Rastrigin:

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 - Rastrigin(x) 2-D:

def objfunc(x1_training, x2_training):
        return  operator * (10 * dim + x1_training** 2 - 10 * np.cos(2 * np.pi * x1_tra
```

```
                                    + x2_training** 2 - 10 * np.cos(2 * np.pi * x2_trai
                          )

# Constraints:
lb = -5.12
ub = +5.12

# Input array dimension(s):
dim = 2

# 2-D inputs' parameter bounds:
param = {'x1_training': ('cont', [lb, ub]),
             'x2_training': ('cont', [lb, ub])}

# True y bounds:
operator = -1
y_global_orig = 0 # targets global minimum

# Test data:
x1_test = np.linspace(lb, ub, n_test)
x2_test = np.linspace(lb, ub, 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,
                                                                  jac = True,
                                                                  bounds=self.parameter_
                                               start_points_arr)
```

```
                                               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

```

```
1623409277.7587016
```

```
### 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     [-0.84969467  2.25612281].       -20.33436270766351     -19.908403246996286
init     [-5.1188288  -2.02411446].       -33.07414982069084     -19.908403246996286
init     [-3.61721968 -4.17445279].       -53.347974723929894    -19.908403246996286
init     [-3.21269544 -1.58145816].       -39.218472310354045    -19.908403246996286
init     [-1.05710106  0.39748336].       -19.908403246996286    -19.908403246996286
1        [ 5.12 -5.12].           -57.849427451571785    -19.908403246996286
2        [5.12 5.12].     -57.849427451571785    -19.908403246996286
3        [-5.12  5.12].           -57.849427451571785    -19.908403246996286
4        [ 4.14436454 -0.00562717].       -21.021468411280992    -19.908403246996286
5        [ 0.90453736 -3.67597578].       -30.56174647213838     -19.908403246996286
6        [1.14408866 5.12      ].       -34.05943735904623     -19.908403246996286
7        [-5.12        1.53252144].       -51.06529057237367     -19.908403246996286
8        [2.38165269 2.25031261].       -38.116194699068366    -19.908403246996286
9        [-1.92500688  5.12      ].       -33.72010378814381     -19.908403246996286
10       [ 1.44342922 -0.8583475 ].       -25.901270823337516    -19.908403246996286
11       [ 5.12       -2.21418283].       -41.59580835606468     -19.908403246996286
12       [5.12       2.19156504].       -40.13803107224641     -19.908403246996286
13       [-1.04126178 -5.12      ].       -30.343128540457087    -19.908403246996286
14       [ 2.495817 -5.12     ].       -55.15036255666753     -19.908403246996286
15       [-0.83442789 -2.1119796 ].       -12.472032567698271    -12.472032567698271
16       [-2.94556588  2.81479221].       -23.21912772665249     -12.472032567698271
17       [ 2.86110468 -2.41116811].       -36.054463425160016    -12.472032567698271
18       [-5.12 -5.12].           -57.849427451571785    -12.472032567698271
19       [-2.86751477  0.63246584].       -28.623555274481788    -12.472032567698271
20       [0.65935445 0.95530994].       -17.131484189258828    -12.472032567698271
```

```
### 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     [-0.6554122  -4.85451539].       -43.490296251903594    -18.964539418712707
init     [ 0.50854377 -0.6622987 ].       -35.91861667536101     -18.964539418712707
init     [-0.81543371 -1.73737143].       -20.479562046739524    -18.964539418712707
init     [-3.02439799  1.2213347 ].       -18.964539418712707    -18.964539418712707
init     [-2.05153614 -2.3876887 ].       -28.041315668371354    -18.964539418712707
1        [5.12 5.12].     -57.849427451571785    -18.964539418712707
2        [ 5.12 -5.12].           -57.849427451571785    -18.964539418712707
3        [-0.09385176  5.12      ].       -30.62237948828588     -18.964539418712707
4        [5.12       0.07742125].       -30.090737340815892    -18.964539418712707
5        [-5.12  5.12].           -57.849427451571785    -18.964539418712707
6        [-5.12 -5.12].           -57.849427451571785    -18.964539418712707
7        [2.34189388 2.48319939].       -47.05343956837548     -18.964539418712707
8        [-5.12       -1.23373171].       -39.42641991167889     -18.964539418712707
9        [ 2.61411342 -2.99759227].       -33.35812552791467     -18.964539418712707
10       [-0.74543942  2.31208203].       -29.990471773955868    -18.964539418712707
11       [-5.12        2.08178927].       -34.55020643103296     -18.964539418712707
12       [-2.52725945  4.36068499].       -61.66363741628898     -18.964539418712707
13       [ 5.12       -2.32585987].       -48.92231468233234     -18.964539418712707
14       [2.36218141 5.12      ].       -50.983860472041414    -18.964539418712707
15       [5.12       2.42919667].       -53.85237451500107     -18.964539418712707
16       [ 2.01461056 -5.12      ].       -33.025476858990146    -18.964539418712707
17       [ 2.83816644 -0.24273295].       -22.39702171163043     -18.964539418712707
18       [-2.8042642 -5.12     ].       -43.444766639140596    -18.964539418712707
19       [-2.25650069 -0.20127923].       -22.527017005249654    -18.964539418712707
20       [-3.99951994 -2.95154699].       -25.167682923559983    -18.964539418712707
```

```
### 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     [0.52017052 2.1314337 ].       -27.953858411008774    -10.607662635789808
init     [-2.14113547  0.11087468].       -10.607662635789808    -10.607662635789808
init     [4.02377681 4.05804123].       -33.42749829480097     -10.607662635789808
init     [-3.83400642 -2.99783293].       -28.650953928965198    -10.607662635789808
init     [-4.59297584 -0.6061072 ].       -57.6631355589384      -10.607662635789808
1        [ 5.12 -5.12].           -57.849427451571785    -10.607662635789808
2        [-5.12  5.12].           -57.849427451571785    -10.607662635789808
3        [ 0.29058652 -5.12      ].       -41.531730491109414    -10.607662635789808
4        [ 5.12       -0.40159497].       -47.23465990242375     -10.607662635789808
5        [ 1.34926676 -1.51099296].       -39.92029433036741     -10.607662635789808
6        [-1.23870341  5.12      ].       -39.74990992200105     -10.607662635789808
7        [-3.18894141  2.50496675].       -42.69633104507217     -10.607662635789808
8        [-5.12 -5.12].           -57.849427451571785    -10.607662635789808
9        [1.57482567 5.12      ].       -50.319822241391805    -10.607662635789808
10       [-1.29118604 -2.45786364].       -39.918841952151645    -10.607662635789808
```

```
11       [2.99709432 1.3850341 ].         -28.405087646262942    -10.607662635789808
12       [ 2.85580978 -3.62581611].       -42.167714195843985    -10.607662635789808
13       [-2.30601269 -5.12      ].       -47.689585725068994    -10.607662635789808
14       [5.12       2.19425441].         -40.30804730641291     -10.607662635789808
15       [-0.65816312  0.23394885].       -24.93645297356877     -10.607662635789808
16       [-5.12        2.00549937].       -32.95271062624369     -10.607662635789808
17       [ 5.12      -2.6409384].         -52.227968205520575    -10.607662635789808
18       [5.12 5.12].     -57.849427451571785    -10.607662635789808
19       [-1.0927624   2.88659271].       -13.610623206429532    -10.607662635789808
20       [-3.03930389  5.12      ].       -38.465465721764886    -10.607662635789808
```

```python
### 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     [4.78238555 0.48365823].         -51.03163809010808     -14.323038259018315
init     [4.84028785 2.19971578].         -39.78645699016559     -14.323038259018315
init     [ 2.02474316 -2.90724357].       -14.323038259018315    -14.323038259018315
init     [ 4.87705042 -5.05620219].       -52.80627247106233     -14.323038259018315
init     [-2.52946061 -0.66773471].       -41.61497868486559     -14.323038259018315
1        [-5.12  5.12].           -57.849427451571785    -14.323038259018315
2        [0.36040841 5.12      ].         -45.44859926058701     -14.323038259018315
3        [-5.12 -5.12].           -57.849427451571785    -14.323038259018315
4        [-0.95955594 -5.12      ].       -30.166606192601527    -14.323038259018315
5        [0.85383826 1.14532953].         -9.856891235727582     -9.856891235727582
6        [-5.12        1.50098216].       -51.17747076414409     -9.856891235727582
7        [3.91416534 5.12      ].         -45.66479784828887     -9.856891235727582
8        [-1.9785673   2.82544573].       -17.423568731164455    -9.856891235727582
9        [-5.12       -1.96263023].       -33.05102503274147     -9.856891235727582
10       [ 0.30793382 -1.28261887].       -27.335353766193386    -9.856891235727582
11       [ 5.12      -2.2531874].         -44.201824125923935    -9.856891235727582
12       [ 1.82466825 -5.12      ].       -37.73280560268178     -9.856891235727582
13       [2.02595061 2.72645231].         -23.14480719730869     -9.856891235727582
14       [-2.35560543  5.12      ].       -50.63267437342252     -9.856891235727582
15       [-2.64736472 -3.36997513].       -51.2206014786411      -9.856891235727582
16       [ 2.44506428 -0.45937645].       -45.275554911690634    -9.856891235727582
17       [-0.59688998  1.36720899].       -37.1456044502214      -9.856891235727582
18       [ 0.07186359 -3.22172625].       -19.61972628253676     -9.856891235727582
19       [-3.38761896  2.82456281].       -42.547616871449435    -9.856891235727582
20       [ 3.0553742  -3.08867166].       -20.986680782269925    -9.856891235727582
```

```python
### 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     [-2.84678993  3.79629882].       -33.93442008827236     -7.8108627039749745
init     [-3.00319585  4.2865757 ].       -39.673876075575784    -7.8108627039749745
init     [-0.11866943  1.14425716].       -7.8108627039749745    -7.8108627039749745
init     [2.72289645 0.1886002 ].         -25.38160395721669     -7.8108627039749745
init     [-2.08076286 -3.19773462].       -22.589982116319675    -7.8108627039749745
1        [ 5.12 -5.12].           -57.849427451571785    -7.8108627039749745
2        [5.12 5.12].     -57.849427451571785    -7.8108627039749745
3        [-5.12       -0.04630553].       -29.34712917841543     -7.8108627039749745
4        [1.00367564 5.12      ].         -29.934745246375115    -7.8108627039749745
5        [-5.12 -5.12].           -57.849427451571785    -7.8108627039749745
6        [ 1.09960658 -5.12      ].       -32.02917361927805     -7.8108627039749745
7        [ 5.12       -1.64920241].       -47.562903928514835    -7.8108627039749745
8        [5.12       1.86221319].         -35.911784866528144    -7.8108627039749745
9        [ 0.60373748 -1.88128451].       -24.50895436813786     -7.8108627039749745
10       [-2.19546384  0.01761772].       -11.52163286653667     -7.8108627039749745
11       [2.18624659 2.78752583].         -26.314468334677066    -7.8108627039749745
12       [-5.12        2.61662352].       -53.20469431514221     -7.8108627039749745
13       [-5.12       -2.41197079].       -53.251297452232095    -7.8108627039749745
14       [ 2.81735734 -3.13556692].       -27.075952033869953    -7.8108627039749745
15       [-1.68829697 -5.12      ].       -45.555583369738955    -7.8108627039749745
16       [-0.55454463  2.77887495].       -35.64379646494397     -7.8108627039749745
17       [-5.12  5.12].           -57.849427451571785    -7.8108627039749745
18       [-0.05287257  0.20248114].       -7.6490134469723685    -7.6490134469723685
19       [-2.75503539  1.29648775].       -31.8343249293343      -7.6490134469723685
20       [-1.91691125 -1.19956819].       -13.329623151978797    -7.6490134469723685
```

```
### 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     [ 4.02288795 -1.72052679].       -31.08835710146886     -17.28954482757088
init     [ 3.28938622 -4.69302655].       -58.797867722203385    -17.28954482757088
init     [-4.0175956   0.97333314].       -17.28954482757088     -17.28954482757088
init     [ 0.30532979 -0.83141193].       -19.296253155889353    -17.28954482757088
init     [-1.68542362  1.25459899].       -28.65063093627617     3-17.28954482757088
1        [5.12 5.12].     -57.849427451571785    -17.28954482757088
2        [-4.83496274 -5.12      ].       -57.21317932538919     -17.28954482757088
3        [-5.12  5.12].           -57.849427451571785    -17.28954482757088
4        [0.39537975 5.12      ].         -46.997190601467736    -17.28954482757088
5        [-0.76672355 -5.12      ].       -38.46373950913987     -17.28954482757088
6        [2.84227155 1.91251786].         -17.731018693618715    -17.28954482757088
7        [-2.93733582 -2.12046024].       -16.61959010715603     -16.61959010715603
8        [5.12       1.12270975].         -33.013105721990755    -16.61959010715603
9        [-5.12       -1.49037628].       -51.127659055493346    -16.61959010715603
10       [-2.34799559  4.0390988 ].       -37.90314509322524     -16.61959010715603
11       [ 0.9685314  -2.98176844].       -10.089371496466573    -10.089371496466573
12       [0.64484531 2.21179159].         -29.06688343548177     -10.089371496466573
13       [-5.12        2.42517762].       -53.721326410135255    -10.089371496466573
14       [2.74689379 4.13821319].         -38.40505001294136     -10.089371496466573
15       [ 2.28528606 -0.11338293].       -19.86644345320579     -10.089371496466573
16       [-1.08449465 -2.5365513 ].       -28.72412302946712 3   -10.089371496466573
17       [ 5.12       -3.46067913].       -60.5973690576684      -10.089371496466573
18       [ 1.80215005 -2.45052388].       -35.55513364164354     -10.089371496466573
```

```
19       [-2.81092547 -4.00805478].       -30.243360569695188    -10.089371496466573
20       [-2.51078787 -0.57245277].       -45.590375588107314    -10.089371496466573
```

```
### 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     [-4.33860312  2.86636843].       -45.646133072936244    -22.52235437888213
init     [-0.63068947  2.28828342].       -34.83012662845338     -22.52235437888213
init     [4.8946126  0.39419771].         -44.09657005662876     -22.52235437888213
init     [ 0.01147355 -4.38219639].       -36.613209822404315    -22.52235437888213
init     [-2.37118484e+00 -1.20319155e-03].       -22.52235437888213     -22.5223543
1        [-5.12 -5.12].           -57.849427451571785    -22.52235437888213
2        [ 5.12 -5.12].           -57.849427451571785    -22.52235437888213
3        [3.59739669 5.12      ].         -60.05120463923109     -22.52235437888213
4        [ 1.41112868 -0.96393528].       -21.657165478950468    -21.657165478950468
5        [-5.12       -1.46589911].       -50.84490930413446     -21.657165478950468
6        [-1.80753104  5.12      ].       -38.655309625327945    -21.657165478950468
7        [2.33785804 2.13340309].         -28.573094433316356    -21.657165478950468
8        [ 3.58916393 -2.53684788].       -57.522188262653614    -21.657165478950468
9        [-2.48043673 -3.05376149].       -35.96769222134215     -21.657165478950468
10       [0.80995522 5.12      ].         -35.902112011377625    -21.657165478950468
11       [-5.12  5.12].           -57.8494274515718      -21.657165478950468
12       [5.12       2.99248418].         -37.89082341791584     -21.657165478950468
13       [ 2.32601071 -5.12      ].       -48.93143564198441     -21.657165478950468
14       [-0.55082996 -1.43388006].       -41.00311989601389     -21.657165478950468
15       [-2.34099077 -5.12      ].       -49.815676500557025    -21.657165478950468
16       [-5.12        0.84038443].       -34.2523131162538      -21.657165478950468
17       [0.85097494 0.69471686].         -18.683601659275595    -18.683601659275595
18       [2.76182342e+00 1.62731562e-03].       -16.88598953939842     -16.8859895
19       [-2.52420025  1.81675733].       -35.48424380045234     -16.88598953939842
20       [ 1.26275904 -2.76486912].       -29.106981160028443    -16.88598953939842
```

```
### 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     [3.82391708 4.79785639].         -50.20079446939181     -13.871821018360485
init     [3.78055209 0.31596228].         -36.5114251593508      -13.871821018360485
init     [-2.73686192 -5.00327624].       -43.34985765011677     -13.871821018360485
init     [-0.7119993  -0.99992207].       -13.871821018360485    -13.871821018360485
init     [ 0.23218863 -0.22126801].       -17.190590355445654    -13.871821018360485
1        [-5.12  5.12].           -57.849427451571785    -13.871821018360485
```

```
2        [ 5.12 -5.12].           -57.849427451571785    -13.871821018360485
3        [-5.12       -0.06242896].       -29.688108487892283    -13.871821018360485
4        [-0.66933007  4.17617456].       -38.268791016751       -13.871821018360485
5        [ 1.1839806  -4.40960853].       -45.24635181965112     -13.871821018360485
6        [-2.62602926  1.70391172].       -39.68004517161082     -13.871821018360485
7        [-5.12      -3.1295391].       -41.852175967908025    -13.871821018360485
8        [1.62768714 2.34329026].       -40.62271504061769     -13.871821018360485
9        [ 5.12     -2.111193].       -35.72471352565967     -13.871821018360485
10       [-2.59386882 -1.8380546 ].       -33.1624807774898      -13.871821018360485
11       [5.12       2.44906925].       -54.41498505416354     -13.871821018360485
12       [ 2.13260867 -1.86114711].       -14.857206142266792    -13.871821018360485
13       [-5.12        2.43375741].       -53.99414881080518     -13.871821018360485
14       [1.38469941 5.12      ].       -48.33071356264601     -13.871821018360485
15       [-2.63534835  5.12      ].       -52.46645952667072     -13.871821018360485
16       [-0.67851125 -3.04210305].       -24.40488687280375     -13.871821018360485
17       [-5.12 -5.12].           -57.849427451571785    -13.871821018360485
18       [ 3.23550706 -3.37196817].       -47.864407467592976    -13.871821018360485
19       [-0.47232632  1.43373965].       -41.273721383778096    -13.871821018360485
20       [ 1.66324715 -0.59534289].       -36.56440218162068     -13.871821018360485
```

```
### 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     [-5.01376866  0.01919582].       -25.248289026162446    -24.454800313488693
init     [-0.04328148 -3.74958562].       -24.454800313488693    -24.454800313488693
init     [-3.66478248 -2.88195916].       -39.46509426509438     -24.454800313488693
init     [-0.83447623 -2.57944404].       -31.06766397812992     -24.454800313488693
init     [-4.25922917 -1.58209393].       -49.922543556206975    -24.454800313488693
1        [3.51080105 5.12      ].       -61.22741827329939     -24.454800313488693
2        [ 4.97700533 -1.02611641].       -26.06202024938157     -24.454800313488693
3        [-2.52403041  5.12      ].       -55.18167347624602     -24.454800313488693
4        [0.61751971 1.54839588].       -39.71556524383652     -24.454800313488693
5        [ 4.2427189 -5.12      ].       -56.468051987276       -24.454800313488693
6        [5.12       2.12548252].       -36.39279201650744     -24.454800313488693
7        [-5.12        3.12506535].       -41.62258355309121     -24.454800313488693
8        [ 2.10287467 -1.47886137].       -28.53834962156197     -24.454800313488693
9        [-5.12 -5.12].           -57.849427451571785    -24.454800313488693
10       [-2.36275482  1.37428892].       -41.0173319424346      -24.454800313488693
11       [0.47303229 5.12      ].       -49.005261506506486    -24.454800313488693
12       [-2.04427685 -5.12      ].       -33.48826741638594     -24.454800313488693
13       [ 1.51205998 -5.12      ].       -51.18234352935599     -24.454800313488693
14       [3.02469045 0.93500646].       -10.965374527767734    -10.965374527767734
15       [-0.72458289 -0.51303365].       -32.34493493437702     -10.965374527767734
16       [-5.12  5.12].           -57.849427451571785    -10.965374527767734
17       [ 5.12      -2.94884462].       -38.13251660495474     -10.965374527767734
18       [2.58458053 2.76478723].       -42.017138665638434    -10.965374527767734
19       [-1.00108305  3.18957388].       -17.469654333241053    -10.965374527767734
20       [3.35971123 0.24880194].       -37.634535317638246    -10.965374527767734
```

```
### 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     [ 2.77832339 -4.90750004].      -41.674330194390116    -13.839458310244165
init     [1.36855793 2.54775176].        -44.69639719065837     -13.839458310244165
init     [-0.01528819 -2.81808235].      -13.839458310244165    -13.839458310244165
init     [-3.09183626  2.66783449].      -33.23221510904937     -13.839458310244165
init     [-3.38830503 -4.2154003 ].      -54.73014366983691     -13.839458310244165
1        [ 5.12      -0.5682251].        -48.34278335032869     -13.839458310244165
2        [5.12 5.12].     -57.849427451571785    -13.839458310244165
3        [-5.12       -0.60617675].      -47.14818167122235     -13.839458310244165
4        [-5.12  5.12].           -57.849427451571785    -13.839458310244165
5        [-0.8379435  5.12     ].        -34.378138143812066    -13.839458310244165
6        [-1.23131113 -0.10568775].      -12.480746380355079    -12.480746380355079
7        [ 2.00292574 -0.37934979].      -21.41896664552759     -12.480746380355079
8        [-0.38439185 -5.12      ].      -46.548257325985105    -12.480746380355079
9        [ 5.12       -3.59828261].      -60.02547526453966     -12.480746380355079
10       [4.57050405 2.23720805].        -54.12647113202378     -12.480746380355079
11       [2.14162684 5.12      ].        -37.21613079039564     -12.480746380355079
12       [-2.40379207 -1.6441248 ].      -42.8816993237165      -12.480746380355079
13       [-5.12        2.04202706].      -33.44121548192435     -12.480746380355079
14       [ 2.21713263 -2.45296822].      -38.44879899011462     -12.480746380355079
15       [-0.9466073  1.5704207].        -22.956734240607865    -12.480746380355079
16       [-5.12 -5.12].           -57.849427451571785    -12.480746380355079
17       [ 0.07816176 -0.6834172 ].      -15.717596556880348    -12.480746380355079
18       [-2.80733215  5.12      ].      -43.28094675730979     -12.480746380355079
19       [-2.88970169  0.57134913].      -29.995497270452663    -12.480746380355079
20       [-5.12      -2.6720571].        -50.768476582354886    -12.480746380355079
```

```
### 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     [-3.27403839 -4.92057353].      -47.655641290890955    -10.679755252484755
init     [-0.37664229  2.30332343].      -35.87889240695259     -10.679755252484755
init     [-0.81711509 -0.14922651].      -10.679755252484755    -10.679755252484755
init     [-4.98912446 -0.12931474].      -28.05462905574235     -10.679755252484755
init     [4.52410012 3.59214172].        -71.62694632141611     -10.679755252484755
1        [ 5.12       -4.19757348].      -53.309530921692996    -10.679755252484755
2        [-5.12  5.12].           -57.849427451571785    -10.679755252484755
3        [ 0.96012019 -5.12      ].      -30.15883754762058     -10.679755252484755
4        [ 3.08643828 -0.51789495].      -31.170104877741366    -10.679755252484755
5        [1.20979293 5.12      ].        -37.888814093608474    -10.679755252484755
6        [-1.93526615  5.12      ].      -33.48579392674555     -10.679755252484755
7        [-0.89792848 -2.64272   ].      -26.018894786903587    -10.679755252484755
8        [-3.4830931   2.27188446].      -48.607738550470664    -10.679755252484755
9        [-5.12       -2.81628559].      -42.81069741853823     -10.679755252484755
10       [5.12      0.6503182].          -45.20929354866007     -10.679755252484755
```

```
11       [ 2.21431934 -2.88804458].       -23.394624654774354    -10.679755252484755
12       [2.11111433 1.90834962].         -12.05102427590678     -10.679755252484755
13       [-2.69537145 -0.92832216].       -22.489338055326584    -10.679755252484755
14       [ 0.73157855 -0.4668894 ].       -31.692439990512668    -10.679755252484755
15       [ 5.12       -1.79840595].       -39.1642161864332 8    -10.679755252484755
16       [ 3.03776991 -5.12       ].      -38.433033436284795    -10.679755252484755
17       [-1.00816207 -5.12       ].      -29.954251731609475    -10.679755252484755
18       [-5.12 -5.12].           -57.849427451571785    -10.679755252484755
19       [1.95067909 2.97127228].         -13.272406355275303    -10.679755252484755
20       [-1.6722066   0.52361912].       -37.65611843816105     -10.679755252484755
```

```
### 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     [-3.54137249  2.45810889].       -57.903242869085595    -40.51116653209555
init     [-2.42365424  0.34549139].       -40.51116653209555     -40.51116653209555
init     [-4.97075238  4.28796936].       -55.62655915398208     -40.51116653209555
init     [ 4.10332011 -4.77776458].       -49.962803461970296    -40.51116653209555
init     [ 4.6791612  -3.71497655].       -62.183891474990624    -40.51116653209555
1        [5.12 5.12].     -57.849427451571785    -40.51116653209555
2        [-5.12 -5.12].           -57.849427451571785    -40.51116653209555
3        [2.51309592 0.83213308].         -32.03971426626734     -32.03971426626734
4        [0.33253154 5.12       ].        -43.991598753921245    -32.03971426626734
5        [-0.48618694 -4.11927945].       -39.846598671493474    -32.03971426626734
6        [-5.12       -1.54443991].       -50.92270382455786     -32.03971426626734
7        [ 5.12       -0.37881497].       -46.30672922060628     -32.03971426626734
8        [ 1.63812548 -1.9816146 ].       -23.141446916470205    -23.141446916470205
9        [-0.04356885  2.14282559].       -8.729516183363952     -8.729516183363952
10       [2.70818235 3.4913327 ].         -52.10618312240411     -8.729516183363952
11       [-2.63746706 -2.72194224].       -42.61505981707079     -8.729516183363952
12       [-2.28788975  5.12       ].      -46.51741172574315     -8.729516183363952
13       [5.12       2.25424055].         -44.27272450222781     -8.729516183363952
14       [ 0.0037311 -0.3262923].         -14.721330502388067    -8.729516183363952
15       [ 1.66611538 -5.12       ].      -46.73062167571054     -8.729516183363952
16       [-2.54732531 -5.12       ].      -54.97473172846729     -8.729516183363952
17       [-5.12        0.84440909].       -34.047653349556896    -8.729516183363952
18       [-0.7808367   2.68240734].       -30.000037051648487    -8.729516183363952
19       [0.5127096  1.34737847].         -37.790234939259534    -8.729516183363952
20       [-0.5211275  -1.60016792].       -40.82809429425107     -8.729516183363952
```

```
### 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     [ 2.84367268 -2.68757791].       -33.58019830007169     -17.435826639425656
init     [3.32061217 4.76927179].         -56.857057997503354    -17.435826639425656
init     [ 4.83943541 -0.47667971].       -48.211919361679726    -17.435826639425656
init     [1.11659482 2.82139151].         -17.435826639425656    -17.435826639425656
init     [1.45012065 2.27346667].         -38.25352329493884     -17.435826639425656
1        [-5.12       -2.72662613].       -47.82255372239195     -17.435826639425656
2        [-5.12  5.12].           -57.849427451571785    -17.435826639425656
3        [-1.07701199 -5.12      ].       -31.23270596882761     -17.435826639425656
4        [-2.85654782  1.12632981].       -16.21112054978481     -16.21112054978481
5        [-1.154889  5.12    ].           -34.63188133754804     -16.21112054978481
6        [-0.62085659 -1.40877295].       -38.024559266150035    -16.21112054978481
7        [ 5.12 -5.12].           -57.849427451571785    -16.21112054978481
8        [-5.12       1.7378335].         -42.70847828668707     -16.21112054978481
9        [5.12       2.46276694].         -54.717535612015695    -16.21112054978481
10       [ 1.73841757 -5.12      ].       -42.673912520821744    -16.21112054978481
11       [-3.90824115 -5.12      ].       -45.815513030575275    -16.21112054978481
12       [-1.25798002  2.43117962].       -37.07391719340609     -16.21112054978481
13       [ 1.88141228 -0.24378477].       -15.858585300123591    -15.858585300123591
14       [-3.23791011 -0.94253001].       -21.258418036226548    -15.858585300123591
15       [0.95611055 5.12      ].         -30.216690836228075    -15.858585300123591
16       [-2.30297224 -3.0776452 ].       -29.209429517284445    -15.858585300123591
17       [-3.20380788  3.62576427].       -47.585821405077965    -15.858585300123591
18       [ 5.12       -2.74078372].       -47.01536166541292     -15.858585300123591
19       [-5.12       -0.42609353].       -48.04731724732528     -15.858585300123591
20       [ 0.51107486 -3.1800649 ].       -36.09570624703747     -15.858585300123591
```

```
### 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     [0.14277984 2.79721013].         -18.683085263052178    -10.423838604848608
init     [ 3.7931795  -5.03759925].       -47.36348784446708     -10.423838604848608
init     [-1.94830412  4.68586229].       -40.19779334078636     -10.423838604848608
init     [ 0.13431513 -1.86076749].       -10.423838604848608    -10.423838604848608
init     [ 0.40140736 -2.85434939].       -30.353548735049138    -10.423838604848608
1        [-5.12 -5.12].           -57.8494274515718      -10.423838604848608
2        [5.12 5.12].     -57.849427451571785    -10.423838604848608
3        [-5.12        0.35555346].       -45.207642852354454    -10.423838604848608
4        [ 5.12       -0.06379677].       -29.721475124888244    -10.423838604848608
5        [-5.12  5.12].           -57.849427451571785    -10.423838604848608
6        [-1.64656857  0.04127342].       -19.098167256925223    -10.423838604848608
7        [-1.68580028 -5.12      ].       -45.69191789552137     -10.423838604848608
8        [1.91874718 0.23697281].         -14.195264550334736    -10.423838604848608
9        [1.75616124 5.12      ].         -41.621790330862304    -10.423838604848608
10       [-3.36992531 -2.3260542 ].       -48.20779673906814     -10.423838604848608
11       [3.48152774 2.41512991].         -56.498180801428376    -10.423838604848608
12       [ 3.28653767 -2.13814022].       -31.18475038164678     -10.423838604848608
13       [-3.14137462  2.27899541].       -30.566374914781573    -10.423838604848608
14       [ 1.13105582 -5.12      ].       -33.40703924772944     -10.423838604848608
15       [ 5.12       -2.95114372].       -38.1014370278825      -10.423838604848608
16       [ 0.28544129 -0.04890574].       -12.760765122036727    -10.423838604848608
17       [-5.12       2.7306299].         -47.595110257946494    -10.423838604848608
18       [-0.92695044  1.65967373].       -20.024288270649013    -10.423838604848608
```

```
19       [-1.0865402  -1.60055036].       -23.254370993180814    -10.423838604848608
20       [-5.12       -2.31626272].       -48.33395381347558     -10.423838604848608
```

```
### 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     [ 3.57189322 -3.28810573].       -54.938487770767075    -7.990765314336182
init     [-4.56332069 -1.41784631].       -60.750198753157726    -7.990765314336182
init     [-2.29989449  0.3072023 ].       -31.984997246800887    -7.990765314336182
init     [-1.9873903  -2.00218256].       -7.990765314336182     -7.990765314336182
init     [-3.97576933 -2.5610341 ].       -41.754957769694336    -7.990765314336182
1        [4.80390202 4.79749377].         -59.83100237180094     -7.990765314336182
2        [-5.12  5.12].           -57.849427451571785    -7.990765314336182
3        [-0.15101206  5.12      ].       -33.12122955022906     -7.990765314336182
4        [2.15573482 1.04431635].         -10.5403695712771      -7.990765314336182
5        [-0.33203224 -5.12      ].       -43.96399510457497     -7.990765314336182
6        [5.12       0.33805777].         -44.29383160417213     -7.990765314336182
7        [ 0.62332395 -1.50563915].       -39.79434481153279     -7.990765314336182
8        [-5.12 -5.12].           -57.849427451571785    -7.990765314336182
9        [-5.12        1.87029541].       -35.563728771287046    -7.990765314336182
10       [-0.06815161  2.21604192].       -13.700881289266224    -7.990765314336182
11       [-2.49508483  3.43905833].       -57.323623699278095    -7.990765314336182
12       [2.21825638 3.42305389].         -43.51053400352421     -7.990765314336182
13       [ 5.12 -5.12].           -57.849427451571785    -7.990765314336182
14       [-2.63547612 -5.12      ].       -52.461097427974124    -7.990765314336182
15       [ 2.03069657 -5.12      ].       -33.23386493691041     -7.990765314336182
16       [ 2.88188576 -0.6198523 ].       -28.61523665850344     -7.990765314336182
17       [5.12       2.48546397].         -55.060565689044005    -7.990765314336182
18       [-1.40748939 -2.66961784].       -42.30415811045853     -7.990765314336182
19       [0.3037588  0.81115457].         -20.315544156287167    -7.990765314336182
20       [ 5.12       -1.87138279].       -35.51824000642459     -7.990765314336182
```

```
### 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     [-2.83349935  0.23719262].       -22.27210456874814     -22.27210456874814
init     [ 0.51918292 -4.65303603].       -57.57021076017139     -22.27210456874814
init     [-1.42613673 -2.83565116].       -33.89145899403749     -22.27210456874814
init     [ 1.9325559  -3.44339021].       -35.85029586225333     -22.27210456874814
init     [-4.39987336  4.51595121].       -77.78800881964571     -22.27210456874814
1        [5.12 5.12].     -57.849427451571785    -22.27210456874814
2        [5.12       0.13331511].         -32.25032968272954     -22.27210456874814
```

```
3        [0.53988362 3.56704876].         -51.82862209140108     -22.27210456874814
4        [-5.12 -5.12].           -57.849427451571785    -22.27210456874814
5        [ 5.12 -5.12].           -57.849427451571785    -22.27210456874814
6        [1.26312866 0.0161978 ].         -12.471464098341075    -12.471464098341075
7        [-5.12       -1.65387318].       -47.339260009061995    -12.471464098341075
8        [3.37883226 2.40302338].         -52.63065484701844     -12.471464098341075
9        [-5.12        1.50296771].       -51.18188721612175     -12.471464098341075
10       [ 5.12       -2.35309448].       -50.49579770917917     -12.471464098341075
11       [-1.5829003  5.12      ].        -50.10411319109356     -12.471464098341075
12       [-0.67732573  1.00521315].       -15.883809934358975    -12.471464098341075
13       [-2.37072037 -5.12      ].       -51.423424087600736    -12.471464098341075
14       [2.42515527 5.12      ].         -53.72058180143779     -12.471464098341075
15       [ 2.68542534 -0.95832132].       -22.417763963744918    -12.471464098341075
16       [-2.26750676  2.47831826].       -42.28876315341566 5   -12.471464098341075
17       [-0.06521417 -1.03454494].       -2.1369774407517266    -2.1369774407517266
18       [-0.07171695 -1.03862795].       -2.375168584529092     -2.1369774407517266
19       [-0.07139232 -1.03848877].       -2.363870502294061     -2.1369774407517266
20       [-0.07123581 -1.03841959].       -2.358404089069163     -2.1369774407517266
```

```
### 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     [-2.10263037  0.31320838].       -20.395145364684023    -20.395145364684023
init     [-3.15882714 -4.42470033].       -53.03732051200137     -20.395145364684023
init     [2.93873111 1.60085526].         -29.989224812583537    -20.395145364684023
init     [1.40821398 0.77417363].         -29.451989415882437    -20.395145364684023
init     [-4.71999574 -1.45598869].       -55.89242173757483     -20.395145364684023
1        [ 5.12 -5.12].           -57.849427451571785    -20.395145364684023
2        [-5.12  5.12].           -57.849427451571785    -20.395145364684023
3        [-0.24352154  5.12      ].       -38.57707497365628     -20.395145364684023
4        [5.12 5.12].     -57.849427451571785    -20.395145364684023
5        [ 1.0773245  -3.51439942].       -34.62792051863994     -20.395145364684023
6        [ 5.12       -1.25380004].       -40.73546899864028     -20.395145364684023
7        [-5.12        1.83350688].       -37.27702105856103     -20.395145364684023
8        [-2.31927677  3.11405903].       -31.752962772227757    -20.395145364684023
9        [-1.18603141 -1.98920002].       -11.474681619626478    -11.474681619626478
10       [2.32596067 4.36235783].         -55.521548414876854    -11.474681619626478
11       [5.12       2.01153028].         -32.99719903869182     -11.474681619626478
12       [-0.58774766 -5.12      ].       -47.78842167778669     -11.474681619626478
13       [ 2.51459548 -1.52792814].       -48.4621668388941      -11.474681619626478
14       [ 2.54666739 -5.12      ].       -54.983410370000556    -11.474681619626478
15       [-0.08716349  2.3663121 ].       -23.74433419206523     -11.474681619626478
16       [-5.12 -5.12].           -57.849427451571785    -11.474681619626478
17       [-0.17364977 -0.83493825].       -11.024870561954287    -11.024870561954287
18       [-2.62641666  5.12      ].       -52.83062609846043     -11.024870561954287
19       [-2.21834819 -2.04159075].       -17.453011336479925    -11.024870561954287
20       [-0.14040589 -1.56795455].       -25.22586905478475     -11.024870561954287
```

```
### 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     [1.53983224 0.05584255].       -22.67190580753611      -22.67190580753611
init     [ 3.87687906 -3.25795609].     -38.990099416711985     -22.67190580753611
init     [3.60686662 2.56139557].       -56.66448698782129      -22.67190580753611
init     [1.70088108 4.99604939].       -40.894059318256296     -22.67190580753611
init     [-2.48864335 -4.83014733].     -54.6725749848372       -22.67190580753611
1        [-5.12        2.10205608].     -35.32979086565658      -22.67190580753611
2        [-2.46249901  5.12      ].     -54.71229979245777      -22.67190580753611
3        [-2.31060816 -0.7059126 ].     -32.28877161798881      -22.67190580753611
4        [ 1.04805997 -5.12      ].     -30.475617869606744     -22.67190580753611
5        [-5.12       -2.45342784].     -54.518931499239585     -22.67190580753611
6        [-0.69696296  2.25257333].     -28.99260352957237      -22.67190580753611
7        [ 5.12       -0.35164303].     -45.009422871296316     -22.67190580753611
8        [5.12 5.12].     -57.849427451571785     -22.67190580753611
9        [ 0.15291013 -2.40801192].     -28.468642227877105     -22.67190580753611
10       [-5.12  5.12].         -57.849427451571785     -22.67190580753611
11       [ 5.12 -5.12].         -57.849427451571785     -22.67190580753611
12       [-5.12 -5.12].         -57.849427451571785     -22.67190580753611
13       [-2.8683334   1.88955141].     -17.34184291669915      -17.34184291669915
14       [-5.12       -0.05581402].     -29.53646947128712      -17.34184291669915
15       [ 2.49768863 -1.57549268].     -47.61553995135146      -17.34184291669915
16       [1.22676146 2.30578465].       -28.80037240444558      -17.34184291669915
17       [-0.33016957  0.00783609].     -14.948062421257708     -14.948062421257708
18       [-0.28956939  5.12      ].     -41.46924811073556      -14.948062421257708
19       [-2.1622082  -2.62671669].     -33.32856278230591      -14.948062421257708
20       [ 2.76420942 -5.12      ].     -45.67394900653106      -14.948062421257708
```

```
### 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     [-4.12125592  2.6751971 ].     -41.43485596167127      -25.867003842388073
init     [-2.59135515 -3.70553152].     -51.60126035043229      -25.867003842388073
init     [-1.72598719 -4.27008445].     -43.97430127029199      -25.867003842388073
init     [1.76104531 3.13952049].       -25.867003842388073     -25.867003842388073
init     [4.9432772  1.38916592].       -44.66580306903559      -25.867003842388073
1        [ 5.12 -5.12].         -57.849427451571785     -25.867003842388073
2        [ 1.77595123 -1.68635391].     -28.267708038638098     -25.867003842388073
3        [5.12 5.12].     -57.849427451571785     -25.867003842388073
4        [-1.28669651  5.12      ].     -42.86563596868251      -25.867003842388073
5        [-5.12       -0.97841456].     -29.97383902098907      -25.867003842388073
6        [-1.20080606  0.35666675].     -24.73866483123347      -24.73866483123347
7        [ 1.54014175 -5.12      ].     -50.98036323268903      -24.73866483123347
8        [ 5.12       -1.82150314].     -37.89952918813169      -24.73866483123347
9        [-5.12 -5.12].         -57.849427451571785     -24.73866483123347
10       [-5.12  5.12].         -57.849427451571785     -24.73866483123347
11       [1.81534113 0.74360693].       -20.258854448428945     -20.258854448428945
```

```
12       [2.02695247 5.12      ].        -33.176300347125        -20.258854448428945
13       [-0.86367682  2.59088871].      -29.32134135335746      -20.258854448428945
14       [-0.58629637 -1.85536879].      -26.204445674808625     -20.258854448428945
15       [-3.21509619  0.05298775].      -18.71323526159539      -18.71323526159539
16       [ 3.23908302 -3.2786277 ].      -42.3447101212353       -18.71323526159539
17       [-5.12       -2.80415832].      -43.45044321307102      -18.71323526159539
18       [-5.12        0.93875125].      -30.537372240881766     -18.71323526159539
19       [ 3.37936763 -0.28953454].      -41.224946957674504     -18.71323526159539
20       [3.66247252 3.14299304].        -42.29034777468806      -18.71323526159539
```

```
### 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     [0.9024594  4.07258857].        -20.24255447774017      -17.388691338534382
init     [4.00927467 3.23417577].        -35.55852254546393      -17.388691338534382
init     [-4.75249064  1.96359764].      -36.54582989269194      -17.388691338534382
init     [-1.24230715  0.18955208].      -17.388691338534382     -17.388691338534382
init     [ 1.61742301 -3.13497377].      -33.22932870179905      -17.388691338534382
1        [-5.12 -5.12].          -57.849427451571785     -17.388691338534382
2        [ 5.12       -0.92646119].      -30.83167446343571      -17.388691338534382
3        [ 5.12 -5.12].          -57.849427451571785     -17.388691338534382
4        [-2.70368883  5.12      ].      -49.10357473157433      -17.388691338534382
5        [-1.34011158 -5.12      ].      -46.084798729665046     -17.388691338534382
6        [-5.12       -1.49922021].      -51.17225492646024      -17.388691338534382
7        [1.91181202 0.46610421].        -25.142609685867193     -17.388691338534382
8        [-2.03027328 -2.26094685].      -20.101518101659103     -17.388691338534382
9        [-1.34692706  2.54188332].      -43.651672434230726     -17.388691338534382
10       [-5.12  5.12].          -57.849427451571785     -17.388691338534382
11       [5.12 5.12].    -57.849427451571785     -17.388691338534382
12       [ 2.24150581 -5.12      ].      -43.415609430404515     -17.388691338534382
13       [5.12       1.32200513].        -45.04385866728544      -17.388691338534382
14       [ 0.18501646 -1.30649305].      -21.24611905786738      -17.388691338534382
15       [2.36236382 5.12      ].        -50.993448371041794     -17.388691338534382
16       [ 3.90976933 -2.66464311].      -39.06081733236839      -17.388691338534382
17       [-3.10053389  0.11466059].      -14.040853765393695     -14.040853765393695
18       [1.35406371 2.30367622].        -36.53189822448121      -14.040853765393695
19       [-0.32998131  5.12      ].      -43.85010886634531      -14.040853765393695
20       [-3.34844745 -3.46708979].      -58.81843584464441      -14.040853765393695
```

```
end_approx = time.time()
end_approx

time_approx = end_approx - start_approx
time_approx

start_exact = time.time()
start_exact
```

```
1623410092.7201362
```

```
### 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   	 [-0.84969467  2.25612281]. 	 -20.33436270766351 	 -19.908403246996286
init   	 [-5.1188288  -2.02411446]. 	 -33.07414982069084 	 -19.908403246996286
init   	 [-3.61721968 -4.17445279]. 	 -53.347974723929894 	 -19.908403246996286
init   	 [-3.21269544 -1.58145816]. 	 -39.218472310354045 	 -19.908403246996286
init   	 [-1.05710106  0.39748336]. 	 -19.908403246996286 	 -19.908403246996286
1      	 [ 4.42339399 -4.97713589]. 	 -63.30510968121598 	 -19.908403246996286
2      	 [4.8613414  1.39882906]. 	 -47.197115901892985 	 -19.908403246996286
3      	 [-4.0285811   4.97365835]. 	 -41.26421026895091 	 -19.908403246996286
4      	 [2.22135298 4.99022003]. 	 -38.0653370455428 	 -19.908403246996286
5      	 [ 0.36872699 -4.71183435]. 	 -51.49938954286396 	 -19.908403246996286
6      	 [ 2.26614624 -0.61084583]. 	 -34.19245837215231 	 -19.908403246996286
7      	 [4.84065641 5.10484999]. 	 -56.19107909595234 	 -19.908403246996286
8      	 [-4.21694827  1.88119384]. 	 -31.918906927805402 	 -19.908403246996286
9      	 [-0.81189684  4.89349523]. 	 -32.970428938558264 	 -19.908403246996286
10     	 [ 9.54978273e-04 -1.83450725e+00]. 	 -8.30185701149347 	 -8.30185701
11     	 [ 4.15547819 -1.86256638]. 	 -28.643542761159143 	 -8.30185701149347
12     	 [1.50169581 1.07776804]. 	 -24.58634356896986 	 -8.30185701149347
13     	 [ 2.40096702 -3.28345583]. 	 -46.75810352553396 	 -8.30185701149347
14     	 [2.62317966 3.00806909]. 	 -33.09387991447062 	 -8.30185701149347
15     	 [-1.22807086 -3.31429846]. 	 -35.05022607036226 	 -8.30185701149347
16     	 [-2.60116523  0.56209798]. 	 -44.377366501501314 	 -8.30185701149347
17     	 [-5.11900722 -5.022035  ]. 	 -54.18851475415071 	 -8.30185701149347
18     	 [-0.2083012  -0.97953904]. 	 -8.495270213723526 	 -8.30185701149347
19     	 [-2.31754441  3.95927362]. 	 -35.49016557832631 	 -8.30185701149347
20     	 [0.79953907 3.82374633]. 	 -27.72810554414676 	 -8.30185701149347
```

```
### 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   	 [-0.6554122  -4.85451539]. 	 -43.490296251903594 	 -18.964539418712707
init   	 [ 0.50854377 -0.6622987 ]. 	 -35.91861667536101 	 -18.964539418712707
init   	 [-0.81543371 -1.73737143]. 	 -20.479562046739524 	 -18.964539418712707
init   	 [-3.02439799  1.2213347 ]. 	 -18.964539418712707 	 -18.964539418712707
init   	 [-2.05153614 -2.3876887 ]. 	 -28.041315668371354 	 -18.964539418712707
1      	 [4.36389158 4.51702655]. 	 -75.950626279408 	 -18.964539418712707
2      	 [ 4.48423034 -3.74346998]. 	 -64.48301965911325 	 -18.964539418712707
3      	 [-1.81881041  5.10238239]. 	 -37.15105201697424 	 -18.964539418712707
4      	 [4.34503103 0.43953058]. 	 -53.98179453555729 	 -18.964539418712707
5      	 [-5.0413191   4.76055462]. 	 -57.75021097488111 	 -18.964539418712707
6      	 [-4.8744542  -4.31604188]. 	 -59.37321683709654 	 -18.964539418712707
```

```
7        [0.24643879 2.62993693].        -33.60192070246863      -18.964539418712707
8        [-4.89853329 -1.02091201].      -27.08842837239992      -18.964539418712707
9        [ 1.85392694 -4.19549834].      -31.605691743931096     -18.964539418712707
10       [-5.08467007  1.24116964].      -38.22187383691047      -18.964539418712707
11       [1.62299757 4.54944054].        -60.01237085482423      -18.964539418712707
12       [1.84803514 1.29777618].        -22.278911929756767     -18.964539418712707
13       [ 2.86303703 -2.28020611].      -28.762751540592845     -18.964539418712707
14       [-2.93853275  3.14620617].      -23.201142979166534     -18.964539418712707
15       [-1.32061146  0.64091136].      -32.77732030303480 6    -18.964539418712707
16       [ 5.02583217 -1.31251147].      -40.94060618149455      -18.964539418712707
17       [-2.75588542 -4.48231814].      -57.25472178168528 4    -18.964539418712707
18       [-2.79681167 -0.86912272].      -18.873305038605448     -18.873305038605448
19       [4.6154328  2.16262933].        -48.2442300276324 4     -18.873305038605448
20       [-0.1125684  -3.01107645].      -11.502193025965456     -11.502193025965456
```

```
### 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     [0.52017052 2.1314337 ].        -27.953858411008774     -10.607662635789808
init     [-2.14113547  0.11087468].      -10.607662635789808     -10.607662635789808
init     [4.02377681 4.05804123].        -33.42749829480097      -10.607662635789808
init     [-3.83400642 -2.99783293].      -28.650953928965198     -10.607662635789808
init     [-4.59297584 -0.6061072 ].      -57.6631355589384       -10.607662635789808
1        [ 3.48734552 -4.65410759].      -69.45782440995153      -10.607662635789808
2        [-4.98603203  3.95760033].      -40.914367327747186     -10.607662635789808
3        [ 4.52457904 -0.15917927].      -44.97640314727789      -10.607662635789808
4        [-0.57642942 -4.7418849 ].      -52.19634115894448      -10.607662635789808
5        [-1.42281295  4.75805437].      -53.00446296713395      -10.607662635789808
6        [ 1.31868059 -1.28112651].      -29.506132511923816     -10.607662635789808
7        [1.65231998 4.89973065].        -44.416614141149786     -10.607662635789808
8        [-4.68395133 -4.89977121].      -61.897309241378935     -10.607662635789808
9        [-1.11691791 -1.97064683].      -7.8796719152898085     -7.8796719152898085
10       [-2.41452284  1.97071276].      -28.47458377401539      -7.8796719152898085
11       [2.58771592 1.07672233].        -27.514495077271388     -7.8796719152898085
12       [ 4.91617166 -2.44479709].      -50.90559701459573      -7.8796719152898085
13       [-0.65813369  0.19542846].      -22.566200870029704     -7.8796719152898085
14       [4.71831013 1.91816128].        -39.2129693337483       -7.8796719152898085
15       [-4.91625828  1.60753634].      -45.90924899335475      -7.8796719152898085
16       [-1.93021222 -2.06808393].      -9.849877227999693      -7.8796719152898085
17       [ 1.79743982 -3.16496593].      -25.218753469214302     -7.8796719152898085
18       [-2.45455158 -4.94553136].      -50.65807767451396      -7.8796719152898085
19       [-0.45091283 -1.68270436].      -36.666385860560396     -7.8796719152898085
20       [-3.87966379  4.96768157].      -42.65990680699781      -7.8796719152898085
```

```
### 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     [4.78238555 0.48365823].         -51.03163809010808    -14.323038259018315
init     [4.84028785 2.19971578].         -39.78645699016559    -14.323038259018315
init     [ 2.02474316 -2.90724357].       -14.323038259018315   -14.323038259018315
init     [ 4.87705042 -5.05620219].       -52.80627247106233    -14.323038259018315
init     [-2.52946061 -0.66773471].       -41.61497868486559    -14.323038259018315
1        [-1.31242285  4.65120347].       -52.99501268754465    -14.323038259018315
2        [-1.79792293 -4.9979256 ].       -35.246844737137096   -14.323038259018315
3        [-5.1107157   4.88035335].       -54.956006925958356   -14.323038259018315
4        [0.49528365 1.43722479].         -41.53869473449281    -14.323038259018315
5        [2.61954547 4.41377834].         -62.22075788333776    -14.323038259018315
6        [-4.46867611 -3.06681654].       -50.049755346011274   -14.323038259018315
7        [-5.08816049  1.34138555].       -44.61561614337628    -14.323038259018315
8        [ 1.4128747  -5.01726856].       -45.76668327811586    -14.323038259018315
9        [ 1.99683481 -0.91902225].       -6.100622971156982    -6.100622971156982
10       [-0.12955962 -2.81923883].       -16.884807615728224   -6.100622971156982
11       [ 4.76323948 -2.33585237].       -52.45019350840946    -6.100622971156982
12       [-2.5459631   2.28305054].       -43.34185972030765    -6.100622971156982
13       [2.52811071 0.86699862].         -30.28066245615318    -6.100622971156982
14       [ 0.54113171 -1.14565121].       -25.176554693957844   -6.100622971156982
15       [4.65501383 4.3387058 ].         -71.40307029735814    -6.100622971156982
16       [-4.97137958 -0.65943059].       -40.69915900029997    -6.100622971156982
17       [-2.62766968 -2.9459918 ].       -33.105249786072434   -6.100622971156982
18       [ 2.63393002 -1.19616813].       -31.713571767856102   -6.100622971156982
19       [-4.39328907 -5.07008141].       -63.79582965617792    -6.100622971156982
20       [0.57369212 4.92214771].         -44.67645515732715    -6.100622971156982
```

```
### 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     [-2.84678993  3.79629882].       -33.93442008827236    -7.8108627039749745
init     [-3.00319585  4.2865757 ].       -39.673876075575784   -7.8108627039749745
init     [-0.11866943  1.14425716].       -7.8108627039749745   -7.8108627039749745
init     [2.72289645 0.1886002 ].         -25.38160395721669    -7.8108627039749745
init     [-2.08076286 -3.19773462].       -22.589982116319675   -7.8108627039749745
1        [ 4.32895605 -5.09732646].       -61.29478907190488    -7.8108627039749745
2        [4.48759904 4.77928109].         -71.12030870314028    -7.8108627039749745
3        [-4.99191113  0.34486967].       -40.6650834440818     -7.8108627039749745
4        [1.09890332 4.03071434].         -19.509403576262443   -7.8108627039749745
5        [ 0.40473436 -5.0026839 ].       -43.45349053170776    -7.8108627039749745
6        [-4.90719846 -2.99028963].       -34.69337862271707    -7.8108627039749745
7        [ 4.99789072 -1.69210937].       -41.40072125086606    -7.8108627039749745
8        [ 0.25167886 -1.71738855].       -25.152979368157432   -7.8108627039749745
9        [4.94463942 1.40316983].         -45.22286008564386    -7.8108627039749745
10       [-2.47806055 -0.2369829 ].       -35.285105616343955   -7.8108627039749745
11       [-2.59117085 -5.07051563].       -51.793498551629234   -7.8108627039749745
12       [-5.07359353  2.93394012].       -36.24862419143108    -7.8108627039749745
13       [2.30209178 2.33373954].         -38.98295106989639    -7.8108627039749745
14       [ 1.98851241 -2.83803699].       -16.780949125479555   -7.8108627039749745
15       [-0.35282096  4.89667092].       -42.15672478029163    -7.8108627039749745
```

```
16       [-0.7522715   2.48972322].       -36.60107644784611     -7.8108627039749745
17       [-0.12796772  0.27724306].       -14.85863078407601     -7.8108627039749745
18       [ 2.35284444 -1.15007701].       -27.006111777618557    -7.8108627039749745
19       [-3.18156905  2.05101163].       -20.66991503219672     -7.8108627039749745
20       [1.78710006 4.92457531].         -36.23723769096235     -7.8108627039749745
```

```
### 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     [ 4.02288795 -1.72052679].       -31.08835710146886     -17.28954482757088
init     [ 3.28938622 -4.69302655].       -58.797867722203385    -17.28954482757088
init     [-4.0175956   0.97333314].       -17.28954482757088     -17.28954482757088
init     [ 0.30532979 -0.83141193].       -19.296253155889353    -17.28954482757088
init     [-1.68542362  1.25459899].       -28.650630936276173    -17.28954482757088
1        [4.93241742 4.79914066].         -55.20982233788753     -17.28954482757088
2        [-4.03046901 -5.01672984].       -41.65014747128707     -17.28954482757088
3        [0.76103835 4.69388342].         -45.372020949792976    -17.28954482757088
4        [-0.45226025 -3.945031  ].       -35.911831531312416    -17.28954482757088
5        [-4.5691026   4.85971735].       -67.20513965063277     -17.28954482757088
6        [-5.00301739 -1.82665255].       -33.73646542523921     -17.28954482757088
7        [3.08507637 1.39545192].         -30.77892734437885     -17.28954482757088
8        [-2.5451768  -2.11462915].       -33.03291347840022     -17.28954482757088
9        [-1.74746547  4.17511061].       -36.11071531168787     -17.28954482757088
10       [0.76142682 1.45539439].         -31.59041266151839     -17.28954482757088
11       [ 1.45544401 -2.27706232].       -38.60620639083284     -17.28954482757088
12       [5.07448183 0.44559609].         -46.445551216079615    -17.28954482757088
13       [-5.08765454  2.0689077 ].       -32.56600472011117     -17.28954482757088
14       [2.91694592 3.48326814].         -41.91750656826394     -17.28954482757088
15       [ 1.26413592 -5.0284348 ].       -37.92938989718779     -17.28954482757088
16       [-3.42751718 -0.00343358].       -30.730963530924512    -17.28954482757088
17       [ 4.90093551 -3.21938767].       -44.34745439628051     -17.28954482757088
18       [ 2.3927977  -0.16243155].       -28.339198215609677    -17.28954482757088
19       [-3.07033367  2.51236469].       -36.66943771070963     -17.28954482757088
20       [-1.89301223 -4.8318944 ].       -34.184813503661445    -17.28954482757088
```

```
### 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     [-4.33860312  2.86636843].       -45.646133072936244    -22.52235437888213
init     [-0.63068947  2.28828342].       -34.83012662845338     -22.52235437888213
init     [4.8946126  0.39419771].         -44.09657005662876     -22.52235437888213
init     [ 0.01147355 -4.38219639].       -36.613209822404315    -22.52235437888213
init     [-2.37118484e+00 -1.20319155e-03].       -22.52235437888213     -22.5223543
```

```
1      [-5.10538874 -4.17523358].      -51.084190064364634     -22.52235437888213
2      [4.61256095 5.03612112].        -64.49624735465329      -22.52235437888213
3      [ 4.19516762 -5.11810643].      -53.046362925673584     -22.52235437888213
4      [ 1.33502396 -1.02148072].      -18.008364266544717     -18.008364266544717
5      [1.98099381 2.65837202].        -26.506850185908945     -18.008364266544717
6      [-2.16075612  5.04321612].      -35.15119723418208      -18.008364266544717
7      [-4.7513011  -1.23270559].      -42.92817262507963      -18.008364266544717
8      [ 3.94437426 -2.25349681].      -31.46061249954035      -18.008364266544717
9      [1.16384929 4.7966166 ].        -36.32225644587605      -18.008364266544717
10     [-1.20885437 -2.24554936].      -23.667660617322973     -18.008364266544717
11     [-2.51449543 -4.83053823].      -54.76819589272778      -18.008364266544717
12     [5.03923336 2.54702745].        -51.75014159070215      -18.008364266544717
13     [ 2.1801667  -3.30936549].      -35.100862191640935     -18.008364266544717
14     [2.67985127 0.38523906].        -39.10729421594518      -18.008364266544717
15     [-4.52270243  4.54957566].      -80.5706827294871       -18.008364266544717
16     [-0.36388026 -0.05675437].      -17.32491319730984      -17.32491319730984
17     [-4.99936967  1.51212822].      -47.251286232484475     -17.32491319730984
18     [-3.01469763 -2.51770406].      -35.408040387486025     -17.32491319730984
19     [-2.63770203  1.77405165].      -35.08408980974093      -17.32491319730984
20     [ 1.29793018 -5.11408499].      -43.26570813408104      -17.32491319730984
```

```
### 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   [3.82391708 4.79785639].        -50.20079446939181      -13.871821018360485
init   [3.78055209 0.31596228].        -36.5114251593508       -13.871821018360485
init   [-2.73686192 -5.00327624].      -43.34985765011677      -13.871821018360485
init   [-0.7119993  -0.99992207].      -13.871821018360485     -13.871821018360485
init   [ 0.23218863 -0.22126801].      -17.190590355445654     -13.871821018360485
1      [-5.11238257  5.00673475].      -53.60395082170822      -13.871821018360485
2      [ 2.56577802 -4.57597654].      -65.562842188195        -13.871821018360485
3      [-5.01085567 -0.73166081].      -36.81699162045986      -13.871821018360485
4      [-1.23481775  3.92215305].      -27.128141935657005     -13.871821018360485
5      [-3.39026727  2.00338745].      -33.225564400282494     -13.871821018360485
6      [1.51944588 2.6337832 ].        -45.84125401155391      -13.871821018360485
7      [-0.50247222 -3.3111631 ].      -44.964167247731105     -13.871821018360485
8      [-4.77476384 -4.0012935 ].      -47.25936230881446      -13.871821018360485
9      [ 4.32579014 -2.89084826].      -43.91455205952295      -13.871821018360485
10     [-2.45114727 -1.53105431].      -47.69508780271624      -13.871821018360485
11     [4.84983403 2.74878684].        -45.28353092995875      -13.871821018360485
12     [ 2.07765374 -1.31380166].      -21.11185341835642      -13.871821018360485
13     [ 5.11323445 -5.03131037].      -54.078129634596635     -13.871821018360485
14     [-1.26079714  1.22301922].      -22.076125048178284     -13.871821018360485
15     [0.2059003   4.65698126].        -44.51176512967983      -13.871821018360485
16     [-2.61045836  4.89988187].      -50.42427301177963      -13.871821018360485
17     [ 0.1520891  -4.68052426].      -40.387247117414205     -13.871821018360485
18     [-4.87441993  2.95918624].      -35.79851418813992      -13.871821018360485
19     [ 5.06936937 -1.27326204].      -39.711030456937614     -13.871821018360485
20     [0.99456667 0.71287151].        -13.814924987023188     -13.814924987023188
```

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

```
### 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 	 [-5.01376866  0.01919582]. 	 -25.24828902616244 	 -24.454800313488693
init 	 [-0.04328148 -3.74958562]. 	 -24.454800313488693 	 -24.454800313488693
init 	 [-3.66478248 -2.88195916]. 	 -39.46509426509438 	 -24.454800313488693
init 	 [-0.83447623 -2.57944404]. 	 -31.06766397812992 	 -24.454800313488693
init 	 [-4.25922917 -1.58209393]. 	 -49.922543556206975 	 -24.454800313488693
1      	 [3.11029524 4.9495987 ]. 	 -36.97642606974743 	 -24.454800313488693
2      	 [ 4.97700533 -1.02611641]. 	 -26.06202024938157 	 -24.454800313488693
3      	 [-2.33124512  3.93000982]. 	 -36.71709267058873 	 -24.454800313488693
4      	 [1.31296979 1.13842648]. 	 -20.423887262816194 	 -20.423887262816194
5      	 [ 3.81870885 -4.87051037]. 	 -47.25134756789021 	 -20.423887262816194
6      	 [-1.56970408  0.80666665]. 	 -28.685134061901998 	 -20.423887262816194
7      	 [4.73691787 1.85573731]. 	 -40.53759113633525 	 -20.423887262816194
8      	 [-5.06696237  3.48486233]. 	 -58.64529333264212 	 -20.423887262816194
9      	 [ 2.25507922 -1.46157585]. 	 -37.25064915147307 	 -20.423887262816194
10     	 [-4.77457463 -4.93780479]. 	 -56.39443082766477 	 -20.423887262816194
11     	 [0.14594062 4.16108897]. 	 -25.953322583342157 	 -20.423887262816194
12     	 [ 0.01598073 -5.08207082]. 	 -27.17842355721745 	 -20.423887262816194
13     	 [-2.17301164 -4.54928398]. 	 -50.2914718988691 	 -20.423887262816194
14     	 [2.07381684 2.97312751]. 	 -14.338839450762492 	 -14.338839450762492
15     	 [4.99785591 5.11534255]. 	 -53.659340631129886 	 -14.338839450762492
16     	 [-0.29107874 -0.74733833]. 	 -23.362962532669524 	 -14.338839450762492
17     	 [ 4.50629898 -2.92321655]. 	 -49.985460630989245 	 -14.338839450762492
18     	 [-3.65278584  1.43479846]. 	 -50.30935016037346 	 -14.338839450762492
19     	 [ 1.87347176 -3.67422405]. 	 -34.5902654573019 	 -14.338839450762492
20     	 [-0.83810217  2.10269432]. 	 -11.877020578351724 	 -11.877020578351724
```

```
### 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 	 [ 2.77832339 -4.90750004]. 	 -41.674330194390116 	 -13.839458310244165
init 	 [1.36855793 2.54775176]. 	 -44.69639719065837 	 -13.839458310244165
init 	 [-0.01528819 -2.81808235]. 	 -13.839458310244165 	 -13.839458310244165
init 	 [-3.09183626  2.66783449]. 	 -33.23221510904937 	 -13.839458310244165
init 	 [-3.38830503 -4.2154003 ]. 	 -54.73014366983691 	 -13.839458310244165
1      	 [ 4.23507693 -0.431621  ]. 	 -46.2770687630113 	 -13.839458310244165
2      	 [4.74478086 4.86802142]. 	 -59.784126747807164 	 -13.839458310244165
3      	 [-4.98218436  0.09559441]. 	 -26.644130043331373 	 -13.839458310244165
4      	 [-0.96866344  5.10031529]. 	 -29.066224550208638 	 -13.839458310244165
5      	 [-1.07106721 -0.80473677]. 	 -9.403456188541279 	 -9.403456188541279
6      	 [-0.21023307 -5.06620597]. 	 -34.09073870767831 	 -9.403456188541279
7      	 [-4.64973396  4.86109759]. 	 -64.714437987482 	 -9.403456188541279
8      	 [ 1.69147387 -0.77785559]. 	 -25.319832804208247 	 -9.403456188541279
```

```
9        [ 4.74428194 -3.44141813].       -64.04096585220329     -9.403456188541279
10       [5.08116922 2.28338441].         -44.38694231636987     -9.403456188541279
11       [-2.82345036 -0.79073572].       -21.61255174358596     -9.403456188541279
12       [2.13394352 4.69246785].         -43.44684206466981     -9.403456188541279
13       [-1.26229872  1.31451006].       -28.03652575659897     -9.403456188541279
14       [-4.67059871 -2.41225009].       -60.93616704167654     -9.403456188541279
15       [-4.82300452  2.23896475].       -43.15368921599071     -9.403456188541279
16       [ 2.04476302 -2.53058008].       -30.793788587920115    -9.403456188541279
17       [-1.13081955 -2.22976056].       -18.174484753136333    -9.403456188541279
18       [2.52668193 1.05220058].         -27.88411737679069     -9.403456188541279
19       [0.20451314 0.27174141].         -18.65820407199581     -9.403456188541279
20       [-0.72662049  3.48877742].       -44.13839493445833     -9.403456188541279
```

```
### 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     [-3.27403839 -4.92057353].       -47.655641290890955    -10.679755252484755
init     [-0.37664229  2.30332343].       -35.87889240695259     -10.679755252484755
init     [-0.81711509 -0.14922651].       -10.679755252484755    -10.679755252484755
init     [-4.98912446 -0.12931474].       -28.05462905574235     -10.679755252484755
init     [4.52410012 3.59214172].         -71.62694632141611     -10.679755252484755
1        [ 3.39619778 -4.31739077].       -62.230362468246824    -10.679755252484755
2        [-3.97579834  4.78684114].       -46.542034180630544    -10.679755252484755
3        [ 4.931333   -0.42545154].       -44.33834986185242     -10.679755252484755
4        [-0.42268955 -2.93333409].       -28.490864310297965    -10.679755252484755
5        [0.73969381 5.08396191].         -38.40048119269832     -10.679755252484755
6        [2.28258149 0.30555921].         -30.75684878013545     -10.679755252484755
7        [-3.70079875  2.03704169].       -31.157469768014984    -10.679755252484755
8        [-4.81401576 -2.63371139].       -52.87016994280418     -10.679755252484755
9        [ 0.68997711 -5.01338344].       -39.327999760054304    -10.679755252484755
10       [-1.47124223  4.39000996].       -58.97947078467569     -10.679755252484755
11       [-2.75051529 -0.78520666].       -25.955404634035425    -10.679755252484755
12       [ 2.22237313 -2.06585807].       -18.32356790543367     -10.679755252484755
13       [2.35833133 2.34360878].         -42.89565742187748     -10.679755252484755
14       [ 0.36501166 -0.46540981].       -36.72826028143761     -10.679755252484755
15       [-1.55072052  0.76735836].       -31.40155389396293     -10.679755252484755
16       [-2.08426949 -2.94862605].       -14.92434655895212     -10.679755252484755
17       [ 4.66349498 -2.067656  ].       -42.08497191666144     -10.679755252484755
18       [-5.11656722 -4.75235687].       -61.180447054973435    -10.679755252484755
19       [-5.06197412  2.18895669].       -37.421608015540365    -10.679755252484755
20       [ 4.88509342 -5.05913643].       -52.63640561558164     -10.679755252484755
```

```
### 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     [-3.54137249  2.45810889].      -57.903242869085595    -40.51116653209555
init     [-2.42365424  0.34549139].      -40.51116653209555     -40.51116653209555
init     [-4.97075238  4.28796936].      -55.62655915398208     -40.51116653209555
init     [ 4.10332011 -4.77776458].      -49.962803461970296    -40.51116653209555
init     [ 4.6791612  -3.71497655].      -62.183891474990624    -40.51116653209555
1        [4.56681598 5.1125623 ].        -68.52421939280718     -40.51116653209555
2        [-1.429116   -5.07020608].      -47.731094352895866    -40.51116653209555
3        [2.4003401  0.24428674].        -33.56512328478563     -33.56512328478563
4        [0.24346729 4.34138446].        -43.928059226736345    -33.56512328478563
5        [-4.47959654 -3.34109879].      -66.56409552980816     -33.56512328478563
6        [ 1.06241937 -2.98131602].      -10.845075972986498    -10.845075972986498
7        [5.06480093 1.74041899].        -40.10038904260732     -10.845075972986498
8        [-1.09003781 -2.34624776].      -23.936557472431417    -10.845075972986498
9        [-4.78184137 -0.11679336].      -33.4661956488988      -10.845075972986498
10       [-2.2382317   4.97247451].      -39.145615858522085    -10.845075972986498
11       [0.00894914 1.73777303].        -13.80322725535334     -10.845075972986498
12       [ 1.58072459 -4.65015533].      -58.73363184192738     -10.845075972986498
13       [2.54136531 3.47107719].        -58.006385821682386    -10.845075972986498
14       [ 4.37141807 -0.52591061].      -56.16379879230416     -10.845075972986498
15       [ 0.61736489 -0.91869582].      -19.90386656831897     -10.845075972986498
16       [ 2.10463681 -2.08778697].      -12.355868157317605    -10.845075972986498
17       [-1.11469594  2.86736483].      -15.226755781333043    -10.845075972986498
18       [-3.91662165 -5.06347114].      -43.104552987872864    -10.845075972986498
19       [-2.58018261 -1.92241974].      -30.27527779296744     -10.845075972986498
20       [0.54713589 0.89828309].        -22.64461028496768     -10.845075972986498
```

```
### 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     [ 2.84367268 -2.68757791].      -33.58019830007169     -17.435826639425656
init     [3.32061217 4.76927179].        -56.857057997503354    -17.435826639425656
init     [ 4.83943541 -0.47667971].      -48.211919361679726    -17.435826639425656
init     [1.11659482 2.82139151].        -17.435826639425656    -17.435826639425656
init     [1.45012065 2.27346667].        -38.25352329493884     -17.435826639425656
1        [-4.77366547 -2.51495235].      -57.58729610569417     -17.435826639425656
2        [-4.65374571  4.09568012].      -55.87107425121984     -17.435826639425656
3        [-1.27060472 -4.09193524].      -31.271886524351807    -17.435826639425656
4        [-2.10094624  1.06573776].      -8.335652949817936     -8.335652949817936
5        [ 4.86268922 -5.06347529].      -53.5658403863605      -8.335652949817936
6        [-5.09729204  0.63523074].      -44.79913732627612     -8.335652949817936
7        [-0.96020581  4.57337119].      -41.10473946134023     -8.335652949817936
8        [ 0.28480628 -1.36186149].      -30.569255140213812    -8.335652949817936
9        [ 1.31600496 -4.67140141].      -52.323399452247806    -8.335652949817936
10       [4.69005593 2.62048695].        -59.81026203807347 5    -8.335652949817936
11       [-2.31171513 -1.69127824].      -35.59212933809745     -8.335652949817936
12       [-4.6036405  -4.99499112].      -64.102009656859       -8.335652949817936
13       [-2.15710968  2.2784746 ].      -26.11344186501026     -8.335652949817936
14       [1.82868854 0.16050324].        -13.293153857264013    -8.335652949817936
15       [-0.42990146  0.76590682].      -28.81926948689457     -8.335652949817936
16       [-2.97460849  0.59536936].      -27.587493726219805    -8.335652949817936
```

```
    17       [-3.18242114  5.10631514].        -44.23196492926981      -8.335652949817936
    18       [0.88141446 4.83296117].          -31.804434280697322     -8.335652949817936
    19       [ 5.00393882 -2.51207676].        -51.32422010016859      -8.335652949817936
    20       [2.98942295 0.25129508].          -19.10324517218459      -8.335652949817936
```

```
### 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     [0.14277984 2.79721013].          -18.683085263052178     -10.423838604848608
    init     [ 3.7931795  -5.03759925].        -47.36348784446708      -10.423838604848608
    init     [-1.94830412  4.68586229].        -40.19779334078636      -10.423838604848608
    init     [ 0.13431513 -1.86076749].        -10.423838604848608     -10.423838604848608
    init     [ 0.40140736 -2.85434939].        -30.353548735049138     -10.423838604848608
    1        [-4.41622269 -4.73809363].        -71.34622836697065      -10.423838604848608
    2        [5.05912775 0.29982017].          -39.4462875989013       -10.423838604848608
    3        [-4.20980556  0.51637465].        -45.43749551773853      -10.423838604848608
    4        [4.53544959 5.1124022 ].          -68.85187656736286      -10.423838604848608
    5        [1.31360407 0.18385684].          -21.61289552814893      -10.423838604848608
    6        [-4.94809879  4.29013902].        -55.911359496868954     -10.423838604848608
    7        [-1.12649164 -5.00366347].        -29.303794038681126     -10.423838604848608
    8        [-2.65717024 -2.21344302].        -35.19061913995456      -10.423838604848608
    9        [ 3.17519888 -2.28675294].        -33.07114501118153      -10.423838604848608
    10       [1.34298568 5.01025395].          -42.44271087505727      -10.423838604848608
    11       [2.64620399 2.34957814].          -44.44842202071649      -10.423838604848608
    12       [-0.82427784  1.06061752].        -8.02150276828544       -8.02150276828544
    13       [ 5.10402074 -2.14030754].        -36.33349317142212      -8.02150276828544
    14       [-2.7309059   2.45732687].        -44.335842084340364     -8.02150276828544
    15       [-1.82397385 -0.17043247].        -14.079857503896577     -8.02150276828544
    16       [ 1.34255645 -4.68101175].        -53.40770861969442      -8.02150276828544
    17       [-5.05395073 -1.21803576].        -35.60019563613481      -8.02150276828544
    18       [-0.32442331 -0.54961933].        -34.43285117006625      -8.02150276828544
    19       [5.04108374 2.69311828].          -46.49512008472046      -8.02150276828544
    20       [ 2.04906264 -0.68284468].        -19.231730591838726     -8.02150276828544
```

```
### 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     [ 3.57189322 -3.28810573].        -54.938487770767075     -7.990765314336182
    init     [-4.56332069 -1.41784631].        -60.750198753157726     -7.990765314336182
    init     [-2.29989449  0.3072023 ].        -31.984997246800887     -7.990765314336182
    init     [-1.9873903  -2.00218256].        -7.990765314336182      -7.990765314336182
    init     [-3.97576933 -2.5610341 ].        -41.754957769694336     -7.990765314336182
    1        [4.80390202 4.79749377].          -59.83100237180094      -7.990765314336182
```

```
    2        [-4.16810549  5.02944823].       -47.91763842306193     -7.990765314336182
    3        [1.77549009 1.28607595].         -25.458963304224234    -7.990765314336182
    4        [ 0.0186844  -5.05314554].       -26.15582323479906     -7.990765314336182
    5        [0.8415407  4.63451672].         -43.38305067787293     -7.990765314336182
    6        [4.74515621 0.10172366].         -34.80511306555563     -7.990765314336182
    7        [ 0.29876634 -1.73615948].       -26.988399128696233    -7.990765314336182
    8        [-4.49663156  2.25072382].       -55.32869213523205     -7.990765314336182
    9        [-0.6354838   2.22969794].       -30.693519656466428    -7.990765314336182
    10       [-3.14028177 -5.00531997].       -38.559595068237925    -7.990765314336182
    11       [4.35286877 2.19708583].         -46.53358510860255     -7.990765314336182
    12       [ 2.14326382 -0.43652095].       -27.784309763291844    -7.990765314336182
    13       [-1.33154135  4.92630338].       -41.99669838150662     -7.990765314336182
    14       [-1.27347191 -3.25498348].       -33.999161616155384    -7.990765314336182
    15       [ 1.1820377  -4.83194368].       -35.67909945970859     -7.990765314336182
    16       [ 3.65779494 -5.02638532].       -54.25585029088287     -7.990765314336182
    17       [2.54820992 3.54309845].         -58.22722367638867     -7.990765314336182
    18       [ 5.10684557 -1.7214604 ].       -42.996951321010485    -7.990765314336182
    19       [-5.0091895  -4.37129392].       -61.12136353809342     -7.990765314336182
    20       [-0.64469737 -0.66885139].       -31.887734301035515    -7.990765314336182
```

```python
### 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     [-2.83349935  0.23719262].       -22.27210456874814     -22.27210456874814
    init     [ 0.51918292 -4.65303603].       -57.57021076017139     -22.27210456874814
    init     [-1.42613673 -2.83565116].       -33.89145899403749     -22.27210456874814
    init     [ 1.9325559  -3.44339021].       -35.85029586225333     -22.27210456874814
    init     [-4.39987336  4.51595121].       -77.78800881964571     -22.27210456874814
    1        [4.06080895 4.40655317].         -54.95421034473918     -22.27210456874814
    2        [-5.11995443 -4.68844075].       -64.67592250239862     -22.27210456874814
    3        [4.12206639 0.06030118].         -20.504089236344484    -20.504089236344484
    4        [-0.03509414  4.1160457 ].       -19.727689166446165    -19.727689166446165
    5        [ 5.114123   -4.06066837].       -45.82362087025153     -19.727689166446165
    6        [1.067671   0.75885408].         -12.050143088129088    -12.050143088129088
    7        [-4.8151146  -0.94280278].       -30.734971672906735    -12.050143088129088
    8        [-1.08364727  2.23789723].       -16.772393777505908    -12.050143088129088
    9        [2.29272256 2.57546838].         -43.438496776294095    -12.050143088129088
    10       [ 0.70788506 -1.07237015].       -15.28260299623733     -12.050143088129088
    11       [-4.43655313  1.38642616].       -58.38111369531196     -12.050143088129088
    12       [-2.74014071 -4.77533768].       -49.34600536042029     -12.050143088129088
    13       [-2.18903166  4.72133595].       -45.136420238311864    -12.050143088129088
    14       [ 5.09216366 -1.25194212].       -39.24985633508995     -12.050143088129088
    15       [-1.0378687   0.17694279].       -6.9594099744550375    -6.9594099744550375
    16       [-3.92887994 -2.44184673].       -41.72048353372222     -6.9594099744550375
    17       [1.3475999 5.1161228].           -46.29165412396723     -6.9594099744550375
    18       [ 1.93548231 -1.06180449].       -6.428550755028452     -6.428550755028452
    19       [4.71531302 1.8709138 ].         -41.00951744229297     -6.428550755028452
    20       [ 2.0497107  -0.54349293].       -24.609456140694668    -6.428550755028452
```

```python
### EXACT GP EI GRADIENTS

np.random.seed(run_num_17)
```

```
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     [-2.10263037  0.31320838].      -20.395145364684023    -20.395145364684023
init     [-3.15882714 -4.42470033].      -53.03732051200137     -20.395145364684023
init     [2.93873111 1.60085526].        -29.989224812583537    -20.395145364684023
init     [1.40821398 0.77417363].        -29.451989415882437    -20.395145364684023
init     [-4.71999574 -1.45598869].      -55.89242173757483     -20.395145364684023
1        [ 4.96008736 -4.44720822].      -64.14783177084814     -20.395145364684023
2        [-4.48458015  4.87453914].      -66.7751306072647      -20.395145364684023
3        [0.68918634 4.75681217].        -46.403080170128334    -20.395145364684023
4        [4.92477561 4.97364465].        -50.22369669128301     -20.395145364684023
5        [ 0.92966222 -4.61914041].      -40.48806820146371     -20.395145364684023
6        [ 4.79588687 -1.08221629].      -32.63321099942499     -20.395145364684023
7        [-2.20574447  2.92472705].      -21.77210909560035     -20.395145364684023
8        [ 1.59821581 -2.01953576].      -24.86361094180775     -20.395145364684023
9        [-0.97041093 -1.73567117].      -15.025664805835008    -15.025664805835008
10       [-4.82154921  1.60543049].      -49.36402141551662     -15.025664805835008
11       [4.80003902 1.052409  ].        -31.592733600635157    -15.025664805835008
12       [0.11942379 2.3153242 ].        -22.050731500846105    -15.025664805835008
13       [3.10171227 4.31089192].        -43.911234780118356    -15.025664805835008
14       [-5.01335854 -4.35057553].      -60.003544944852315    -15.025664805835008
15       [-1.48546923  5.04244211].      -47.944658933536985    -15.025664805835008
16       [-1.1828003  -3.49033229].      -39.465053136266384    -15.025664805835008
17       [-0.32976851 -0.48736285].      -35.11954930011615     -15.025664805835008
18       [ 3.00950416 -4.69886968].      -44.311952056203545    -15.025664805835008
19       [ 3.3111116  -1.78432043].      -35.75361177200671     -15.025664805835008
20       [-2.8447112  -1.23221407].      -22.889718122727896    -15.025664805835008
```

```
### 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     [1.53983224 0.05584255].        -22.67190580753611     -22.67190580753611
init     [ 3.87687906 -3.25795609].      -38.990099416711985    -22.67190580753611
init     [3.60686662 2.56139557].        -56.66448698782129     -22.67190580753611
init     [1.70088108 4.99604939].        -40.894059318256296    -22.67190580753611
init     [-2.48864335 -4.83014733].      -54.6725749848372      -22.67190580753611
1        [-5.00718904  2.9174042 ].      -34.91005808676357     -22.67190580753611
2        [-2.99572465 -0.54340826].      -28.90362514523323     -22.67190580753611
3        [-1.26375514  2.97482686].      -21.434681387522748    -21.434681387522748
4        [ 1.15010421 -4.55177438].      -45.64435873691351     -21.434681387522748
5        [-5.01779201 -3.225694  ].      -44.12449775607688     -21.434681387522748
6        [-0.43969771 -2.28090657].      -36.61635977988647     -21.434681387522748
7        [5.06768234 0.33261668].        -41.6436796565958      -21.434681387522748
8        [4.84471484 4.90128037].        -53.750611251185354    -21.434681387522748
9        [-3.63865114  4.5345895 ].      -70.0063579582363      -21.434681387522748
10       [0.1294679 1.204589 ].          -11.783292362366176    -11.783292362366176
```

```
11       [-4.75628954  0.32958045].       -47.13025261167572      -11.783292362366176
12       [ 4.26756046 -5.04406482].       -55.13661721500662      -11.783292362366176
13       [ 2.22094431 -2.09730848].       -19.327387184001488     -11.783292362366176
14       [-1.10508167  4.9870455 ].       -28.22653216153722      -11.783292362366176
15       [-1.18914502  0.28137564].       -19.720743548177467     -11.783292362366176
16       [0.89354652 2.04221127].         -7.473459291543399      -7.473459291543399
17       [-2.60833948 -2.74606128].       -42.36295583914121      -7.473459291543399
18       [-2.75615379  1.65352663].       -35.641134641437745     -7.473459291543399
19       [0.53187661 3.01969957].         -29.27807883705011З     -7.473459291543399
20       [ 2.87441501 -0.43159573].       -30.494011686301285     -7.473459291543399
```

```
### 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     [-4.12125592  2.6751971 ].       -41.43485596167127      -25.867003842388073
init     [-2.59135515 -3.70553152].       -51.60126035043229      -25.867003842388073
init     [-1.72598719 -4.27008445].       -43.97430127029199      -25.867003842388073
init     [1.76104531 3.13952049].         -25.867003842388073     -25.867003842388073
init     [4.9432772  1.38916592].         -44.66580306903559      -25.867003842388073
1        [ 4.21710172 -4.56788076].       -65.70127216515351      -25.867003842388073
2        [ 1.33148367 -1.37898766].       -35.81946957386545      -25.867003842388073
3        [-4.47356312 -0.83164219].       -45.659055854674875     -25.867003842388073
4        [-1.70726049  4.62995106].       -53.85214051685652      -25.867003842388073
5        [-1.1560595   0.80460077].       -13.054438144741923     -13.054438144741923
6        [4.33309282 4.60431349].         -72.89021732044269      -13.054438144741923
7        [ 1.5490079  -4.78091704].       -52.855855758060706     -13.054438144741923
8        [ 3.89443714 -0.76552688].       -26.898811484711675     -13.054438144741923
9        [-4.99722686  4.77192639].       -56.37175387068849      -13.054438144741923
10       [-4.84280999 -3.83232357].       -47.68806114025933      -13.054438144741923
11       [1.92748351 0.90273708].         -7.360253484560014      -7.360253484560014
12       [-0.69529354 -1.12264264].       -17.938756616211073     -7.360253484560014
13       [0.10402916 1.78829826].         -12.886848044881196     -7.360253484560014
14       [1.50220491 4.78266425].         -53.09155995105475      -7.360253484560014
15       [-2.8301877   0.41341393].       -31.909270093313516     -7.360253484560014
16       [ 5.06188914 -2.74423977].       -44.26200525490793      -7.360253484560014
17       [-1.12053562  2.06249008].       -9.003824290781543      -7.360253484560014
18       [2.49727064 0.69529328].         -40.088358724884884     -7.360253484560014
19       [-0.38704907 -2.58186341].       -43.10751723009094      -7.360253484560014
20       [0.58929849 0.95864954].         -20.068715280426684     -7.360253484560014
```

```
### 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     [0.9024594  4.07258857].        -20.24255447774017     -17.388691338534382
init     [4.00927467 3.23417577].        -35.55852254546393     -17.388691338534382
init     [-4.75249064  1.96359764].      -36.54582989269194     -17.388691338534382
init     [-1.24230715  0.18955208].      -17.388691338534382    -17.388691338534382
init     [ 1.61742301 -3.13497377].      -33.22932870179905     -17.388691338534382
1        [-5.04038248 -4.33114855].      -59.36512603389513     -17.388691338534382
2        [-2.96489388  5.04868064].      -34.98620614086819     -17.388691338534382
3        [ 5.11316294 -4.88092784].      -55.06148040152972     -17.388691338534382
4        [ 3.68851601 -0.58351791].      -46.36787148871594     -17.388691338534382
5        [-1.62883463 -4.26048482].      -48.36180151832993     -17.388691338534382
6        [-4.98848541 -0.90526799].      -27.450398333143248    -17.388691338534382
7        [0.93645439 1.02242281].        -2.8079216448068234    -2.8079216448068234
8        [-1.06609337  2.48258368].      -28.089915626264425    -2.8079216448068234
9        [-2.5351291  -1.74802429].      -39.36399797660107     -2.8079216448068234
10       [ 0.8545737  -0.41008551].      -23.236739034957935    -2.8079216448068234
11       [1.67686636 1.62961902].        -36.76554998753772     -2.8079216448068234
12       [ 2.06952897 -5.12      ].      -34.146831784777255    -2.8079216448068234
13       [-5.12        4.38793399].      -65.80041903280606     -2.8079216448068234
14       [2.52507625 5.08432136].        -53.47341881184989     -2.8079216448068234
15       [4.95325427 1.6352905 ].        -44.23656112704768     -2.8079216448068234
16       [-0.63220912  4.85767325].      -44.479391014623005    -2.8079216448068234
17       [4.83010635 4.98828829].        -53.41661893108656     -2.8079216448068234
18       [ 4.02218806 -2.67672933].      -37.882722340183825    -2.8079216448068234
19       [-2.93959251  1.04239   ].      -10.792076232574544    -2.8079216448068234
20       [ 0.22909672 -1.93232082].      -13.367335583846376    -2.8079216448068234
```

```
end_exact = time.time()
end_exact

time_exact = end_exact - start_exact
time_exact
```

```
82.01042890548706
```

```
### 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
```

```
(2.5234887430171615, 2.116479226101828)
```

```
### 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
```

```
(3.1089279907189096, 2.442537715098709)
```

```
### 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
```

```
(2.6108506059478387, 2.064286267887516)
```

```
### 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
```

```
(2.288170828405886, 1.8083908923838952)
```

```
### 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
```

```
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
```

```
(2.034576678342154, 2.6985808939996834)
```

```
### 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
```

```
(2.3114825426797987, 3.34424594224657)
```

```
### 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
```

```
(2.8264842567634267, 2.8521455348071294)
```

```
### 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
```

```
(2.6984850096526367, 2.6257495285396604)
```

```
### 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
```

```
(2.39474253794569, 2.4746054894126215)
```

```
### 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
```

```
(2.5241871672711627, 2.2410773013304173)
```

```
### 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
```

```
(2.489149658474126, 2.7029938766754724)
```

```
### 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
```

```
(2.1667099483116363, 2.383711149601142)
```

```
### 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
```

```
(2.763711012992627, 2.1205418514449623)
```

```
### 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
```

```
(2.546375238661026, 2.0821257819171355)
```

```
### 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
```

```
(2.355212606187029, 3.2370678736916036)
```

```
### 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
```

```
(0.7593924197776527, 1.8607491248253725)
```

```
### 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
```

```
(2.400153680851681, 2.70975972608701)
```

```
### 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
```

```
(2.70458168750813, 2.011357983188474)
```

```
### 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
```

```
(2.929231041510643, 1.9960943729846712)
```

```
### 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
```

```
(2.641971206249818, 1.0324445815031589)
```

```
# 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_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_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_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_11[slice13],
          simple_regret_approx_12[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_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],
```

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

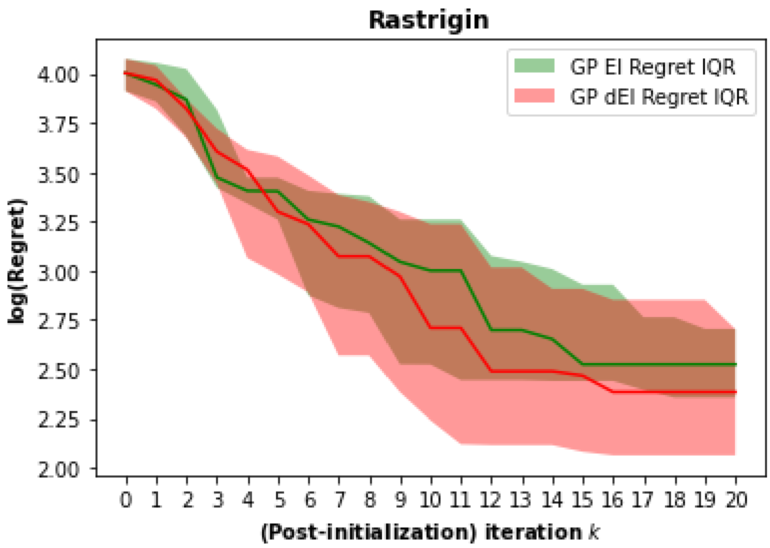
```
### Summarize arrays: 'exact'

lower_exact = [lower_exact1,
```

```
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_exact12,
            upper_exact13,
            upper_exact14,
            upper_exact15,
            upper_exact16,
            upper_exact17,
            upper_exact18,
            upper_exact19,
            upper_exact20,
            upper_exact21]
```

```
### Visualize!

title = 'Rastrigin'

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
```

```
(814.9613211154938, 82.01042890548706)
```

```
1
```