrogate_exact_18, Acquisition_new(util_grad_exact), objfunc, param)
exact_18.run(init_evals=n_init, max_iter=iters)

```

```
  Evaluation       Proposed point            Current eval.          Best eval.
  init     [4.75561363 7.58180061].         -49.215059064668324    -8.150075223157177
  init     [8.17902206 2.72760338].         -8.150075223157177     -8.150075223157177
  init     [ 7.78349603 11.25204429].       -106.98107856533271    -8.150075223157177
  init     [ 4.99152501 14.81843172].       -196.08601750690153    -8.150075223157177
  init     [-1.14547366  0.42458888].       -71.23649635555944     -8.150075223157177
  1        [-4.83474958 11.77354131].       -35.58412906267954     -8.150075223157177
  2        [-1.88826854  6.70399181].       -14.630614325969386    -8.150075223157177
  3        [-0.51657686 14.28710066].       -73.5610643915935      -8.150075223157177
  4        [4.18472296 0.83236175].         -5.757617883721708     -5.757617883721708
  5        [ 0.86842496 10.0227023 ].       -44.37192956080076     -5.757617883721708
  6        [-5.          3.18813342].       -208.702097778099      -5.757617883721708
  7        [1.84462578 4.06147988].         -7.7144510287816       -5.757617883721708
  8        [ 9.59675026 14.67960992].       -145.8800819268564     -5.757617883721708
  9        [8.66313736 6.69467935].         -25.968187639214978    -5.757617883721708
```

```
10      [-5.          7.88814383].       -99.19917553327659      -5.757617883721708
11      [7.22480662 0.        ].         -17.199447285302824     -5.757617883721708
12      [5.19197717 3.808651  ].         -21.13637250329181      -5.757617883721708
13      [-1.12533381  3.82887264].       -31.15910927858084      -5.757617883721708
14      [-4.68199713 14.92551895].       -11.552320795324627     -5.757617883721708
15      [ 3.82478451 11.07643055].       -88.55869087071808      -5.757617883721708
16      [-2.26717971 10.08322737].       -3.876517881589809      -3.876517881589809
17      [1.52644442 6.30964627].         -16.369839276929937     -3.876517881589809
18      [1.87680645 1.16582288].         -12.407364659722182     -3.876517881589809
19      [-5.  0.].       -308.12909601160663     -3.876517881589809
20      [9.97696352 2.55733096].         -2.003744773764411      -2.003744773764411
```

```
### EXACT GP EI GRADIENTS

np.random.seed(run_num_19)
surrogate_exact_19 = dGaussianProcess(cov_func, optimize=opt)

exact_19 = dGPGO(surrogate_exact_19, Acquisition_new(util_grad_exact), objfunc, param)
exact_19.run(init_evals=n_init, max_iter=iters)

```

```
Evaluation       Proposed point          Current eval.           Best eval.
init    [-3.53699597 11.41874575].       -4.475608269502271      -4.475608269502271
init    [-1.2959304   2.07197531].       -51.13947793770689      -4.475608269502271
init    [-0.02830155  1.24499348].       -42.63973921760034      -4.475608269502271
init    [ 5.07965622 12.09890697].       -131.17235664506808     -4.475608269502271
init    [9.74112872 9.53491102].         -46.84458913925112      -4.475608269502271
1       [8.6773951  0.80876842].         -4.184707152540867      -4.184707152540867
2       [4.45041553 5.47999855].         -23.54840201771013      -4.184707152540867
3       [-4.04838436  6.32106816].       -71.9700715917133       -4.184707152540867
4       [-8.69860031e-04  1.42821549e+01].       -88.17326654597659      -4.18470715
5       [0.80655347 8.67861441].         -31.685363189083702     -4.184707152540867
6       [4.5267817 0.       ].           -10.309146212628553     -4.184707152540867
7       [ 9.55368033 14.5300422 ].       -143.140599752017       -4.184707152540867
8       [-5.  0.].       -308.12909601160663     -4.184707152540867
9       [8.98634645 4.92281445].         -9.105848201119821      -4.184707152540867
10      [0.80400624 5.10119911].         -16.750598342783768     -4.184707152540867
11      [6.38776327 2.54152106].         -21.61398856083099      -4.184707152540867
12      [-4.26795547 14.34625502].       -6.510901919506036      -4.184707152540867
13      [6.65649654 7.59676408].         -60.76127542886388      -4.184707152540867
14      [-5.          9.61899984].       -70.00383005346774      -4.184707152540867
15      [3.09847994 2.21049321].         -0.4164861410776428     -0.4164861410776428
16      [2.87865522 2.85419458].         -0.8612833800100894     -0.4164861410776428
17      [-1.98482839 10.07211907].       -6.300440821720379      -0.4164861410776428
18      [3.66391216 8.65411004].         -47.25691856580857 6    -0.4164861410776428
19      [ 3.02409105 14.90354405].       -157.59347477537295     -0.4164861410776428
20      [2.8461419  1.89221928].         -1.203896878362313      -0.4164861410776428
```

```
### EXACT GP EI GRADIENTS

np.random.seed(run_num_20)
surrogate_exact_20 = dGaussianProcess(cov_func, optimize=opt)

exact_20 = dGPGO(surrogate_exact_20, Acquisition_new(util_grad_exact), objfunc, param)
exact_20.run(init_evals=n_init, max_iter=iters)

```

```
Evaluation       Proposed point            Current eval.          Best eval.
init     [ 3.82196202 13.46570592].        -138.5264349938869     -14.042667401507376
init     [ 8.37296094 12.23756216].        -115.63178540512689    -14.042667401507376
init     [-4.46165622 10.37636373].        -35.66708529307584     -14.042667401507376
init     [0.68021413 7.77766418].          -25.307769914281764    -14.042667401507376
init     [4.86927198 2.90775327].          -14.042667401507376    -14.042667401507376
1        [-4.88337277  1.15554411].        -258.03874991509736    -14.042667401507376
2        [-1.84310627 14.89552828].        -37.92416090431381     -14.042667401507376
3        [9.98998478 0.35020336].          -8.875777366291523     -8.875777366291523
4        [9.82049302 5.23462252].          -6.926835976089437     -6.926835976089437
5        [0.11401177 1.25905543].          -40.34403021450308     -6.926835976089437
6        [6.34167483 7.90009461].          -65.79565375919643     -6.926835976089437
7        [-5.          5.51340856].        -149.00489585112476    -6.926835976089437
8        [-0.53405313 11.49764199].        -39.52473478318865     -6.926835976089437
9        [5.78794253 0.        ].          -19.693729446216174    -6.926835976089437
10       [-0.56018414  4.58448812].        -23.64577960232005     -6.926835976089437
11       [7.94295741 3.09170569].          -11.652630026865006    -6.926835976089437
12       [3.53895381 5.13608054].          -11.072114646842019    -6.926835976089437
13       [-5.         13.92763768].        -23.349545126428623    -6.926835976089437
14       [9.82106318 7.9963632 ].          -27.837869899793276    -6.926835976089437
15       [2.71099301 9.86405986].          -53.53718705475601     -6.926835976089437
16       [2.69605974 0.        ].          -8.347647162761653     -6.926835976089437
17       [ 9.5753511  14.80706293].        -149.3983233662887     -6.926835976089437
18       [-2.55613798  8.18045993].        -9.45991547733541      -6.926835976089437
19       [6.9362799  4.81790452].          -30.890336248422418    -6.926835976089437
20       [ 5.75900204 10.82833027].        -112.58763977748158    -6.926835976089437
```

```python
end_exact = time.time()
end_exact

time_exact = end_exact - start_exact
time_exact
```

```
775.5369441509247
```

```python
### Simple regret minimization: run number = 1

approx_output_1 = np.append(np.min(approx_1.GP.y[0:n_init]),approx_1.GP.y[n_init:(n_ini
exact_output_1 = np.append(np.min(exact_1.GP.y[0:n_init]),exact_1.GP.y[n_init:(n_init+i

regret_approx_1 = np.log(-approx_output_1 + y_global_orig)
regret_exact_1 = np.log(-exact_output_1 + y_global_orig)

simple_regret_approx_1 = min_max_array(regret_approx_1)
simple_regret_exact_1 = min_max_array(regret_exact_1)

min_simple_regret_approx_1 = min(simple_regret_approx_1)
min_simple_regret_exact_1 = min(simple_regret_exact_1)

min_simple_regret_approx_1, min_simple_regret_exact_1
```

```
(1.4140401206065185, 1.0935804752363887)
```

```python
### Simple regret minimization: run number = 2

```

```
approx_output_2 = np.append(np.min(approx_2.GP.y[0:n_init]),approx_2.GP.y[n_init:(n_ini
exact_output_2 = np.append(np.min(exact_2.GP.y[0:n_init]),exact_2.GP.y[n_init:(n_init+i

regret_approx_2 = np.log(-approx_output_2 + y_global_orig)
regret_exact_2 = np.log(-exact_output_2 + y_global_orig)

simple_regret_approx_2 = min_max_array(regret_approx_2)
simple_regret_exact_2 = min_max_array(regret_exact_2)

min_simple_regret_approx_2 = min(simple_regret_approx_2)
min_simple_regret_exact_2 = min(simple_regret_exact_2)

min_simple_regret_approx_2, min_simple_regret_exact_2
```

```
(-0.45755271593560126, -0.5327401667959443)
```

```
### Simple regret minimization: run number = 3

approx_output_3 = np.append(np.min(approx_3.GP.y[0:n_init]),approx_3.GP.y[n_init:(n_ini
exact_output_3 = np.append(np.min(exact_3.GP.y[0:n_init]),exact_3.GP.y[n_init:(n_init+i

regret_approx_3 = np.log(-approx_output_3 + y_global_orig)
regret_exact_3 = np.log(-exact_output_3 + y_global_orig)

simple_regret_approx_3 = min_max_array(regret_approx_3)
simple_regret_exact_3 = min_max_array(regret_exact_3)

min_simple_regret_approx_3 = min(simple_regret_approx_3)
min_simple_regret_exact_3 = min(simple_regret_exact_3)

min_simple_regret_approx_3, min_simple_regret_exact_3
```

```
(-0.28844349713038764, -0.38423333281658867)
```

```
### Simple regret minimization: run number = 4

approx_output_4 = np.append(np.min(approx_4.GP.y[0:n_init]),approx_4.GP.y[n_init:(n_ini
exact_output_4 = np.append(np.min(exact_4.GP.y[0:n_init]),exact_4.GP.y[n_init:(n_init+i

regret_approx_4 = np.log(-approx_output_4 + y_global_orig)
regret_exact_4 = np.log(-exact_output_4 + y_global_orig)

simple_regret_approx_4 = min_max_array(regret_approx_4)
simple_regret_exact_4 = min_max_array(regret_exact_4)

min_simple_regret_approx_4 = min(simple_regret_approx_4)
min_simple_regret_exact_4 = min(simple_regret_exact_4)

min_simple_regret_approx_4, min_simple_regret_exact_4
```

```
(0.9092818714485758, -0.029980791108480058)
```

```
### Simple regret minimization: run number = 5

```

```

approx_output_5 = np.append(np.min(approx_5.GP.y[0:n_init]),approx_5.GP.y[n_init:(n_ini
exact_output_5 = np.append(np.min(exact_5.GP.y[0:n_init]),exact_5.GP.y[n_init:(n_init+i

regret_approx_5 = np.log(-approx_output_5 + y_global_orig)
regret_exact_5 = np.log(-exact_output_5 + y_global_orig)

simple_regret_approx_5 = min_max_array(regret_approx_5)
simple_regret_exact_5 = min_max_array(regret_exact_5)

min_simple_regret_approx_5 = min(simple_regret_approx_5)
min_simple_regret_exact_5 = min(simple_regret_exact_5)

min_simple_regret_approx_5, min_simple_regret_exact_5
```

```
(1.3273120831432876, 0.5522698135370556)
```

```
### Simple regret minimization: run number = 6

approx_output_6 = np.append(np.min(approx_6.GP.y[0:n_init]),approx_6.GP.y[n_init:(n_ini
exact_output_6 = np.append(np.min(exact_6.GP.y[0:n_init]),exact_6.GP.y[n_init:(n_init+i

regret_approx_6 = np.log(-approx_output_6 + y_global_orig)
regret_exact_6 = np.log(-exact_output_6 + y_global_orig)

simple_regret_approx_6 = min_max_array(regret_approx_6)
simple_regret_exact_6 = min_max_array(regret_exact_6)

min_simple_regret_approx_6 = min(simple_regret_approx_6)
min_simple_regret_exact_6 = min(simple_regret_exact_6)

min_simple_regret_approx_6, min_simple_regret_exact_6
```

```
(0.842517075174006, -1.4254655049645975)
```

```
### Simple regret minimization: run number = 7

approx_output_7 = np.append(np.min(approx_7.GP.y[0:n_init]),approx_7.GP.y[n_init:(n_ini
exact_output_7 = np.append(np.min(exact_7.GP.y[0:n_init]),exact_7.GP.y[n_init:(n_init+i

regret_approx_7 = np.log(-approx_output_7 + y_global_orig)
regret_exact_7 = np.log(-exact_output_7 + y_global_orig)

simple_regret_approx_7 = min_max_array(regret_approx_7)
simple_regret_exact_7 = min_max_array(regret_exact_7)

min_simple_regret_approx_7 = min(simple_regret_approx_7)
min_simple_regret_exact_7 = min(simple_regret_exact_7)

min_simple_regret_approx_7, min_simple_regret_exact_7
```

```
(0.19066991806673148, -2.874883800877702)
```

```
### Simple regret minimization: run number = 8
```

```

approx_output_8 = np.append(np.min(approx_8.GP.y[0:n_init]),approx_8.GP.y[n_init:(n_ini
exact_output_8 = np.append(np.min(exact_8.GP.y[0:n_init]),exact_8.GP.y[n_init:(n_init+i

regret_approx_8 = np.log(-approx_output_8 + y_global_orig)
regret_exact_8 = np.log(-exact_output_8 + y_global_orig)

simple_regret_approx_8 = min_max_array(regret_approx_8)
simple_regret_exact_8 = min_max_array(regret_exact_8)

min_simple_regret_approx_8 = min(simple_regret_approx_8)
min_simple_regret_exact_8 = min(simple_regret_exact_8)

min_simple_regret_approx_8, min_simple_regret_exact_8
```

```
(1.8496984829216314, 1.0842581625679653)
```

```
### Simple regret minimization: run number = 9

approx_output_9 = np.append(np.min(approx_9.GP.y[0:n_init]),approx_9.GP.y[n_init:(n_ini
exact_output_9 = np.append(np.min(exact_9.GP.y[0:n_init]),exact_9.GP.y[n_init:(n_init+i

regret_approx_9 = np.log(-approx_output_9 + y_global_orig)
regret_exact_9 = np.log(-exact_output_9 + y_global_orig)

simple_regret_approx_9 = min_max_array(regret_approx_9)
simple_regret_exact_9 = min_max_array(regret_exact_9)

min_simple_regret_approx_9 = min(simple_regret_approx_9)
min_simple_regret_exact_9 = min(simple_regret_exact_9)

min_simple_regret_approx_9, min_simple_regret_exact_9
```

```
(0.5311505318004143, 0.815807298232431)
```

```
### Simple regret minimization: run number = 10

approx_output_10 = np.append(np.min(approx_10.GP.y[0:n_init]),approx_10.GP.y[n_init:(n_
exact_output_10 = np.append(np.min(exact_10.GP.y[0:n_init]),exact_10.GP.y[n_init:(n_ini

regret_approx_10 = np.log(-approx_output_10 + y_global_orig)
regret_exact_10 = np.log(-exact_output_10 + y_global_orig)

simple_regret_approx_10 = min_max_array(regret_approx_10)
simple_regret_exact_10 = min_max_array(regret_exact_10)

min_simple_regret_approx_10 = min(simple_regret_approx_10)
min_simple_regret_exact_10 = min(simple_regret_exact_10)

min_simple_regret_approx_10, min_simple_regret_exact_10
```

```
(0.7270146338323442, -5.437914608852267)
```

```
### Simple regret minimization: run number = 11
### Simple regret minimization: run number = 11

approx_output_11 = np.append(np.min(approx_11.GP.y[0:n_init]),approx_11.GP.y[n_init:(n_
exact_output_11 = np.append(np.min(exact_11.GP.y[0:n_init]),exact_11.GP.y[n_init:(n_ini

regret_approx_11 = np.log(-approx_output_11 + y_global_orig)
regret_exact_11 = np.log(-exact_output_11 + y_global_orig)

simple_regret_approx_11 = min_max_array(regret_approx_11)
simple_regret_exact_11 = min_max_array(regret_exact_11)

min_simple_regret_approx_11 = min(simple_regret_approx_11)
min_simple_regret_exact_11 = min(simple_regret_exact_11)

min_simple_regret_approx_11, min_simple_regret_exact_11
```

```
(0.39575423934463444, -0.14968453976348015)
```

```
### Simple regret minimization: run number = 12

approx_output_12 = np.append(np.min(approx_12.GP.y[0:n_init]),approx_12.GP.y[n_init:(n_
exact_output_12 = np.append(np.min(exact_12.GP.y[0:n_init]),exact_12.GP.y[n_init:(n_ini

regret_approx_12 = np.log(-approx_output_12 + y_global_orig)
regret_exact_12 = np.log(-exact_output_12 + y_global_orig)

simple_regret_approx_12 = min_max_array(regret_approx_12)
simple_regret_exact_12 = min_max_array(regret_exact_12)

min_simple_regret_approx_12 = min(simple_regret_approx_12)
min_simple_regret_exact_12 = min(simple_regret_exact_12)

min_simple_regret_approx_12, min_simple_regret_exact_12
```

```
(0.7191542780910578, -2.272091418115509)
```

```
### Simple regret minimization: run number = 13

approx_output_13 = np.append(np.min(approx_13.GP.y[0:n_init]),approx_13.GP.y[n_init:(n_
exact_output_13 = np.append(np.min(exact_13.GP.y[0:n_init]),exact_13.GP.y[n_init:(n_ini

regret_approx_13 = np.log(-approx_output_13 + y_global_orig)
regret_exact_13 = np.log(-exact_output_13 + y_global_orig)

simple_regret_approx_13 = min_max_array(regret_approx_13)
simple_regret_exact_13 = min_max_array(regret_exact_13)

min_simple_regret_approx_13 = min(simple_regret_approx_13)
min_simple_regret_exact_13 = min(simple_regret_exact_13)

min_simple_regret_approx_13, min_simple_regret_exact_13
```

```
(0.047568844859905, -0.9953373835742909)
```

```
### Simple regret minimization: run number = 14

approx_output_14 = np.append(np.min(approx_14.GP.y[0:n_init]),approx_14.GP.y[n_init:(n_
exact_output_14 = np.append(np.min(exact_14.GP.y[0:n_init]),exact_14.GP.y[n_init:(n_ini

regret_approx_14 = np.log(-approx_output_14 + y_global_orig)
regret_exact_14 = np.log(-exact_output_14 + y_global_orig)

simple_regret_approx_14 = min_max_array(regret_approx_14)
simple_regret_exact_14 = min_max_array(regret_exact_14)

min_simple_regret_approx_14 = min(simple_regret_approx_14)
min_simple_regret_exact_14 = min(simple_regret_exact_14)

min_simple_regret_approx_14, min_simple_regret_exact_14
```

```
(0.4762878090317656, -0.24320123819126943)
```

```
### Simple regret minimization: run number = 15

approx_output_15 = np.append(np.min(approx_15.GP.y[0:n_init]),approx_15.GP.y[n_init:(n_
exact_output_15 = np.append(np.min(exact_15.GP.y[0:n_init]),exact_15.GP.y[n_init:(n_ini

regret_approx_15 = np.log(-approx_output_15 + y_global_orig)
regret_exact_15 = np.log(-exact_output_15 + y_global_orig)

simple_regret_approx_15 = min_max_array(regret_approx_15)
simple_regret_exact_15 = min_max_array(regret_exact_15)

min_simple_regret_approx_15 = min(simple_regret_approx_15)
min_simple_regret_exact_15 = min(simple_regret_exact_15)

min_simple_regret_approx_15, min_simple_regret_exact_15
```

```
(1.0888676626860307, 0.10330154852064004)
```

```
### Simple regret minimization: run number = 16

approx_output_16 = np.append(np.min(approx_16.GP.y[0:n_init]),approx_16.GP.y[n_init:(n_
exact_output_16 = np.append(np.min(exact_16.GP.y[0:n_init]),exact_16.GP.y[n_init:(n_ini

regret_approx_16 = np.log(-approx_output_16 + y_global_orig)
regret_exact_16 = np.log(-exact_output_16 + y_global_orig)

simple_regret_approx_16 = min_max_array(regret_approx_16)
simple_regret_exact_16 = min_max_array(regret_exact_16)

min_simple_regret_approx_16 = min(simple_regret_approx_16)
min_simple_regret_exact_16 = min(simple_regret_exact_16)

min_simple_regret_approx_16, min_simple_regret_exact_16
```

```
(1.1462234586538276, -0.353750483354487)
```

```
### Simple regret minimization: run number = 17

approx_output_17 = np.append(np.min(approx_17.GP.y[0:n_init]),approx_17.GP.y[n_init:(n_
exact_output_17 = np.append(np.min(exact_17.GP.y[0:n_init]),exact_17.GP.y[n_init:(n_ini

regret_approx_17 = np.log(-approx_output_17 + y_global_orig)
regret_exact_17 = np.log(-exact_output_17 + y_global_orig)

simple_regret_approx_17 = min_max_array(regret_approx_17)
simple_regret_exact_17 = min_max_array(regret_exact_17)

min_simple_regret_approx_17 = min(simple_regret_approx_17)
min_simple_regret_exact_17 = min(simple_regret_exact_17)

min_simple_regret_approx_17, min_simple_regret_exact_17
```

```
(0.48388485945608994, 1.3255993329913722)
```

```
### Simple regret minimization: run number = 18

approx_output_18 = np.append(np.min(approx_18.GP.y[0:n_init]),approx_18.GP.y[n_init:(n_
exact_output_18 = np.append(np.min(exact_18.GP.y[0:n_init]),exact_18.GP.y[n_init:(n_ini

regret_approx_18 = np.log(-approx_output_18 + y_global_orig)
regret_exact_18 = np.log(-exact_output_18 + y_global_orig)

simple_regret_approx_18 = min_max_array(regret_approx_18)
simple_regret_exact_18 = min_max_array(regret_exact_18)

min_simple_regret_approx_18 = min(simple_regret_approx_18)
min_simple_regret_exact_18 = min(simple_regret_exact_18)

min_simple_regret_approx_18, min_simple_regret_exact_18
```

```
(1.6733882078753337, 0.47365805230309505)
```

```
### Simple regret minimization: run number = 19

approx_output_19 = np.append(np.min(approx_19.GP.y[0:n_init]),approx_19.GP.y[n_init:(n_
exact_output_19 = np.append(np.min(exact_19.GP.y[0:n_init]),exact_19.GP.y[n_init:(n_ini

regret_approx_19 = np.log(-approx_output_19 + y_global_orig)
regret_exact_19 = np.log(-exact_output_19 + y_global_orig)

simple_regret_approx_19 = min_max_array(regret_approx_19)
simple_regret_exact_19 = min_max_array(regret_exact_19)

min_simple_regret_approx_19 = min(simple_regret_approx_19)
min_simple_regret_exact_19 = min(simple_regret_exact_19)

min_simple_regret_approx_19, min_simple_regret_exact_19
```

```
(-2.559174494760639, -3.9846398779506007)
```

```
### Simple regret minimization: run number = 20

approx_output_20 = np.append(np.min(approx_20.GP.y[0:n_init]),approx_20.GP.y[n_init:(n_
exact_output_20 = np.append(np.min(exact_20.GP.y[0:n_init]),exact_20.GP.y[n_init:(n_ini

regret_approx_20 = np.log(-approx_output_20 + y_global_orig)
regret_exact_20 = np.log(-exact_output_20 + y_global_orig)

simple_regret_approx_20 = min_max_array(regret_approx_20)
simple_regret_exact_20 = min_max_array(regret_exact_20)

min_simple_regret_approx_20 = min(simple_regret_approx_20)
min_simple_regret_exact_20 = min(simple_regret_exact_20)

min_simple_regret_approx_20, min_simple_regret_exact_20
```

```
(1.5958961782176881, 1.8762459772083349)
```

```
# Iteration1 :

slice1 = 0

approx1 = [simple_regret_approx_1[slice1],
       simple_regret_approx_2[slice1],
       simple_regret_approx_3[slice1],
       simple_regret_approx_4[slice1],
       simple_regret_approx_5[slice1],
       simple_regret_approx_6[slice1],
       simple_regret_approx_7[slice1],
       simple_regret_approx_8[slice1],
       simple_regret_approx_9[slice1],
       simple_regret_approx_10[slice1],
       simple_regret_approx_11[slice1],
       simple_regret_approx_12[slice1],
       simple_regret_approx_13[slice1],
       simple_regret_approx_14[slice1],
       simple_regret_approx_15[slice1],
       simple_regret_approx_16[slice1],
       simple_regret_approx_17[slice1],
       simple_regret_approx_18[slice1],
       simple_regret_approx_19[slice1],
       simple_regret_approx_20[slice1]]

exact1 = [simple_regret_exact_1[slice1],
       simple_regret_exact_2[slice1],
       simple_regret_exact_3[slice1],
       simple_regret_exact_4[slice1],
       simple_regret_exact_5[slice1],
       simple_regret_exact_6[slice1],
       simple_regret_exact_7[slice1],
       simple_regret_exact_8[slice1],
       simple_regret_exact_9[slice1],
       simple_regret_exact_10[slice1],
       simple_regret_exact_11[slice1]
       simple_regret_exact_11[slice1],
       simple_regret_exact_12[slice1],
       simple_regret_exact_13[slice1],
       simple_regret_exact_14[slice1],
       simple_regret_exact_15[slice1],
       simple_regret_exact_16[slice1],
       simple_regret_exact_17[slice1],
       simple_regret_exact_18[slice1],
       simple_regret_exact_19[slice1],
       simple_regret_exact_20[slice1]]

approx1_results = pd.DataFrame(approx1).sort_values(by=[0], ascending=False)
exact1_results = pd.DataFrame(exact1).sort_values(by=[0], ascending=False)

### Best simple regret minimization IQR - approx:
lower_approx1 = np.asarray(approx1_results[4:5][0])[0]
median_approx1 = np.asarray(approx1_results[9:10][0])[0]
upper_approx1 = np.asarray(approx1_results[14:15][0])[0]

lower_exact1 = np.asarray(exact1_results[4:5][0])[0]
median_exact1 = np.asarray(exact1_results[9:10][0])[0]
upper_exact1 = np.asarray(exact1_results[14:15][0])[0]
```

```
# Iteration11 :

slice11 = 10

approx11 = [simple_regret_approx_1[slice11],
       simple_regret_approx_2[slice11],
       simple_regret_approx_3[slice11],
       simple_regret_approx_4[slice11],
       simple_regret_approx_5[slice11],
       simple_regret_approx_6[slice11],
       simple_regret_approx_7[slice11],
       simple_regret_approx_8[slice11],
       simple_regret_approx_9[slice11],
       simple_regret_approx_10[slice11],
       simple_regret_approx_11[slice11],
       simple_regret_approx_12[slice11],
       simple_regret_approx_13[slice11],
       simple_regret_approx_14[slice11],
       simple_regret_approx_15[slice11],
       simple_regret_approx_16[slice11],
       simple_regret_approx_17[slice11],
       simple_regret_approx_18[slice11],
       simple_regret_approx_19[slice11],
       simple_regret_approx_20[slice11]]

exact11 = [simple_regret_exact_1[slice11],
       simple_regret_exact_2[slice11],
       simple_regret_exact_3[slice11],
       simple_regret_exact_4[slice11],
       simple_regret_exact_5[slice11],
       simple_regret_exact_6[slice11],
       simple_regret_exact_7[slice11],
       simple_regret_exact_8[slice11],
       simple_regret_exact_9[slice11],
       simple_regret_exact_10[slice11],
       simple_regret_exact_11[slice11],
       simple_regret_exact_12[slice11],
       simple_regret_exact_13[slice11],
       simple_regret_exact_14[slice11],
       simple_regret_exact_15[slice11],
       simple_regret_exact_16[slice11],
       simple_regret_exact_17[slice11],
       simple_regret_exact_18[slice11],
       simple_regret_exact_19[slice11],
       simple_regret_exact_20[slice11]]

approx11_results = pd.DataFrame(approx11).sort_values(by=[0], ascending=False)
exact11_results = pd.DataFrame(exact11).sort_values(by=[0], ascending=False)

### Best simple regret minimization IQR - approx:
lower_approx11 = np.asarray(approx11_results[4:5][0])[0]
median_approx11 = np.asarray(approx11_results[9:10][0])[0]
upper_approx11 = np.asarray(approx11_results[14:15][0])[0]

lower_exact11 = np.asarray(exact11_results[4:5][0])[0]
median_exact11 = np.asarray(exact11_results[9:10][0])[0]
upper_exact11 = np.asarray(exact11_results[14:15][0])[0]
```

```
# Iteration21 :

slice21 = 20

approx21 = [simple_regret_approx_1[slice21],
       simple_regret_approx_2[slice21],
       simple_regret_approx_3[slice21],
       simple_regret_approx_4[slice21],
       simple_regret_approx_5[slice21],
       simple_regret_approx_6[slice21],
       simple_regret_approx_7[slice21],
       simple_regret_approx_8[slice21],
       simple_regret_approx_9[slice21],
       simple_regret_approx_10[slice21],
       simple_regret_approx_11[slice21],
       simple_regret_approx_12[slice21],
       simple_regret_approx_13[slice21],
       simple_regret_approx_14[slice21],
       simple_regret_approx_15[slice21],
       simple_regret_approx_16[slice21],
       simple_regret_approx_17[slice21],
       simple_regret_approx_18[slice21],
       simple_regret_approx_19[slice21],
       simple_regret_approx_20[slice21]]

exact21 = [simple_regret_exact_1[slice21],
       simple_regret_exact_2[slice21],
       simple_regret_exact_3[slice21],
       simple_regret_exact_4[slice21],
       simple_regret_exact_5[slice21],
       simple_regret_exact_6[slice21],
       simple_regret_exact_7[slice21],
       simple_regret_exact_8[slice21],
       simple_regret_exact_9[slice21],
       simple_regret_exact_10[slice21],
       simple_regret_exact_11[slice21],
       simple_regret_exact_12[slice21],
       simple_regret_exact_13[slice21],
       simple_regret_exact_14[slice21],
       simple_regret_exact_15[slice21],
       simple_regret_exact_16[slice21],
       simple_regret_exact_17[slice21],
       simple_regret_exact_18[slice21],
       simple_regret_exact_19[slice21],
       simple_regret_exact_20[slice21]]

approx21_results = pd.DataFrame(approx21).sort_values(by=[0], ascending=False)
exact21_results = pd.DataFrame(exact21).sort_values(by=[0], ascending=False)

### Best simple regret minimization IQR - approx:
lower_approx21 = np.asarray(approx21_results[4:5][0])[0]
median_approx21 = np.asarray(approx21_results[9:10][0])[0]
upper_approx21 = np.asarray(approx21_results[14:15][0])[0]

lower_exact21 = np.asarray(exact21_results[4:5][0])[0]
median_exact21 = np.asarray(exact21_results[9:10][0])[0]
upper_exact21 = np.asarray(exact21_results[14:15][0])[0]
```

```
# Iteration2 :

slice2 = 1

approx2 = [simple_regret_approx_1[slice2],
       simple_regret_approx_2[slice2],
       simple_regret_approx_3[slice2],
       simple_regret_approx_4[slice2],
       simple_regret_approx_5[slice2],
       simple_regret_approx_6[slice2],
       simple_regret_approx_7[slice2],
       simple_regret_approx_8[slice2],
       simple_regret_approx_9[slice2],
       simple_regret_approx_10[slice2],
       simple_regret_approx_11[slice2],
       simple_regret_approx_12[slice2],
       simple_regret_approx_13[slice2],
       simple_regret_approx_14[slice2],
       simple_regret_approx_15[slice2],
       simple_regret_approx_16[slice2],
       simple_regret_approx_17[slice2],
       simple_regret_approx_18[slice2],
       simple_regret_approx_19[slice2],
       simple_regret_approx_20[slice2]]
```

```
exact2 = [simple_regret_exact_1[slice2],
       simple_regret_exact_2[slice2],
       simple_regret_exact_3[slice2],
       simple_regret_exact_4[slice2],
       simple_regret_exact_5[slice2],
       simple_regret_exact_6[slice2],
       simple_regret_exact_7[slice2],
       simple_regret_exact_8[slice2],
       simple_regret_exact_9[slice2],
       simple_regret_exact_10[slice2],
       simple_regret_exact_11[slice2],
       simple_regret_exact_12[slice2],
       simple_regret_exact_13[slice2],
       simple_regret_exact_14[slice2],
       simple_regret_exact_15[slice2],
       simple_regret_exact_16[slice2],
       simple_regret_exact_17[slice2],
       simple_regret_exact_18[slice2],
       simple_regret_exact_19[slice2],
       simple_regret_exact_20[slice2]]

approx2_results = pd.DataFrame(approx2).sort_values(by=[0], ascending=False)
exact2_results = pd.DataFrame(exact2).sort_values(by=[0], ascending=False)

### Best simple regret minimization IQR - approx:
lower_approx2 = np.asarray(approx2_results[4:5][0])[0]
median_approx2 = np.asarray(approx2_results[9:10][0])[0]
upper_approx2 = np.asarray(approx2_results[14:15][0])[0]

lower_exact2 = np.asarray(exact2_results[4:5][0])[0]
median_exact2 = np.asarray(exact2_results[9:10][0])[0]
upper_exact2 = np.asarray(exact2_results[14:15][0])[0]
```

```
# Iteration12 :

slice12 = 11

approx12 = [simple_regret_approx_1[slice12],
       simple_regret_approx_2[slice12],
       simple_regret_approx_3[slice12],
       simple_regret_approx_4[slice12],
       simple_regret_approx_5[slice12],
       simple_regret_approx_6[slice12],
       simple_regret_approx_7[slice12],
       simple_regret_approx_8[slice12],
       simple_regret_approx_9[slice12],
       simple_regret_approx_10[slice12],
       simple_regret_approx_11[slice12],
       simple_regret_approx_12[slice12],
       simple_regret_approx_13[slice12],
       simple_regret_approx_14[slice12],
       simple_regret_approx_15[slice12],
       simple_regret_approx_16[slice12],
       simple_regret_approx_17[slice12],
       simple_regret_approx_18[slice12],
       simple_regret_approx_19[slice12],
       simple_regret_approx_20[slice12]]

exact12 = [simple_regret_exact_1[slice12],
       simple_regret_exact_2[slice12],
       simple_regret_exact_3[slice12],
       simple_regret_exact_4[slice12],
       simple_regret_exact_5[slice12],
       simple_regret_exact_6[slice12],
       simple_regret_exact_7[slice12],
       simple_regret_exact_8[slice12],
       simple_regret_exact_9[slice12],
       simple_regret_exact_10[slice12],
       simple_regret_exact_11[slice12],
       simple_regret_exact_12[slice12],
       simple_regret_exact_13[slice12],
       simple_regret_exact_14[slice12],
       simple_regret_exact_15[slice12],
       simple_regret_exact_16[slice12],
       simple_regret_exact_17[slice12],
       simple_regret_exact_18[slice12],
       simple_regret_exact_19[slice12],
       simple_regret_exact_20[slice12]]

approx12_results = pd.DataFrame(approx12).sort_values(by=[0], ascending=False)
exact12_results = pd.DataFrame(exact12).sort_values(by=[0], ascending=False)

### Best simple regret minimization IQR - approx:
lower_approx12 = np.asarray(approx12_results[4:5][0])[0]
median_approx12 = np.asarray(approx12_results[9:10][0])[0]
upper_approx12 = np.asarray(approx12_results[14:15][0])[0]

lower_exact12 = np.asarray(exact12_results[4:5][0])[0]
median_exact12 = np.asarray(exact12_results[9:10][0])[0]
upper_exact12 = np.asarray(exact12_results[14:15][0])[0]
```

```
# Iteration3 :

slice3 = 2

approx3 = [simple_regret_approx_1[slice3],
       simple_regret_approx_2[slice3],
       simple_regret_approx_3[slice3],
       simple_regret_approx_4[slice3],
       simple_regret_approx_5[slice3],
       simple_regret_approx_6[slice3],
       simple_regret_approx_7[slice3],
       simple_regret_approx_8[slice3],
       simple_regret_approx_9[slice3],
       simple_regret_approx_10[slice3],
       simple_regret_approx_11[slice3],
       simple_regret_approx_12[slice3],
       simple_regret_approx_13[slice3],
       simple_regret_approx_14[slice3],
       simple_regret_approx_15[slice3],
       simple_regret_approx_16[slice3],
       simple_regret_approx_17[slice3],
       simple_regret_approx_18[slice3],
       simple_regret_approx_19[slice3],
       simple_regret_approx_20[slice3]]

exact3 = [simple_regret_exact_1[slice3],
       simple_regret_exact_2[slice3],
       simple_regret_exact_3[slice3],
       simple_regret_exact_4[slice3],
       simple_regret_exact_5[slice3],
       simple_regret_exact_6[slice3],
       simple_regret_exact_7[slice3],
       simple_regret_exact_8[slice3],
       simple_regret_exact_9[slice3],
       simple_regret_exact_10[slice3],
       simple_regret_exact_11[slice3],
       simple_regret_exact_12[slice3],
       simple_regret_exact_13[slice3],
       simple_regret_exact_14[slice3],
       simple_regret_exact_15[slice3],
       simple_regret_exact_16[slice3],
       simple_regret_exact_17[slice3],
       simple_regret_exact_18[slice3],
       simple_regret_exact_19[slice3],
       simple_regret_exact_20[slice3]]

approx3_results = pd.DataFrame(approx3).sort_values(by=[0], ascending=False)
exact3_results = pd.DataFrame(exact3).sort_values(by=[0], ascending=False)

### Best simple regret minimization IQR - approx:
lower_approx3 = np.asarray(approx3_results[4:5][0])[0]
median_approx3 = np.asarray(approx3_results[9:10][0])[0]
upper_approx3 = np.asarray(approx3_results[14:15][0])[0]

lower_exact3 = np.asarray(exact3_results[4:5][0])[0]
median_exact3 = np.asarray(exact3_results[9:10][0])[0]
upper_exact3 = np.asarray(exact3_results[14:15][0])[0]
```

```
# Iteration13 :

slice13 = 12

approx13 = [simple_regret_approx_1[slice13],
       simple_regret_approx_2[slice13],
       simple_regret_approx_3[slice13],
       simple_regret_approx_4[slice13],
       simple_regret_approx_5[slice13],
       simple_regret_approx_6[slice13],
       simple_regret_approx_7[slice13],
       simple_regret_approx_8[slice13],
       simple_regret_approx_9[slice13],
       simple_regret_approx_10[slice13],
       simple_regret_approx_10[slice13],
       simple_regret_approx_11[slice13],
       simple_regret_approx_12[slice13],
       simple_regret_approx_13[slice13],
       simple_regret_approx_14[slice13],
       simple_regret_approx_15[slice13],
       simple_regret_approx_16[slice13],
       simple_regret_approx_17[slice13],
       simple_regret_approx_18[slice13],
       simple_regret_approx_19[slice13],
       simple_regret_approx_20[slice13]]

exact13 = [simple_regret_exact_1[slice13],
       simple_regret_exact_2[slice13],
       simple_regret_exact_3[slice13],
       simple_regret_exact_4[slice13],
       simple_regret_exact_5[slice13],
       simple_regret_exact_6[slice13],
       simple_regret_exact_7[slice13],
       simple_regret_exact_8[slice13],
       simple_regret_exact_9[slice13],
       simple_regret_exact_10[slice13],
       simple_regret_exact_11[slice13],
       simple_regret_exact_12[slice13],
       simple_regret_exact_13[slice13],
       simple_regret_exact_14[slice13],
       simple_regret_exact_15[slice13],
       simple_regret_exact_16[slice13],
       simple_regret_exact_17[slice13],
       simple_regret_exact_18[slice13],
       simple_regret_exact_19[slice13],
       simple_regret_exact_20[slice13]]

approx13_results = pd.DataFrame(approx13).sort_values(by=[0], ascending=False)
exact13_results = pd.DataFrame(exact13).sort_values(by=[0], ascending=False)

### Best simple regret minimization IQR - approx:
lower_approx13 = np.asarray(approx13_results[4:5][0])[0]
median_approx13 = np.asarray(approx13_results[9:10][0])[0]
upper_approx13 = np.asarray(approx13_results[14:15][0])[0]

lower_exact13 = np.asarray(exact13_results[4:5][0])[0]
median_exact13 = np.asarray(exact13_results[9:10][0])[0]
upper_exact13 = np.asarray(exact13_results[14:15][0])[0]
```

```
# Iteration4 :

slice4 = 3

approx4 = [simple_regret_approx_1[slice4],
       simple_regret_approx_2[slice4],
       simple_regret_approx_3[slice4],
       simple_regret_approx_4[slice4],
       simple_regret_approx_5[slice4],
       simple_regret_approx_6[slice4],
       simple_regret_approx_7[slice4],
       simple_regret_approx_8[slice4],
       simple_regret_approx_9[slice4],
       simple_regret_approx_10[slice4],
       simple_regret_approx_11[slice4],
       simple_regret_approx_12[slice4],
       simple_regret_approx_13[slice4],
       simple_regret_approx_14[slice4],
       simple_regret_approx_15[slice4],
       simple_regret_approx_16[slice4],
       simple_regret_approx_17[slice4],
       simple_regret_approx_18[slice4],
       simple_regret_approx_19[slice4],
       simple_regret_approx_20[slice4]]

exact4 = [simple_regret_exact_1[slice4],
       simple_regret_exact_2[slice4],
       simple_regret_exact_3[slice4],
       simple_regret_exact_4[slice4],
       simple_regret_exact_5[slice4],
       simple_regret_exact_6[slice4],
       simple_regret_exact_7[slice4],
       simple_regret_exact_8[slice4],
       simple_regret_exact_9[slice4],
       simple_regret_exact_10[slice4],
       simple_regret_exact_11[slice4],
       simple_regret_exact_12[slice4],
       simple_regret_exact_13[slice4],
       simple_regret_exact_14[slice4],
       simple_regret_exact_15[slice4],
       simple_regret_exact_16[slice4],
       simple_regret_exact_17[slice4],
       simple_regret_exact_18[slice4],
       simple_regret_exact_19[slice4],
       simple_regret_exact_20[slice4]]

approx4_results = pd.DataFrame(approx4).sort_values(by=[0], ascending=False)
exact4_results = pd.DataFrame(exact4).sort_values(by=[0], ascending=False)

### Best simple regret minimization IQR - approx:
lower_approx4 = np.asarray(approx4_results[4:5][0])[0]
median_approx4 = np.asarray(approx4_results[9:10][0])[0]
upper_approx4 = np.asarray(approx4_results[14:15][0])[0]

lower_exact4 = np.asarray(exact4_results[4:5][0])[0]
median_exact4 = np.asarray(exact4_results[9:10][0])[0]
upper_exact4 = np.asarray(exact4_results[14:15][0])[0]
```

```
# Iteration14 :

slice14 = 13

approx14 = [simple_regret_approx_1[slice14],
       simple_regret_approx_2[slice14],
       simple_regret_approx_3[slice14],
       simple_regret_approx_3[slice14],
       simple_regret_approx_4[slice14],
       simple_regret_approx_5[slice14],
       simple_regret_approx_6[slice14],
       simple_regret_approx_7[slice14],
       simple_regret_approx_8[slice14],
       simple_regret_approx_9[slice14],
       simple_regret_approx_10[slice14],
       simple_regret_approx_11[slice14],
       simple_regret_approx_12[slice14],
       simple_regret_approx_13[slice14],
       simple_regret_approx_14[slice14],
       simple_regret_approx_15[slice14],
       simple_regret_approx_16[slice14],
       simple_regret_approx_17[slice14],
       simple_regret_approx_18[slice14],
       simple_regret_approx_19[slice14],
       simple_regret_approx_20[slice14]]

exact14 = [simple_regret_exact_1[slice14],
       simple_regret_exact_2[slice14],
       simple_regret_exact_3[slice14],
       simple_regret_exact_4[slice14],
       simple_regret_exact_5[slice14],
       simple_regret_exact_6[slice14],
       simple_regret_exact_7[slice14],
       simple_regret_exact_8[slice14],
       simple_regret_exact_9[slice14],
       simple_regret_exact_10[slice14],
       simple_regret_exact_11[slice14],
       simple_regret_exact_12[slice14],
       simple_regret_exact_13[slice14],
       simple_regret_exact_14[slice14],
       simple_regret_exact_15[slice14],
       simple_regret_exact_16[slice14],
       simple_regret_exact_17[slice14],
       simple_regret_exact_18[slice14],
       simple_regret_exact_19[slice14],
       simple_regret_exact_20[slice14]]

approx14_results = pd.DataFrame(approx14).sort_values(by=[0], ascending=False)
exact14_results = pd.DataFrame(exact14).sort_values(by=[0], ascending=False)

### Best simple regret minimization IQR - approx:
lower_approx14 = np.asarray(approx14_results[4:5][0])[0]
median_approx14 = np.asarray(approx14_results[9:10][0])[0]
upper_approx14 = np.asarray(approx14_results[14:15][0])[0]

lower_exact14 = np.asarray(exact14_results[4:5][0])[0]
median_exact14 = np.asarray(exact14_results[9:10][0])[0]
upper_exact14 = np.asarray(exact14_results[14:15][0])[0]
```

```
# Iteration5 :

slice5 = 4
```

```

approx5 = [simple_regret_approx_1[slice5],
       simple_regret_approx_2[slice5],
       simple_regret_approx_3[slice5],
       simple_regret_approx_4[slice5],
       simple_regret_approx_5[slice5],
       simple_regret_approx_6[slice5],
       simple_regret_approx_7[slice5],
       simple_regret_approx_8[slice5],
       simple_regret_approx_9[slice5],
       simple_regret_approx_10[slice5],
       simple_regret_approx_11[slice5],
       simple_regret_approx_12[slice5],
       simple_regret_approx_13[slice5],
       simple_regret_approx_14[slice5],
       simple_regret_approx_15[slice5],
       simple_regret_approx_16[slice5],
       simple_regret_approx_17[slice5],
       simple_regret_approx_18[slice5],
       simple_regret_approx_19[slice5],
       simple_regret_approx_20[slice5]]

exact5 = [simple_regret_exact_1[slice5],
       simple_regret_exact_2[slice5],
       simple_regret_exact_3[slice5],
       simple_regret_exact_4[slice5],
       simple_regret_exact_5[slice5],
       simple_regret_exact_6[slice5],
       simple_regret_exact_7[slice5],
       simple_regret_exact_8[slice5],
       simple_regret_exact_9[slice5],
       simple_regret_exact_10[slice5],
       simple_regret_exact_11[slice5],
       simple_regret_exact_12[slice5],
       simple_regret_exact_13[slice5],
       simple_regret_exact_14[slice5],
       simple_regret_exact_15[slice5],
       simple_regret_exact_16[slice5],
       simple_regret_exact_17[slice5],
       simple_regret_exact_18[slice5],
       simple_regret_exact_19[slice5],
       simple_regret_exact_20[slice5]]

approx5_results = pd.DataFrame(approx5).sort_values(by=[0], ascending=False)
exact5_results = pd.DataFrame(exact5).sort_values(by=[0], ascending=False)

### Best simple regret minimization IQR - approx:
lower_approx5 = np.asarray(approx5_results[4:5][0])[0]
median_approx5 = np.asarray(approx5_results[9:10][0])[0]
upper_approx5 = np.asarray(approx5_results[14:15][0])[0]

lower_exact5 = np.asarray(exact5_results[4:5][0])[0]
median_exact5 = np.asarray(exact5_results[9:10][0])[0]
upper_exact5 = np.asarray(exact5_results[14:15][0])[0]
```

```
# Iteration15 :

slice15 = 14

approx15 = [simple_regret_approx_1[slice15],
       simple_regret_approx_2[slice15],
       simple_regret_approx_3[slice15],
       simple_regret_approx_4[slice15],
       simple_regret_approx_5[slice15],
       simple_regret_approx_6[slice15],
       simple_regret_approx_7[slice15],
       simple_regret_approx_8[slice15],
       simple_regret_approx_9[slice15],
       simple_regret_approx_10[slice15],
       simple_regret_approx_11[slice15],
       simple_regret_approx_12[slice15],
       simple_regret_approx_13[slice15],
       simple_regret_approx_14[slice15],
       simple_regret_approx_15[slice15],
       simple_regret_approx_16[slice15],
       simple_regret_approx_17[slice15],
       simple_regret_approx_18[slice15],
       simple_regret_approx_19[slice15],
       simple_regret_approx_20[slice15]]

exact15 = [simple_regret_exact_1[slice15],
       simple_regret_exact_2[slice15],
       simple_regret_exact_3[slice15],
       simple_regret_exact_4[slice15],
       simple_regret_exact_5[slice15],
       simple_regret_exact_6[slice15],
       simple_regret_exact_7[slice15],
       simple_regret_exact_8[slice15],
       simple_regret_exact_9[slice15],
       simple_regret_exact_10[slice15],
       simple_regret_exact_11[slice15],
       simple_regret_exact_12[slice15],
       simple_regret_exact_13[slice15],
       simple_regret_exact_14[slice15],
       simple_regret_exact_15[slice15],
       simple_regret_exact_16[slice15],
       simple_regret_exact_17[slice15],
       simple_regret_exact_18[slice15],
       simple_regret_exact_19[slice15],
       simple_regret_exact_20[slice15]]

approx15_results = pd.DataFrame(approx15).sort_values(by=[0], ascending=False)
exact15_results = pd.DataFrame(exact15).sort_values(by=[0], ascending=False)

### Best simple regret minimization IQR - approx:
lower_approx15 = np.asarray(approx15_results[4:5][0])[0]
median_approx15 = np.asarray(approx15_results[9:10][0])[0]
upper_approx15 = np.asarray(approx15_results[14:15][0])[0]

```

```
lower_exact15 = np.asarray(exact15_results[4:5][0])[0]
median_exact15 = np.asarray(exact15_results[9:10][0])[0]
upper_exact15 = np.asarray(exact15_results[14:15][0])[0]
```

```
# Iteration6 :

slice6 = 5

approx6 = [simple_regret_approx_1[slice6],
       simple_regret_approx_2[slice6],
       simple_regret_approx_3[slice6],
       simple_regret_approx_4[slice6],
       simple_regret_approx_5[slice6],
       simple_regret_approx_6[slice6],
       simple_regret_approx_7[slice6],
       simple_regret_approx_8[slice6],
       simple_regret_approx_9[slice6],
       simple_regret_approx_10[slice6],
       simple_regret_approx_11[slice6],
       simple_regret_approx_12[slice6],
       simple_regret_approx_13[slice6],
       simple_regret_approx_14[slice6],
       simple_regret_approx_15[slice6],
       simple_regret_approx_16[slice6],
       simple_regret_approx_17[slice6],
       simple_regret_approx_18[slice6],
       simple_regret_approx_19[slice6],
       simple_regret_approx_20[slice6]]

exact6 = [simple_regret_exact_1[slice6],
       simple_regret_exact_2[slice6],
       simple_regret_exact_3[slice6],
       simple_regret_exact_4[slice6],
       simple_regret_exact_5[slice6],
       simple_regret_exact_6[slice6],
       simple_regret_exact_7[slice6],
       simple_regret_exact_8[slice6],
       simple_regret_exact_9[slice6],
       simple_regret_exact_10[slice6],
       simple_regret_exact_11[slice6],
       simple_regret_exact_12[slice6],
       simple_regret_exact_13[slice6],
       simple_regret_exact_14[slice6],
       simple_regret_exact_15[slice6],
       simple_regret_exact_16[slice6],
       simple_regret_exact_17[slice6],
       simple_regret_exact_18[slice6],
       simple_regret_exact_19[slice6],
       simple_regret_exact_20[slice6]]

approx6_results = pd.DataFrame(approx6).sort_values(by=[0], ascending=False)
exact6_results = pd.DataFrame(exact6).sort_values(by=[0], ascending=False)

### Best simple regret minimization IQR - approx:
lower_approx6 = np.asarray(approx6_results[4:5][0])[0]
```

```
lower_approx6 = np.asarray(approx6_results[4:5][0])[0]
median_approx6 = np.asarray(approx6_results[9:10][0])[0]
upper_approx6 = np.asarray(approx6_results[14:15][0])[0]

lower_exact6 = np.asarray(exact6_results[4:5][0])[0]
median_exact6 = np.asarray(exact6_results[9:10][0])[0]
upper_exact6 = np.asarray(exact6_results[14:15][0])[0]
```

```
# Iteration16 :

slice16 = 15

approx16 = [simple_regret_approx_1[slice16],
       simple_regret_approx_2[slice16],
       simple_regret_approx_3[slice16],
       simple_regret_approx_4[slice16],
       simple_regret_approx_5[slice16],
       simple_regret_approx_6[slice16],
       simple_regret_approx_7[slice16],
       simple_regret_approx_8[slice16],
       simple_regret_approx_9[slice16],
       simple_regret_approx_10[slice16],
       simple_regret_approx_11[slice16],
       simple_regret_approx_12[slice16],
       simple_regret_approx_13[slice16],
       simple_regret_approx_14[slice16],
       simple_regret_approx_15[slice16],
       simple_regret_approx_16[slice16],
       simple_regret_approx_17[slice16],
       simple_regret_approx_18[slice16],
       simple_regret_approx_19[slice16],
       simple_regret_approx_20[slice16]]

exact16 = [simple_regret_exact_1[slice16],
       simple_regret_exact_2[slice16],
       simple_regret_exact_3[slice16],
       simple_regret_exact_4[slice16],
       simple_regret_exact_5[slice16],
       simple_regret_exact_6[slice16],
       simple_regret_exact_7[slice16],
       simple_regret_exact_8[slice16],
       simple_regret_exact_9[slice16],
       simple_regret_exact_10[slice16],
       simple_regret_exact_11[slice16],
       simple_regret_exact_12[slice16],
       simple_regret_exact_13[slice16],
       simple_regret_exact_14[slice16],
       simple_regret_exact_15[slice16],
       simple_regret_exact_16[slice16],
       simple_regret_exact_17[slice16],
       simple_regret_exact_18[slice16],
       simple_regret_exact_19[slice16],
       simple_regret_exact_20[slice16]]

approx16_results = pd.DataFrame(approx16).sort_values(by=[0], ascending=False)
```

```
exact16_results = pd.DataFrame(exact16).sort_values(by=[0], ascending=False)

### Best simple regret minimization IQR - approx:
lower_approx16 = np.asarray(approx16_results[4:5][0])[0]
median_approx16 = np.asarray(approx16_results[9:10][0])[0]
upper_approx16 = np.asarray(approx16_results[14:15][0])[0]

lower_exact16 = np.asarray(exact16_results[4:5][0])[0]
median_exact16 = np.asarray(exact16_results[9:10][0])[0]
upper_exact16 = np.asarray(exact16_results[14:15][0])[0]
```

```
# Iteration7 :

slice7 = 6

approx7 = [simple_regret_approx_1[slice7],
       simple_regret_approx_2[slice7],
       simple_regret_approx_3[slice7],
       simple_regret_approx_4[slice7],
       simple_regret_approx_5[slice7],
       simple_regret_approx_6[slice7],
       simple_regret_approx_7[slice7],
       simple_regret_approx_8[slice7],
       simple_regret_approx_9[slice7],
       simple_regret_approx_10[slice7],
       simple_regret_approx_11[slice7],
       simple_regret_approx_12[slice7],
       simple_regret_approx_13[slice7],
       simple_regret_approx_14[slice7],
       simple_regret_approx_15[slice7],
       simple_regret_approx_16[slice7],
       simple_regret_approx_17[slice7],
       simple_regret_approx_18[slice7],
       simple_regret_approx_19[slice7],
       simple_regret_approx_20[slice7]]

exact7 = [simple_regret_exact_1[slice7],
       simple_regret_exact_2[slice7],
       simple_regret_exact_3[slice7],
       simple_regret_exact_4[slice7],
       simple_regret_exact_5[slice7],
       simple_regret_exact_6[slice7],
       simple_regret_exact_7[slice7],
       simple_regret_exact_8[slice7],
       simple_regret_exact_9[slice7],
       simple_regret_exact_10[slice7],
       simple_regret_exact_11[slice7],
       simple_regret_exact_12[slice7],
       simple_regret_exact_13[slice7],
       simple_regret_exact_14[slice7],
       simple_regret_exact_15[slice7],
       simple_regret_exact_16[slice7],
       simple_regret_exact_17[slice7],
       simple_regret_exact_18[slice7],
       simple_regret_exact_19[slice7],
       simple_regret_exact_19[slice7],
       simple_regret_exact_20[slice7]]

approx7_results = pd.DataFrame(approx7).sort_values(by=[0], ascending=False)
exact7_results = pd.DataFrame(exact7).sort_values(by=[0], ascending=False)

### Best simple regret minimization IQR - approx:
lower_approx7 = np.asarray(approx7_results[4:5][0])[0]
median_approx7 = np.asarray(approx7_results[9:10][0])[0]
upper_approx7 = np.asarray(approx7_results[14:15][0])[0]

lower_exact7 = np.asarray(exact7_results[4:5][0])[0]
median_exact7 = np.asarray(exact7_results[9:10][0])[0]
upper_exact7 = np.asarray(exact7_results[14:15][0])[0]
```

```
# Iteration17 :

slice17 = 16

approx17 = [simple_regret_approx_1[slice17],
       simple_regret_approx_2[slice17],
       simple_regret_approx_3[slice17],
       simple_regret_approx_4[slice17],
       simple_regret_approx_5[slice17],
       simple_regret_approx_6[slice17],
       simple_regret_approx_7[slice17],
       simple_regret_approx_8[slice17],
       simple_regret_approx_9[slice17],
       simple_regret_approx_10[slice17],
       simple_regret_approx_11[slice17],
       simple_regret_approx_12[slice17],
       simple_regret_approx_13[slice17],
       simple_regret_approx_14[slice17],
       simple_regret_approx_15[slice17],
       simple_regret_approx_16[slice17],
       simple_regret_approx_17[slice17],
       simple_regret_approx_18[slice17],
       simple_regret_approx_19[slice17],
       simple_regret_approx_20[slice17]]

exact17 = [simple_regret_exact_1[slice17],
       simple_regret_exact_2[slice17],
       simple_regret_exact_3[slice17],
       simple_regret_exact_4[slice17],
       simple_regret_exact_5[slice17],
       simple_regret_exact_6[slice17],
       simple_regret_exact_7[slice17],
       simple_regret_exact_8[slice17],
       simple_regret_exact_9[slice17],
       simple_regret_exact_10[slice17],
       simple_regret_exact_11[slice17],
       simple_regret_exact_12[slice17],
       simple_regret_exact_13[slice17],
       simple_regret_exact_14[slice17],
       simple_regret_exact_15[slice17],
       simple_regret_exact_16[slice17],
       simple_regret_exact_17[slice17],
       simple_regret_exact_18[slice17],
       simple_regret_exact_19[slice17],
       simple_regret_exact_20[slice17]]

approx17_results = pd.DataFrame(approx17).sort_values(by=[0], ascending=False)
exact17_results = pd.DataFrame(exact17).sort_values(by=[0], ascending=False)

### Best simple regret minimization IQR - approx:
lower_approx17 = np.asarray(approx17_results[4:5][0])[0]
median_approx17 = np.asarray(approx17_results[9:10][0])[0]
upper_approx17 = np.asarray(approx17_results[14:15][0])[0]

lower_exact17 = np.asarray(exact17_results[4:5][0])[0]
median_exact17 = np.asarray(exact17_results[9:10][0])[0]
upper_exact17 = np.asarray(exact17_results[14:15][0])[0]
```

```
# Iteration8 :

slice8 = 7

approx8 = [simple_regret_approx_1[slice8],
       simple_regret_approx_2[slice8],
       simple_regret_approx_3[slice8],
       simple_regret_approx_4[slice8],
       simple_regret_approx_5[slice8],
       simple_regret_approx_6[slice8],
       simple_regret_approx_7[slice8],
       simple_regret_approx_8[slice8],
       simple_regret_approx_9[slice8],
       simple_regret_approx_10[slice8],
       simple_regret_approx_11[slice8],
       simple_regret_approx_12[slice8],
       simple_regret_approx_13[slice8],
       simple_regret_approx_14[slice8],
       simple_regret_approx_15[slice8],
       simple_regret_approx_16[slice8],
       simple_regret_approx_17[slice8],
       simple_regret_approx_18[slice8],
       simple_regret_approx_19[slice8],
       simple_regret_approx_20[slice8]]

exact8 = [simple_regret_exact_1[slice8],
       simple_regret_exact_2[slice8],
       simple_regret_exact_3[slice8],
       simple_regret_exact_4[slice8],
       simple_regret_exact_5[slice8],
       simple_regret_exact_6[slice8],
       simple_regret_exact_7[slice8],
       simple_regret_exact_8[slice8],
       simple_regret_exact_9[slice8],
       simple_regret_exact_10[slice8],
       simple_regret_exact_11[slice8],
       simple_regret_exact_12[slice8],
       simple_regret_exact_13[slice8],
       simple_regret_exact_14[slice8],
       simple_regret_exact_15[slice8],
       simple_regret_exact_16[slice8],
       simple_regret_exact_17[slice8],
       simple_regret_exact_18[slice8],
       simple_regret_exact_19[slice8],
       simple_regret_exact_20[slice8]]

approx8_results = pd.DataFrame(approx8).sort_values(by=[0], ascending=False)
exact8_results = pd.DataFrame(exact8).sort_values(by=[0], ascending=False)

### Best simple regret minimization IQR - approx:
lower_approx8 = np.asarray(approx8_results[4:5][0])[0]
median_approx8 = np.asarray(approx8_results[9:10][0])[0]
upper_approx8 = np.asarray(approx8_results[14:15][0])[0]

lower_exact8 = np.asarray(exact8_results[4:5][0])[0]
median_exact8 = np.asarray(exact8_results[9:10][0])[0]
upper_exact8 = np.asarray(exact8_results[14:15][0])[0]
```

```
# Iteration18 :

slice18 = 17

approx18 = [simple_regret_approx_1[slice18],
       simple_regret_approx_2[slice18],
       simple_regret_approx_3[slice18],
       simple_regret_approx_4[slice18],
       simple_regret_approx_5[slice18],
       simple_regret_approx_6[slice18],
       simple_regret_approx_7[slice18],
       simple_regret_approx_8[slice18],
       simple_regret_approx_9[slice18],
       simple_regret_approx_10[slice18],
       simple_regret_approx_11[slice18],
       simple_regret_approx_12[slice18],
       simple_regret_approx_13[slice18],
       simple_regret_approx_14[slice18],
       simple_regret_approx_15[slice18],
       simple_regret_approx_16[slice18],
       simple_regret_approx_17[slice18],
       simple_regret_approx_18[slice18],
       simple_regret_approx_19[slice18],
       simple_regret_approx_20[slice18]]

exact18 = [simple_regret_exact_1[slice18],
       simple_regret_exact_2[slice18],
       simple_regret_exact_3[slice18],
       simple_regret_exact_4[slice18],
       simple_regret_exact_5[slice18],
       simple_regret_exact_6[slice18],
       simple_regret_exact_7[slice18],
       simple_regret_exact_8[slice18],
       simple_regret_exact_8[slice18],
       simple_regret_exact_9[slice18],
       simple_regret_exact_10[slice18],
       simple_regret_exact_11[slice18],
       simple_regret_exact_12[slice18],
       simple_regret_exact_13[slice18],
       simple_regret_exact_14[slice18],
       simple_regret_exact_15[slice18],
       simple_regret_exact_16[slice18],
       simple_regret_exact_17[slice18],
       simple_regret_exact_18[slice18],
       simple_regret_exact_19[slice18],
       simple_regret_exact_20[slice18]]

approx18_results = pd.DataFrame(approx18).sort_values(by=[0], ascending=False)
exact18_results = pd.DataFrame(exact18).sort_values(by=[0], ascending=False)

### Best simple regret minimization IQR - approx:
lower_approx18 = np.asarray(approx18_results[4:5][0])[0]
median_approx18 = np.asarray(approx18_results[9:10][0])[0]
upper_approx18 = np.asarray(approx18_results[14:15][0])[0]

lower_exact18 = np.asarray(exact18_results[4:5][0])[0]
median_exact18 = np.asarray(exact18_results[9:10][0])[0]
upper_exact18 = np.asarray(exact18_results[14:15][0])[0]
```

```
# Iteration9 :

slice9 = 8

approx9 = [simple_regret_approx_1[slice9],
       simple_regret_approx_2[slice9],
       simple_regret_approx_3[slice9],
       simple_regret_approx_4[slice9],
       simple_regret_approx_5[slice9],
       simple_regret_approx_6[slice9],
       simple_regret_approx_7[slice9],
       simple_regret_approx_8[slice9],
       simple_regret_approx_9[slice9],
       simple_regret_approx_10[slice9],
       simple_regret_approx_11[slice9],
       simple_regret_approx_12[slice9],
       simple_regret_approx_13[slice9],
       simple_regret_approx_14[slice9],
       simple_regret_approx_15[slice9],
       simple_regret_approx_16[slice9],
       simple_regret_approx_17[slice9],
       simple_regret_approx_18[slice9],
       simple_regret_approx_19[slice9],
       simple_regret_approx_20[slice9]]

exact9 = [simple_regret_exact_1[slice9],
       simple_regret_exact_2[slice9],
       simple_regret_exact_3[slice9],
       simple_regret_exact_4[slice9],
       simple_regret_exact_5[slice9],
       simple_regret_exact_6[slice9],
       simple_regret_exact_7[slice9],
       simple_regret_exact_8[slice9],
       simple_regret_exact_9[slice9],
       simple_regret_exact_10[slice9],
       simple_regret_exact_11[slice9],
       simple_regret_exact_12[slice9],
       simple_regret_exact_13[slice9],
       simple_regret_exact_14[slice9],
       simple_regret_exact_15[slice9],
       simple_regret_exact_16[slice9],
       simple_regret_exact_17[slice9],
       simple_regret_exact_18[slice9],
       simple_regret_exact_19[slice9],
       simple_regret_exact_20[slice9]]

approx9_results = pd.DataFrame(approx9).sort_values(by=[0], ascending=False)
exact9_results = pd.DataFrame(exact9).sort_values(by=[0], ascending=False)

### Best simple regret minimization IQR - approx:
lower_approx9 = np.asarray(approx9_results[4:5][0])[0]
median_approx9 = np.asarray(approx9_results[9:10][0])[0]
upper_approx9 = np.asarray(approx9_results[14:15][0])[0]

lower_exact9 = np.asarray(exact9_results[4:5][0])[0]
median_exact9 = np.asarray(exact9_results[9:10][0])[0]
upper_exact9 = np.asarray(exact9_results[14:15][0])[0]
```

```
# Iteration19 :

slice19 = 18

approx19 = [simple_regret_approx_1[slice19],
       simple_regret_approx_2[slice19],
       simple_regret_approx_3[slice19],
       simple_regret_approx_4[slice19],
       simple_regret_approx_5[slice19],
       simple_regret_approx_6[slice19],
       simple_regret_approx_7[slice19],
       simple_regret_approx_8[slice19],
       simple_regret_approx_9[slice19],
       simple_regret_approx_10[slice19],
       simple_regret_approx_11[slice19],
       simple_regret_approx_12[slice19],
       simple_regret_approx_13[slice19],
       simple_regret_approx_14[slice19],
       simple_regret_approx_15[slice19],
       simple_regret_approx_16[slice19],
       simple_regret_approx_17[slice19],
       simple_regret_approx_18[slice19],
       simple_regret_approx_19[slice19],
       simple_regret_approx_20[slice19]]

```

```
exact19 = [simple_regret_exact_1[slice19],
       simple_regret_exact_2[slice19],
       simple_regret_exact_3[slice19],
       simple_regret_exact_4[slice19],
       simple_regret_exact_5[slice19],
       simple_regret_exact_6[slice19],
       simple_regret_exact_7[slice19],
       simple_regret_exact_8[slice19],
       simple_regret_exact_9[slice19],
       simple_regret_exact_10[slice19],
       simple_regret_exact_11[slice19],
       simple_regret_exact_12[slice19],
       simple_regret_exact_13[slice19],
       simple_regret_exact_14[slice19],
       simple_regret_exact_15[slice19],
       simple_regret_exact_16[slice19],
       simple_regret_exact_17[slice19],
       simple_regret_exact_18[slice19],
       simple_regret_exact_19[slice19],
       simple_regret_exact_20[slice19]]

approx19_results = pd.DataFrame(approx19).sort_values(by=[0], ascending=False)
exact19_results = pd.DataFrame(exact19).sort_values(by=[0], ascending=False)

### Best simple regret minimization IQR - approx:
lower_approx19 = np.asarray(approx19_results[4:5][0])[0]
median_approx19 = np.asarray(approx19_results[9:10][0])[0]
upper_approx19 = np.asarray(approx19_results[14:15][0])[0]

lower_exact19 = np.asarray(exact19_results[4:5][0])[0]
median_exact19 = np.asarray(exact19_results[9:10][0])[0]
upper_exact19 = np.asarray(exact19_results[14:15][0])[0]
```

```
# Iteration10 :

slice10 = 9

approx10 = [simple_regret_approx_1[slice10],
       simple_regret_approx_2[slice10],
       simple_regret_approx_3[slice10],
       simple_regret_approx_4[slice10],
       simple_regret_approx_5[slice10],
       simple_regret_approx_6[slice10],
       simple_regret_approx_7[slice10],
       simple_regret_approx_8[slice10],
       simple_regret_approx_9[slice10],
       simple_regret_approx_10[slice10],
       simple_regret_approx_11[slice10],
       simple_regret_approx_12[slice10],
       simple_regret_approx_13[slice10],
       simple_regret_approx_14[slice10],
       simple_regret_approx_15[slice10],
       simple_regret_approx_16[slice10],
       simple_regret_approx_17[slice10],
       simple_regret_approx_18[slice10]
       simple_regret_approx_18[slice10],
       simple_regret_approx_19[slice10],
       simple_regret_approx_20[slice10]]

exact10 = [simple_regret_exact_1[slice10],
       simple_regret_exact_2[slice10],
       simple_regret_exact_3[slice10],
       simple_regret_exact_4[slice10],
       simple_regret_exact_5[slice10],
       simple_regret_exact_6[slice10],
       simple_regret_exact_7[slice10],
       simple_regret_exact_8[slice10],
       simple_regret_exact_9[slice10],
       simple_regret_exact_10[slice10],
       simple_regret_exact_11[slice10],
       simple_regret_exact_12[slice10],
       simple_regret_exact_13[slice10],
       simple_regret_exact_14[slice10],
       simple_regret_exact_15[slice10],
       simple_regret_exact_16[slice10],
       simple_regret_exact_17[slice10],
       simple_regret_exact_18[slice10],
       simple_regret_exact_19[slice10],
       simple_regret_exact_20[slice10]]

approx10_results = pd.DataFrame(approx10).sort_values(by=[0], ascending=False)
exact10_results = pd.DataFrame(exact10).sort_values(by=[0], ascending=False)

### Best simple regret minimization IQR - approx:
lower_approx10 = np.asarray(approx10_results[4:5][0])[0]
median_approx10 = np.asarray(approx10_results[9:10][0])[0]
upper_approx10 = np.asarray(approx10_results[14:15][0])[0]

lower_exact10 = np.asarray(exact10_results[4:5][0])[0]
median_exact10 = np.asarray(exact10_results[9:10][0])[0]
upper_exact10 = np.asarray(exact10_results[14:15][0])[0]
```

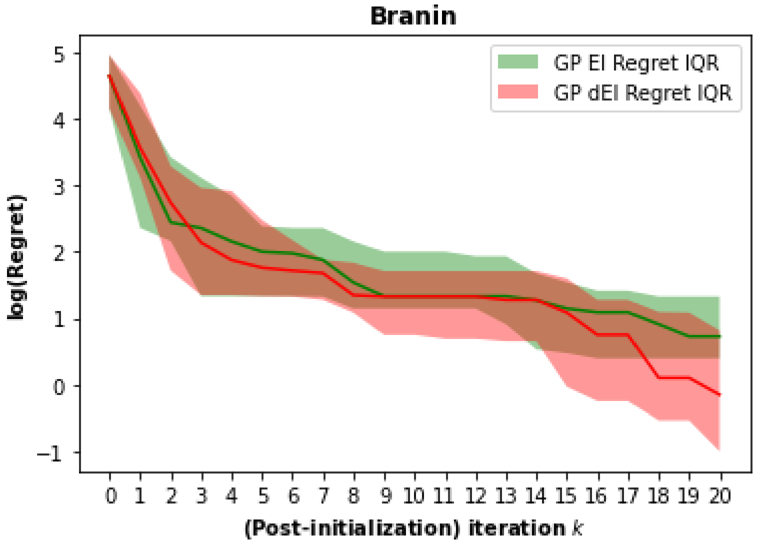
```
# Iteration20 :

slice20 = 19

approx20 = [simple_regret_approx_1[slice20],
       simple_regret_approx_2[slice20],
       simple_regret_approx_3[slice20],
       simple_regret_approx_4[slice20],
       simple_regret_approx_5[slice20],
       simple_regret_approx_6[slice20],
       simple_regret_approx_7[slice20],
       simple_regret_approx_8[slice20],
       simple_regret_approx_9[slice20],
       simple_regret_approx_10[slice20],
       simple_regret_approx_11[slice20],
       simple_regret_approx_12[slice20],
       simple_regret_approx_13[slice20],
       simple_regret_approx_14[slice20],
       simple_regret_approx_15[slice20],
       simple_regret_approx_16[slice20],
       simple_regret_approx_17[slice20],
       simple_regret_approx_18[slice20],
       simple_regret_approx_19[slice20],
       simple_regret_approx_20[slice20]]

exact20 = [simple_regret_exact_1[slice20],
       simple_regret_exact_2[slice20],
       simple_regret_exact_3[slice20],
       simple_regret_exact_4[slice20],
       simple_regret_exact_5[slice20],
       simple_regret_exact_6[slice20],
       simple_regret_exact_7[slice20],
       simple_regret_exact_8[slice20],
       simple_regret_exact_9[slice20],
       simple_regret_exact_10[slice20],
       simple_regret_exact_11[slice20],
       simple_regret_exact_12[slice20],
       simple_regret_exact_13[slice20],
       simple_regret_exact_14[slice20],
       simple_regret_exact_15[slice20],
       simple_regret_exact_16[slice20],
       simple_regret_exact_17[slice20],
       simple_regret_exact_18[slice20],
       simple_regret_exact_19[slice20],
       simple_regret_exact_20[slice20]]

approx20_results = pd.DataFrame(approx20).sort_values(by=[0], ascending=False)
exact20_results = pd.DataFrame(exact20).sort_values(by=[0], ascending=False)

### Best simple regret minimization IQR - approx:
lower_approx20 = np.asarray(approx20_results[4:5][0])[0]
median_approx20 = np.asarray(approx20_results[9:10][0])[0]
upper_approx20 = np.asarray(approx20_results[14:15][0])[0]

lower_exact20 = np.asarray(exact20_results[4:5][0])[0]
median_exact20 = np.asarray(exact20_results[9:10][0])[0]
upper_exact20 = np.asarray(exact20_results[14:15][0])[0]
```

```
### Summarize arrays: 'Loser'

lower_approx = [lower_approx1,
            lower_approx2,
            lower_approx3,
            lower_approx4,
            lower_approx5,
            lower_approx6,
            lower_approx7,
            lower_approx8,
            lower_approx9,
            lower_approx10,
            lower_approx11,
            lower_approx12,
            lower_approx13,
            lower_approx14,
            lower_approx15,
            lower_approx16,
            lower_approx17,
            lower_approx18,
            lower_approx19,
            lower_approx20,
            lower_approx21]

median_approx = [median_approx1,
            median_approx2,
            median_approx3,
            median_approx4,
            median_approx5,
            median_approx6,
            median_approx7,
            median_approx8,
            median_approx9,
            median_approx10,
            median_approx11,
            median_approx12,
            median_approx13,
            median_approx14,
            median_approx15,
            median_approx16,
            median_approx17,
            median_approx18,
            median_approx19,
            median_approx20,
            median_approx21]

upper_approx = [upper_approx1,
            upper_approx2,
            upper_approx3,
            upper_approx4,
            upper_approx5,
            upper_approx6,
            upper_approx7,
            upper_approx8,
            upper_approx9,
            upper_approx10,
            upper_approx11,
            upper_approx12,
            upper_approx13,
            upper_approx14,
            upper_approx15,
            upper_approx16,
            upper_approx17,
            upper_approx18,
            upper_approx19,
            upper_approx20,
            upper_approx21]
```

```
### Summarize arrays: 'exact'
### Summarize arrays: - exact

lower_exact = [lower_exact1,
            lower_exact2,
            lower_exact3,
            lower_exact4,
            lower_exact5,
            lower_exact6,
            lower_exact7,
            lower_exact8,
            lower_exact9,
            lower_exact10,
            lower_exact11,
            lower_exact12,
            lower_exact13,
            lower_exact14,
            lower_exact15,
            lower_exact16,
            lower_exact17,
            lower_exact18,
            lower_exact19,
            lower_exact20,
            lower_exact21]

median_exact = [median_exact1,
            median_exact2,
            median_exact3,
            median_exact4,
            median_exact5,
            median_exact6,
            median_exact7,
            median_exact8,
            median_exact9,
            median_exact10,
            median_exact11,
            median_exact12,
            median_exact13,
            median_exact14,
            median_exact15,
            median_exact16,
            median_exact17,
            median_exact18,
            median_exact19,
            median_exact20,
            median_exact21]

upper_exact = [upper_exact1,
            upper_exact2,
            upper_exact3,
            upper_exact4,
            upper_exact5,
            upper_exact6,
            upper_exact7,
            upper_exact8,
            upper_exact9,
            upper_exact10,
            upper_exact11,
            upper_exact12,
            upper_exact13,
            upper_exact14,
            upper_exact15,
            upper_exact16,
            upper_exact17,
            upper_exact18,
            upper_exact19,
            upper_exact20,
            upper_exact21]
```

```
### Visualize!

title = 'Branin'

plt.figure()

plt.plot(median_approx, color = 'Green')
plt.plot(median_exact, color = 'Red')

xstar = np.arange(0, iters+1, step=1)
plt.fill_between(xstar, lower_approx, upper_approx, facecolor = 'Green', alpha=0.4, lab
plt.fill_between(xstar, lower_exact, upper_exact, facecolor = 'Red', alpha=0.4, label='

plt.title(title, weight = 'bold', family = 'Arial')
plt.xlabel('(Post-initialization) iteration $\it{k}$', weight = 'bold', family = 'Arial
plt.ylabel('log(Regret)', weight = 'bold', family = 'Arial')
plt.legend(loc=1) # add plot legend

### Make the x-ticks integers, not floats:
count = len(xstar)
plt.xticks(np.arange(count), np.arange(0, count))
plt.show() #visualize!
```

```
findfont: Font family ['Arial'] not found. Falling back to DejaVu Sans.
findfont: Font family ['Arial'] not found. Falling back to DejaVu Sans.
```



```
time_approx, time_exact
```

```
time_approx, time_exact
```

```
(805.8598399162292, 775.5369441509247)
```