Rosenbrock:

GP EI: derivation of exact partial-order GP EI derivatives wrt **x1**, **x2**, **x3**, **x4**

```
#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
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 = 40
opt = True
```

```
### Objective Function - Rosenbrock(x) 4-D:

def objfunc(x1_training, x2_training, x3_training, x4_training):
            return  operator * (
                100 * (x2_training - x1_training ** 2) ** 2 + (x1_training - 1) ** 2 +
```

```
                100 * (x3_training - x2_training ** 2) ** 2 + (x2_training - 1) ** 2 +
                100 * (x4_training - x3_training ** 2) ** 2 + (x3_training - 1) ** 2
                )

# Constraints:
lb = -2.048
ub = +2.048

# Input array dimension(s):
dim = 4

# 4-D inputs' parameter bounds:
param = {'x1_training': ('cont', [lb, ub]),
                 'x2_training': ('cont', [lb, ub]),
                 'x3_training': ('cont', [lb, ub]),
                 'x4_training': ('cont', [lb, ub])}

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

# Test data:
x1_test = np.linspace(lb, ub, n_test)
x2_test = np.linspace(lb, ub, n_test)
x3_test = np.linspace(lb, ub, n_test)
x4_test = np.linspace(lb, ub, n_test)
Xstar = np.column_stack((x1_test, x2_test, x3_test, x4_test))

Xstar_d = np.column_stack((x1_test, x2_test, x3_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)))
```

```
        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)
        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

```

```
1623407545.547943
```

```
### 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     [-0.33987787  0.90244913 -2.04753152 -0.80964578].       -3394.123441866637
init     [-1.44688787 -1.66978112 -1.28507817 -0.63258326].       -3615.357417364757
init     [-0.42284043  0.15899334 -0.33097927  0.75865907].       -59.369079192709535
init     [-1.21056359  1.54876902 -1.93582042  0.69823492].       -2823.036073483468
init     [-0.33871953  0.24039354 -1.4729751  -1.2365763 ].       -1404.6293821132788
1        [-0.0490961   0.0026703   0.52114175  1.27606514].       -130.3799970253568
2        [ 2.048 -2.048  2.048 -2.048].           -8265.420843724798     -59.369079192709535
3        [-2.048  2.048  2.048 -2.048].           -4829.447006924799     -59.369079192709535
4        [ 2.048 -2.048 -2.048  2.048].           -8273.612843724799     -59.369079192709535
5        [2.048 2.048 2.048 2.048].       -1385.2811701248       -59.369079192709535
6        [-2.048 -2.048  2.048  2.048].           -4837.639006924799     -59.369079192709535
7        [ 2.048       2.048       0.89949977 -2.048     ].       -2364.74423245248
8        [-2.048  2.048  2.048  2.048].           -1393.4731701248       -59.369079192709535
9        [ 2.048 -2.048  2.048  2.048].           -4829.447006924799     -59.369079192709535
10       [ 2.048 -2.048 -2.048 -2.048].           -11709.586680524799    -59.369079192709535
11       [-0.04486943  0.26245645  1.33015464  0.00220004].       -479.8765479489312
12       [ 2.048  2.048 -2.048  2.048].           -4829.447006924799     -59.369079192709535
13       [-2.048 -2.048  2.048 -2.048].           -8273.612843724797     -59.369079192709535
14       [0.39277217 0.66508874 0.36925183 0.93955179].           -92.02035347587906
15       [-0.3665318  -0.09024272 -0.1746368   2.048     ].       -419.8520995192605
16       [0.63988049 1.57185141 0.14918416 0.78089091].           -732.8030854206354
17       [ 1.59540661 -0.05364952  0.14712847  1.38299872].       -865.0662183090815
```

18	[1.92664281 1.32621188 -2.02981529 -1.67272649].	-5370.451989240503	
19	[-1.38940861 -0.20044123 0.5322671 0.92704531].	-527.0965669464073	
20	[1.89620782 -0.14839589 -1.53922942 0.68600274].	-1937.398128539917	
21	[0.75241121 -0.58888911 0.39295842 1.31362989].	-270.9503393195635	
22	[0.08259405 1.94781598 1.35341898 -1.16334885].	-1871.305496177096	
23	[1.8307373 0.45568333 -1.22143559 -0.95524121].	-1647.634169421071	
24	[-0.06298569 0.3710531 1.78740995 1.56979854].	-551.8556278894023	
25	[0.79734412 -0.83463689 -1.80414999 -1.60545318].	-3215.219531243629	
26	[-0.1582911 -0.68994974 1.6102458 -1.19194178].	-1616.834034875373	
27	[-0.22926075 1.4608557 1.28246287 -1.03341029].	-989.8946435008918	
28	[2.0062999 -1.50296352 1.08688969 -0.29540852].	-3418.823764054943	
29	[-0.79647557 0.5219737 -0.52585479 -1.74939235].	-481.2109398861377	
30	[-1.67298376 -0.62270762 -1.80623959 1.82093967].	-1877.551098909314	
31	[1.55656109 -0.57199926 0.1756522 1.30094652].	-1064.001858558137	
32	[-1.98935079 1.82470542 0.7078706 -0.53078398].	-1258.384739357216	
33	[-0.06966701 0.83377973 -0.92724163 1.71662838].	-410.2453655200145	
34	[0.56787983 -0.70452346 0.6459641 -2.00969127].	-699.9448688633819	
35	[0.49704515 0.93111689 1.6178312 0.1594768].	-707.9393288226192	
36	[0.87715693 -0.5909966 -1.62091187 -1.99652745].	-2720.678153604498	
37	[-1.08501802 1.32287606 -1.67441492 1.95053983].	-1259.169117865957	
38	[-1.73681667 0.33276035 0.5322298 -0.61463567].	-826.8067343838653	
39	[1.69661485 -1.31722849 -1.50498054 0.6533503].	-3082.082077592921	
40	[-0.38001736 -1.73073496 1.72122227 -1.51184018].	-2525.930515971026	

```
1 ### ESTIMATED GP EI GRADIENTS
2
3 np.random.seed(run_num_2)
4 surrogate_approx_2 = GaussianProcess(cov_func, optimize=opt)
5
6 approx_2 = GPGO(surrogate_approx_2, Acquisition(util_grad_approx), objfunc, param)
7 approx_2.run(init_evals=n_init, max_iter=iters)
8
```

Evaluation	Proposed point	Current eval.	Best eval.
init	[-0.26216488 -1.94180615 0.20341751 -0.26491948].	-1696.976458203069	
init	[-0.32617348 -0.69494857 -1.20975919 0.48853388].	-455.3157381798458	
init	[-0.82061446 -0.95507548 0.49616418 0.11936602].	-291.5008468259235	
init	[-1.49676054 0.05561598 -1.29253431 1.16873277].	-682.7190038869136	
init	[1.4498828 -0.02360591 1.41951584 -1.72177213].	-2051.02408589455	
1	[-1.86778037 0.18049011 2.00280848 0.51009437].	-2718.247471569953	
2	[2.048 2.048 -2.048 2.048].	-4829.447006924799	-291.5008468
3	[-2.048 2.048 -2.048 -2.048].	-8273.612843724799	-291.5008468
4	[2.048 2.048 2.048 2.048].	-1385.2811701248	-291.50084682592353
5	[2.048 -2.048 2.048 2.048].	-4829.447006924799	-291.5008468
6	[2.048 -2.048 -2.048 -2.048].	-11709.586680524799	-291.5008468
7	[2.048 2.048 -2.048 -2.048].	-8265.4208437248	-291.5008468
8	[2.048 -2.048 -2.048 2.048].	-8273.612843724803	-291.5008468
9	[-2.048 -2.048 -2.048 -2.048].	-11717.778680524798	-291.5008468
10	[-2.048 2.048 2.048 -2.048].	-4829.447006924799	-291.5008468
11	[-2.048 -2.048 2.048 -2.048].	-8273.612843724797	-291.5008468
12	[-2.048 -2.048 -2.048 2.048].	-8281.8048437248	-291.5008468
13	[-2.048 2.048 -2.048 2.048].	-4837.639006924799	-291.5008468
14	[0.18190488 2.048 0.3253766 0.25448714].	-1907.277812219361	
15	[-2.048 -2.048 2.048 2.048].	-4837.639006924799	-291.5008468
16	[-2.048 2.048 1.32278726 2.048].	-1304.610065264430	
17	[0.3123691 -0.03488486 0.27467449 2.048].	-400.3992688360985	
18	[2.048 2.048 2.048 -2.048].	-4821.255006924799	-291.5008468
19	[2.048 -2.048 2.048 -2.048].	-8265.420843724798	-291.5008468

```
20       [ 2.048       0.05069028 -0.01457087  0.46404581].       -1741.525845293071
21       [ 0.04462302  0.26846403 -2.048      -0.09134914].       -2303.992180749520
22       [0.27344559 0.455541   2.048      2.048     ].          -815.8207189943965
23       [-0.54044064  0.20470646  0.04701984 -2.048     ].       -425.0160411278497
24       [-2.048      -0.02157205 -0.3367943  -0.11652646].       -1806.144440287191
25       [-1.24663827 -0.00408505 -0.43085572  1.90844862].       -566.2723726921901
26       [-0.14563468  1.84388246 -0.275985   -2.02519151].       -2128.652409333823
27       [ 0.51129315 -0.04674546  1.82266245 -0.02252549].       -1461.572585880421
28       [-0.03619105 -0.9793058   0.0457563   2.048     ].       -604.0445860660286
29       [-0.38043746 -0.47436586 -0.14851725 -1.90468684].       -428.9143078391985
30       [ 1.82143212  0.73311298 -2.04727183 -1.88527067].       -5038.575769519586
31       [-0.47465946  0.42283806 -0.85595954 -1.68383769].       -700.8748323801497
32       [-0.68527262 -0.58563323  1.51289546  0.58564071].       -543.9353731860382
33       [0.55223936 0.46147771 1.27238896 0.21156894].           -313.3316330394328
34       [ 1.32173857  2.04339184 -0.29528961 -2.00670871].       -2448.850338994838
35       [-0.34823756 -0.05469126  0.43322284  1.88093636].       -311.5685252340465
36       [-0.03706493  0.9732862  -0.44071488  1.85148322].       -564.9168528854626
37       [ 1.28905706  0.32105178 -1.58817841 -1.93695353].       -2461.505686335076
38       [-1.87234884 -1.86175433  1.40728674 -0.66191861].       -4019.649521955219
39       [-1.75988164  0.58497024  1.75923584  2.00126065].       -959.8960097161256
40       [-0.05047861  0.37549186  0.96577463  1.93190576].       -183.26676877682718
```

```
1 ### ESTIMATED GP EI GRADIENTS
2
3 np.random.seed(run_num_3)
4 surrogate_approx_3 = GaussianProcess(cov_func, optimize=opt)
5
6 approx_3 = GPGO(surrogate_approx_3, Acquisition(util_grad_approx), objfunc, param)
7 approx_3.run(init_evals=n_init, max_iter=iters)
8
```

```
Evaluation       Proposed point           Current eval.          Best eval.
init     [ 0.20806821  0.85257348 -0.85645419  0.04434987].       -367.7787956485136
init     [ 1.60951073  1.62321649 -1.53360257 -1.19913317].       -3099.341888277455
init     [-1.83719034 -0.24244288 -1.92562704 -0.17681111].       -3229.929027395333
init     [ 0.61089402 -0.90731609  0.72194008  0.3721741 ].       -171.0831542888061
init     [-1.94977021  0.24106634 -0.98610198 -0.3477455 ].       -1564.280611971214
1        [-2.048  2.048  2.048  2.048].           -1393.4731701248       -171.083154
2        [-2.048 -2.048  2.048 -2.048].           -8273.612843724797     -171.083154
3        [ 2.048 -2.048 -2.048  2.048].           -8273.612843724799     -171.083154
4        [2.048 2.048 2.048 2.048].       -1385.2811701248       -171.0831542888061
5        [-0.35345106  2.048       2.048      -2.048     ].       -4731.147173102215
6        [ 2.048      -2.048       0.13981966 -2.048     ].       -5979.124878992427
7        [-2.048 -2.048  2.048  2.048].           -4837.639006924799     -171.083154
8        [ 2.048 -2.048  2.048  2.048].           -4829.447006924799     -171.083154
9        [ 0.05316528  2.048      -2.048       2.048     ].       -4786.856558033743
10       [-2.048  2.048 -2.048 -2.048].           -8273.612843724799     -171.083154
11       [-2.048 -2.048 -2.048  2.048].           -8281.8048437248       -171.083154
12       [ 2.048       0.42594084  2.048      -0.53999331].       -4012.349349721207
13       [-0.04134587  0.1948249   0.61106596  2.048     ].       -318.8867289083563
14       [-0.4678158 -2.048     -2.048     -2.048    ].           -8327.868564032846
15       [-2.048       0.53467127  2.048       0.01790337].       -3404.637565487796
16       [ 2.048       0.71210795 -0.20409272  2.048     ].       -1668.321634095314
17       [-0.03729572 -2.048       0.33790576  0.40183461].       -1926.260232663489
18       [-2.048       1.10706136 -0.28643562  2.048     ].       -1579.181748752010
19       [ 0.25492497 -0.07302655  0.71993566  1.23744619].       -106.47150583456444
20       [ 1.49239847  0.49268641  1.60406172 -0.66830816].       -1537.673541011973
21       [0.48719489 1.84629116 0.39103139 0.64259933].           -1194.883662581495
```

```
22       [ 1.2299417  -1.91096248  1.4133729   1.99191587].       -1681.931499110462
23       [ 0.00550859 -0.87056605  1.11029903  1.51331117].       -100.5842275200595
24       [-0.06327872 -0.49098943  1.10261026  1.83591466].       -140.55156137310982
25       [-1.92125593  0.53080971 -1.93881188  1.72743047].       -1922.030053518997
26       [1.4605204  1.90918656 1.41415443 0.55762362].           -711.8857692267823
27       [ 1.0047818  -0.63259091 -1.96439414 -1.85587058].       -4106.041253375681
28       [ 0.98492504  1.17178375 -0.22453846 -1.97200959].       -669.856486331395
29       [ 1.10614155  0.98623714  1.95858435 -0.70898671].       -2169.50428747307
30       [ 0.94417659 -2.0345752   0.21589641 -1.32198484].       -2592.7693977126632
31       [ 1.69980461  1.46618113 -0.57797751  1.0150917 ].       -996.1300770680594
32       [-0.7850228   1.3368284   1.82024652  0.72471013].       -726.0825390658513
33       [-0.90364346 -1.40881445 -0.01203534 -1.32845648].       -1080.9212981095382
34       [ 1.23334546 -0.45172328  0.65487468 -1.43597748].       -759.5865546206971
35       [0.74524341 1.03355023 1.41376536 0.64124483].           -219.31817541039308
36       [-1.76099243  0.62045344 -0.09891249  1.63585396].       -912.1566142299126
37       [ 0.74426082 -0.067332   -0.27796404 -1.07523988].       -182.24072574245946
38       [0.1481922  1.19051691 2.038487   1.78777272].           -737.5559124439973
39       [-0.29245864 -0.98054645 -0.12680832 -0.08144759].       -239.90148247899936
40       [ 0.94174472  1.35912663  0.63783186 -1.03478984].       -376.65477667517473
```

```
1 ### ESTIMATED GP EI GRADIENTS
2
3 np.random.seed(run_num_4)
4 surrogate_approx_4 = GaussianProcess(cov_func, optimize=opt)
5
6 approx_4 = GPGO(surrogate_approx_4, Acquisition(util_grad_approx), objfunc, param)
7 approx_4.run(init_evals=n_init, max_iter=iters)
8
```

```
Evaluation       Proposed point           Current eval.          Best eval.
init     [1.91295422 0.19346329 1.93611514 0.87988631].           -2387.047477734054
init     [ 0.80989727 -1.16289743  1.95082017 -2.02248087].       -3769.0203854501506
init     [-1.01178424 -0.26709388  1.14435245 -1.23828193].       -936.5671269056027
init     [ 1.48682029  1.98000917 -1.37690218  0.39867983].       -3042.500138281922
init     [-2.01119294 -0.46460403 -1.8671204   1.87045056].       -2747.9169599921197
1        [-2.048  2.048  2.048  2.048].           -1393.4731701248       -936.5671269
2        [ 2.048 -2.048 -2.048  2.048].           -8273.612843724799     -936.5671269
3        [-2.048  2.048 -2.048 -2.048].           -8273.612843724799     -936.5671269
4        [-2.048 -2.048  2.048  2.048].           -4837.639006924799     -936.5671269
5        [ 0.01575257 -2.048      -2.048      -2.048     ].       -8232.353242429861
6        [ 2.048  2.048  2.048 -2.048].           -4821.255006924799     -936.5671269
7        [-2.048  2.048  2.048 -2.048].           -4829.447006924799     -936.5671269
8        [ 2.048       0.73200004 -2.048      -2.048     ].       -5773.465902750279
9        [-1.09601054  2.048      -0.67176639  2.048     ].       -2702.80551795556
10       [-2.048 -2.048  2.048 -2.048].           -8273.612843724797     -936.5671269
11       [ 2.048 -2.048  2.048  2.048].           -4829.447006924799     -936.5671269
12       [0.66116829 2.048      2.048      2.048     ].           -1183.1214514734775
13       [-0.28151045 -2.048      -0.05260156  0.59319317].       -2303.0400014507027
14       [-2.048       0.44849204  0.40482431  0.17278744].       -1417.2158768631639
15       [ 2.048      -2.048      -0.04026037 -0.76597681].       -5760.180837294365
16       [-2.048      -2.048      -2.048      -0.04886604].       -9621.59195594188
17       [ 2.048       0.59387418 -0.27488977  2.048     ].       -1727.6335017846045
18       [-0.03004513  2.048       0.91355985 -0.24276783].       -1613.6260512468998
19       [-0.45086815  0.12775493  2.048       2.048     ].       -877.9684801032818
20       [-0.15138668  0.64645298 -0.09549713 -2.048     ].       -491.0623555759303
21       [ 0.10188111  0.16515724 -2.048       0.48456236].       -1820.0852185987146
22       [-2.048      -0.33483485 -0.37828314 -2.048     ].       -2568.421703134025
23       [ 0.33784187  2.048      -0.22811678 -2.048     ].       -2773.8237975556917
```

```
24       [-0.7049028  -0.01108954  2.048       0.14702181].       -2088.260756294008
25       [ 2.048  2.048 -2.048  2.048].           -4829.447006924799     -491.062355
26       [-0.07877363 -0.32061305  0.12502611  2.048     ].       -427.455965971639
27       [-0.07930562 -0.31974969  0.12833224  2.048     ].       -427.0764268057542
28       [-1.83504592  0.57272321  0.36396613  1.78672103].       -1063.4262741118819
29       [ 0.24012772 -0.73428516  0.54309972  1.94719065].       -339.5008764584295
30       [-1.22276768  0.7543545  -0.37217706 -1.22703467].       -336.8270745465934
31       [ 1.32638545  1.39105073  1.36951294 -1.83395094].       -1421.9880420840564
32       [2.04198784 1.97328923 1.95800832 0.59914916].           -1906.4297312958206
33       [-0.87122222  2.04692871  0.45878489 -0.82666467].       -1670.4616530704325
34       [-0.33218589  0.75505643  0.15995792 -1.25506911].       -224.93573110971306
35       [ 0.53907861  0.80105769 -0.13294576  1.03329548].       -190.74687989954148
36       [ 0.60276928  1.49878963  0.26705322 -1.09257692].       -657.1050982695266
37       [1.17707935 1.45523333 0.8341324  0.29449966].           -181.61011713003785
38       [-0.08973767  1.69880134 -0.38553909 -0.15508577].       -1368.9319491647348
39       [ 0.74821824  1.78306464 -1.78430824  0.91549893].       -3136.318146218995
40       [ 1.01737435 -1.68893479  1.62333127 -1.74129535].       -2816.089596716538
```

```
1 ### ESTIMATED GP EI GRADIENTS
2
3 np.random.seed(run_num_5)
4 surrogate_approx_5 = GaussianProcess(cov_func, optimize=opt)
5
6 approx_5 = GPGO(surrogate_approx_5, Acquisition(util_grad_approx), objfunc, param)
7 approx_5.run(init_evals=n_init, max_iter=iters)
8
```

```
Evaluation       Proposed point           Current eval.          Best eval.
init     [-1.13871597  1.51851953 -1.20127834  1.71463028].       -1252.0157320257063
init     [-0.04746777  0.45770286  1.08915858  0.07544008].       -222.91752416760144
init     [-0.83230515 -1.27909385 -1.71728376  0.97665145].       -1918.291410223084
init     [-0.24039742 -1.39956278  1.55622208 -0.92534185].       -1356.5528565009568
init     [-0.35129336 -0.8352566   0.52751527  0.32701567].       -100.45329155979191
1        [ 2.048  2.048 -2.048 -2.048].           -8265.4208437248       -100.4532915
2        [ 2.048      -2.048       1.01658064  2.048     ].       -5019.751351509129
3        [-2.048  2.048  2.048 -2.048].           -4829.447006924799     -100.4532915
4        [2.048 2.048 2.048 2.048].       -1385.2811701248       -100.45329155979191
5        [-2.048      -0.78793203  2.048       2.048     ].       -3160.1944342868937
6        [ 2.048 -2.048 -2.048 -2.048].           -11709.586680524799    -100.4532915
7        [-2.048     0.874058 -2.048    -2.048   ].       -5808.357426017814     -100
8        [ 2.048  2.048  2.048 -2.048].           -4821.255006924799     -100.4532915
9        [ 2.048  2.048 -2.048  2.048].           -4829.447006924799     -100.4532915
10       [-2.048     -2.048     -0.3237166 -2.048    ].           -6421.671297259764
11       [ 2.048 -2.048  2.048 -2.048].           -8265.420843724798     -100.4532915
12       [-1.07641781  2.048       2.048       2.048     ].       -1006.9221316360523
13       [2.048      0.33814809 0.06721482 0.12593329].           -1491.0965688074964
14       [-0.45247217  1.96602508 -0.63840851 -0.19789703].       -2380.899556956846
15       [ 1.80499795  0.05340004 -1.61819437  0.9156279 ].       -1588.1307824603793
16       [-0.18792405 -0.41670073  0.6961813   1.12251394].       -91.93205825222469
17       [-2.00979377  1.45046548  1.91636392  1.91286092].       -993.4236305943057
18       [0.42690547 0.44271243 1.85937242 1.71486174].           -588.4415514651023
19       [ 0.6927607   0.97949297 -1.32818922  1.7809417 ].       -553.8106806602497
20       [ 0.86866932  0.5076431  -1.82363683 -0.78598627].       -2138.083968068359
21       [ 1.94994465 -0.35355171 -1.53741326 -1.48909156].       -3496.9851843806637
22       [ 0.1473329  -1.7247022  -1.41938966  0.59671559].       -2450.7700890745095
23       [-1.37505296 -1.32512161  1.12550758 -1.87113142].       -2069.6453850731227
24       [0.75502349 0.74256749 1.84303871 0.46968966].           -1027.436879763098
25       [ 1.32475783 -0.8915516  -0.21088425 -0.78154481].       -874.9477914351841
```

```
26       [ 1.15487158 -0.834299    1.05481692  0.2559162 ].       -559.694281885498
27       [-0.3107496   1.66980759 -0.84551918  0.41760558].       -1582.353544675686
28       [-1.07255585 -0.11466322 -1.92503864  1.03190207].       -1264.7418425813808
29       [ 1.28272724 -1.39964435 -0.12724209 -1.20591634].       -1518.928737142175
30       [-1.20365786  0.31174043  0.0297163   1.38762166].       -328.3195031335746
31       [-1.6109904   0.8590569   0.45648285 -1.77930314].       -711.5941755565317
32       [-0.13522102 -2.00678675  0.85543304  0.21808611].       -1452.834671435960
33       [-1.3997181   0.46429525 -0.96362403  1.3682138 ].       -391.7571103590943
34       [-0.61004237  1.06835669  0.87587592 -2.02996793].       -840.5237144476857
35       [-0.02002553 -0.03632827  0.21463406  1.80302799].       -316.1073483903574
36       [-0.31390776  0.79157762  1.3836017   1.46509244].       -127.43678379889838
37       [-1.38834774  1.34558751  1.64297275  0.83116248].       -391.9264398338271
38       [-1.00480104 -1.63478392  0.45813346 -1.68836468].       -1561.230607409748
39       [-0.03997327  1.87412011  1.35086415 -0.98530202].       -1609.480912063545
40       [-0.49490898 -0.61553645 -0.38107717  1.124754  ].       -234.496171179621
```

```
### 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     [ 1.60915518 -0.68821072  1.31575449 -1.87721062].       -2450.567714994702
init     [-1.60703824  0.38933325  0.12213192 -0.33256477].       -501.1308148976547
init     [-0.67416945  0.50183959 -0.25337272  0.96617311].       -111.72625934007624
init     [0.07387714 0.32300483 0.59537447 2.00795862].          -309.07782450458865
init     [ 1.31013917 -0.35552897  1.54119232  1.32611864].       -741.7784291682051
1        [ 2.048       2.048      -2.048      -0.16855141].       -6272.235654950075
2        [-2.048 -2.048 -2.048  2.048].           -8281.8048437248       -111.726259
3        [-0.31768625 -2.048      -2.048      -2.048     ].       -8275.376423859392
4        [-2.048 -2.048  2.048 -2.048].           -8273.612843724797     -111.726259
5        [ 2.048 -2.048 -2.048  2.048].           -8273.612843724799     -111.726259
6        [-2.048  2.048 -2.048  2.048].           -4837.639006924799     -111.726259
7        [-2.048  2.048  2.048  2.048].           -1393.4731701248       -111.726259
8        [-2.048  2.048 -2.048 -2.048].           -8273.612843724799     -111.726259
9        [-2.048 -2.048  2.048  2.048].           -4837.639006924799     -111.726259
10       [-0.89746264  2.048       2.048      -2.048     ].       -4517.490718018506
11       [2.048 2.048 2.048 2.048].       -1385.2811701248       -111.72625934007624
12       [ 2.048      2.048      2.048     -0.8927154].           -3512.395725008554
13       [ 2.048 -2.048  2.048  2.048].           -4829.447006924799     -111.726259
14       [-0.10283764  2.048       0.25893936  0.88721162].       -2033.949610524387
15       [ 0.28919478 -0.27590698 -0.320587   -0.29969978].       -48.740768932182
16       [ 0.03514236 -0.01962668 -1.27974723  0.16995172].       -386.5293127271378
17       [-0.1433454  -1.65451769  0.71253363 -0.09213704].       -735.0218045780138
18       [-0.71508768 -1.06298624  1.48028271 -0.75758851].       -1137.112846776038
19       [ 0.94214099 -1.57354766 -0.74644094  1.15478747].       -1689.576343784932
20       [1.8909089  0.28940522 0.85682419 1.06069847].           -1151.6121108178431
21       [ 1.89109499 -0.64451329  0.19246157  1.0232119 ].       -1887.8498992613063
22       [-1.48289855  1.9603708   0.88014879  1.04657615].       -898.0695517217156
23       [0.33873613 1.06893801 0.88105688 1.03917139].           -105.25928491494601
24       [-1.61922811  1.23481322  2.02363183  1.9101849 ].       -702.6303179178191
25       [ 0.15810568  1.97152737 -0.44674798 -1.31645352].       -2490.5484677296968
26       [-0.14395639  0.83083395  0.48969449 -1.25888407].       -295.85482689009353
27       [-1.04837812 -1.02335261  0.62386577  1.73596787].       -658.211927766007
```

```
28       [-0.95422001 -0.54737772 -0.44854888 -1.1889234 ].       -470.0820820835665
29       [ 1.96181723 -1.56127344  0.47070345 -0.43759404].       -3364.882590614902
30       [ 1.33981188  0.49177975 -1.7059087  -1.64398766].       -2630.928106036733
31       [-2.04650648  1.44371507 -1.98550922  1.9251675 ].       -2834.811304710729
32       [-1.06066992  1.23904023 -1.04353373 -0.39191513].       -894.0769169135324
33       [-0.35702929  1.09433763 -0.51713564 -1.56045156].       -725.7734812165357
34       [ 1.85564941 -0.68981499  0.2899382  -1.66918129].       -2023.3102026099407
35       [ 0.62057554  0.30413893  1.58954197 -1.24066722].       -1645.0079068362775
36       [ 0.78309714  1.36094253 -1.70864839  1.38494553].       -1566.8381453248671
37       [-0.17324013  1.34467199 -0.60828857  1.01354448].       -802.2419906375629
38       [-0.07967532 -0.56041942 -0.53123136 -0.08364451].       -122.90596687337347
39       [ 0.18867716  1.98328736 -1.20424388  1.44766517].       -3025.401390343327
40       [ 1.41638326 -0.57018461 -1.27636987 -0.21078604].       -1266.5657344000517
```

```
### 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     [-1.73544125  1.14654737 -0.25227579  0.91531337].       -675.0078283253545
init     [ 1.95784504  0.15767909  0.00458942 -1.75287856].       -1660.8385633941034
init     [-9.48473938e-01 -4.81276619e-04  7.34126064e-01  1.24411509e+00].       -18
init     [-0.48766512 -1.77792472 -0.86775563  1.67769509].       -2128.346574385609
init     [-1.17397359 -0.19610025  1.76621986 -1.94601276].       -3119.0824429471622
1        [2.048 2.048 2.048 2.048].       -1385.2811701248       -189.504451516756
2        [ 2.048  2.048 -2.048  2.048].           -4829.447006924799     -189.5044515
3        [-2.048 -2.048 -2.048 -2.048].           -11717.778680524798    -189.5044515
4        [ 2.048 -2.048  2.048  2.048].           -4829.447006924799     -189.5044515
5        [-0.76407837  2.048      -2.048      -2.048     ].       -8021.155976492433
6        [-2.048 -2.048  2.048  2.048].           -4837.639006924799     -189.5044515
7        [ 2.048      -2.048      -2.048      -0.37337285].       -9899.31793886073
8        [-2.048  2.048  2.048  2.048].           -1393.4731701248       -189.5044515
9        [ 2.048 -2.048  2.048 -2.048].           -8265.420843724798     -189.5044515
10       [ 2.048  2.048  2.048 -2.048].           -4821.255006924799     -189.5044515
11       [-2.048  2.048 -2.048  2.048].           -4837.639006924799     -189.5044515
12       [0.23281682 2.048      0.58845417 0.34515969].           -1699.5940462166436
13       [-2.048       2.048       2.048      -0.81077604].       -3437.8937007665054
14       [ 2.048  2.048 -2.048 -2.048].           -8265.4208437248       -189.5044515
15       [-0.00297184  0.30307921  2.048       2.048     ].       -855.0867475003914
16       [-0.48933097 -2.048       0.75506552 -0.40143431].       -1812.037447386919
17       [-0.14976047  0.22071614 -2.048       0.51479887].       -1808.6487825828783
18       [ 2.048      -0.24637753 -0.25108972  2.048     ].       -2379.907852848156
19       [ 1.57724724 -1.43419739  1.0000704   1.67615355].       -1701.786289068928
20       [ 1.30254556  0.86457258 -1.9212007   1.68184436].       -1193.738652806666
21       [-0.09917108 -0.39295794 -0.47187016 -1.06023994].       -225.34614242670636
22       [-1.7492046  -0.13259573  0.13386717  0.59127733].       -1062.9030231974123
23       [ 0.92456199 -0.14870572  1.83949158 -0.27586851].       -1772.2872617658254
24       [1.09755539 0.06000333 0.40179989 0.72821809].           -180.2471447384382
25       [-0.4259852   1.41530979  1.2169289   0.21733628].       -375.96080556137713
26       [ 0.36525072  1.67723443  1.54157949 -0.61270067].       -1294.6877030790056
27       [ 1.69057861  0.50218671  0.64506916 -1.67216904].       -1007.3907003780754
28       [-0.38966076  0.92146266  0.23916367  1.98130549].       -469.1687000785605
29       [ 1.29833014 -1.73994269 -1.37322401 -2.00185139].       -4634.594219963464
```

```
30       [-1.37706882 -1.78891816 -1.78812293  0.9528631 ].       -4371.451001526354
31       [-1.34271262  0.64109026 -0.54198534 -0.83191675].       -360.4993437402680
32       [ 1.10499416  1.00488362  1.58181641 -0.42751198].       -896.0299290520137
33       [ 0.28125263  1.49538692 -1.05695459  0.37020686].       -1345.846818341517
34       [-1.4033467   0.70593538  0.53780605 -0.57026152].       -239.7351017763423
35       [1.1833041  1.98553896 0.30785905 1.872358  ].           -1672.687784979123
36       [-0.28567618 -0.1544846  -1.63190891  1.11871168].       -528.1675091303345
37       [ 0.60536907  1.13095515  0.8477285  -1.307918  ].       -487.9394065197489
38       [-0.60314838  0.30745523 -1.78135793 -0.53039286].       -1734.684786652268
39       [-0.88676012 -1.24914685 -1.92922663 -0.26139304].       -3235.922544427390
40       [ 0.37336764  0.79031767  1.15239768 -0.30730899].       -338.1158489177733
```

```
### 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     [1.52956683 1.91914255 1.51222084 0.12638491].          -957.0835831104033
init     [-1.09474477 -2.0013105  -0.28479972 -0.39996883].      -2902.49459403014
init     [ 0.09287545 -0.0885072   0.22674012  0.17770913].      -9.936266004845226
init     [ 1.06862828  0.86988626  0.49021787 -0.30272811].      -44.27508698465467
init     [-0.86394869  1.94091107 -0.68086151 -1.15179085].      -2389.276657793903
1        [ 2.048      -0.23418835 -2.048       2.048     ].      -2875.923734680736
2        [ 2.048 -2.048  2.048  2.048].           -4829.447006924799     -9.936266004
3        [-2.048      0.8073882  2.048      2.048    ].          -1813.12396018947
4        [ 2.048      -0.77107626 -2.048      -2.048     ].      -7073.972948181394
5        [ 2.048 -2.048  2.048 -2.048].           -8265.420843724798     -9.936266004
6        [-2.048  2.048 -2.048  2.048].           -4837.639006924799     -9.936266004
7        [-2.048       0.14199525  2.048      -2.048     ].      -5961.093756570441
8        [-1.77290669 -1.83187185  1.97229377  1.78585986].      -3125.898820336325
9        [ 0.25243216  0.65132557 -0.14650052  1.22684164].      -214.38936491229418
10       [ 1.61594834  1.90294575 -1.45125373  0.11664432].      -3026.167872742066
11       [-1.37465317 -1.27719736 -1.7759979   1.52146912].      -2448.929554584813
12       [0.84504065 0.08611254 0.72124982 0.70162738].          -94.62044362780631
13       [-1.76613656 -0.49780973 -2.03029569 -1.76491398].      -5312.05401906433
14       [0.118439   1.92378152 0.60865592 1.89654088].          -1555.610460922084
15       [ 0.30464581 -0.18781366  0.41570432 -0.22592534].      -40.48245078294073
16       [ 0.91790074  0.6354423   0.94951529 -1.81924204].      -774.5000287298005
17       [ 0.27539419 -0.26838424  1.63171554  0.09632702].      -916.1660775772639
18       [ 1.56847741  0.24316373 -0.69881531 -1.01761465].      -779.5107598624428
19       [ 0.35484129 -1.86816284 -0.14451651  1.179078  ].      -1862.721763431639
20       [ 0.79611724  0.59266256 -1.75214302  1.71347332].      -634.3946569537112
21       [ 0.33342699  0.85744112 -1.39013062 -0.70905999].      -1211.338605679206
22       [ 1.96447046 -1.1253884  -1.00393877 -0.3542284 ].      -3195.046020551723
23       [-1.90505287e+00 -3.38086882e-01 -1.13311058e-03 -6.45533451e-01].      -16
24       [ 0.06932344 -0.82987684  1.31480695  1.92426221].      -117.00853922429148
25       [ 0.96841336  1.43567499  1.34995083 -0.09034321].      -441.5271478713890
26       [-1.94029947 -0.86847316 -1.27079134  1.67313391].      -2574.396358696985
27       [ 0.3863695   0.15426794  0.8482894  -1.47726505].      -551.715179290625
28       [ 1.27554452 -1.33772913 -0.69806032 -0.96771911].      -1717.904220368127
29       [ 1.22309047  0.50266748 -1.68360001 -0.00168864].      -1285.477542421708
30       [-1.2854068   1.68828179 -0.69504045  1.83299086].      -1447.865896983306
31       [-1.62809426 -1.9138781   1.87257031 -0.31780317].      -3882.770973027463
```

```
32       [ 0.01115185 -1.04402107 -0.98431878 -1.62553599].       -1221.490099419050
33       [-0.78268382  1.59684608 -0.70097452  1.85061772].       -1344.889269729098
34       [-1.59478187 -1.15323929 -1.63623257  1.79534308].       -2342.388490386157
35       [-0.69785189 -0.2136039  -0.78857984  1.22944205].       -163.1446137386041
36       [-0.12161189  0.78179848  0.84265519  0.85241509].       -67.54368335296114
37       [-1.99255462 -0.04453909  0.05325667 -1.23906119].       -1777.308949498485
38       [-1.84338287  1.14762359  1.95274426 -1.61966174].       -3507.482637637990
39       [ 0.52225599 -0.55066016  0.5964857  -0.79658913].       -211.9953091401932
40       [-0.85267003 -1.01769894 -1.92553885 -0.87200242].       -3294.743289500136
```

```
### 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     [-2.00550747  0.00767833 -0.01731259 -1.49983425].       -1847.6492499405208
init     [-1.46591299 -1.15278366 -0.33379049 -1.03177761].       -1509.7528385404720
init     [-1.70369167 -0.63283757 -1.36488408  1.55057801].       -1586.883400641493
init     [ 1.84714867 -1.88928665  0.81554388  0.2980242 ].       -3591.2693050622433
init     [1.63023716 0.68361819 0.19594356 0.82914273].           -460.72675701107613
1        [-2.048  2.048  2.048  2.048].           -1393.4731701248       -460.7267570
2        [ 2.048  2.048 -2.048 -2.048].           -8265.4208437248       -460.7267570
3        [ 0.745632  2.048     2.048    -2.048   ].       -4582.175544663847     -460
4        [-2.048 -2.048  2.048  2.048].           -4837.639006924799     -460.7267570
5        [ 2.048 -2.048 -2.048 -2.048].           -11709.586680524799    -460.7267570
6        [ 0.62170517  2.048      -2.048       2.048     ].       -4643.882194751861
7        [2.048 2.048 2.048 2.048].       -1385.2811701248       -460.72675701107613
8        [ 2.048 -2.048 -2.048  2.048].           -8273.612843724799     -460.7267570
9        [-2.048  2.048 -2.048 -2.048].           -8273.612843724799     -460.7267570
10       [-2.048 -2.048  2.048 -2.048].           -8273.612843724797     -460.7267570
11       [-2.048       2.048       2.048      -0.80668396].       -3433.799137496894
12       [ 2.048      -0.95145748  2.048      -2.048     ].       -6681.109716094484
13       [-2.048 -2.048 -2.048 -2.048].           -11717.778680524798    -460.7267570
14       [3.28483590e-01 1.07302868e-03 2.04800000e+00 2.04800000e+00].           -88
15       [-2.048       2.048      -0.69351776  2.048     ].       -3108.558131849681
16       [ 0.17594119  0.02738846 -2.048      -0.14495646].       -2313.572463111283
17       [2.048      2.048      0.33949458 0.12864949].           -1949.268462731299
18       [-0.37661966 -0.13446282  2.048       0.07597237].       -2120.037805189611
19       [2.048      0.27236274 0.13244069 2.048     ].           -1953.158724111358
20       [-0.36993991 -2.048      -0.19617604  2.048     ].       -2821.403785609384
21       [ 2.048 -2.048  2.048  2.048].           -4829.447006924799     -460.7267570
22       [0.04008396 2.048      1.08914878 2.048     ].           -1459.261284448879
23       [ 0.34857918  0.19648048 -0.1790552  -2.048     ].       -440.42511608063387
24       [-0.37124428  2.048      -0.20447809 -0.27325156].       -2314.162110380757
25       [-2.048       0.40384289  2.048       2.048     ].       -2263.454562520175
26       [ 1.57214197  0.36938666 -0.88856352 -0.03117535].       -618.6573179161406
27       [ 0.1852866  -2.048       0.27614645 -2.048     ].       -2430.530897116609
28       [1.49748654 0.51290318 1.64385257 0.78999404].           -856.3669505790126
29       [-2.048      -0.18471497  0.26383397  0.57584231].       -1959.720661056867
30       [-2.048 -2.048 -2.048  2.048].           -8281.8048437248       -440.4251160
31       [-1.26346914  1.78296691 -1.07923439  0.06511644].       -1947.691992804965
32       [ 2.048       0.15261037 -0.33715839 -0.46397146].       -1683.493040644359
33       [ 2.04376527  1.03643392 -1.98979889  1.0844969 ].       -2761.585518530733
```

```
34       [ 1.08857292  1.4669949   0.92814686 -1.956494  ].       -952.068037210575
35       [ 0.26887127  0.50500011 -0.13630092 -2.048     ].       -463.1824494648608
36       [-1.22825293  1.74001646  0.68661992  1.13566625].       -603.1303835990793
37       [ 0.22168689 -1.83113965  0.22192424 -0.32233273].       -1356.9898885879918
38       [ 0.24948113  1.19516488 -1.28774391 -1.39570007].       -1804.6230797860508
39       [-1.04158977 -0.41917658  1.97191901 -1.03991655].       -2984.8853118097572
40       [ 1.29290756 -0.74402798  1.47624219  2.00947067].       -674.8997870657305
```

```
### ESTIMATED GP EI GRADIENTS

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     [ 1.11132935 -1.96300002  0.54742317  1.0191007 ].       -2176.436426889387
init     [-0.00611528 -1.12723294 -1.23673451  1.0671338 ].       -787.6933551437477
init     [-1.35532201 -1.68616012  0.75923382  1.85709915].       -1852.2892745062024
init     [-2.0318279   0.04993951  1.28049546  0.46090677].       -1975.7725172569656
init     [ 0.90830978 -0.85247562  1.71120281  0.87890241].       -802.2371849082838
1        [ 2.048  2.048 -2.048 -2.048].           -8265.4208437248       -787.6933551
2        [-2.048 -2.048 -2.048 -2.048].           -11717.778680524798    -787.6933551
3        [-2.048  2.048 -2.048  2.048].           -4837.639006924799     -787.6933551
4        [ 0.47780069  2.048       2.048      -2.048     ].       -4690.900483469856
5        [ 2.048       2.048      -0.02364472  2.048     ].       -2662.217114217972
6        [ 2.048 -2.048 -2.048 -2.048].           -11709.586680524799    -787.6933551
7        [-2.048       2.048      -1.33138042 -2.048     ].       -4989.483281183711
8        [-0.56720242 -2.048       2.048      -2.048     ].       -4931.699360052157
9        [-0.57377178  2.048       2.048       2.048     ].       -1221.4200510491394
10       [-2.048 -2.048 -2.048  2.048].           -8281.8048437248       -787.6933551
11       [ 2.048      -0.56117744 -2.048       2.048     ].       -3293.2874943139245
12       [ 2.048      -0.57575421  1.28315288 -2.048     ].       -3734.491905382689
13       [2.048      1.23206354 2.048      0.55848702].           -2229.7458889033796
14       [-0.1317889   0.56530118 -2.048      -0.06188899].       -2412.8377317794775
15       [-2.048      0.7855277  2.048     -2.048    ].           -5273.806829078712
16       [-0.39565453  2.048       0.18726931  0.31931271].       -1975.1799336029685
17       [ 2.048 -2.048  2.048  2.048].           -4829.447006924799     -787.6933551
18       [ 2.048       2.048      -2.048       1.02940829].       -5370.441404973350
19       [-0.12950152 -0.03638543 -0.35623638 -2.048     ].       -490.2776334980217
20       [-2.048       2.048       0.86913613  2.048     ].       -1743.8240457428547
21       [-2.048      -2.048       2.048       0.26287683].       -5922.58887818761
22       [0.00922389 0.05280625 2.048      2.048     ].           -882.2061634329553
23       [-0.12674352  0.24054943 -0.2983193   2.048     ].       -405.02827231485395
24       [ 2.048      -0.0626217  -0.26329595 -0.04185903].       -1824.341728731095
25       [-0.70612861  0.04459932 -0.53963139  1.51068639].       -204.85687590049463
26       [ 0.27699119 -0.52941187  0.98432743  1.32497665].       -101.84990573719982
27       [-1.09101046 -0.78721212 -0.18121539 -1.02716706].       -576.5269503988607
28       [-1.84683074  0.85525047  0.09269607  1.39674932].       -895.5226497615581
29       [-1.10782382  1.08678957  1.83591093  0.79947953].       -711.0490742497022
30       [ 1.8866903  -0.12687083  0.40402276  1.86449239].       -1665.8947477686202
31       [ 0.08049474 -1.5649912   0.34922937 -0.81227962].       -783.0673403408462
32       [ 1.34205403  2.01618533  1.56150399 -1.66922673].       -2320.0148127725156
33       [0.86506562 1.08373241 0.70537686 0.11630819].           -47.901348809022956
34       [ 1.45206965  1.18898007  1.01897055 -1.15126377].       -579.7917141619665
35       [-0.52727733 -0.75829671  0.98954076 -1.48721947].       -738.3210826217644
```

```
36       [-0.75155978  1.3610957   1.14781785  1.31613751].       -116.29154433379108
37       [2.02714091 0.73764233 1.18728261 1.92488869].           -1205.8810803418426
38       [ 0.15851054 -0.76533319 -0.06448037 -0.23312002].       -115.34819415249515
39       [-1.42267809  1.18387659 -0.61381498  1.74883739].       -673.5228586952437
40       [-0.82339488 -1.42360596  1.4478472   0.39763342].       -773.1006601987814
```

```
### 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     [-1.30961535 -1.96822941 -0.15065692  0.92132937].       -3072.6361628465456
init     [-0.32684604 -0.05969061 -1.99564978 -0.0517259 ].       -2041.908514926983
init     [ 1.80964005  1.43685669  0.94193447 -1.60261705].       -1084.6194797620587
init     [ 1.61343148  1.4629038  -1.37180521  0.54204012].       -1549.0827052305292
init     [-1.96409912 -1.56984415 -0.75215949 -1.40119119].       -4385.783412609677
1        [-2.048  2.048  2.048  2.048].           -1393.4731701248       -1084.619479
2        [ 2.048      -0.91250438  2.048       2.048     ].       -3222.169648290899
3        [-2.048  2.048  2.048 -2.048].           -4829.447006924799     -1084.619479
4        [ 2.048 -2.048  2.048 -2.048].           -8265.420843724798     -1084.619479
5        [ 2.048 -2.048 -2.048  2.048].           -8273.612843724799     -1084.619479
6        [-2.048  2.048 -2.048  2.048].           -4837.639006924799     -1084.619479
7        [ 2.048 -2.048 -2.048 -2.048].           -11709.586680524799    -1084.619479
8        [-2.048  2.048 -2.048 -2.048].           -8273.612843724799     -1084.619479
9        [1.08886219 2.048      2.048      1.24541998].           -1406.8280653788652
10       [ 2.048  2.048 -2.048 -2.048].           -8265.4208437248       -1084.619479
11       [-2.048 -2.048  2.048 -2.048].           -8273.612843724797     -1084.619479
12       [-2.048      -0.82499621  2.048       2.048     ].       -3180.6919127661913
13       [-0.05076564  0.12352942  2.048      -0.29959423].       -2437.1490792312607
14       [ 0.06656225  0.7470947  -0.00641287  2.048     ].       -508.3898752990210
15       [-2.048      -1.18077181 -2.048       2.048     ].       -4558.031738878403
16       [-0.90574639  2.048       0.16575045  0.19860708].       -1781.9851379289514
17       [2.048      2.048      0.05711203 2.048     ].           -2593.478908866238
18       [ 2.048      -0.559388    0.07527866  0.03599057].       -2269.882924180482
19       [ 0.03457948  0.29203048 -0.00744897 -2.048     ].       -431.21972298664735
20       [ 0.89974518  2.048       2.048      -2.048     ].       -4512.882639400185
21       [ 2.048  2.048 -2.048  2.048].           -4829.447006924799     -431.219722
22       [ 0.01805574 -2.048       2.048       2.048     ].       -1352.2409388680248
23       [ 0.24415386 -0.01483553 -0.16655859 -2.048     ].       -437.16818284212917
24       [ 0.15547624 -0.09977176 -0.11715715  2.048     ].       -420.14987269616694
25       [-0.22887801 -0.1728996  -2.048      -2.048     ].       -4345.651842457049
26       [ 0.15234581 -2.048       2.048       0.13182734].       -2551.131648064501
27       [-2.048       0.77177479  0.22070694  2.048     ].       -1595.0929430712267
28       [-2.04131249 -0.0225987   1.47106294 -0.24288256].       -2561.327976178364
29       [ 0.32288174  0.5738954  -0.62374632  1.89578275].       -343.1949329045258
30       [ 1.34553372 -0.06390034 -0.99376889 -1.73517952].       -1197.460412286185
31       [-0.02847015  0.82695272  1.12184468  2.01272622].       -145.41785962811852
32       [ 0.2861566  -1.26320465  0.4953292  -1.70534049].       -688.4113843347533
33       [ 1.48161712  0.84992301  0.13479361 -0.9666187 ].       -313.48243984588396
34       [ 1.59888422 -0.78668892  0.27709365 -0.43610009].       -1159.7048682449756
35       [-0.29021798 -0.77532859 -0.74730451 -1.16249637].       -559.7522407393208
36       [ 0.05360964  1.85146916 -1.71137361  1.11723377].       -3320.132125181345
37       [-1.57643092  1.74419672  1.13557999  0.27545106].       -528.4755615940201
```

```
38       [1.20334288 0.20165943 1.35940674 0.92703019].          -414.8765148103328
39       [-0.1140311  -1.40990782 -0.03519657 -0.91024241].      -702.9354119031176
40       [ 1.91214941 -1.81467173  1.45340086  0.88242509].      -3491.832227753870
```

```
### 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     [-1.416549    0.98324356 -0.9694617   0.13819656].       -553.611160809845
init     [-1.98830095  1.71518775  1.64132804 -1.91110583].       -2800.587326666160
init     [ 1.87166448 -1.48599062 -0.88543907  0.43451672].       -3468.879783124874
init     [ 1.81954616  1.44480478 -2.03874618  0.08694181].       -3716.955397581154
init     [ 0.21314615 -0.05989411  1.0982775  -1.38970418].       -796.5750150214192
1        [-2.048 -2.048  2.048  2.048].           -4837.639006924799     -553.611160
2        [2.048 2.048 2.048 2.048].       -1385.2811701248       -553.611160809845
3        [-2.048 -2.048 -2.048 -2.048].           -11717.778680524798    -553.611160
4        [-2.048  2.048  2.048  2.048].           -1393.4731701248       -553.611160
5        [-2.048 -2.048 -2.048  2.048].           -8281.8048437248       -553.611160
6        [ 2.048  2.048  2.048 -2.048].           -4821.255006924799     -553.611160
7        [-2.048 -2.048  2.048 -2.048].           -8273.612843724797     -553.611160
8        [ 2.048 -2.048  2.048  2.048].           -4829.447006924799     -553.611160
9        [ 2.048 -2.048  2.048 -2.048].           -8265.420843724798     -553.611160
10       [-2.048  2.048 -2.048 -2.048].           -8273.612843724799     -553.611160
11       [-2.048  2.048 -2.048  2.048].           -4837.639006924799     -553.611160
12       [0.04117651 0.47958604 0.3467295  2.048     ].           -397.4504372946813
13       [ 2.048 -2.048 -2.048 -2.048].           -11709.586680524799    -397.450437
14       [0.05649917 2.048      0.71016894 0.38172922].           -1635.619462231906
15       [-2.048      -0.1548642   0.33324272  0.30310824].       -1915.847757453209
16       [ 1.0056466   2.048      -0.80769352 -2.048     ].       -3343.032667162500
17       [ 0.46500357  0.00928405 -2.048       2.048     ].       -894.9684280766043
18       [0.03866676 0.26551502 2.048      2.048     ].           -861.2459969869678
19       [ 2.048      2.048     -0.4124581  2.048     ].           -2939.722130793758
20       [2.048      0.28214534 2.048      0.27954581].           -3453.201131507811
21       [-0.30084609 -2.048       0.45304837  0.12705278].       -1868.914357099802
22       [-0.53249325 -0.03014277 -2.048      -0.10402547].       -2289.906549816028
23       [ 2.048 -2.048 -2.048  2.048].           -8273.612843724799     -397.450437
24       [-0.18882818 -0.31727506 -0.13967788 -2.048     ].       -450.1395502899814
25       [-0.24181274 -0.10292764 -0.04916385  2.048     ].       -425.2623706568773
26       [-2.048       0.77718535  0.36659492  2.048     ].       -1549.237684431107
27       [ 0.31497503 -0.25975828  0.12303257  2.048     ].       -429.2729555580485
28       [-0.05173221  0.07898024 -0.15437385  1.77356224].       -312.6046798645755
29       [ 0.49248519 -0.36364078  0.1241312  -0.46128554].       -62.360242192976955
30       [-1.69090131  1.78595323  1.92890053  1.11252811].       -963.0734459791341
31       [-1.83621103  1.87903779 -0.68162285 -1.26546617].       -2308.192588094473
32       [ 1.92156215  1.40944334  1.51577691 -1.70996176].       -2150.672266003839
33       [ 0.0919898  -0.08568237 -1.11132262 -1.62208438].       -948.8029306342575
34       [ 0.89455542 -1.05435484 -1.37244859 -0.5642034 ].       -1570.071442160326
35       [-1.89442692  1.28025951  1.41878095  0.88925994].       -672.7102189249384
36       [-1.58683532  0.93332869  0.12508988 -0.19690846].       -318.7661115293826
37       [-1.40446844 -0.06409175 -1.4736892   1.11702579].       -757.4511493361595
38       [-0.22075864  0.21300348 -0.04241919  1.59092653].       -259.1978979660242
```

```
39       [0.78784188 0.56687636 1.60738668 0.53712367].          -585.1246700385428
40       [-0.908832   -1.12747314 -1.62090109 -1.33948435].      -2806.611413418693
```

```
### 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     [ 1.13746907 -1.07503116  1.32824487  1.90770872].       -570.621446040192
init     [ 1.93577416 -0.19067188  0.44663793  1.1285566 ].       -1656.450565242978
init     [ 0.58004826  0.90938667 -1.9044904  -0.82555097].       -2770.136415369179
init     [-1.80833283  1.46252162 -0.5207899   0.73665721].       -1066.23070619416
init     [-0.99827733 -0.62430734 -2.00944529 -0.58026483].       -2986.744297402763
1        [-2.048 -2.048  2.048 -2.048].           -8273.612843724797     -570.621446
2        [ 2.048  2.048  2.048 -2.048].           -4821.255006924799     -570.621446
3        [-2.048 -2.048  2.048  2.048].           -4837.639006924799     -570.621446
4        [ 2.048      -2.048       0.37059336 -2.048     ].       -5847.067836379805
5        [-2.048  2.048  2.048 -2.048].           -4829.447006924799     -570.621446
6        [-0.18697538  2.048       2.048       2.048     ].       -1330.162778334060
7        [ 2.048 -2.048 -2.048  2.048].           -8273.612843724799     -570.621446
8        [ 2.048  2.048 -2.048  2.048].           -4829.447006924799     -570.621446
9        [-2.048 -2.048 -2.048  2.048].           -8281.8048437248       -570.621446
10       [-2.048  2.048 -2.048 -2.048].           -8273.612843724799     -570.621446
11       [-0.03981162  0.00450934  2.048      -0.18063442].       -2336.602054800292
12       [-2.048  2.048 -2.048  2.048].           -4837.639006924799     -570.621446
13       [2.048 2.048 2.048 2.048].       -1385.2811701248       -570.621446040192
14       [ 2.048 -2.048  2.048  2.048].           -4829.447006924799     -570.621446
15       [-2.048      -2.048      -1.25877398 -2.048     ].       -8213.438842569785
16       [-0.32794879 -0.2005761   0.16409003  2.048     ].       -423.4060806195413
17       [-2.048       1.06419756  2.048       2.048     ].       -1534.622460268997
18       [ 2.048       2.048      -1.02321284 -2.048     ].       -4147.080825802423
19       [-0.02608303 -2.048       0.2158432   0.4942587 ].       -2033.523074189746
20       [0.30759815 2.048      0.31526354 0.30328507].           -1892.470012837244
21       [ 2.048 -2.048 -2.048 -2.048].           -11709.586680524799    -423.406080
22       [-0.23105505  0.271603    0.37655836 -2.048     ].       -495.88551459455795
23       [ 0.00563474  0.20532564 -1.13214298  2.048     ].       -206.99354023572508
24       [-0.36314152  1.48132021 -0.81174684  1.60396981].       -1180.420334245901
25       [ 1.21516934  0.30106952 -1.50517253  1.22133599].       -508.70625179966345
26       [-1.954652    0.28185696  0.66734217 -0.53116287].       -1391.591581989171
27       [-0.7642521  -0.55159498  0.98612647  0.44030284].       -209.3084449600003
28       [-0.57801656 -0.89346951 -0.48374123  0.33465638].       -324.3432054652136
29       [-1.55232093 -0.83414065 -0.91616803  1.68889364].       -1397.811662780371
30       [ 1.25552684  0.98228179 -0.01194804  0.60380272].       -168.2404221329065
31       [ 1.86866832  1.80735286  0.88218609 -1.31606209].       -1292.319542893279
32       [-0.60816258 -1.48369319  1.19762397 -0.3389362 ].       -767.5440279170598
33       [ 0.9372928  -0.43945546  1.53552677  0.75343429].       -613.6859570054377
34       [-0.66508622  0.66638952  0.87959599 -0.22493756].       -126.61151527058537
35       [-0.07484835  0.02857783 -1.34884795  0.8967764 ].       -274.95003109985123
36       [0.96418378 0.51201859 1.579638   1.98711879].           -217.4114104120697
37       [-0.03576572  1.04072633  0.05068127 -0.63198527].       -256.8777483425788
38       [-1.69813972  1.3940542   0.19151705 -1.17115017].       -682.7768916967663
39       [-0.2194799   1.51724609 -1.91465659  1.68166585].       -2397.839733596574
40       [-0.2367588   1.06158386 -0.87110555 -1.24039047].       -905.0561957506699
```

```
### 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     [ 0.05711194  1.11888405  1.5172718  -2.0150397 ].       -1996.4555487728376
init     [-0.77932165  1.87434492  0.05372605 -0.744307  ].       -1417.9597465038332
init     [ 0.16056294 -1.14173976  1.25534764 -0.64612505].       -635.6383005184798
init     [ 0.15928873 -2.02394097  0.70923255 -1.18774062].       -1863.0160368222728
init     [ 1.7717559  -0.51509351  1.0339079   1.07781735].       -1397.2995170993763
1        [-2.048 -2.048 -2.048  2.048].           -8281.8048437248       -635.6383005
2        [ 2.048  2.048 -2.048  2.048].           -4829.447006924799     -635.6383005
3        [-2.048       0.19015951  2.048       2.048     ].       -2479.773608431594
4        [ 2.048       0.13381568 -2.048      -2.048     ].       -5983.328365860246
5        [-2.048      -0.89703731 -2.048      -2.048     ].       -7324.766917562415
6        [-2.048  2.048 -2.048  2.048].           -4837.639006924799     -635.6383005
7        [ 2.048 -2.048 -2.048  2.048].           -8273.612843724799     -635.6383005
8        [2.048 2.048 2.048 2.048].       -1385.2811701248       -635.6383005184798
9        [-2.048 -2.048  2.048 -2.048].           -8273.612843724797     -635.6383005
10       [-0.05222314 -2.048       2.048       2.048     ].       -1353.3681815099549
11       [ 2.048 -2.048  2.048 -2.048].           -8265.420843724798     -635.6383005
12       [ 2.048       2.048       0.4283162  -0.52864382].       -1932.160369297066
13       [-2.048  2.048 -2.048 -2.048].           -8273.612843724799     -635.6383005
14       [-2.048       2.048       2.048      -0.21178993].       -2874.1774548197186
15       [-0.18596147  2.048       0.57182985  2.048     ].       -2016.4931333612099
16       [-2.03022918e-03  1.37755447e-01 -2.04800000e+00  4.45042371e-01].       -18
17       [-2.048      -2.048       0.8889762   0.92617433].       -5009.594808535263
18       [ 0.6232987  -2.048      -2.048      -0.91509642].       -7119.609541343871
19       [ 2.048 -2.048  2.048  2.048].           -4829.447006924799     -635.6383005
20       [ 2.048  2.048  2.048 -2.048].           -4821.255006924799     -635.6383005
21       [0.11536007 0.06270944 2.048      0.21100589].           -2007.4910473264222
22       [ 0.01481728 -0.33890332 -0.23658632  2.048     ].       -424.9610797451489
23       [-2.048       0.40820797 -0.3414399   0.33358286].       -1475.4151549972482
24       [ 0.76148406  2.048      -2.048      -0.60405336].       -6425.048858017759
25       [-2.048       0.66637456  0.83653907 -2.048     ].       -2024.500482228336
26       [ 2.048      -0.44452611 -0.02236963 -0.20696218].       -2165.2475707317885
27       [ 0.10133251 -0.37903082 -0.2183316   2.048     ].       -432.58568317757556
28       [2.048      0.61235701 0.19607215 2.048     ].           -1691.9615551863224
29       [0.27633111 1.37392905 0.39697562 1.34790169].           -533.300532385822
30       [ 1.26069459 -1.83565215 -0.15960944  0.48859879].       -2449.5116865562186
31       [ 0.41185065  0.14704051 -0.06999008 -1.30219395].       -173.9576811616067
32       [ 0.75138373 -1.42916731  1.02061145 -1.76094883].       -1293.3492207589115
33       [ 0.67610656 -1.34656751 -1.92469412  0.22596233].       -2946.698074040383
34       [ 1.7588149  -0.78739493 -1.93213096 -0.323968  ].       -3815.7850490240958
35       [-1.68214821  0.54217423 -0.6400704   0.01722999].       -635.9778142669621
36       [ 0.13865086 -2.01545562 -1.0926892  -1.01367946].       -3572.723172473937
37       [ 1.82238011 -1.1194483  -1.31534249 -1.06360491].       -3422.564769724855
38       [ 1.94381142  1.54603653 -1.04249393  0.23850074].       -1754.0253780514327
39       [ 0.12795405 -1.82551184 -0.52381359  1.82879896].       -2079.0517374677465
40       [ 0.86929247 -1.73849869 -0.68259678  0.13072152].       -2016.3556549384557
```

```
### 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	[1.42875729 -1.31524229 -1.82532827 -0.56713852].		-3924.324416600484
init	[-0.9199578 0.12288092 -0.79495612 -0.80087302].		-330.9329217994366
init	[-1.59030773 -1.02441364 1.71061206 -0.96605449].		-2832.720279494954
init	[0.89200102 1.49796878 1.25779756 -1.18558481].		-912.8737542795088
init	[-1.36297254 -1.85669062 -1.88652621 -1.2278546].		-6538.09169222112
1	[-2.048 2.048 -2.048 2.048].	-4837.639006924799	-330.9329217
2	[2.048 -2.048 2.048 2.048].	-4829.447006924799	-330.9329217
3	[2.048 2.048 -0.61717139 2.048].	-3058.4255477661233	
4	[-2.048 2.048 2.048 2.048].	-1393.4731701248	-330.9329217
5	[2.048 -2.048 2.048 -2.048].	-8265.420843724798	-330.9329217
6	[-2.048 2.048 -2.048 -2.048].	-8273.612843724799	-330.9329217
7	[2.048 2.048 -2.048 -2.048].	-8265.4208437248	-330.9329217
8	[-2.048 -2.048 0.06336705 2.048].	-6040.345130709454	
9	[-2.048 2.048 2.048 -2.048].	-4829.447006924799	-330.9329217
10	[0.78148219 1.171678 2.048 2.048].	-538.8911694302681	
11	[2.048 -2.048 -2.048 2.048].	-8273.612843724799	-330.9329217
12	[-0.41727943 0.7286777 -0.08004661 0.60203821].	-106.81386008776019	
13	[-0.42371819 0.73697565 -0.09322079 0.57974363].	-107.46990013897214	
14	[-0.42282605 0.73423031 -0.09505403 0.57912959].	-106.86165148800532	
15	[-0.42294488 0.73374591 -0.09577858 0.57828517].	-106.6886528388227	
16	[-0.42293975 0.7333189 -0.09602753 0.57807904].	-106.56576860193543	
17	[-0.42299246 0.73312964 -0.09616131 0.57789781].	-106.49862465015939	
18	[-0.63800826 1.47861284 0.78310532 0.49005971].	-316.1949707099658	
19	[-0.16822643 0.33475194 -0.33065132 0.9251936].	-99.13173520839142	
20	[0.26688853 0.20936951 -0.6210786 1.35714168].	-144.27218723033783	
21	[-0.3085223 0.5010192 -0.42658561 0.61100899].	-84.78837818867335	
22	[0.17772393 0.1991617 0.26683836 1.71255526].	-279.22774861285706	
23	[-0.43166503 2.02717788 -0.53820554 0.10309167].	-2507.8924983256666	
24	[-0.40048572 0.13615201 0.13819333 0.63708062].	-43.131128264833166	
25	[-0.24109876 1.33577466 1.78268983 -1.85506578].	-2698.6619273246047	
26	[-0.40492792 0.15454405 0.10815035 0.66986354].	-47.52136236925384	
27	[-0.18451579 0.28219844 0.86944369 0.80690847].	-70.73280373721906	
28	[-0.46733727 0.10240915 -0.61502547 0.94540471].	-78.20493126216306	
29	[1.81050034 0.47345101 1.28230842 0.59391931].	-1009.8154756480709	
30	[1.71257789 2.00766758 2.04074396 0.40774124].	-1895.645937778416	
31	[-0.52169408 0.19912293 0.11130422 0.71416331].	-54.04245125354495	
32	[-0.50564748 0.19940494 0.10039974 0.71204968].	-53.6777056698185	
33	[-1.20652286 0.1849986 0.06717931 0.66327582].	-211.37610860589191	
34	[0.14477367 0.39221492 0.25811707 0.9599007].	-96.31604791279474	
35	[-0.82829673 0.53154598 0.3162951 1.0737626].	-101.34450578089117	
36	[-1.41747201 1.65781696 -1.49014706 -0.13005055].	-2373.8451177593115	
37	[0.06343646 -0.22959915 0.21978558 0.62914234].	-44.98413780442068	
38	[-2.04536301 1.59960423 -1.37306242 1.3677816].	-2255.6066494918728	
39	[1.75116601 -1.15766379 -0.02397232 -1.2176205].	-2125.1864546978345	
40	[-0.38595516 0.67898545 0.91808058 0.81727359].	-51.079035130825176	

```
### ESTIMATED GP EI GRADIENTS

np.random.seed(run_num_16)
```

```
surrogate_approx_16 = GaussianProcess(cov_func, optimize=opt)

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

```

Evaluation	Proposed point	Current eval.	Best eval.
init	[-1.13339974 0.09487705 0.20767317 -1.86121441].		-514.1403917485846
init	[-0.57045469 -1.13426046 0.77302236 -1.37735608].		-636.540695155299
init	[-1.75994935 1.80638048 0.26083893 -1.72854338].		-1399.5661404272828
init	[0.91193553 -1.3989799 -1.02284777 -0.8458762].		-1753.4526459251956
init	[0.80531749 -0.1463743 -1.16710546 -0.21779963].		-460.1166598787994
1	[2.048 2.048 2.048 2.048].	-1385.2811701248	-460.11665987879945
2	[-2.048 2.048 -2.048 2.048].	-4837.639006924799	-460.116659
3	[-2.048 -2.048 2.048 2.048].	-4837.639006924799	-460.116659
4	[2.048 2.048 2.048 -2.048].	-4821.255006924799	-460.116659
5	[2.048 -2.048 2.048 2.048].	-4829.447006924799	-460.116659
6	[2.048 2.048 -2.048 2.048].	-4829.447006924799	-460.116659
7	[-2.048 -2.048 -2.048 2.048].	-8281.8048437248	-460.116659
8	[-2.048 2.048 2.048 2.048].	-1393.4731701248	-460.116659
9	[2.048 2.048 -2.048 -2.048].	-8265.4208437248	-460.116659
10	[2.048 -2.048 2.048 -2.048].	-8265.420843724798	-460.116659
11	[-2.048 -2.048 -2.048 -2.048].	-11717.778680524798	-460.116659
12	[2.048 -2.048 -2.048 2.048].	-8273.612843724799	-460.116659
13	[-2.048 -2.048 2.048 -2.048].	-8273.612843724797	-460.116659
14	[-0.07853879 0.45636083 2.048 0.29509575].	-1881.669332016716	
15	[-2.048 2.048 -2.048 -2.048].	-8273.612843724799	-460.116659
16	[-0.00878823 0.2056628 -0.09919498 2.048].	-424.4951645074444	
17	[0.19361197 2.048 -0.15621246 0.27605862].	-2306.334648365382	
18	[-2.048 -0.05311158 0.02817756 0.38551326].	-1830.2628885331026	
19	[-2.048 2.048 2.048 -2.048].	-4829.447006924799	-424.495164
20	[2.048 -2.048 -2.048 -2.048].	-11709.586680524799	-424.495164
21	[2.048 0.04960034 0.31583126 0.39907].	-1739.1059185886857	
22	[-0.11028388 0.11778902 -2.048 0.28426392].	-1966.3909137025285	
23	[-0.07583815 -2.048 0.3820319 0.49391588].	-1898.0691005385204	
24	[0.37006987 0.21942287 0.17857812 -2.048].	-436.6565406565057	
25	[0.00221686 2.048 2.048 2.048].	-1343.9447383106664	
26	[0.23246641 0.14036431 0.22815856 -2.048].	-448.0380647900797	
27	[0.02702661 0.18776617 -0.42671713 -2.048].	-525.8111183534426	
28	[0.01781265 0.14273488 2.048 -2.048].	-4312.589124282656	
29	[-2.048 0.65997344 0.56479965 2.048].	-1559.3593042542104	
30	[0.47251379 -0.10838687 -0.02157398 2.048].	-432.90076959927205	
31	[0.35855098 -0.0605761 -0.16931152 2.048].	-417.2438331280042	
32	[0.16100656 0.15823216 -0.50539254 -2.048].	-564.1399666632592	
33	[2.048 0.17641733 -2.048 0.11636404].	-3720.6429312222062	
34	[0.3616728 -0.1267279 0.1051908 2.048].	-424.814902532517	
35	[-0.95071835 -0.2289704 -0.04594312 -1.93639198].	-511.4883243438568	
36	[2.048 2.048 2.048 0.39626386].	-2367.129974975508	
37	[2.048 0.17433706 0.11280404 -2.048].	-2043.918358927217	
38	[1.23092008 1.05604526 0.35017173 1.98018267].	-425.1432446295908	
39	[1.3844554 0.50295026 -0.30971323 2.02838352].	-607.0844287384073	
40	[-0.44208983 1.99194991 0.15971221 0.97938317].	-1867.703353468525	

```
### ESTIMATED GP EI GRADIENTS

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

```

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

```

```
Evaluation       Proposed point           Current eval.          Best eval.
init     [-0.84105215  0.12528335 -1.26353086 -1.76988013].       -1340.061030292479
init     [1.17549244 0.64034211 0.56328559 0.30966945].           -57.67854505088549
init     [-1.8879983  -0.58239548  1.82551833 -1.80205699].       -4588.535377497626
init     [ 1.49111646  1.545382   -1.83831075  0.62430665].       -2599.959730636226
init     [ 0.21197361  0.39941429 -0.06746675 -0.88888049].       -99.66183844529772
1        [ 2.048 -2.048  2.048  2.048].           -4829.447006924799     -57.67854505
2        [ 2.048       1.05649593  1.32151056 -2.048     ].       -2429.744400124792
3        [-0.55463032  2.048       2.048       2.048     ].       -1228.831722957927
4        [-2.048 -2.048 -2.048  2.048].           -8281.8048437248       -57.67854505
5        [ 2.048 -2.048 -2.048 -2.048].           -11709.586680524799    -57.67854505
6        [-2.048  2.048 -2.048  2.048].           -4837.639006924799     -57.67854505
7        [-2.048 -2.048  2.048  2.048].           -4837.639006924799     -57.67854505
8        [ 2.048 -2.048 -2.048  2.048].           -8273.612843724799     -57.67854505
9        [ 2.048 -2.048  2.048 -2.048].           -8265.420843724798     -57.67854505
10       [2.048      2.048      2.048      1.08031649].           -1894.310907441302
11       [ 0.30382411 -0.02747498  0.47507381  0.38525282].       -28.294446786821823
12       [ 0.49560715 -0.04984534  0.54591177  0.29434606].       -39.82580398039754
13       [-1.79497026  1.76546405  1.28434291 -0.33941445].       -952.0102989499331
14       [ 0.44883056 -0.92403133  0.20514262 -0.09834762].       -175.36017592138887
15       [ 0.75888275 -0.05732555  0.24496585  0.1993185 ].       -49.62563349753145
16       [ 0.5984243  -0.28933658  0.28377512  0.43315051].       -60.69216986210886
17       [0.54629375 0.25760332 0.68991895 0.80974456].           -51.04186686389741
18       [-0.03022147  0.23917661  0.72690777  0.58983567].       -52.61934257570242
19       [0.70176649 0.09840648 0.62015951 0.71873139].           -65.00776643742648
20       [ 0.63602298 -0.2385767   0.07758818  0.1236843 ].       -45.30261806493653
21       [0.20652337 0.26942335 0.49977194 0.4775316 ].           -29.99207945865354
22       [ 0.71529338  1.93480924 -1.53844117 -1.96326752].       -4874.764210509927
23       [-0.1947031   0.19405784 -0.18651802  0.86895635].       -80.5319508091907
24       [-1.84837319 -1.71996318 -1.54439013 -1.82221233].       -6457.873681581815
25       [-0.15921705 -0.00415956  1.37229052  0.65249987].       -342.34884854638193
26       [0.66745558 0.55369299 0.20408384 0.51294665].           -25.376392217711714
27       [ 1.30152593  0.68015254 -1.79257626 -0.43179959].       -1948.0563208246786
28       [0.67016107 0.07701834 0.81742873 0.9239938 ].           -87.23597287235853
29       [-0.03552169 -0.95576019 -0.82394689 -0.23735467].       -485.62770529342293
30       [-0.78940489  0.54425439  0.44427943 -1.97967394].       -480.4916998108777
31       [-1.55320406 -1.06992981  0.03407838 -1.14225885].       -1478.5284734919987
32       [0.86155625 0.68655148 0.84149486 0.31404081].           -29.682939792261852
33       [1.0551574  0.1735419  0.56276816 0.78510025].           -139.51339155203266
34       [0.25209015 0.88142615 0.22734012 0.45672361].           -114.67134219639271
35       [0.2098577  0.21289626 0.38236609 0.6687167 ].           -43.1382159248404
36       [-0.50407692  0.22632679  0.17504305  1.97008084].       -381.2946431419823
37       [ 0.81130009 -1.51128184 -0.00363526  1.44855886].       -1211.1614914606882
38       [-0.43708156 -0.27418816 -0.9320137  -0.23362957].       -252.01085519104976
39       [-0.88266509 -0.44579317 -1.7473574   0.2161308 ].       -1346.8737024339018
40       [0.38792439 0.2615397  0.43435461 0.02195894].           -18.644334187699467
```

```
### 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     [ 0.61593289  0.02233702  1.55075162 -1.30318244].       -1629.4185602457633
init     [1.44274665 1.02455823 0.68035243 1.99841976].          -361.44679331797425
init     [-0.99545734 -1.93205893  0.5559055   1.42259154].      -2000.4660253561508
init     [ 0.96737126 -1.96277407 -1.59087358 -0.82852355].      -4947.239458116178
init     [ 0.7658299   1.48122033 -1.23439366  0.64384627].      -1338.1489953342577
1        [-2.048  2.048  2.048  2.048].          -1393.4731701248       -361.4467933
2        [-2.048  2.048 -2.048 -2.048].          -8273.612843724799     -361.4467933
3        [-2.048 -2.048  2.048 -2.048].          -8273.612843724797     -361.4467933
4        [ 2.048 -2.048  2.048  2.048].          -4829.447006924799     -361.4467933
5        [-2.048       0.39550459 -2.048       2.048     ].       -2408.6442678719914
6        [-2.048  2.048  2.048 -2.048].          -4829.447006924799     -361.4467933
7        [ 2.048  2.048 -2.048 -2.048].          -8265.4208437248       -361.4467933
8        [-2.048 -2.048 -2.048 -2.048].          -11717.778680524798    -361.4467933
9        [ 2.048      -0.87618063 -2.048       2.048     ].       -3838.3646036629693
10       [2.048      2.048      2.048      0.36580616].          -2390.3586525056033
11       [ 2.048 -2.048  2.048 -2.048].          -8265.420843724798     -361.4467933
12       [-2.048       0.5111653   0.33528765  0.20084355].       -1367.8516486916708
13       [-0.06749412  0.3716805   2.048       2.048     ].       -841.5268785235213
14       [ 2.048  2.048  2.048 -2.048].          -4821.255006924799     -361.4467933
15       [ 2.048  2.048 -2.048  2.048].          -4829.447006924799     -361.4467933
16       [-1.00836931 -2.048      -2.048       2.048     ].       -5319.217376428902
17       [-0.73923716  2.048      -0.12900526  2.048     ].       -2512.598345442087
18       [-2.048      -0.74217147  2.048       2.048     ].       -3135.1201067931274
19       [ 2.048      -0.12134098  0.27989241  0.35868773].       -1880.244372998437
20       [-0.20567022  0.28808095 -0.34036074 -2.048     ].       -495.94319523647704
21       [-0.1193788   2.048       0.21548701 -0.76652986].       -2065.7697426972773
22       [ 0.48981752 -0.18164856  0.16862714  2.048     ].       -429.82367542653634
23       [2.048 2.048 2.048 2.048].       -1385.2811701248       -361.44679331797425
24       [-2.048       2.048      -2.048       0.80064351].       -5528.670069851621
25       [-0.25529919  0.53518375 -1.76903814 -0.33690932].       -1655.6380022936917
26       [-0.72041966  0.43906847  1.85603283 -0.21836083].       -1623.2039978519113
27       [-2.048       0.08735412  0.11912231 -2.048     ].       -2124.1087474200167
28       [ 0.9745277  -2.0178505  -0.29741346  1.9385005 ].       -3142.629658146597
29       [1.43581515 1.70427941 0.34660858 2.02461876].          -1030.898856822082
30       [ 1.65420563  0.46876487 -1.15497708 -1.23229633].       -1367.1275184848027
31       [ 0.48235683 -0.32152824 -0.13029186 -1.97700488].       -437.0615453746011
32       [-1.29054067 -1.01825456 -1.0846477   0.74140891].       -1202.9159813869048
33       [-0.04444255 -1.69905763 -0.81028648  0.21299648].       -1687.5219031566799
34       [ 1.35400996 -1.89016219 -1.89577984 -0.06905842].       -5735.540400611233
35       [0.80445969 0.53349022 0.88330485 1.98262236].           -181.98014634206487
36       [-1.30456047 -1.72741921 -1.73383645 -0.6779548 ].       -4779.299516062846
37       [1.50756889 1.89855338 0.12349852 2.01809109].          -1628.7136941839224
38       [-0.25372346  0.13469482  1.91152524  1.5248555 ].       -815.4309755580788
39       [-1.01118677  1.56552334 -1.07636716 -0.43010379].       -1534.6863178834344
40       [-0.65068038  1.10009825  0.05709799 -1.60403925].       -440.72721207383967
```

```
### 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     [-1.64850237  1.07007884 -1.03654206 -1.48221261].       -1412.165722504925
init     [-0.69039488 -1.70803378  0.70441812  1.2558082 ].       -1034.97990396323
init     [ 1.97731088  0.55566637 -1.16357834  0.20081636].       -1480.569995327432
init     [ 1.86613588e-01 -1.08922440e+00 -1.58217895e+00 -1.39564217e-03].       -15
init     [-1.4249639   0.13388219 -0.46282028  0.77138896].       -422.6519691366176
1        [ 2.048 -2.048  2.048 -2.048].           -8265.420843724798     -422.651969
2        [0.83916702 2.048      2.048      2.048     ].           -1104.126146388082
3        [ 0.60943483  2.048       2.048      -2.048     ].       -4640.742289311788
4        [-2.048 -2.048  2.048 -2.048].           -8273.612843724797     -422.651969
5        [-2.048  2.048 -2.048  2.048].           -4837.639006924799     -422.651969
6        [ 2.048 -2.048  2.048  2.048].           -4829.447006924799     -422.651969
7        [-2.048       2.048       2.048       0.70015539].       -2153.718531679629
8        [-2.048 -2.048 -2.048 -2.048].           -11717.778680524798    -422.651969
9        [-2.048 -2.048 -2.048  2.048].           -8281.8048437248       -422.651969
10       [ 2.048 -2.048 -2.048 -2.048].           -11709.586680524799    -422.651969
11       [ 2.048 -2.048 -2.048  2.048].           -8273.612843724799     -422.651969
12       [ 2.048  2.048 -2.048 -2.048].           -8265.4208437248       -422.651969
13       [ 2.048  2.048 -2.048  2.048].           -4829.447006924799     -422.651969
14       [-2.048      -0.60808661  2.048       2.048     ].       -3061.578011536028
15       [2.048      0.64261857 2.048      0.30303525].           -3045.304545880172
16       [-2.048       1.30097593  2.048      -2.048     ].       -4756.8851440609
17       [-0.05554178  2.048      -0.0987187   0.42514964].       -2281.847894393856
18       [ 0.19661126 -0.23700152  0.10816946 -2.048     ].       -435.07674103750577
19       [0.43477045 0.04775423 0.09928276 2.048     ].           -420.3769069031035
20       [-2.048  2.048 -2.048 -2.048].           -8273.612843724799     -420.376906
21       [-2.048      -0.10652455 -0.01300213 -0.17740439].       -1864.466020148789
22       [-0.14661242 -0.67318025  2.048      -0.07349956].       -2129.232422095484
23       [-0.51394201  0.18525105 -0.49302349  2.048     ].       -359.39259584762084
24       [ 0.04223414  0.22998796 -0.3853382  -2.048     ].       -510.2967694207748
25       [2.048      0.3242387  0.27704515 2.048     ].           -1891.354691811500
26       [-1.53372405 -0.03763685  0.13687502  1.97089071].       -962.3519514274669
27       [-0.04871763  0.21398955 -2.048       2.048     ].       -914.5444455566166
28       [ 1.54130277 -1.40096691  0.44854182  0.28603579].       -1662.608158615334
29       [-0.27694401 -0.87943929  1.25330768 -1.56633965].       -1103.828799523978
30       [-0.63808557  0.03132333 -0.59613659 -0.75571869].       -179.40284795662322
31       [-0.05173137  0.70642684  1.09726491 -0.37487361].       -335.7968436330318
32       [ 0.71374153  2.02105291  1.40017248 -0.46276058].       -1537.641421353224
33       [1.17135501 0.69985291 0.40068972 0.5299697 ].           -60.10743371792253
34       [ 1.47564194  0.02636049 -0.28983168 -0.78361601].       -549.3029618091022
35       [-2.02182177 -0.70114881  1.40662951 -0.08419685].       -2814.800496493994
36       [ 1.59068772  1.16671535 -0.40371861 -1.88168526].       -917.8516733150443
37       [ 0.35769955 -1.72257422  0.15444364 -0.55293506].       -1175.446641753262
38       [ 0.37183303 -1.45548263  0.73367031 -1.47898748].       -859.1859036805786
39       [-1.23597818 -0.09473091  0.75009555  0.3176916 ].       -330.3961499944753
40       [ 1.02747587 -1.86636766 -0.64418959 -1.95688314].       -3130.985945873225
```

```
### 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     [0.36098376 1.62903543 1.60370987 1.29367031].           -499.4296468486902
init     [-1.90099626  0.78543905 -0.49692286  0.07582083].       -937.6489569337288
```

```
init     [ 0.6469692  -1.25398951 -0.93259202  0.8954099 ].       -916.294647636079
init     [ 1.15918278  1.43494201  1.12740309 -1.897823  ].       -1092.029190829674
init     [-1.57002246  1.02924575 -1.06816219 -1.00431457].       -1129.867964278779
1        [-2.048 -2.048  2.048 -2.048].           -8273.612843724797     -499.429646
2        [ 2.048  2.048 -2.048  2.048].           -4829.447006924799     -499.429646
3        [ 2.048 -2.048  2.048  2.048].           -4829.447006924799     -499.429646
4        [ 2.048 -2.048 -2.048 -2.048].           -11709.586680524799    -499.429646
5        [-2.048      -1.31218948  2.048       2.048     ].       -3519.181909230178
6        [-2.048 -2.048 -2.048  2.048].           -8281.8048437248       -499.429646
7        [ 2.048 -2.048  2.048 -2.048].           -8265.420843724798     -499.429646
8        [ 2.048  2.048 -2.048 -2.048].           -8265.4208437248       -499.429646
9        [-2.048  2.048  2.048 -2.048].           -4829.447006924799     -499.429646
10       [-2.048 -2.048 -2.048 -2.048].           -11717.778680524798    -499.429646
11       [-2.048  2.048 -2.048  2.048].           -4837.639006924799     -499.429646
12       [-2.048  2.048  2.048  2.048].           -1393.4731701248       -499.429646
13       [ 2.048 -2.048 -2.048  2.048].           -8273.612843724799     -499.429646
14       [2.048 2.048 2.048 2.048].       -1385.2811701248       -499.42964684869025
15       [2.048      0.33222157 0.34513924 0.24223786].           -1500.568523063980
16       [-0.17531963 -0.01006597  2.048      -0.03965709].       -2215.697948272316
17       [-2.048  2.048 -2.048 -2.048].           -8273.612843724799     -499.429646
18       [-0.13630837  0.47773194 -0.03357712  2.048     ].       -449.5372127249809
19       [ 0.06416532  0.28150094 -2.048       0.02864992].       -2206.160081415771
20       [-0.29064905 -2.048       0.2339401   0.07333681].       -2034.771542772617
21       [ 2.048       2.048       2.048      -0.51091938].       -3138.531790164403
22       [-1.64017903e-01  1.48217285e-03 -2.92206995e-02 -2.04800000e+00].       -4
23       [-0.03419113  2.048       0.32453199  2.048     ].       -2296.489515799192
24       [-0.18129873  2.048       0.0556355  -0.57412625].       -2155.636963961828
25       [ 0.04155987 -0.47044916  0.12063162  2.048     ].       -440.6541161105476
26       [-2.048       0.26333078  0.08099687  2.048     ].       -1972.693706051193
27       [ 0.04678099 -0.35812747  0.00551397 -2.048     ].       -437.6742928880651
28       [-0.1176829   0.52535607  1.25800185  2.04523084].       -145.54357766986558
29       [0.63753141 1.39167152 0.03172582 0.39546635].           -476.76064164378056
30       [-1.87196263 -0.79279469 -0.91608427 -0.35078485].       -2241.778048936678
31       [-0.04876514  0.40945054  1.30688952  0.19211095].       -377.6802644324403
32       [-1.04514726 -0.18482258  0.96996067 -1.16480678].       -699.6403016208243
33       [ 0.24018362 -0.11406655 -0.02327053  0.19288687].       -9.646826040884186
34       [-1.7380717  -1.16787012 -1.20464654  0.92155594].       -2459.434183140090
35       [ 0.37665898  1.49622433 -0.6456347  -0.86106668].       -1182.006847432322
36       [-0.83423098  0.87355459 -1.03564615  1.6428321 ].       -366.747503263408
37       [ 0.99818601  0.88294004  1.42843553 -0.2987412 ].       -590.7562504994663
38       [ 0.13639    -1.98625677 -1.18880173  1.99566803].       -3086.135235801925
39       [-0.37295724 -0.45068957 -1.20087908  0.35091761].       -359.810068893671
40       [ 1.40096206 -0.67822828 -0.22506665 -2.00215088].       -1170.257381297159
```

```
end_approx = time.time()
end_approx

time_approx = end_approx - start_approx
time_approx

start_exact = time.time()
start_exact
```

```
1623408371.1556969
```

```
### EXACT GP EI GRADIENTS

np.random.seed(run_num-1)
```

```
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	[-0.33987787 0.90244913 -2.04753152 -0.80964578].		-3394.123441866637
init	[-1.44688787 -1.66978112 -1.28507817 -0.63258326].		-3615.357417364757
init	[-0.42284043 0.15899334 -0.33097927 0.75865907].		-59.36907919270953
init	[-1.21056359 1.54876902 -1.93582042 0.69823492].		-2823.036073483468
init	[-0.33871953 0.24039354 -1.4729751 -1.2365763].		-1404.629382113278
1	[0.47982557 -0.71006248 0.11082999 1.58081884].		-353.9044857665011
2	[-0.06378109 1.47176573 1.35331664 0.61093559].		-431.9228759760618
3	[1.37005676 0.81765656 1.71318718 -1.88527122].		-2545.546979632746
4	[-1.81053107 -1.47940754 1.78238716 1.59146846].		-2545.8382380521
5	[1.9232277 -1.98832647 -1.16650153 -1.56722413].		-6727.49381067573
6	[1.56126543 -1.53158254 0.54665558 -0.91048865].		-2052.24859507814
7	[-1.91249677 0.95620178 0.92652261 -1.68362348].		-1384.494575236834
8	[1.79630348 1.92819261 -0.80082844 1.48474429].		-2286.402613249769
9	[1.41062424 0.50059934 -1.20225809 0.78433658].		-481.84120359572677
10	[-0.47896694 1.97024478 -1.57773591 -1.22917739].		-4676.218592985641
11	[-0.80177084 0.10432746 -1.2657979 -1.17103934].		-970.2835994391748
12	[-1.21299939 -0.61601458 0.78020353 1.73140592].		-585.3749170898446
13	[1.93343763 -1.00515763 1.02813215 -0.89606058].		-2636.316924015028
14	[0.16408403 -1.87745056 -1.08510878 -1.96427878].		-3488.188043504941
15	[1.21927521 -1.3441189 -0.25420212 -0.95510121].		-1337.1274101009928
16	[0.71957744 -0.08618859 1.86832512 -0.97928598].		-2382.807999161919
17	[-0.85563904 -0.0953072 0.53507817 0.88022639].		-136.2633926140205
18	[1.48287921 1.62432131 1.87436746 0.5222742].		-987.3782986646119
19	[-0.05953229 -1.85141643 -0.82367161 0.90781219].		-2169.380666875191
20	[1.89620782 -0.14839589 -1.53922942 0.68600274].		-1937.398128539917
21	[0.75241121 -0.58888911 0.39295842 1.31362989].		-270.9503393195663
22	[-1.91527024 1.88567789 1.75165396 1.9022421].		-789.0627595795792
23	[1.8307373 0.45568333 -1.22143559 -0.95524121].		-1647.6341694210719
24	[0.8051578 1.44958647 -0.0529613 -1.67199703].		-810.1388967930725
25	[0.79734412 -0.83463689 -1.80414999 -1.60545318].		-3215.219531243629
26	[-0.1582911 -0.68994974 1.6102458 -1.19194178].		-1616.8340348753732
27	[-0.22926075 1.4608557 1.28246287 -1.03341029].		-989.8946435008918
28	[2.0062999 -1.50296352 1.08688969 -0.29540852].		-3418.8237640549432
29	[-0.79647557 0.5219737 -0.52585479 -1.74939235].		-481.21093988613774
30	[0.66119611 -0.1938156 0.35461529 0.57051867].		-71.6059468847258
31	[1.55656109 -0.57199926 0.1756522 1.30094652].		-1064.001858558137
32	[-1.98935079 1.82470542 0.7078706 -0.53078398].		-1258.384739357216
33	[-0.06966701 0.83377973 -0.92724163 1.71662838].		-410.2453655200145
34	[0.56787983 -0.70452346 0.6459641 -2.00969127].		-699.9448688633819
35	[0.49704515 0.93111689 1.6178312 0.1594768].		-707.9393288226192
36	[0.87715693 -0.5909966 -1.62091187 -1.99652745].		-2720.678153604498
37	[-1.08501802 1.32287606 -1.67441492 1.95053983].		-1259.1691178659573
38	[-1.73681667 0.33276035 0.5322298 -0.61463567].		-826.8067343838653
39	[1.69661485 -1.31722849 -1.50498054 0.6533503].		-3082.082077592921
40	[-0.38001736 -1.73073496 1.72122227 -1.51184018].		-2525.930515971026

```
### 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     [-0.26216488 -1.94180615  0.20341751 -0.26491948].       -1696.976458203069
init     [-0.32617348 -0.69494857 -1.20975919  0.48853388].       -455.3157381798458
init     [-0.82061446 -0.95507548  0.49616418  0.11936602].       -291.5008468259235
init     [-1.49676054  0.05561598 -1.29253431  1.16873277].       -682.7190038869136
init     [ 1.4498828  -0.02360591  1.41951584 -1.72177213].       -2051.02408589455
1        [-0.4632877   1.20273901  0.32769712 -1.38322494].       -447.6512250019873
2        [-1.79751416 -0.52112824 -0.26988812  1.94096178].       -1797.9483083066798
3        [ 0.94656668 -1.40026063  1.68602863 -2.04011953].       -2925.242198220676
4        [-0.36783662 -1.88731676  1.04396437  1.21112717].       -1054.8119304069996
5        [ 0.49207579 -0.1936526   1.21953714 -1.80202412].       -1242.389386909326
6        [-1.91545461 -0.03437501  0.5961899  -2.03028164].       -1985.7779986308444
7        [ 1.20412991 -0.51609684  1.79301296  1.64643946].       -868.568370989524
8        [ 0.9497619  -1.00087775  2.01275464 -1.90179618].       -4013.1500050237373
9        [ 1.91280873  0.05792566 -1.4610237  -1.74473588].       -3023.791522215833
10       [ 0.6711098   0.67319442 -0.29503953 -0.49728085].       -96.9852621015022
11       [ 1.62443908 -1.07297197  0.55047237  1.34488749].       -1527.2605334054315
12       [ 0.70328035  0.79253662 -1.11663844  0.54625177].       -366.9920724173906
13       [ 0.8543915  -1.08588577 -0.49915066 -1.03232031].       -782.2437551489709
14       [-0.96817674  1.00253367  0.23548821 -1.14406601].       -207.99406890784175
15       [-1.20514898 -0.59463617  0.02209828 -1.99340305].       -835.9401338477081
16       [-1.79245809  0.56510131  0.01560574 -0.60648902].       -756.0808185096352
17       [0.88403941 0.44212308 1.24926468 0.08105246].           -341.878459426937
18       [-0.99411718 -0.98352436  0.78232123  0.01078764].       -436.32643069587914
19       [-1.00984619 -2.04537038 -1.48067386  0.56031501].       -4433.688298840226
20       [ 0.55207279 -0.01956725  1.4668598  -0.13039796].       -747.8208786411448
21       [-1.93895196  1.55137364  0.69722405 -0.50223336].       -886.5665471151541
22       [ 0.87913442  1.08383489 -0.5164871  -0.88012047].       -429.534789456539
23       [ 1.68243189 -0.29617387  1.20898237  0.845374  ].       -1143.5479302987378
24       [-1.50413177  0.89525845  1.82922983  0.42630423].       -1152.0157493412569
25       [ 1.08619    -1.14804293  0.95918544  1.11887641].       -563.341210484766
26       [1.92373852 1.90161802 1.15448994 1.02248265].           -940.9949911919929
27       [ 0.89051522  1.59091218 -0.85470426 -0.72790011].       -1426.463825694853
28       [ 1.7952232   0.98917322 -0.00676112  1.436821  ].       -804.0658058564687
29       [ 0.63523663  1.67930549  1.90252766 -1.75774148].       -3139.9508366843415
30       [ 1.82143212  0.73311298 -2.04727183 -1.88527067].       -5038.575769519586
31       [-0.47465946  0.42283806 -0.85595954 -1.68383769].       -700.8748323801497
32       [-0.68527262 -0.58563323  1.51289546  0.58564071].       -543.9353731860382
33       [0.55223936 0.46147771 1.27238896 0.21156894].           -313.3316330394328
34       [ 1.32173857  2.04339184 -0.29528961 -2.00670871].       -2448.8503389948387
35       [-0.34823756 -0.05469126  0.43322284  1.88093636].       -311.56852523404655
36       [0.51395761 1.42586065 1.73124137 1.4637363 ].           -380.1695492899531
37       [ 1.28905706  0.32105178 -1.58817841 -1.93695353].       -2461.505686335076
38       [-1.87234884 -1.86175433  1.40728674 -0.66191861].       -4019.649521955219
39       [-1.75988164  0.58497024  1.75923584  2.00126065].       -959.8960097161256
40       [-0.05047861  0.37549186  0.96577463  1.93190576].       -183.26676877682718
```

```
### 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)
```

```
8
```

```
Evaluation       Proposed point           Current eval.          Best eval.
init     [ 0.20806821  0.85257348 -0.85645419  0.04434987].       -367.7787956485136
init     [ 1.60951073  1.62321649 -1.53360257 -1.19913317].       -3099.341888277455
init     [-1.83719034 -0.24244288 -1.92562704 -0.17681111].       -3229.929027395333
init     [ 0.61089402 -0.90731609  0.72194008  0.3721741 ].       -171.0831542888061
init     [-1.94977021  0.24106634 -0.98610198 -0.3477455 ].       -1564.280611971214
1        [-1.67027913  0.62837106  0.23691576 -0.56703073].       -516.3694433953648
2        [-0.29100799 -0.48807482  1.00190111 -1.60624191].       -776.2427879756497
3        [ 1.73122643 -0.6484088  -0.01911714  1.34680056].       -1533.906024958559
4        [ 0.2331452   0.18954322  1.52098534 -2.02444137].       -2105.567470381272
5        [0.13068371 1.75340124 0.50169946 1.20414234].           -1055.6545691701
6        [ 1.20186817 -1.28642743 -1.13023645 -1.33955666].       -2216.156072663885
7        [1.75427745 1.64208478 0.672074   0.7129784 ].           -623.7616288616542
8        [-1.16664765  0.47902952  0.0920567   0.11458401].       -86.60324424339615
9        [ 2.04115885  1.62476609 -1.34334081 -1.03870914].       -3047.933279194406
10       [-0.79508563  1.14428579  1.85786343  1.29587167].       -525.0293915730558
11       [-0.49210466  0.13453559 -0.0550971   1.12595775].       -131.8782605794016
12       [ 0.15608471  1.42891248  1.70018876 -1.78973927].       -2400.928425310443
13       [ 1.70127945  0.89285159  1.02292007 -0.09786345].       -537.125759480679
14       [-0.15926852 -0.36244133 -1.73771274 -2.03864869].       -2933.713458154941
15       [ 1.49026914  0.9714593  -0.44400853  0.61778337].       -368.7135592455129
16       [-0.46262741  0.05837161  1.68184685  0.06103135].       -1053.578087303314
17       [ 0.16200584 -0.70881686  0.67231605  1.16645499].       -111.6911386063480
18       [-2.03453222 -1.33622661 -1.70069197  0.05220796].       -5042.119631809486
19       [ 1.33696319 -1.63880317  0.39971969  0.30947677].       -1706.172914318579
20       [ 1.49239847  0.49268641  1.60406172 -0.66830816].       -1537.673541011973
21       [ 1.29844154 -1.80613562  0.68428634  0.57604784].       -1893.217386799778
22       [ 1.2299417  -1.91096248  1.4133729   1.99191587].       -1681.931499110462
23       [ 0.00550859 -0.87056605  1.11029903  1.51331117].       -100.5842275200599
24       [ 1.03821088 -1.81361364 -1.15747931  1.09327608].       -2832.013290095302
25       [-1.92125593  0.53080971 -1.93881188  1.72743047].       -1922.030053518997
26       [1.4605204  1.90918656 1.41415443 0.55762362].           -711.8857692267823
27       [ 1.0047818  -0.63259091 -1.96439414 -1.85587058].       -4106.041253375681
28       [ 0.98492504  1.17178375 -0.22453846 -1.97200959].       -669.856486331395
29       [ 1.10614155  0.98623714  1.95858435 -0.70898671].       -2169.50428747307
30       [ 0.94417659 -2.0345752   0.21589641 -1.32198484].       -2592.769397712663
31       [ 1.69980461  1.46618113 -0.57797751  1.0150917 ].       -996.1300770680594
32       [-0.7850228   1.3368284   1.82024652  0.72471013].       -726.0825390658513
33       [-0.90364346 -1.40881445 -0.01203534 -1.32845648].       -1080.921298109538
34       [ 1.23334546 -0.45172328  0.65487468 -1.43597748].       -759.5865546206971
35       [0.74524341 1.03355023 1.41376536 0.64124483].           -219.3181754103930
36       [-1.76099243  0.62045344 -0.09891249  1.63585396].       -912.1566142299126
37       [ 0.74426082 -0.067332   -0.27796404 -1.07523988].       -182.2407257424594
38       [1.36215247 1.66444399 1.29525051 0.73108159].           -311.5110661199934
39       [-0.29245864 -0.98054645 -0.12680832 -0.08144759].       -239.9014824789936
40       [ 0.94174472  1.35912663  0.63783186 -1.03478984].       -376.6547766751747
```

```
1 ### EXACT GP EI GRADIENTS
2
3 np.random.seed(run_num_4)
4 surrogate_exact_4 = dGaussianProcess(cov_func, optimize=opt)
5
6 exact_4 = dGPGO(surrogate_exact_4, Acquisition_new(util_grad_exact), objfunc, param)
7 exact_4.run(init_evals=n_init, max_iter=iters)
8
```

```
Evaluation       Proposed point           Current eval.          Best eval.
init     [1.91295422 0.19346329 1.93611514 0.87988631].          -2387.047477734054
init     [ 0.80989727 -1.16289743  1.95082017 -2.02248087].      -3769.020385450150
init     [-1.01178424 -0.26709388  1.14435245 -1.23828193].      -936.5671269056027
init     [ 1.48682029  1.98000917 -1.37690218  0.39867983].      -3042.500138281922
init     [-2.01119294 -0.46460403 -1.8671204   1.87045056].      -2747.916959992119
1        [ 0.18515006  0.09995911  0.56365156 -0.40347466].      -84.75946556482826
2        [ 1.08643695 -1.32704877  0.75746856  1.22708601].      -777.5868539622428
3        [ 0.50108145  1.44873455 -1.9222709   0.09215335].      -3067.494422071646
4        [ 0.97015307 -0.14843719 -1.90712713 -0.08913273].      -1889.173868533498
5        [ 0.5719512   0.52770101  0.01636585 -0.84311832].      -83.39656774588515
6        [ 1.42806536 -1.31189144 -0.42005859 -1.21391197].      -1782.389365571046
7        [ 1.26503965 -0.57328862  0.00821153 -1.87677415].      -838.511022269754
8        [ 1.32407951 -1.69087568 -1.18515526  1.20351454].      -2837.888629380975
9        [-1.2736561  -1.35513827 -0.00896676 -0.65844098].      -1282.091224084182
10       [-1.30788384  1.6352187   1.93744968 -0.47135138].      -1846.533547228594
11       [ 0.73892166 -0.30564     1.8830846  -1.98827539].      -3458.202946512108
12       [ 0.84366127 -0.65834373 -0.05004698 -0.57847503].      -248.72419145669912
13       [0.48281039 0.89101288 1.25443128 1.89276724].          -75.02371382812068
14       [-0.62352509  0.24447576  0.3249728  -0.91682565].      -117.31504127076755
15       [ 1.63775139  0.76920136 -0.76521696 -1.80540063].      -1125.32595778395
16       [-1.86465068 -0.92999789 -1.25515852 -1.55335209].      -3387.497726089871
17       [-1.49687757  1.64977007 -1.01799232 -2.00936642].      -2371.816139376084
18       [-1.44955162 -0.24929692  0.99857842  0.63114793].      -661.1309921254697
19       [ 0.43568332  1.10448031 -1.26447195  1.42079957].      -709.4880074006844
20       [1.95857626 1.06291297 1.70378865 0.91546935].          -1198.36545447881
21       [-0.64020879 -0.32548536  1.10358982  0.69077064].      -185.85029281947786
22       [ 0.51955275 -1.53918616 -1.81726474  1.82769867].      -2311.956689208271
23       [-1.68797962 -1.89587099  0.53229563 -1.84784893].      -3659.270796405538
24       [0.19058347 0.6163165  1.52408089 0.05787768].          -678.6412482177482
25       [ 1.95414706 -2.00910552  0.17623568  0.51214857].      -4920.277230184521
26       [0.29278859 0.48443353 0.89325068 0.69622496].          -61.07998569163262
27       [-1.68297999  1.07256271 -1.17005342 -1.60752089].      -1746.051797666213
28       [-1.83504592  0.57272321  0.36396613  1.78672103].      -1063.426274111881
29       [-1.82621733  1.3648309   1.38733792  0.83316977].      -538.2029210893229
30       [-1.22276768  0.7543545  -0.37217706 -1.22703467].      -336.8270745465934
31       [ 1.32638545  1.39105073  1.36951294 -1.83395094].      -1421.988042084056
32       [-0.76125084  0.12963873 -0.62036409 -0.95263865].      -246.209503333022
33       [-0.87122222  2.04692871  0.45878489 -0.82666467].      -1670.461653070432
34       [-0.33218589  0.75505643  0.15995792 -1.25506911].      -224.93573110971306
35       [ 0.53907861  0.80105769 -0.13294576  1.03329548].      -190.74687989954148
36       [ 0.60276928  1.49878963  0.26705322 -1.09257692].      -657.1050982695266
37       [1.17707935 1.45523333 0.8341324  0.29449966].          -181.61011713003785
38       [-0.08973767  1.69880134 -0.38553909 -0.15508577].      -1368.931949164734
39       [ 0.74821824  1.78306464 -1.78430824  0.91549893].      -3136.318146218995
40       [ 1.01737435 -1.68893479  1.62333127 -1.74129535].      -2816.089596716538
```

```
### EXACT GP EI GRADIENTS

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

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

```

```
Evaluation       Proposed point           Current eval.          Best eval.
init     [-1.13871597  1.51851953 -1.20127834  1.71463028].      -1252.015732025706
```

init	[-0.04746777 0.45770286 1.08915858 0.07544008].	-222.9175241676014
init	[-0.83230515 -1.27909385 -1.71728376 0.97665145].	-1918.291410223084
init	[-0.24039742 -1.39956278 1.55622208 -0.92534185].	-1356.5528565009568
init	[-0.35129336 -0.8352566 0.52751527 0.32701567].	-100.45329155979191
1	[0.31810707 -2.04127366 0.06337582 0.57260104].	-2185.7438722645497
2	[0.93417407 -0.95713874 0.04788205 0.51617412].	-441.3555380991531
3	[-0.44059633 1.95260162 0.98851101 1.32304877].	-1121.7490227731053
4	[-0.69240508 -1.69821421 1.05569041 0.3397459].	-878.6270483712931
5	[-0.62007949 -0.59387599 -1.14516255 0.71394031].	-365.539811259935
6	[-0.77079598 1.5938349 -1.67587878 -1.06932543].	-3392.021959788046
7	[0.08440215 0.09678137 -0.68591607 1.2573758].	-115.56245509730319
8	[-0.05157625 -0.67724132 1.83878851 0.29903289].	-1191.2661216102438
9	[0.5080949 1.05243414 0.54879247 -1.08623221].	-287.2533091457605
10	[-0.91327357 1.24112404 1.78933104 -0.59901106].	-1471.6531056871854
11	[1.16854508 0.27037259 0.45464854 -1.32530474].	-370.0513563419816
12	[-1.62959076 1.6064982 -0.65160286 1.3152801].	-1244.2646588595474
13	[-1.18632711 2.02375049 0.31679644 -0.89218401].	-1570.711430108154
14	[-1.96247991 1.30933862 0.78091536 1.12353327].	-768.6132013299593
15	[1.80499795 0.05340004 -1.61819437 0.9156279].	-1588.1307824603791
16	[-1.61562475 1.12448258 0.36308353 1.13101629].	-409.0966974363194
17	[-2.00979377 1.45046548 1.91636392 1.91286092].	-993.4236305943057
18	[-0.01686843 -1.65674521 0.67476732 0.43664698].	-711.3131631801381
19	[0.6927607 0.97949297 -1.32818922 1.7809417].	-553.8106806602497
20	[0.86866932 0.5076431 -1.82363683 -0.78598627].	-2138.083968068359
21	[1.94994465 -0.35355171 -1.53741326 -1.48909156].	-3496.9851843806637
22	[0.1473329 -1.7247022 -1.41938966 0.59671559].	-2450.7700890745095
23	[-1.37505296 -1.32512161 1.12550758 -1.87113142].	-2069.6453850731227
24	[0.75502349 0.74256749 1.84303871 0.46968966].	-1027.436879763098
25	[1.32475783 -0.8915516 -0.21088425 -0.78154481].	-874.9477914351841
26	[1.15487158 -0.834299 1.05481692 0.2559162].	-559.694281885498
27	[-0.3107496 1.66980759 -0.84551918 0.41760558].	-1582.353544675686
28	[-1.07255585 -0.11466322 -1.92503864 1.03190207].	-1264.7418425813808
29	[1.28272724 -1.39964435 -0.12724209 -1.20591634].	-1518.9287371421751
30	[-1.20365786 0.31174043 0.0297163 1.38762166].	-328.3195031335746
31	[-1.6109904 0.8590569 0.45648285 -1.77930314].	-711.5941755565317
32	[-0.13522102 -2.00678675 0.85543304 0.21808611].	-1452.8346714359607
33	[-1.3997181 0.46429525 -0.96362403 1.3682138].	-391.7571103590943
34	[-0.61004237 1.06835669 0.87587592 -2.02996793].	-840.5237144476857
35	[-1.12910189 1.09708521 1.24874948 -1.89400768].	-1200.554415721937
36	[-0.31390776 0.79157762 1.3836017 1.46509244].	-127.43678379889838
37	[-1.38834774 1.34558751 1.64297275 0.83116248].	-391.92643983382715
38	[-1.00480104 -1.63478392 0.45813346 -1.68836468].	-1561.230607409748
39	[-0.03997327 1.87412011 1.35086415 -0.98530202].	-1609.4809120635457
40	[-0.49490898 -0.61553645 -0.38107717 1.124754].	-234.496171179621

```
### 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	[1.60915518 -0.68821072 1.31575449 -1.87721062].	-2450.5677149947023
init	[-1.60703824 0.38933325 0.12213192 -0.33256477].	-501.1308148976547
init	[-0.67416945 0.50183959 -0.25337272 0.96617311].	-111.72625934007624

```
init     [0.07387714 0.32300483 0.59537447 2.00795862].          -309.0778245045865
init     [ 1.31013917 -0.35552897  1.54119232  1.32611864].      -741.7784291682051
1        [ 0.89019247  1.79138369 -0.60698719 -1.00911474].      -1748.991187524451
2        [-1.14213156  0.58137556  0.92917136  0.01684862].      -163.6614524559984
3        [ 1.75433426  0.20493592 -1.15829832  0.97976468].      -988.298019147344
4        [-0.79474745 -0.73843181  0.53990645 -1.68693428].      -585.5828092278404
5        [-1.00900642 -1.66327098 -0.76191645 -2.04296494].      -2666.422191033631
6        [ 0.04661632  0.04813664  1.80385034 -1.54175126].      -2627.028746889713
7        [-1.74092098 -0.28326251  0.05425427 -0.54902337].      -1138.893020966359
8        [-0.62606658 -0.58785451  0.50109625  1.05568426].      -168.5725461690538
9        [-0.44297594  0.21604389  1.09977188 -1.18858878].      -688.7294077105599
10       [-0.39873712 -1.5352345  -0.18194884 -1.01176645].      -1050.594876515039
11       [ 1.0579262   0.73488137  2.01182067 -0.80229802].      -2584.45765645506
12       [-0.86349482 -0.11862378  0.83160281  1.00745874].      -156.2593189829688
13       [-2.02192246  1.2686919  -0.68755355  1.22030758].      -1390.567077328647
14       [-1.90383044 -1.71776218  1.30135346  2.00336669].      -3151.598091384067
15       [-1.33861412  1.23201025 -0.96307082 -0.24365355].      -793.3806495510584
16       [-1.20433764 -1.7806771  -1.09423599  0.66747467].      -2908.121430619087
17       [-1.86392832  1.13826433  0.68259296  1.42921976].      -684.3704195020284
18       [-0.71508768 -1.06298624  1.48028271 -0.75758851].      -1137.112846776038
19       [ 0.94214099 -1.57354766 -0.74644094  1.15478747].      -1689.576343784932
20       [1.8909089  0.28940522 0.85682419 1.06069847].          -1151.612110817843
21       [ 1.89109499 -0.64451329  0.19246157  1.0232119 ].      -1887.849899261306
22       [-1.48289855  1.9603708   0.88014879  1.04657615].      -898.0695517217156
23       [0.33873613 1.06893801 0.88105688 1.03917139].          -105.2592849149460
24       [-1.61922811  1.23481322  2.02363183  1.9101849 ].      -702.6303179178191
25       [ 0.15810568  1.97152737 -0.44674798 -1.31645352].      -2490.548467729696
26       [-0.14395639  0.83083395  0.48969449 -1.25888407].      -295.8548268900935
27       [-1.04837812 -1.02335261  0.62386577  1.73596787].      -658.211927766007
28       [-0.95422001 -0.54737772 -0.44854888 -1.1889234 ].      -470.0820820835665
29       [ 1.96181723 -1.56127344  0.47070345 -0.43759404].      -3364.882590614902
30       [ 1.33981188  0.49177975 -1.7059087  -1.64398766].      -2630.928106036733
31       [-2.04650648  1.44371507 -1.98550922  1.9251675 ].      -2834.811304710729
32       [-1.06066992  1.23904023 -1.04353373 -0.39191513].      -894.0769169135324
33       [-0.35702929  1.09433763 -0.51713564 -1.56045156].      -725.7734812165357
34       [ 1.85564941 -0.68981499  0.2899382  -1.66918129].      -2023.310202609940
35       [ 0.62057554  0.30413893  1.58954197 -1.24066722].      -1645.007906836277
36       [ 0.78309714  1.36094253 -1.70864839  1.38494553].      -1566.838145324867
37       [-0.17324013  1.34467199 -0.60828857  1.01354448].      -802.2419906375629
38       [-0.07967532 -0.56041942 -0.53123136 -0.08364451].      -122.9059668733734
39       [ 0.18867716  1.98328736 -1.20424388  1.44766517].      -3025.401390343327
40       [ 1.41638326 -0.57018461 -1.27636987 -0.21078604].      -1266.565734400051
```

```
1 ### EXACT GP EI GRADIENTS
2
3 np.random.seed(run_num_7)
4 surrogate_exact_7 = dGaussianProcess(cov_func, optimize=opt)
5
6 exact_7 = dGPGO(surrogate_exact_7, Acquisition_new(util_grad_exact), objfunc, param)
7 exact_7.run(init_evals=n_init, max_iter=iters)
8
```

```
Evaluation       Proposed point           Current eval.          Best eval.
init     [-1.73544125  1.14654737 -0.25227579  0.91531337].       -675.0078283253545
init     [ 1.95784504  0.15767909  0.00458942 -1.75287856].       -1660.838563394103
init     [-9.48473938e-01 -4.81276619e-04  7.34126064e-01  1.24411509e+00].       -18
init     [-0.48766512 -1.77792472 -0.86775563  1.67769509].       -2128.346574385609
init     [-1.17397359 -0.19610025  1.76621986 -1.94601276].       -3119.082442947162
```

1	[1.3841121 1.10038019 -0.7618778 0.29747336].	-466.9154980081061
2	[-0.46462418 -1.01739574 -0.63627658 -1.21236035].	-701.8724792881038
3	[-1.46678761 0.45704385 1.94337688 -0.92807062].	-2808.720764811721
4	[0.88038975 0.42831332 -1.19910716 -0.98216741].	-794.0017025703881
5	[1.1357617 -0.39564146 -0.88860615 1.99053848].	-543.1082177327809
6	[-1.69941301 0.52314958 0.44742864 -1.63861248].	-908.2124698408273
7	[-0.89608508 -0.10949347 -1.06104869 0.94295889].	-210.81764308842642
8	[0.22065276 1.61933938 -0.32876869 -0.63556608].	-1175.6105963434723
9	[0.88589231 1.97042035 -2.00774073 0.91601832].	-4590.454863252045
10	[2.0309589 -0.60175451 -0.05136673 -1.39521684].	-2451.256149086253
11	[0.99588388 1.9687159 -1.0789958 -1.90873983].	-3500.057706047198
12	[-1.46915487 -0.99321291 1.31221829 0.19699703].	-1246.592726306397
13	[-0.40008565 -0.92090559 -0.11573164 -1.38791558].	-413.0030842698699
14	[-0.8124053 -1.03583082 1.46649275 -0.71060039].	-1129.3673394317823
15	[0.44977671 1.70397743 0.71330121 -1.63729986].	-1166.6723629796802
16	[0.29053641 0.21600328 -0.96070099 1.09997824].	-111.30510281931322
17	[-1.50508243 -0.57055302 0.39945733 -1.0990935].	-972.2626464144971
18	[-1.24710295 0.95762102 -1.73052056 -1.50459248].	-2773.546119260599
19	[1.57724724 -1.43419739 1.0000704 1.67615355].	-1701.786289068928
20	[1.30254556 0.86457258 -1.9212007 1.68184436].	-1193.738652806666
21	[-0.09917108 -0.39295794 -0.47187016 -1.06023994].	-225.34614242670636
22	[0.1774199 1.74048623 0.48743567 -0.60968492].	-1011.4502620486527
23	[-0.31346838 -0.90223615 -1.67879094 1.18131619].	-1002.0194042116342
24	[1.09755539 0.06000333 0.40179989 0.72821809].	-180.2471447384382
25	[-0.4259852 1.41530979 1.2169289 0.21733628].	-375.96080556137713
26	[0.36525072 1.67723443 1.54157949 -0.61270067].	-1294.6877030790056
27	[1.69057861 0.50218671 0.64506916 -1.67216904].	-1007.3907003780754
28	[-1.74730148 -1.48302988 -1.77713273 0.9247701].	-4159.123649942196
29	[1.29833014 -1.73994269 -1.37322401 -2.00185139].	-4634.594219963464
30	[-1.37706882 -1.78891816 -1.78812293 0.9528631].	-4371.451001526354
31	[-1.34271262 0.64109026 -0.54198534 -0.83191675].	-360.49934374026805
32	[1.10499416 1.00488362 1.58181641 -0.42751198].	-896.0299290520137
33	[0.28125263 1.49538692 -1.05695459 0.37020686].	-1345.846818341517
34	[-1.4033467 0.70593538 0.53780605 -0.57026152].	-239.7351017763423
35	[1.1833041 1.98553896 0.30785905 1.872358].	-1672.687784979123
36	[-0.28567618 -0.1544846 -1.63190891 1.11871168].	-528.1675091303345
37	[0.60536907 1.13095515 0.8477285 -1.307918].	-487.9394065197489
38	[-0.60314838 0.30745523 -1.78135793 -0.53039286].	-1734.6847866522687
39	[-0.88676012 -1.24914685 -1.92922663 -0.26139304].	-3235.9225444273907
40	[0.37336764 0.79031767 1.15239768 -0.30730899].	-338.11584891777335

```
1 ### EXACT GP EI GRADIENTS
2
3 np.random.seed(run_num_8)
4 surrogate_exact_8 = dGaussianProcess(cov_func, optimize=opt)
5
6 exact_8 = dGPGO(surrogate_exact_8, Acquisition_new(util_grad_exact), objfunc, param)
7 exact_8.run(init_evals=n_init, max_iter=iters)
8
```

Evaluation	Proposed point	Current eval.	Best eval.
init	[1.52956683 1.91914255 1.51222084 0.12638491].	-957.0835831104033	
init	[-1.09474477 -2.0013105 -0.28479972 -0.39996883].	-2902.49459403014	
init	[0.09287545 -0.0885072 0.22674012 0.17770913].	-9.936266004845226	
init	[1.06862828 0.86988626 0.49021787 -0.30272811].	-44.27508698465467	
init	[-0.86394869 1.94091107 -0.68086151 -1.15179085].	-2389.276657793903	
1	[-1.63681569 0.94682494 1.80081066 1.81583128].	-593.1371362732267	
2	[2.0241536 -0.6796886 -1.83302627 0.24753545].	-3789.198897659005	

3	[-1.25453515 0.6570893 -1.93700116 1.94051986].	-987.1154357131456
4	[1.61665153 -1.94173538 1.76024883 0.7901011].	-3021.590236385825
5	[1.61690965 0.28689875 0.72519754 -1.78512954].	-1118.110716909711
6	[0.38306717 0.28451289 0.20902794 1.06419821].	-109.19990161751558
7	[-0.70002706 1.15319997 2.04529378 -2.01252415].	-3937.9004813257
8	[-1.77290669 -1.83187185 1.97229377 1.78585986].	-3125.898820336325
9	[1.16552122 1.89824598 -1.54249073 0.81909326].	-2927.810244765299
10	[0.10812383 -0.51758335 -2.03090377 -1.90582035].	-4205.305325106837
11	[1.35410927 -1.84798102 0.27464388 1.23773193].	-2485.474846971902
12	[-0.64051136 0.65756381 -1.43646501 1.19262726].	-439.9530863865007
13	[-0.14971546 0.50775422 0.63444506 0.46089506].	-39.77905313856528
14	[-0.06069706 1.17252573 -1.39191537 0.21232906].	-1206.5725167417554
15	[1.94676455 0.73444973 0.5608725 -0.22515299].	-963.9095824792869
16	[-1.77262273 -2.03776117 -0.16499829 -1.11504552].	-4695.995287267862
17	[0.20993332 0.5569241 0.8493887 0.54120142].	-59.4705694381346
18	[1.56847741 0.24316373 -0.69881531 -1.01761465].	-779.5107598624428
19	[0.35484129 -1.86816284 -0.14451651 1.179078].	-1862.7217634316391
20	[0.79611724 0.59266256 -1.75214302 1.71347332].	-634.3946569537112
21	[0.06425197 1.96587622 0.90111521 1.97688995].	-1400.6239448847728
22	[1.96447046 -1.1253884 -1.00393877 -0.3542284].	-3195.0460205517234
23	[-1.90505287e+00 -3.38086882e-01 -1.13311058e-03 -6.45533451e-01].	-16
24	[0.06932344 -0.82987684 1.31480695 1.92426221].	-117.00853922429148
25	[1.94956957 1.98306302 -1.11386725 -2.04089036].	-3960.2663481731424
26	[-1.94029947 -0.86847316 -1.27079134 1.67313391].	-2574.3963586969853
27	[0.3863695 0.15426794 0.8482894 -1.47726505].	-551.715179290625
28	[1.27554452 -1.33772913 -0.69806032 -0.96771911].	-1717.9042203681279
29	[1.22309047 0.50266748 -1.68360001 -0.00168864].	-1285.4775424217082
30	[-1.2854068 1.68828179 -0.69504045 1.83299086].	-1447.8658969833066
31	[-1.62809426 -1.9138781 1.87257031 -0.31780317].	-3882.7709730274632
32	[0.01115185 -1.04402107 -0.98431878 -1.62553599].	-1221.4900994190507
33	[-0.78268382 1.59684608 -0.70097452 1.85061772].	-1344.8892697299098
34	[-1.59478187 -1.15323929 -1.63623257 1.79534308].	-2342.3884903861576
35	[-0.69785189 -0.2136039 -0.78857984 1.22944205].	-163.14461373860414
36	[-0.12161189 0.78179848 0.84265519 0.85241509].	-67.54368335296114
37	[-1.99255462 -0.04453909 0.05325667 -1.23906119].	-1777.308949498485
38	[-1.84338287 1.14762359 1.95274426 -1.61966174].	-3507.4826376379906
39	[0.52225599 -0.55066016 0.5964857 -0.79658913].	-211.9953091401932
40	[0.0662418 0.0768072 1.03349554 0.33770486].	-161.19477034947843

```
1 ### EXACT GP EI GRADIENTS
2
3 np.random.seed(run_num_9)
4 surrogate_exact_9 = dGaussianProcess(cov_func, optimize=opt)
5
6 exact_9 = dGPGO(surrogate_exact_9, Acquisition_new(util_grad_exact), objfunc, param)
7 exact_9.run(init_evals=n_init, max_iter=iters)
8
```

Evaluation	Proposed point	Current eval.	Best eval.
init	[-2.00550747 0.00767833 -0.01731259 -1.49983425].	-1847.6492499405208
init	[-1.46591299 -1.15278366 -0.33379049 -1.03177761].	-1509.7528385404728
init	[-1.70369167 -0.63283757 -1.36488408 1.55057801].	-1586.883400641493
init	[1.84714867 -1.88928665 0.81554388 0.2980242].	-3591.2693050622433
init	[1.63023716 0.68361819 0.19594356 0.82914273].	-460.72675701107613
1	[1.52944985 1.66411173 1.05790663 -0.6270896].	-644.1181418882177
2	[0.36691068 1.35867799 -0.37546878 -1.8128426].	-1027.4883118151865
3	[1.91082423 -1.54166834 -1.15314338 -0.10428515].	-4160.216262988813
4	[1.8941372 -1.08190004 1.39260512 1.90549007].	-2190.903816870602

5	[-0.77748103 -1.36916658 1.33978041 1.79642139].	-427.0198710621125
6	[0.43540538 1.01380899 -1.90982985 -1.97512192].	-4101.0253414933
7	[-0.53922054 0.42288161 -1.65103556 1.59059735].	-475.2116658321254
8	[-1.50294447 -1.23674694 0.70406135 1.57347334].	-1417.570625638267
9	[0.93681476 -0.46414781 0.86557214 -1.01655719].	-536.263637801552
10	[1.2087894 -0.20026847 -1.51486286 0.34947216].	-904.0741546959051
11	[0.77052113 1.14973778 -1.5407734 -0.35713227].	-1602.8351337940483
12	[-0.57410591 -1.15922655 -1.47822393 -1.45918156].	-2359.4391147680835
13	[0.94016196 -1.35872001 0.53165354 -0.50916564].	-744.2032650905579
14	[-1.37454807 -0.82899214 1.16351592 0.83737527].	-797.3177408401876
15	[1.79796123 1.63136227 1.09539079 -0.19522935].	-697.3152311893124
16	[-1.22972822 1.34093876 -0.72463508 1.99718201].	-864.1278242321243
17	[0.9448242 -1.63675234 -0.80095414 0.52073276].	-1862.4463434291495
18	[-0.45272639 0.24594195 -1.37697682 -1.70214641].	-1509.8399823581478
19	[-0.82137149 -0.0716439 1.76308515 0.19802656].	-1216.8510398242117
20	[0.15589795 0.83672932 0.91785406 -0.67661491].	-302.24803204814896
21	[-0.39164466 -0.83932236 0.5986962 -1.42650476].	-423.74812423174933
22	[-0.30577308 1.59623401 -0.09521891 -0.07634192].	-928.4523825651771
23	[1.02842179 -1.06051989 -1.41518039 -0.12016884].	-1554.5173837772443
24	[1.089408 1.54077152 -0.65082177 -1.69000317].	-1377.213943907618
25	[-1.10293183 -1.23710514 -1.70721259 -0.99154435].	-3192.762557425501
26	[1.76694719 -0.66753681 1.5391013 0.67821062].	-1845.1897093751204
27	[1.55595475 -0.04915379 1.86520262 1.8685069].	-1218.6819553360144
28	[-0.16132052 1.87369501 0.42650489 1.87382795].	-1581.3357400333223
29	[-1.97729446 -0.30654351 0.20585548 0.68478427].	-1831.3879932990244
30	[0.53323654 -0.34523495 -0.90883618 1.48485433].	-194.40211150767075
31	[-1.26346914 1.78296691 -1.07923439 0.06511644].	-1947.6919928049658
32	[-0.24277585 0.08353605 -1.35071151 1.57855697].	-198.34783344424238
33	[0.79555608 -1.11052001 -1.44027944 -1.73988948].	-2484.0683153676932
34	[1.08857292 1.4669949 0.92814686 -1.956494].	-952.068037210575
35	[0.62710801 -1.37936141 1.87632722 0.78688539].	-1068.1794944818805
36	[-1.22825293 1.74001646 0.68661992 1.13566625].	-603.1303835990793
37	[0.22168689 -1.83113965 0.22192424 -0.32233273].	-1356.9898885879918
38	[0.24948113 1.19516488 -1.28774391 -1.39570007].	-1804.6230797860508
39	[-1.04158977 -0.41917658 1.97191901 -1.03991655].	-2984.8853118097572
40	[1.29290756 -0.74402798 1.47624219 2.00947067].	-674.8997870657305

```
1 ### EXACT GP EI GRADIENTS
2
3 np.random.seed(run_num_10)
4 surrogate_exact_10 = dGaussianProcess(cov_func, optimize=opt)
5
6 exact_10 = dGPGO(surrogate_exact_10, Acquisition_new(util_grad_exact), objfunc, param)
7 exact_10.run(init_evals=n_init, max_iter=iters)
8
```

Evaluation	Proposed point	Current eval.	Best eval.
init	[1.11132935 -1.96300002 0.54742317 1.0191007].		-2176.436426889387
init	[-0.00611528 -1.12723294 -1.23673451 1.0671338].		-787.6933551437477
init	[-1.35532201 -1.68616012 0.75923382 1.85709915].		-1852.2892745062024
init	[-2.0318279 0.04993951 1.28049546 0.46090677].		-1975.7725172569656
init	[0.90830978 -0.85247562 1.71120281 0.87890241].		-802.2371849082838
1	[-0.81633257 -1.58112005 1.34627871 -1.85591268].		-1994.0093456433356
2	[1.07342562 1.39217912 -1.61809444 0.04354584].		-1940.366717616157
3	[-1.43008917 -1.54327785 -0.09024326 -0.99359866].		-2012.6496423727694
4	[-0.36623728 0.66682619 -0.29800586 0.3236784].		-92.71025043943006
5	[0.94867509 -2.00602812 -1.86802941 0.10104421].		-5481.717992918238
6	[-1.60488548 -0.26364021 -1.58688116 -0.53910029].		-2030.3006701793129

7	[-1.79898003 -0.90849921 0.74067811 1.84460761].	-1898.183921518177
8	[1.79248298 1.92137503 1.88100286 -0.11875547].	-1834.248967261895
9	[0.09221679 1.75138084 0.80469637 1.1725351].	-844.7044503699319
10	[0.25385502 -0.56137304 -1.02847848 -0.94794006].	-629.0912858575092
11	[-1.08914015 -0.86926364 0.27263031 -0.44518991].	-481.2092416961812
12	[-1.47012591 -0.73192944 0.84283469 -0.53623689].	-1011.021074482824
13	[-0.95001787 1.76383678 1.45961371 -1.58350427].	-1730.8910014293738
14	[0.90499836 1.10141346 1.28603698 0.54893986].	-130.69915689422186
15	[-1.79535365 1.85121901 -2.01339918 -0.3682827].	-5121.14606532571
16	[-0.35390856 -0.84421579 -0.44549616 1.32577765].	-362.535168310281
17	[0.82928393 -0.24396943 1.8187735 0.20578126].	-1360.8839400930592
18	[-0.64646938 1.72808184 -1.32611196 -1.14608907].	-2883.670679994826
19	[1.77563892 -0.90998601 -0.54683595 1.45418423].	-1979.8175305753468
20	[1.48857256 0.88094911 -0.84988613 -0.64237975].	-632.4811382199022
21	[-0.38511662 2.04739557 0.37558093 -1.28295932].	-2023.214311202851
22	[-0.70082677 -1.96912104 -0.68708509 1.5540179].	-2820.396617607148
23	[0.01423615 -0.75885146 -2.00485552 -1.56799862].	-3858.6710755636245
24	[-1.295827 1.12800061 -0.65775574 2.02362631].	-664.0812506500738
25	[-1.74537717 -0.72709154 -2.02712967 -0.4807217].	-4203.558594573616
26	[0.27699119 -0.52941187 0.98432743 1.32497665].	-101.84990573719982
27	[-1.09101046 -0.78721212 -0.18121539 -1.02716706].	-576.5269503988607
28	[-1.84683074 0.85525047 0.09269607 1.39674932].	-895.5226497615581
29	[-1.10782382 1.08678957 1.83591093 0.79947953].	-711.0490742497022
30	[1.8866903 -0.12687083 0.40402276 1.86449239].	-1665.8947477686202
31	[0.08049474 -1.5649912 0.34922937 -0.81227962].	-783.0673403408462
32	[1.34205403 2.01618533 1.56150399 -1.66922673].	-2320.0148127725156
33	[0.86506562 1.08373241 0.70537686 0.11630819].	-47.901348809022956
34	[1.45206965 1.18898007 1.01897055 -1.15126377].	-579.7917141619665
35	[-0.52727733 -0.75829671 0.98954076 -1.48721947].	-738.3210826217644
36	[-0.75155978 1.3610957 1.14781785 1.31613751].	-116.29154433379108
37	[2.02714091 0.73764233 1.18728261 1.92488869].	-1205.8810803418426
38	[0.15851054 -0.76533319 -0.06448037 -0.23312002].	-115.34819415249515
39	[-1.42267809 1.18387659 -0.61381498 1.74883739].	-673.5228586952437
40	[-0.82339488 -1.42360596 1.4478472 0.39763342].	-773.1006601987814

```
### 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	[-1.30961535 -1.96822941 -0.15065692 0.92132937].	-3072.6361628465456
init	[-0.32684604 -0.05969061 -1.99564978 -0.0517259].	-2041.908514926983
init	[1.80964005 1.43685669 0.94193447 -1.60261705].	-1084.6194797620587
init	[1.61343148 1.4629038 -1.37180521 0.54204012].	-1549.0827052305292
init	[-1.96409912 -1.56984415 -0.75215949 -1.40119119].	-4385.783412609677
1	[1.83952342 1.99341396 -0.66333061 -1.06547331].	-2574.640397190391
2	[-0.50547507 -1.49511739 -0.59683872 0.57939828].	-1124.6348986589117
3	[1.75448183 0.59941317 1.722094 -1.87233294].	-3141.9815488207764
4	[1.79525537 -0.77193875 0.07904719 -0.83005758].	-1697.18081013564
5	[0.08235597 1.20351735 -1.38078313 -1.82955343].	-2346.083214352141
6	[-1.96031946 0.99863466 -1.22105629 0.79968669].	-1362.5401457795631
7	[1.32122758 -1.57765635 1.52175549 -0.96298413].	-2280.010270656378
8	[1.35923303 -1.24567748 -0.76853718 -1.15385311].	-1807.7681589140548

9	[0.28340268 -1.24036592 -1.99635595 -1.88565185].	-4885.425833430284
10	[0.00333095 -1.11448453 0.85634639 -0.32611189].	-256.8153273481155
11	[1.45797774 -1.45580172 -0.7585601 -0.09951625].	-2165.842668064896
12	[0.69901164 1.51409958 -0.05184924 -0.14638403].	-658.4411064717772
13	[0.861473 -0.43280607 0.46623865 0.24889164].	-148.2846556243542
14	[-1.70615562 -0.5940618 0.52920458 0.51918329].	-1247.434752084834
15	[-0.37830153 1.4817917 -1.66154637 -1.03228681].	-3114.964527163649
16	[-1.78058023 -0.56425788 1.90591314 0.77617674].	-2473.700423571399
17	[-0.45001579 1.91236701 -0.65876847 1.49172643].	-2272.642899710617
18	[0.12524724 -1.95276106 1.81441315 1.78963802].	-1022.908886961554
19	[1.09908957 -0.43405758 0.61859251 0.40766764].	-290.4150274779587
20	[0.36103998 0.36582851 0.27979328 -0.22574481].	-18.248042387909766
21	[-1.97672755 1.41145903 0.92901769 -1.88026378].	-1497.662562738795
22	[1.75283337 1.24073503 1.75641149 1.78801317].	-509.6266415521887
23	[0.51683445 0.5694324 1.20893699 0.63536788].	-156.12259244782246
24	[1.72074709 0.38575114 -0.43076513 0.28116466].	-700.6234481962055
25	[-1.52142811 -1.53003023 0.99039518 0.91216498].	-1673.870762111767
26	[-1.42079613 -1.38032598 -1.31102534 1.61117467].	-2207.811526999247
27	[-0.88544558 0.3188642 0.27497829 -0.10211445].	-32.34306084978883
28	[-0.70003402 -2.0232322 -1.40257619 1.08192701].	-3748.485117019409
29	[-1.77840172 -1.11020889 -1.99052581 -0.45637993].	-4838.110916540896
30	[1.34553372 -0.06390034 -0.99376889 -1.73517952].	-1197.460412286185
31	[-0.28513591 1.70128455 -0.35440022 0.21816641].	-1322.719056943745
32	[0.2861566 -1.26320465 0.4953292 -1.70534049].	-688.4113843347533
33	[1.48161712 0.84992301 0.13479361 -0.9666187].	-313.4824398458839
34	[1.59888422 -0.78668892 0.27709365 -0.43610009].	-1159.704868244975
35	[-0.29021798 -0.77532859 -0.74730451 -1.16249637].	-559.7522407393208
36	[0.05360964 1.85146916 -1.71137361 1.11723377].	-3320.132125181345
37	[-1.57643092 1.74419672 1.13557999 0.27545106].	-528.4755615940201
38	[1.20334288 0.20165943 1.35940674 0.92703019].	-414.8765148103328
39	[-0.1140311 -1.40990782 -0.03519657 -0.91024241].	-702.9354119031176
40	[1.91214941 -1.81467173 1.45340086 0.88242509].	-3491.832227753870

```
1 ### EXACT GP EI GRADIENTS
2
3 np.random.seed(run_num_12)
4 surrogate_exact_12 = dGaussianProcess(cov_func, optimize=opt)
5
6 exact_12 = dGPGO(surrogate_exact_12, Acquisition_new(util_grad_exact), objfunc, param)
7 exact_12.run(init_evals=n_init, max_iter=iters)
8
```

Evaluation	Proposed point	Current eval.	Best eval.
init	[-1.416549 0.98324356 -0.9694617 0.13819656].		-553.611160809845
init	[-1.98830095 1.71518775 1.64132804 -1.91110583].		-2800.587326666160
init	[1.87166448 -1.48599062 -0.88543907 0.43451672].		-3468.879783124874
init	[1.81954616 1.44480478 -2.03874618 0.08694181].		-3716.955397581154
init	[0.21314615 -0.05989411 1.0982775 -1.38970418].		-796.5750150214192
1	[1.8444842 1.09558026 1.3312379 -0.38240132].		-998.7882904772749
2	[1.66129593 0.87073672 0.5816198 0.10572224].		-366.0502544279664
3	[1.44989773 0.86205149 1.70214109 1.20890877].		-531.5430041745518
4	[0.74784519 -0.87913124 0.68208437 1.38327915].		-295.7002912929996
5	[0.62898362 -0.72891718 -1.22436402 -0.53895578].		-858.129586262932
6	[-1.70973589 0.68633941 -0.37528478 -0.0720933].		-585.8498109295922
7	[-1.38708707 1.21965328 0.90740933 0.51329934].		-98.63944356233895
8	[1.86689185 -1.93569271 2.04039957 1.10854027].		-4173.482040007393
9	[-0.11129628 1.41579596 -0.63186892 0.16351787].		-901.6165373650053
10	[2.02281781 -1.96033515 -1.49693588 -0.58386694].		-7328.152102380102

11	[-1.64158863 0.88855344 -1.86185311 -1.04514634].	-3079.912385424591
12	[-1.57870863 1.8513935 -0.303098 1.40030235].	-1613.205379457244
13	[1.6802252 -1.8192295 -0.01445017 1.38147967].	-3460.334063745846
14	[-1.16680679 1.27539561 1.70702369 1.91886518].	-105.67273264192974
15	[-1.18355784 -1.03286419 -1.59607503 -0.21520693].	-2080.241860998364
16	[0.96578729 0.9654606 -0.43353335 1.07326823].	-267.042357821699
17	[-1.59936356 0.05202077 -1.68099862 -1.36931641].	-2686.1678997041663
18	[-1.93196624 1.1203768 -0.36986941 0.15225688].	-956.9261517599865
19	[0.7072878 0.6559028 -1.10801994 0.4179542].	-309.2552412962974
20	[-1.78114909 -0.69041045 -1.87445089 -0.16824127].	-3419.402257969233
21	[2.03701021 0.84977593 1.88576638 -0.40654752].	-2796.318245152882
22	[-1.93415834 1.60900614 -1.1198731 -0.56720358].	-2175.221159522027
23	[-1.2991774 -0.66905575 1.72276999 -0.06727684].	-1647.9491977105322
24	[0.86179156 1.80144989 -1.16981236 1.63366197].	-2073.7536293335856
25	[0.47150925 -0.19129195 1.71696357 -0.56016155].	-1532.3791517552193
26	[-0.92510619 0.21502146 1.34712889 1.00119717].	-280.9257146505063
27	[0.73593244 -0.93145875 1.79902655 -1.02205019].	-2121.7026042201296
28	[0.58013731 -1.25518042 1.01801776 0.76034697].	-297.32048885186435
29	[0.49248519 -0.36364078 0.1241312 -0.46128554].	-62.360242192976955
30	[-1.69090131 1.78595323 1.92890053 1.11252811].	-963.0734459791341
31	[-1.83621103 1.87903779 -0.68162285 -1.26546617].	-2308.192588094473
32	[1.92156215 1.40944334 1.51577691 -1.70996176].	-2150.6722660038395
33	[0.0919898 -0.08568237 -1.11132262 -1.62208438].	-948.8029306342575
34	[-0.1542413 -0.44101939 0.49396321 1.62342477].	-224.51899594338653
35	[-1.89442692 1.28025951 1.41878095 0.88925994].	-672.7102189249384
36	[-1.58683532 0.93332869 0.12508988 -0.19690846].	-318.7661115293826
37	[-1.40446844 -0.06409175 -1.4736892 1.11702579].	-757.4511493361595
38	[-0.22075864 0.21300348 -0.04241919 1.59092653].	-259.1978979660242
39	[0.78784188 0.56687636 1.60738668 0.53712367].	-585.1246700385428
40	[-0.908832 -1.12747314 -1.62090109 -1.33948435].	-2806.611413418693

```
1 ### EXACT GP EI GRADIENTS
2
3 np.random.seed(run_num_13)
4 surrogate_exact_13 = dGaussianProcess(cov_func, optimize=opt)
5
6 exact_13 = dGPGO(surrogate_exact_13, Acquisition_new(util_grad_exact), objfunc, param)
7 exact_13.run(init_evals=n_init, max_iter=iters)
8
```

Evaluation	Proposed point	Current eval.	Best eval.
init	[1.13746907 -1.07503116 1.32824487 1.90770872].	-570.621446040192
init	[1.93577416 -0.19067188 0.44663793 1.1285566].	-1656.450565242978
init	[0.58004826 0.90938667 -1.9044904 -0.82555097].	-2770.1364153691793
init	[-1.80833283 1.46252162 -0.5207899 0.73665721].	-1066.23070619416
init	[-0.99827733 -0.62430734 -2.00944529 -0.58026483].	-2986.7442974027663
1	[1.8660317 -2.04795071 -1.03637524 0.86930505].	-5812.2868484529
2	[-1.99093343 -1.57541334 -1.27558555 1.5579621].	-4501.431269415862
3	[-1.7084272 0.04375793 -1.84819292 -1.21433545].	-3329.025935220234
4	[1.20977231 0.68296146 -1.42178533 -0.28413332].	-955.0616727742835
5	[0.73870568 0.44692721 0.01696619 0.92994009].	-92.08194989676083
6	[-1.66395119 -0.80730287 -1.84496721 0.13029281].	-2992.267160837443
7	[-1.81799706 -1.22894338 -1.59166549 1.27851928].	-3195.085476637445
8	[0.13435789 -1.36913607 1.48107155 -1.66219771].	-1701.200626347697
9	[1.49280048 -1.10459031 1.9369463 0.58144526].	-2172.9421677440087
10	[-1.76484225 1.52616774 -1.96596454 -0.41212094].	-3943.275554929836
11	[-1.02843701 -0.74279767 -1.33933376 -0.48197864].	-1212.340125948955
12	[1.90955169 -1.50253651 1.13409201 1.80691212].	-2811.5983744767746

```
13       [-0.15542593  0.77543219 -0.32110269 -0.50675733].       -181.8473682162925
14       [ 0.50991521  0.22600793 -1.46246783  1.59299728].       -265.8925968256384
15       [ 0.98814807 -1.58991753  0.55997626  1.08080517].       -1111.631134123834
16       [-0.23675551  0.13543324 -0.00873668  0.81396943].       -70.24023282169031
17       [0.15455253 1.64946528 1.57544879 0.02949059].           -998.3859062868203
18       [ 1.24746474  1.90552036  0.8662623  -0.00290784].       -834.2343974023479
19       [-0.13364844  0.3340454  -0.34606933  1.03498846].       -118.2462124513074
20       [-1.29091789 -0.9704555   1.94901953  1.88212017].       -1174.140145161568
21       [ 1.53455042  1.97398803 -0.19739056 -1.64338008].       -1976.3005900039768
22       [-0.54607875  1.50796225 -0.51658438  1.23602916].       -1023.9376977250108
23       [-1.69647431 -0.13849044 -0.85437537 -1.77819124].       -1627.3331486126021
24       [-0.36314152  1.48132021 -0.81174684  1.60396981].       -1180.4203342459014
25       [ 1.21516934  0.30106952 -1.50517253  1.22133599].       -508.7062517996634
26       [-1.74787857  0.92605445  0.44786136  1.67306986].       -694.7452976294811
27       [-0.7642521  -0.55159498  0.98612647  0.44030284].       -209.3084449600003
28       [-0.57801656 -0.89346951 -0.48374123  0.33465638].       -324.3432054652136
29       [-1.55232093 -0.83414065 -0.91616803  1.68889364].       -1397.811662780371
30       [ 1.25552684  0.98228179 -0.01194804  0.60380272].       -168.2404221329065
31       [ 1.86866832  1.80735286  0.88218609 -1.31606209].       -1292.3195428932797
32       [-0.60816258 -1.48369319  1.19762397 -0.3389362 ].       -767.5440279170598
33       [ 0.9372928  -0.43945546  1.53552677  0.75343429].       -613.6859570054377
34       [-0.66508622  0.66638952  0.87959599 -0.22493756].       -126.61151527058537
35       [-0.07484835  0.02857783 -1.34884795  0.8967764 ].       -274.95003109985123
36       [0.96418378 0.51201859 1.579638   1.98711879].           -217.4114104120697
37       [-0.03576572  1.04072633  0.05068127 -0.63198527].       -256.8777483425788
38       [-1.69813972  1.3940542   0.19151705 -1.17115017].       -682.7768916967663
39       [-0.2194799   1.51724609 -1.91465659  1.68166585].       -2397.8397335966574
40       [-0.2367588   1.06158386 -0.87110555 -1.24039047].       -905.0561957506699
```

```
### 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     [ 0.05711194  1.11888405  1.5172718  -2.0150397 ].       -1996.4555487728376
init     [-0.77932165  1.87434492  0.05372605 -0.744307  ].       -1417.9597465038332
init     [ 0.16056294 -1.14173976  1.25534764 -0.64612505].       -635.6383005184798
init     [ 0.15928873 -2.02394097  0.70923255 -1.18774062].       -1863.0160368222728
init     [ 1.7717559  -0.51509351  1.0339079   1.07781735].       -1397.2995170993763
1        [ 0.99332651 -0.37717683  2.01270476 -0.19604562].       -2342.516103819445
2        [-0.97786591 -1.99526909  0.19679333 -1.95785246].       -2715.3884769596207
3        [-2.03313668 -0.93462245 -0.00775355 -1.32427274].       -2835.7413982504854
4        [0.0398087  0.42157067 0.33954241 1.06185617].           -111.54909229033513
5        [-0.26998621 -0.47520129 -0.51714112 -1.39745282].       -368.5151759305819
6        [ 0.04971751 -0.84388258  0.45380972  1.62586983].       -284.52532843827254
7        [ 0.98214632  0.84754107 -1.43565246  1.56748659].       -495.6539924024287
8        [-1.76432072  1.75373419  1.76096656 -0.17849612].       -1441.8355353693546
9        [-0.25013786  0.58131844  0.74150156  1.21362059].       -89.06448998253701
10       [ 0.52069338 -0.46443599  1.37184847 -0.8653678 ].       -945.0703467819066
11       [ 1.03203137 -0.02147058 -1.41823377  0.92455831].       -444.3425429441072
12       [ 1.39730886 -0.01876222 -0.41538171  1.80457934].       -675.413538554842
13       [-1.83683859 -2.00726755 -1.26585638 -0.1341597 ].       -6023.245600609232
14       [-1.39208894  1.36606172 -0.85438521 -0.2751843 ].       -883.147772405993
```

15	[-1.2605655 0.59505477 1.88176971 -1.84227427].	-3236.255717077692
16	[-0.70663261 -0.788236 1.25868644 -0.82944904].	-795.1983064011594
17	[1.2943011 -1.65891147 1.28764649 0.76212883].	-1413.573229450830
18	[-0.85939116 1.79062222 0.63035876 -0.70415994].	-899.798599656544
19	[1.54956234 0.2443581 1.59720447 -0.44513779].	-1600.51201005209
20	[-1.59175313 0.59075894 -1.18001661 0.91206296].	-645.9947182472462
21	[-0.02537064 1.72531768 0.18305542 -1.16579367].	-1223.984452437756
22	[-1.26484967 1.1756304 0.57576666 -0.62946688].	-180.70171640777548
23	[1.07007622 1.97358001 0.14598518 -1.34343711].	-1662.104814084645
24	[0.64750203 -1.51918159 1.91248765 -1.53072617].	-3090.576598668522
25	[-1.10900046 -1.28899246 1.80112137 -1.10237781].	-2535.884617135554
26	[-0.44884526 -0.20569578 -1.34067759 -1.77079305].	-1490.086931665832
27	[-1.64083647 -0.08188128 0.53807534 1.50857328].	-954.8339772285511
28	[-0.32302589 0.4282859 -0.1091979 1.49674112].	-242.83247253126913
29	[0.27633111 1.37392905 0.39697562 1.34790169].	-533.300532385822
30	[1.26069459 -1.83565215 -0.15960944 0.48859879].	-2449.511686556218
31	[0.41185065 0.14704051 -0.06999008 -1.30219395].	-173.9576811616067
32	[0.75138373 -1.42916731 1.02061145 -1.76094883].	-1293.349220758911
33	[0.67610656 -1.34656751 -1.92469412 0.22596233].	-2946.698074040383
34	[1.7588149 -0.78739493 -1.93213096 -0.323968].	-3815.785049024095
35	[-1.68214821 0.54217423 -0.6400704 0.01722999].	-635.9778142669621
36	[0.13865086 -2.01545562 -1.0926892 -1.01367946].	-3572.723172473937
37	[1.82238011 -1.1194483 -1.31534249 -1.06360491].	-3422.564769724855
38	[1.94381142 1.54603653 -1.04249393 0.23850074].	-1754.025378051432
39	[0.12795405 -1.82551184 -0.52381359 1.82879896].	-2079.051737467746
40	[0.86929247 -1.73849869 -0.68259678 0.13072152].	-2016.355654938455

```
### 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	[1.42875729 -1.31524229 -1.82532827 -0.56713852].	-3924.3244166004847
init	[-0.9199578 0.12288092 -0.79495612 -0.80087302].	-330.9329217994366
init	[-1.59030773 -1.02441364 1.71061206 -0.96605449].	-2832.720279494954
init	[0.89200102 1.49796878 1.25779756 -1.18558481].	-912.8737542795088
init	[-1.36297254 -1.85669062 -1.88652621 -1.2278546].	-6538.09169222112
1	[0.58460113 -0.32775446 -0.39631899 -0.44883834].	-110.79789128159267
2	[1.0086952 1.22074613 0.84139756 0.31633606].	-61.63957624806721
3	[1.17524651 0.81266714 0.06578144 1.87055775].	-416.9041118681516
4	[-1.37601528 2.01548466 0.00408975 1.70012238].	-1945.0028201943212
5	[1.38289251 2.04145711 -1.69811698 -0.4971158].	-4593.703203221396
6	[1.58556031 -1.16178674 1.82715306 0.55767211].	-2152.927698875604
7	[-1.9457908 -1.62753648 0.90927685 2.04416103].	-3397.158605260719
8	[-1.47661357 0.9182499 -1.01377724 -1.54885025].	-1178.208793385813
9	[-0.02651074 -0.92941848 1.00534467 0.95777283].	-93.57224973285331
10	[-0.18573446 0.83886291 1.4122079 -1.25290689].	-1170.9573549500035
11	[0.57370594 -1.10128151 -1.02032557 0.01089391].	-818.1084051893896
12	[-1.0655448 0.02289176 -1.63283341 0.71785545].	-782.2861555773686
13	[-0.4994855 1.64957649 0.57379538 -1.81250558].	-1118.6781915959964
14	[-1.08497521 0.84636257 0.49280125 -1.81616963].	-444.52535472196007
15	[1.03136549 -0.3112375 0.45622147 -1.36849795].	-452.55693422282735
16	[-1.25906869 -0.08959343 0.7303103 -0.45009119].	-435.7603115490442

17	[0.26669857 -1.68266996 1.01018804 2.03422787].	-749.7571901059255
18	[1.51360165 1.98828678 1.70815557 1.18950089].	-813.6654166220812
19	[0.23371844 0.85794796 -1.13695059 -1.46894332].	-1183.172629401706
20	[-2.01176549 0.83883047 -0.0157342 -0.16909236].	-1094.109201736202
21	[0.48578938 2.0011167 -0.76894845 -0.91579566].	-2821.641308934085
22	[-2.04286803 1.33193982 -1.84903679 1.30743019].	-2583.356388762627
23	[-1.44755121 -2.02169385 1.96860149 -0.83865247].	-4382.196027042846
24	[-1.70112205 1.62661238 -1.56239402 1.62635524].	-2012.158744072306
25	[-0.24109876 1.33577466 1.78268983 -1.85506578].	-2698.661927324604
26	[0.38949032 -1.37982152 -0.45361363 -0.78642501].	-896.9406318394908
27	[1.14701612 -0.79378236 -1.07987956 -1.48697064].	-1448.833233472687
28	[-1.43303668 -1.93865232 -1.95336332 -1.12318036].	-7318.656924336549
29	[1.03832863 -1.16145392 1.70120691 0.87975459].	-924.9044843772957
30	[1.71257789 2.00766758 2.04074396 0.40774124].	-1895.645937778416
31	[1.28582099 -1.49887299 0.06383655 1.77795686].	-1791.964482589178
32	[0.81334184 -0.16141846 -1.32866259 0.48226796].	-422.68469495337126
33	[-0.74818958 0.94692366 -0.04901244 -1.17932449].	-248.2251143656931
34	[-1.92520942 -1.82842073 -0.73296963 0.94338598].	-4760.966359727565
35	[1.06801719 0.30765051 1.6121512 0.52897123].	-729.0455971122695
36	[-1.41747201 1.65781696 -1.49014706 -0.13005055].	-2373.845117759311
37	[-0.4844244 1.37206471 -2.0436346 0.79540728].	-2825.614482426562
38	[-2.04536301 1.59960423 -1.37306242 1.3677816].	-2255.606649491872
39	[1.75116601 -1.15766379 -0.02397232 -1.2176205].	-2125.186454697834
40	[-1.25806394 -1.84417246 -1.25807903 -0.61869265].	-3847.965947792774

```
### EXACT GP EI GRADIENTS

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

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

```

Evaluation	Proposed point	Current eval.	Best eval.
init	[-1.13339974 0.09487705 0.20767317 -1.86121441].	-514.1403917485846
init	[-0.57045469 -1.13426046 0.77302236 -1.37735608].	-636.540695155299
init	[-1.75994935 1.80638048 0.26083893 -1.72854338].	-1399.5661404272828
init	[0.91193553 -1.3989799 -1.02284777 -0.8458762].	-1753.4526459251956
init	[0.80531749 -0.1463743 -1.16710546 -0.21779963].	-460.1166598787994
1	[-0.13108778 0.65969904 0.77810332 -0.96940243].	-302.4994095893768
2	[1.22573358 -1.62776275 -0.59572774 1.86643276].	-2271.007002774895
3	[0.09704324 -1.45330609 -1.31939395 0.64367914].	-1524.0512445841719
4	[1.13175832 0.95270927 0.6939976 -1.85842868].	-563.0362907121892
5	[0.87766015 -0.56988658 1.86040774 0.55181588].	-1265.0478080730422
6	[0.38324944 1.15320791 0.7361225 0.87331715].	-147.9841514380348
7	[1.52761822 1.05335325 1.86698362 1.69312043].	-543.6189085901927
8	[0.25466755 1.95777246 -1.96183909 1.6941066].	-4190.703432069714
9	[-0.63259733 1.23580709 1.66993686 0.07586134].	-810.9774526850983
10	[-0.25142728 0.6963049 -0.15196989 -1.98024137].	-484.95388046833204
11	[1.82271043 -1.73186874 0.86059175 0.55960344].	-3023.3081911634945
12	[-0.62557716 1.77178749 -1.72549473 -0.03016737].	-3472.288969003308
13	[1.35578327 -0.8312143 -0.91182455 -0.38361047].	-1124.1936478797918
14	[0.53777153 0.09512887 0.09037536 0.48936688].	-29.44280902753841
15	[-1.62152026 -1.77709901 1.61597695 2.03243669].	-2227.950104164753
16	[-1.87707151 -1.6480133 -1.77751135 -1.61557787].	-6996.651917829854
17	[1.97250323 1.79523848 0.12675634 0.0337621].	-1400.095758841347
18	[-0.84563711 -1.32015721 -1.96163582 1.97171453].	-2156.13484709627

19	[0.89972157 -1.87913879 -1.5988437 1.66792078].	-3448.549160904544
20	[-1.7958477 1.44665275 0.41079517 1.92766293].	-916.9318079383902
21	[1.41428035 -1.8667259 -1.45290667 1.16687183].	-4036.797553605628
22	[1.53805572 0.16820999 -1.68697563 -1.72537074].	-2874.914984625272
23	[-1.60757224 0.37443735 -1.22386642 1.96901563].	-708.7489798501717
24	[1.94242514 1.39930119 1.09479611 -0.08147982].	-802.8831214032763
25	[-0.37698257 -1.1846033 -0.67927416 1.96327194].	-844.7700735312142
26	[0.76097357 -0.7554043 -1.68858591 1.96761798].	-776.9536659413799
27	[0.56605901 0.76600669 -0.77466918 -1.968011].	-868.1233995856558
28	[-0.33820064 1.50267429 1.45033202 1.42829275].	-305.80554799968826
29	[1.12705548 -1.87015347 -0.18151034 -0.32981433].	-2362.517922860184
30	[0.75743389 1.00544326 0.75450434 -1.43491909].	-427.0136528381095
31	[-1.49994631 0.11419392 1.97504145 0.95615802].	-1716.1129143999592
32	[-1.58705473 -0.11125429 -0.30949866 -1.51669008].	-971.7003995977981
33	[1.16945483 -1.66995561 1.14717628 -1.43539411].	-1956.3697290903986
34	[1.66267478 -1.91018957 -0.49616571 0.87781687].	-3954.3979714222473
35	[1.12235304 -0.08782422 -0.83101542 -1.32571468].	-663.0203749314862
36	[0.61165069 1.29386844 -2.0006338 -1.22259168].	-4174.394427951614
37	[-0.88630481 -1.6866546 -1.46697368 -1.76056431].	-4018.0025992286905
38	[-1.95200933 -2.03845899 1.73236918 -2.03702153].	-6564.664954368759
39	[-0.79495387 0.97135031 -1.58578895 -1.44407326].	-2228.378864665281
40	[-0.44208983 1.99194991 0.15971221 0.97938317].	-1867.703353468525

```
### EXACT GP EI GRADIENTS

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

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

```

Evaluation	Proposed point	Current eval.	Best eval.
init	[-0.84105215 0.12528335 -1.26353086 -1.76988013].		-1340.0610302924795
init	[1.17549244 0.64034211 0.56328559 0.30966945].		-57.67854505088549
init	[-1.8879983 -0.58239548 1.82551833 -1.80205699].		-4588.535377497626
init	[1.49111646 1.545382 -1.83831075 0.62430665].		-2599.959730636226
init	[0.21197361 0.39941429 -0.06746675 -0.88888049].		-99.66183844529772
1	[0.42203482 -2.00904097 1.09846332 1.90827245].		-1400.0503426825965
2	[0.09299641 1.76547411 2.00100021 1.86818336].		-891.7493177411302
3	[1.03524528 2.01791497 0.66692259 -1.76993414].		-1740.6149776893044
4	[1.79060021 -1.63794888 0.78010485 -1.24596241].		-3060.238268647078
5	[-1.98982343 -1.90350839 -1.70179203 1.51123826].		-6489.526096787567
6	[-1.31799397 2.01514826 -0.77630285 1.85071305].		-2512.8348079919515
7	[-1.70500609 0.22690576 -1.80045119 1.65595213].		-1328.4751987733005
8	[1.86062044 -1.55616452 0.75643756 -1.41436721].		-3197.3762049513903
9	[-0.23805299 -1.5834918 0.80621095 0.72568851].		-567.2509761868196
10	[1.61865995 -1.8818946 -2.00305014 1.10989172].		-5961.044846880138
11	[-0.72789992 -0.26871933 1.28346906 0.91848902].		-268.275398760704
12	[0.52172695 -1.90390253 1.43290056 -1.37482193].		-2137.9887660512804
13	[0.69165667 0.09427389 -1.05529447 1.04542536].		-133.60784178022286
14	[-1.98093256 -1.81212483 0.95420026 0.17949747].		-3903.3545575055746
15	[0.88962951 1.27603683 1.37351323 -1.12492908].		-937.0950833794803
16	[-0.67968804 -1.44603439 1.68593438 1.83708789].		-490.7944754048813
17	[-0.82342883 -1.84286467 0.32791642 1.7467644].		-1857.4670492161945
18	[1.81352625 0.97294554 0.21198954 -0.34099275].		-606.5005846326469
19	[1.26977767 1.58392248 1.26092496 0.00980864].		-405.9632770929851
20	[-1.00359125 -1.85000404 1.26277177 0.24557093].		-1476.9998000770709

21	[-0.81518875 1.07441562 0.4356407 0.86881634].	-118.1850292493423
22	[-1.26489958 -1.73404299 1.8423527 -0.61024919].	-2864.10960753177
23	[0.461464 -1.24370359 -0.30735697 -1.17185651].	-723.3642477988516
24	[0.91600734 -0.49890457 0.24883577 -0.18646842].	-188.00510674115222
25	[-0.75299319 0.66639061 0.10740529 1.37821958].	-203.08605659574272
26	[-0.06645649 1.30454928 -1.64569473 1.21466059].	-1520.9682958377457
27	[1.30152593 0.68015254 -1.79257626 -0.43179959].	-1948.0563208246786
28	[0.67016107 0.07701834 0.81742873 0.9239938].	-87.23597287235853
29	[-0.03552169 -0.95576019 -0.82394689 -0.23735467].	-485.62770529342293
30	[-0.78940489 0.54425439 0.44427943 -1.97967394].	-480.4916998108777
31	[-1.55320406 -1.06992981 0.03407838 -1.14225885].	-1478.5284734919987
32	[-0.72040874 -1.13875366 0.00904431 -0.9410444].	-537.7200524503019
33	[-0.99120872 0.74885557 0.52254111 -1.08124627].	-193.2725609651687
34	[0.7461642 0.1028982 0.17156515 0.68392102].	-67.58128471883002
35	[0.2098577 0.21289626 0.38236609 0.6687167].	-43.1382159248404
36	[0.5251741 1.25092555 -0.04996177 0.01012974].	-357.2324422499452
37	[0.81130009 -1.51128184 -0.00363526 1.44855886].	-1211.1614914606882
38	[-0.43708156 -0.27418816 -0.9320137 -0.23362957].	-252.01085519104976
39	[-0.88266509 -0.44579317 -1.7473574 0.2161308].	-1346.8737024339018
40	[1.78607808 0.95245947 -1.33129278 0.76786254].	-1108.720591515662

```
### 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	[0.61593289 0.02233702 1.55075162 -1.30318244].	-1629.4185602457633
init	[1.44274665 1.02455823 0.68035243 1.99841976].	-361.44679331797425
init	[-0.99545734 -1.93205893 0.5559055 1.42259154].	-2000.4660253561508
init	[0.96737126 -1.96277407 -1.59087358 -0.82852355].	-4947.239458116178
init	[0.7658299 1.48122033 -1.23439366 0.64384627].	-1338.1489953342577
1	[-1.26288935 0.87759561 0.57290898 0.13600289].	-64.35538477553783
2	[-1.74446096 1.39314865 -1.6298479 -1.01881068].	-2912.5693898788636
3	[-2.03346864 1.78025771 1.7144816 1.78378342].	-910.0104665352812
4	[1.64588817 -1.61712998 1.57846447 0.25331343].	-2487.5297271973463
5	[1.95846595 -1.61619238 0.37536288 1.15289237].	-3583.047935120242
6	[-1.08847274 -0.9436667 -0.60971513 -1.62513327].	-1087.57814364623
7	[0.93115609 1.78128986 0.00350732 -1.34043722].	-1269.4360904655873
8	[0.47395153 -0.51406193 1.77573504 1.99418243].	-420.53351476551353
9	[1.72008428 -0.94304356 -1.59657826 2.04264977].	-2177.0084861325863
10	[1.73561417 -0.21946425 1.44182572 -0.94212871].	-2153.557278841074
11	[-0.9051035 0.56081845 0.04174753 1.09781341].	-138.99469085785802
12	[-0.10366618 0.58328748 0.52458188 -1.52649207].	-362.40126070791297
13	[-1.80142378 -1.30992486 1.37594861 -1.5878291].	-3311.51273381473
14	[-1.10045743 1.6279866 0.07765978 -1.53921094].	-923.690049821954
15	[-1.36911015 -1.26106733 -1.10497762 -0.42321055].	-1995.092818765097
16	[1.34874724 -0.66725668 -0.11233362 -0.10276976].	-654.7644544144575
17	[-2.03840758 -1.02306318 -0.43795472 -1.41361415].	-3174.879884791438
18	[0.92417415 -1.2766809 1.53075985 0.38222458].	-845.0282471213242
19	[-0.90884815 -1.07971857 1.82699941 -0.71338556].	-2056.863779476655
20	[1.92104964 0.85427448 1.75285377 1.75741926].	-1083.4249072866344
21	[-0.80950738 1.53213968 -1.19707279 -1.54218491].	-2226.7971902190075
22	[-0.77674369 -0.9601174 -1.23380575 -0.73873945].	-1232.3197232684288

```
23       [-0.30424501 -1.41566203  1.07316125 -1.41254683].       -979.2046094688227
24       [-1.37188790e+00 -1.91343383e+00  1.89013899e+00 -3.61196141e-05].       -3(
25       [ 1.03643217  0.37351182 -1.27503057 -0.76182749].       -824.7877668149931
26       [0.48476226 1.63508301 0.47961728 0.28471247].       -678.5739686000503
27       [-1.11289949 -0.01153869  0.60452205  0.40918519].       -198.6348501192675
28       [-0.78606796 -1.73543614  0.22726162  0.37670767].       -1350.987793773297
29       [-0.2812224  -1.06418723 -0.73216872  1.03400931].       -512.1013127048446
30       [ 1.65420563  0.46876487 -1.15497708 -1.23229633].       -1367.127518484802
31       [ 0.45561251 -1.2856581   0.69185979  0.74844961].       -328.2335184294011
32       [-1.29054067 -1.01825456 -1.0846477   0.74140891].       -1202.9159813869048
33       [-0.04444255 -1.69905763 -0.81028648  0.21299648].       -1687.521903156679
34       [ 1.35400996 -1.89016219 -1.89577984 -0.06905842].       -5735.540400611233
35       [-1.95646766  1.00206324  1.19245359  0.86058453].       -842.2963684080944
36       [-1.30456047 -1.72741921 -1.73383645 -0.6779548 ].       -4779.299516062846
37       [1.50756889 1.89855338 0.12349852 2.01809109].       -1628.7136941839224
38       [-0.25372346  0.13469482  1.91152524  1.5248555 ].       -815.4309755580788
39       [-1.01118677  1.56552334 -1.07636716 -0.43010379].       -1534.6863178834344
40       [-0.65068038  1.10009825  0.05709799 -1.60403925].       -440.72721207383967
```

```
### 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     [-1.64850237  1.07007884 -1.03654206 -1.48221261].       -1412.165722504925
init     [-0.69039488 -1.70803378  0.70441812  1.2558082 ].       -1034.97990396323
init     [ 1.97731088  0.55566637 -1.16357834  0.20081636].       -1480.569995327432
init     [ 1.86613588e-01 -1.08922440e+00 -1.58217895e+00 -1.39564217e-03].       -15
init     [-1.4249639   0.13388219 -0.46282028  0.77138896].       -422.65196913661765
1        [-1.34097732  0.17175608 -1.835954    1.33348398].       -1041.7752034870423
2        [-0.47183857 -1.08051498 -1.98692162 -0.26910445].       -2958.5580127023322
3        [ 0.95635377 -0.09333769 -1.3606779  -1.10736771].       -1171.346180444734
4        [-1.87182651 -1.16011981  1.01228468  1.7311966 ].       -2249.1066694975525
5        [-0.82657301  1.95053524  0.9782044  -1.88333914].       -1770.3793486483364
6        [-0.98994087 -0.49943038  0.93011811  1.25314217].       -286.469091399597
7        [-0.35663689 -0.46273206 -0.95591287 -1.52972143].       -776.5691012170619
8        [ 0.69425194 -0.71523599 -1.65736387  2.04458423].       -673.1736856574765
9        [ 0.09943037 -1.35763835  1.3172727  -1.33848286].       -1165.8977469871683
10       [-0.20349371  1.59029297 -0.67841032 -1.64335141].       -1715.7962272986503
11       [ 0.34129079 -1.7134077  -1.60651937  1.63754837].       -2501.6667856875133
12       [-1.04533355  0.76206533 -0.52972482  1.14332856].       -215.25607256241028
13       [ 0.58688618 -0.06914282  0.19680719 -0.94155176].       -118.84679319937815
14       [-1.95802936  0.22155234 -0.41099883  0.89883213].       -1390.6821336092044
15       [ 0.48118595 -0.64255789  0.64678184 -1.72218236].       -543.1455287201588
16       [-0.11892923  0.34503773  0.38281773  0.88220296].       -74.08684814103536
17       [1.00553932 1.91351647 1.76932332 1.640034  ].       -663.0610478743334
18       [ 1.86717929 -0.93208909  0.38747913  1.57727232].       -2183.9647223240972
19       [-0.47227969 -1.06617946 -0.03278646  0.88856394].       -389.25668712151105
20       [ 1.9488119  -1.80752078 -0.48920473 -0.57007702].       -4629.556928154609
21       [ 0.28847414 -0.14003924 -0.42148983  0.93559537].       -85.71549541447347
22       [ 1.94014221 -0.57875769 -2.04442496 -1.87321588].       -6128.625545737622
23       [ 0.51513501 -0.94655805  2.04317749  0.99268864].       -1296.0355377370665
24       [-1.42772146  0.04964094  1.11089206  1.30022216].       -525.6197156403487
```

```
25       [-0.83544783  0.36907882 -0.68553566 -1.27956431].       -391.0364371129658
26       [-1.93097095  1.84046418  0.77938391 -1.19947924].       -1372.4922405179746
27       [ 0.93135423 -0.04191946 -0.28813248 -0.77884995].       -168.12518908427677
28       [ 1.64128941  0.48224145 -1.28233846  1.33374014].       -734.142421431925
29       [-0.27694401 -0.87943929  1.25330768 -1.56633965].       -1103.8287995239782
30       [-0.63808557  0.03132333 -0.59613659 -0.75571869].       -179.40284795662322
31       [-0.05173137  0.70642684  1.09726491 -0.37487361].       -335.7968436330318
32       [ 0.71374153  2.02105291  1.40017248 -0.46276058].       -1537.6414213532244
33       [1.17135501 0.69985291 0.40068972 0.5299697 ].          -60.10743371792253
34       [ 1.47564194  0.02636049 -0.28983168 -0.78361601].       -549.3029618091022
35       [-2.02182177 -0.70114881  1.40662951 -0.08419685].       -2814.800496493994
36       [ 1.59068772  1.16671535 -0.40371861 -1.88168526].       -917.8516733150443
37       [ 0.35769955 -1.72257422  0.15444364 -0.55293506].       -1175.4466417532626
38       [ 0.37183303 -1.45548263  0.73367031 -1.47898748].       -859.1859036805786
39       [-1.23597818 -0.09473091  0.75009555  0.3176916 ].       -330.3961499944753
40       [ 1.02747587 -1.86636766 -0.64418959 -1.95688314].       -3130.985945873225
```

```
1 ### EXACT GP EI GRADIENTS
2
3 np.random.seed(run_num_20)
4 surrogate_exact_20 = dGaussianProcess(cov_func, optimize=opt)
5
6 exact_20 = dGPGO(surrogate_exact_20, Acquisition_new(util_grad_exact), objfunc, param)
7 exact_20.run(init_evals=n_init, max_iter=iters)
8
```

```
Evaluation       Proposed point           Current eval.          Best eval.
init     [0.36098376 1.62903543 1.60370987 1.29367031].          -499.42964684869025
init     [-1.90099626  0.78543905 -0.49692286  0.07582083].       -937.6489569337288
init     [ 0.6469692  -1.25398951 -0.93259202  0.8954099 ].       -916.294647636079
init     [ 1.15918278  1.43494201  1.12740309 -1.897823  ].       -1092.029190829674
init     [-1.57002246  1.02924575 -1.06816219 -1.00431457].       -1129.8679642787797
1        [-0.07788899 -0.69957055  0.04358577 -0.96817632].       -168.73719780046255
2        [ 0.46111926 -1.76732302  1.4924756   1.87484522].       -678.6495892411331
3        [-0.41017193 -1.04370263  0.10465761  1.36986945].       -435.46806705922864
4        [-0.68337633  1.04340582 -0.02893697 -1.71758559].       -457.32636617313904
5        [-0.76890696  1.43771268  0.93871157 -1.25103137].       -656.9194881080607
6        [ 0.87400662  0.25583496 -0.98932062 -1.26131996].       -643.3869394235138
7        [ 0.53628406  0.05885878 -1.79915953 -1.23751637].       -2341.2209459944415
8        [ 1.48676537  0.63274154 -0.16463738  2.03360842].       -685.1791327179618
9        [-0.03479318 -1.72142218  0.08490681 -1.4905188 ].       -1358.8910688226583
10       [-0.71679575  0.1183272   0.44250923 -0.43806832].       -78.21773357648438
11       [ 0.93174174 -1.80043821 -1.11231057  0.9157171 ].       -2630.413075041978
12       [-0.14497107 -1.88923506 -1.24191093  2.04331784].       -2719.37616469843
13       [ 1.77606792  1.00107966  0.41432244 -0.82536097].       -598.5926340581475
14       [ 1.95153786  1.39205014  1.34480077 -1.0042522 ].       -1411.4176282442563
15       [ 0.35184308  0.38504659  0.24691546 -1.49409436].       -250.98565692044613
16       [ 1.9572054   1.55274861  0.42376897 -1.22554584].       -1112.7964838243277
17       [-1.94567723 -2.03159091  0.59649137  1.63019238].       -4811.182622250396
18       [-0.46124676  1.35057717 -0.27348387  0.04303635].       -573.4146307103362
19       [-0.29968836 -0.36181771 -0.56648986  1.97334726].       -348.08624989049326
20       [-0.33426003 -0.39091638  1.4649311  -0.6427726 ].       -979.0992443779695
21       [0.93135994 0.60897708 0.52812793 0.52804205].          -15.739895623084873
22       [ 1.91938095 -1.53492843  1.86274858 -1.29906807].       -5030.332344932889
23       [-0.97155197  1.66050876 -1.54658613  2.02775762].       -1927.7557869919847
24       [-1.14859456  1.41782234 -0.00329371  1.37715507].       -601.8453472714374
25       [1.41562305 0.87537359 1.77578706 0.95501074].          -713.378163906437
26       [ 0.37538949  0.83543473 -2.04526569 -1.58776789].       -4140.755353505755
```

```
27      [-0.45927725 -1.89064134  0.5847654  -0.84621611].      -1487.3603438687906
28      [ 1.14127524 -1.81378399 -0.41349412  1.14333225].      -2447.0589023310777
29      [-1.565668   -1.65420093 -1.89373656 -1.70995388].      -6656.291951536386
30      [-1.87196263 -0.79279469 -0.91608427 -0.35078485].      -2241.778048936678
31      [-0.04876514  0.40945054  1.30688952  0.19211095].      -377.6802644324403
32      [-1.04514726 -0.18482258  0.96996067 -1.16480678].      -699.6403016208243
33      [ 0.24018362 -0.11406655 -0.02327053  0.19288687].      -9.646826040884186
34      [-1.7380717  -1.16787012 -1.20464654  0.92155594].      -2459.4341831400907
35      [ 0.37665898  1.49622433 -0.6456347  -0.86106668].      -1182.0068474323225
36      [-0.83423098  0.87355459 -1.03564615  1.6428321 ].      -366.747503263408
37      [ 0.99818601  0.88294004  1.42843553 -0.2987412 ].      -590.7562504994663
38      [ 0.13639    -1.98625677 -1.18880173  1.99566803].      -3086.1352358019253
39      [-0.37295724 -0.45068957 -1.20087908  0.35091761].      -359.81006888936713
40      [ 1.40096206 -0.67822828 -0.22506665 -2.00215088].      -1170.257381297159
```

```
end_exact = time.time()
end_exact

time_exact = end_exact - start_exact
time_exact
```

```
210.44350624084473
```

```
### 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
```

```
(4.522009786014001, 4.271178127566797)
```

```
### 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)
```

```
14
15 min_simple_regret_approx_2, min_simple_regret_exact_2
```

```
(5.210942844229998, 4.574559029861857)
```

```
1 ### Simple regret minimization: run number = 3
2
3 approx_output_3 = np.append(np.min(approx_3.GP.y[0:n_init]),approx_3.GP.y[n_init:(n_ini
4 exact_output_3 = np.append(np.min(exact_3.GP.y[0:n_init]),exact_3.GP.y[n_init:(n_init+i
5
6 regret_approx_3 = np.log(-approx_output_3 + y_global_orig)
7 regret_exact_3 = np.log(-exact_output_3 + y_global_orig)
8
9 simple_regret_approx_3 = min_max_array(regret_approx_3)
10 simple_regret_exact_3 = min_max_array(regret_exact_3)
11
12 min_simple_regret_approx_3 = min(simple_regret_approx_3)
13 min_simple_regret_exact_3 = min(simple_regret_exact_3)
14
15 min_simple_regret_approx_3, min_simple_regret_exact_3
```

```
(4.610995461278897, 4.461337277261566)
```

```
1 ### Simple regret minimization: run number = 4
2
3 approx_output_4 = np.append(np.min(approx_4.GP.y[0:n_init]),approx_4.GP.y[n_init:(n_ini
4 exact_output_4 = np.append(np.min(exact_4.GP.y[0:n_init]),exact_4.GP.y[n_init:(n_init+i
5
6 regret_approx_4 = np.log(-approx_output_4 + y_global_orig)
7 regret_exact_4 = np.log(-exact_output_4 + y_global_orig)
8
9 simple_regret_approx_4 = min_max_array(regret_approx_4)
10 simple_regret_exact_4 = min_max_array(regret_exact_4)
11
12 min_simple_regret_approx_4 = min(simple_regret_approx_4)
13 min_simple_regret_exact_4 = min(simple_regret_exact_4)
14
15 min_simple_regret_approx_4, min_simple_regret_exact_4
```

```
(5.201862175683879, 4.112184246094226)
```

```
1 ### Simple regret minimization: run number = 5
2
3 approx_output_5 = np.append(np.min(approx_5.GP.y[0:n_init]),approx_5.GP.y[n_init:(n_ini
4 exact_output_5 = np.append(np.min(exact_5.GP.y[0:n_init]),exact_5.GP.y[n_init:(n_init+i
5
6 regret_approx_5 = np.log(-approx_output_5 + y_global_orig)
7 regret_exact_5 = np.log(-exact_output_5 + y_global_orig)
8
9 simple_regret_approx_5 = min_max_array(regret_approx_5)
10 simple_regret_exact_5 = min_max_array(regret_exact_5)
11
12 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
```

```
(4.521049806967185, 4.749811120589589)
```

```
### 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
```

```
(3.8865158242765974, 4.656426686390739)
```

```
### 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
```

```
(5.194328935476146, 4.71227510466372)
```

```
### 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
```

```
(3.7008685662033893, 3.683340470693204)
```

```
### 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
```

```
(6.087740433378212, 5.269928753632324)
```

```
### 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
```

```
(3.8691436628726703, 3.8691436628726703)
```

```
### 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
```

```
(4.97961138856125, 2.904057807859144)
```

```
### 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
```

```
(4.132927927969349, 4.132927927969349)
```

```
### 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
```

```
(4.8411234634771745, 4.251921263980968)
```

```
### 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
```

```
(5.158812057918859, 4.489360713876927)
```

```
### 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
```

```
(3.764244969947392, 4.121304135802509)
```

```
### 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
```

```
(6.033670782631654, 3.3824497046465036)
```

```
### 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)
```

```
11
12 min_simple_regret_approx_17 = min(simple_regret_approx_17)
13 min_simple_regret_exact_17 = min(simple_regret_exact_17)
14
15 min_simple_regret_approx_17, min_simple_regret_exact_17
```

```
(2.9255423030280014, 3.7644092846315886)
```

```
1 ### Simple regret minimization: run number = 18
2
3 approx_output_18 = np.append(np.min(approx_18.GP.y[0:n_init]),approx_18.GP.y[n_init:(n_
4 exact_output_18 = np.append(np.min(exact_18.GP.y[0:n_init]),exact_18.GP.y[n_init:(n_ini
5
6 regret_approx_18 = np.log(-approx_output_18 + y_global_orig)
7 regret_exact_18 = np.log(-exact_output_18 + y_global_orig)
8
9 simple_regret_approx_18 = min_max_array(regret_approx_18)
10 simple_regret_exact_18 = min_max_array(regret_exact_18)
11
12 min_simple_regret_approx_18 = min(simple_regret_approx_18)
13 min_simple_regret_exact_18 = min(simple_regret_exact_18)
14
15 min_simple_regret_approx_18, min_simple_regret_exact_18
```

```
(5.203897595093871, 4.164420610036739)
```

```
1 ### Simple regret minimization: run number = 19
2
3 approx_output_19 = np.append(np.min(approx_19.GP.y[0:n_init]),approx_19.GP.y[n_init:(n_
4 exact_output_19 = np.append(np.min(exact_19.GP.y[0:n_init]),exact_19.GP.y[n_init:(n_ini
5
6 regret_approx_19 = np.log(-approx_output_19 + y_global_orig)
7 regret_exact_19 = np.log(-exact_output_19 + y_global_orig)
8
9 simple_regret_approx_19 = min_max_array(regret_approx_19)
10 simple_regret_exact_19 = min_max_array(regret_exact_19)
11
12 min_simple_regret_approx_19 = min(simple_regret_approx_19)
13 min_simple_regret_exact_19 = min(simple_regret_exact_19)
14
15 min_simple_regret_approx_19, min_simple_regret_exact_19
```

```
(4.096133523042415, 4.096133523042415)
```

```
1 ### Simple regret minimization: run number = 20
2
3 approx_output_20 = np.append(np.min(approx_20.GP.y[0:n_init]),approx_20.GP.y[n_init:(n_
4 exact_output_20 = np.append(np.min(exact_20.GP.y[0:n_init]),exact_20.GP.y[n_init:(n_ini
5
6 regret_approx_20 = np.log(-approx_output_20 + y_global_orig)
7 regret_exact_20 = np.log(-exact_output_20 + y_global_orig)
8
9 simple_regret_approx_20 = min_max_array(regret_approx_20)
10 simple_regret_exact_20 = min_max_array(regret_exact_20)
```

```
11
12 min_simple_regret_approx_20 = min(simple_regret_approx_20)
13 min_simple_regret_exact_20 = min(simple_regret_exact_20)
14
15 min_simple_regret_approx_20, min_simple_regret_exact_20
```

```
(2.266628953568505, 2.266628953568505)
```

```
 1 # Iteration1 :
 2
 3 slice1 = 0
 4
 5 approx1 = [simple_regret_approx_1[slice1],
 6        simple_regret_approx_2[slice1],
 7        simple_regret_approx_3[slice1],
 8        simple_regret_approx_4[slice1],
 9        simple_regret_approx_5[slice1],
10        simple_regret_approx_6[slice1],
11        simple_regret_approx_7[slice1],
12        simple_regret_approx_8[slice1],
13        simple_regret_approx_9[slice1],
14        simple_regret_approx_10[slice1],
15        simple_regret_approx_11[slice1],
16        simple_regret_approx_12[slice1],
17        simple_regret_approx_13[slice1],
18        simple_regret_approx_14[slice1],
19        simple_regret_approx_15[slice1],
20        simple_regret_approx_16[slice1],
21        simple_regret_approx_17[slice1],
22        simple_regret_approx_18[slice1],
23        simple_regret_approx_19[slice1],
24        simple_regret_approx_20[slice1]]
25
26 exact1 = [simple_regret_exact_1[slice1],
27        simple_regret_exact_2[slice1],
28        simple_regret_exact_3[slice1],
29        simple_regret_exact_4[slice1],
30        simple_regret_exact_5[slice1],
31        simple_regret_exact_6[slice1],
32        simple_regret_exact_7[slice1],
33        simple_regret_exact_8[slice1],
34        simple_regret_exact_9[slice1],
35        simple_regret_exact_10[slice1],
36        simple_regret_exact_11[slice1],
37        simple_regret_exact_12[slice1],
38        simple_regret_exact_13[slice1],
39        simple_regret_exact_14[slice1],
40        simple_regret_exact_15[slice1],
41        simple_regret_exact_16[slice1],
42        simple_regret_exact_17[slice1],
43        simple_regret_exact_18[slice1],
44        simple_regret_exact_19[slice1],
45        simple_regret_exact_20[slice1]]
46
```

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

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

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

```
# Iteration11 :

slice11 = 10

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

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

```
        simple_regret_exact_19[slice11],
        simple_regret_exact_20[slice11]]

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

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

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

```
# Iteration21 :

slice21 = 20

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

exact21 = [simple_regret_exact_1[slice21],
        simple_regret_exact_2[slice21],
        simple_regret_exact_3[slice21],
        simple_regret_exact_4[slice21],
        simple_regret_exact_5[slice21],
        simple_regret_exact_6[slice21],
        simple_regret_exact_7[slice21],
        simple_regret_exact_8[slice21],
        simple_regret_exact_9[slice21],
        simple_regret_exact_10[slice21],
        simple_regret_exact_11[slice21],
        simple_regret_exact_12[slice21],
        simple_regret_exact_13[slice21],
        simple_regret_exact_14[slice21],
```

```
        simple_regret_exact_15[slice21],
        simple_regret_exact_16[slice21],
        simple_regret_exact_17[slice21],
        simple_regret_exact_18[slice21],
        simple_regret_exact_19[slice21],
        simple_regret_exact_20[slice21]]

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

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

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

```
# Iteration31 :

slice31 = 30

approx31 = [simple_regret_approx_1[slice31],
        simple_regret_approx_2[slice31],
        simple_regret_approx_3[slice31],
        simple_regret_approx_4[slice31],
        simple_regret_approx_5[slice31],
        simple_regret_approx_6[slice31],
        simple_regret_approx_7[slice31],
        simple_regret_approx_8[slice31],
        simple_regret_approx_9[slice31],
        simple_regret_approx_10[slice31],
        simple_regret_approx_11[slice31],
        simple_regret_approx_12[slice31],
        simple_regret_approx_13[slice31],
        simple_regret_approx_14[slice31],
        simple_regret_approx_15[slice31],
        simple_regret_approx_16[slice31],
        simple_regret_approx_17[slice31],
        simple_regret_approx_18[slice31],
        simple_regret_approx_19[slice31],
        simple_regret_approx_20[slice31]]

exact31 = [simple_regret_exact_1[slice31],
        simple_regret_exact_2[slice31],
        simple_regret_exact_3[slice31],
        simple_regret_exact_4[slice31],
        simple_regret_exact_5[slice31],
        simple_regret_exact_6[slice31],
        simple_regret_exact_7[slice31],
        simple_regret_exact_8[slice31],
        simple_regret_exact_9[slice31],
        simple_regret_exact_10[slice31],
```

```
        simple_regret_exact_11[slice31],
        simple_regret_exact_12[slice31],
        simple_regret_exact_13[slice31],
        simple_regret_exact_14[slice31],
        simple_regret_exact_15[slice31],
        simple_regret_exact_16[slice31],
        simple_regret_exact_17[slice31],
        simple_regret_exact_18[slice31],
        simple_regret_exact_19[slice31],
        simple_regret_exact_20[slice31]]

approx31_results = pd.DataFrame(approx31).sort_values(by=[0], ascending=False)
exact31_results = pd.DataFrame(exact31).sort_values(by=[0], ascending=False)

### Best simple regret minimization IQR - approx:
lower_approx31 = np.asarray(approx31_results[4:5][0])[0]
median_approx31 = np.asarray(approx31_results[9:10][0])[0]
upper_approx31 = np.asarray(approx31_results[14:15][0])[0]

lower_exact31 = np.asarray(exact31_results[4:5][0])[0]
median_exact31 = np.asarray(exact31_results[9:10][0])[0]
upper_exact31 = np.asarray(exact31_results[14:15][0])[0]
```

```
# Iteration41 :

slice41 = 40

approx41 = [simple_regret_approx_1[slice41],
        simple_regret_approx_2[slice41],
        simple_regret_approx_3[slice41],
        simple_regret_approx_4[slice41],
        simple_regret_approx_5[slice41],
        simple_regret_approx_6[slice41],
        simple_regret_approx_7[slice41],
        simple_regret_approx_8[slice41],
        simple_regret_approx_9[slice41],
        simple_regret_approx_10[slice41],
        simple_regret_approx_11[slice41],
        simple_regret_approx_12[slice41],
        simple_regret_approx_13[slice41],
        simple_regret_approx_14[slice41],
        simple_regret_approx_15[slice41],
        simple_regret_approx_16[slice41],
        simple_regret_approx_17[slice41],
        simple_regret_approx_18[slice41],
        simple_regret_approx_19[slice41],
        simple_regret_approx_20[slice41]]

exact41 = [simple_regret_exact_1[slice41],
        simple_regret_exact_2[slice41],
        simple_regret_exact_3[slice41],
        simple_regret_exact_4[slice41],
        simple_regret_exact_5[slice41],
        simple_regret_exact_6[slice41],
```

```
        simple_regret_exact_7[slice41],
        simple_regret_exact_8[slice41],
        simple_regret_exact_9[slice41],
        simple_regret_exact_10[slice41],
        simple_regret_exact_11[slice41],
        simple_regret_exact_12[slice41],
        simple_regret_exact_13[slice41],
        simple_regret_exact_14[slice41],
        simple_regret_exact_15[slice41],
        simple_regret_exact_16[slice41],
        simple_regret_exact_17[slice41],
        simple_regret_exact_18[slice41],
        simple_regret_exact_19[slice41],
        simple_regret_exact_20[slice41]]

approx41_results = pd.DataFrame(approx41).sort_values(by=[0], ascending=False)
exact41_results = pd.DataFrame(exact41).sort_values(by=[0], ascending=False)

### Best simple regret minimization IQR - approx:
lower_approx41 = np.asarray(approx41_results[4:5][0])[0]
median_approx41 = np.asarray(approx41_results[9:10][0])[0]
upper_approx41 = np.asarray(approx41_results[14:15][0])[0]

lower_exact41 = np.asarray(exact41_results[4:5][0])[0]
median_exact41 = np.asarray(exact41_results[9:10][0])[0]
upper_exact41 = np.asarray(exact41_results[14:15][0])[0]
```

```
# Iteration2 :

slice2 = 1

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

exact2 = [simple_regret_exact_1[slice2],
```

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

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

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

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

```
# Iteration12 :

slice12 = 11

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

```
        simple_regret_approx_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]
```

```
# Iteration22 :

slice22 = 21

approx22 = [simple_regret_approx_1[slice22],
        simple_regret_approx_2[slice22],
        simple_regret_approx_3[slice22],
        simple_regret_approx_4[slice22],
        simple_regret_approx_5[slice22],
        simple_regret_approx_6[slice22],
        simple_regret_approx_7[slice22],
        simple_regret_approx_8[slice22],
        simple_regret_approx_9[slice22],
        simple_regret_approx_10[slice22],
        simple_regret_approx_11[slice22],
        simple_regret_approx_12[slice22],
        simple_regret_approx_13[slice22],
        simple_regret_approx_14[slice22],
        simple_regret_approx_15[slice22],
```

```
        simple_regret_approx_16[slice22],
        simple_regret_approx_17[slice22],
        simple_regret_approx_18[slice22],
        simple_regret_approx_19[slice22],
        simple_regret_approx_20[slice22]]

exact22 = [simple_regret_exact_1[slice22],
        simple_regret_exact_2[slice22],
        simple_regret_exact_3[slice22],
        simple_regret_exact_4[slice22],
        simple_regret_exact_5[slice22],
        simple_regret_exact_6[slice22],
        simple_regret_exact_7[slice22],
        simple_regret_exact_8[slice22],
        simple_regret_exact_9[slice22],
        simple_regret_exact_10[slice22],
        simple_regret_exact_11[slice22],
        simple_regret_exact_12[slice22],
        simple_regret_exact_13[slice22],
        simple_regret_exact_14[slice22],
        simple_regret_exact_15[slice22],
        simple_regret_exact_16[slice22],
        simple_regret_exact_17[slice22],
        simple_regret_exact_18[slice22],
        simple_regret_exact_19[slice22],
        simple_regret_exact_20[slice22]]

approx22_results = pd.DataFrame(approx22).sort_values(by=[0], ascending=False)
exact22_results = pd.DataFrame(exact22).sort_values(by=[0], ascending=False)

### Best simple regret minimization IQR - approx:
lower_approx22 = np.asarray(approx22_results[4:5][0])[0]
median_approx22 = np.asarray(approx22_results[9:10][0])[0]
upper_approx22 = np.asarray(approx22_results[14:15][0])[0]

lower_exact22 = np.asarray(exact22_results[4:5][0])[0]
median_exact22 = np.asarray(exact22_results[9:10][0])[0]
upper_exact22 = np.asarray(exact22_results[14:15][0])[0]
```

```
# Iteration32 :

slice32 = 31

approx32 = [simple_regret_approx_1[slice32],
        simple_regret_approx_2[slice32],
        simple_regret_approx_3[slice32],
        simple_regret_approx_4[slice32],
        simple_regret_approx_5[slice32],
        simple_regret_approx_6[slice32],
        simple_regret_approx_7[slice32],
        simple_regret_approx_8[slice32],
        simple_regret_approx_9[slice32],
        simple_regret_approx_10[slice32],
        simple_regret_approx_11[slice32],
```

```
        simple_regret_approx_12[slice32],
        simple_regret_approx_13[slice32],
        simple_regret_approx_14[slice32],
        simple_regret_approx_15[slice32],
        simple_regret_approx_16[slice32],
        simple_regret_approx_17[slice32],
        simple_regret_approx_18[slice32],
        simple_regret_approx_19[slice32],
        simple_regret_approx_20[slice32]]

exact32 = [simple_regret_exact_1[slice32],
        simple_regret_exact_2[slice32],
        simple_regret_exact_3[slice32],
        simple_regret_exact_4[slice32],
        simple_regret_exact_5[slice32],
        simple_regret_exact_6[slice32],
        simple_regret_exact_7[slice32],
        simple_regret_exact_8[slice32],
        simple_regret_exact_9[slice32],
        simple_regret_exact_10[slice32],
        simple_regret_exact_11[slice32],
        simple_regret_exact_12[slice32],
        simple_regret_exact_13[slice32],
        simple_regret_exact_14[slice32],
        simple_regret_exact_15[slice32],
        simple_regret_exact_16[slice32],
        simple_regret_exact_17[slice32],
        simple_regret_exact_18[slice32],
        simple_regret_exact_19[slice32],
        simple_regret_exact_20[slice32]]

approx32_results = pd.DataFrame(approx32).sort_values(by=[0], ascending=False)
exact32_results = pd.DataFrame(exact32).sort_values(by=[0], ascending=False)

### Best simple regret minimization IQR - approx:
lower_approx32 = np.asarray(approx32_results[4:5][0])[0]
median_approx32 = np.asarray(approx32_results[9:10][0])[0]
upper_approx32 = np.asarray(approx32_results[14:15][0])[0]

lower_exact32 = np.asarray(exact32_results[4:5][0])[0]
median_exact32 = np.asarray(exact32_results[9:10][0])[0]
upper_exact32 = np.asarray(exact32_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_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_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]
```

```
# Iteration23 :

slice23 = 22

```

```
approx23 = [simple_regret_approx_1[slice23],
       simple_regret_approx_2[slice23],
       simple_regret_approx_3[slice23],
       simple_regret_approx_4[slice23],
       simple_regret_approx_5[slice23],
       simple_regret_approx_6[slice23],
       simple_regret_approx_7[slice23],
       simple_regret_approx_8[slice23],
       simple_regret_approx_9[slice23],
       simple_regret_approx_10[slice23],
       simple_regret_approx_11[slice23],
       simple_regret_approx_12[slice23],
       simple_regret_approx_13[slice23],
       simple_regret_approx_14[slice23],
       simple_regret_approx_15[slice23],
       simple_regret_approx_16[slice23],
       simple_regret_approx_17[slice23],
       simple_regret_approx_18[slice23],
       simple_regret_approx_19[slice23],
       simple_regret_approx_20[slice23]]

exact23 = [simple_regret_exact_1[slice23],
       simple_regret_exact_2[slice23],
       simple_regret_exact_3[slice23],
       simple_regret_exact_4[slice23],
       simple_regret_exact_5[slice23],
       simple_regret_exact_6[slice23],
       simple_regret_exact_7[slice23],
       simple_regret_exact_8[slice23],
       simple_regret_exact_9[slice23],
       simple_regret_exact_10[slice23],
       simple_regret_exact_11[slice23],
       simple_regret_exact_12[slice23],
       simple_regret_exact_13[slice23],
       simple_regret_exact_14[slice23],
       simple_regret_exact_15[slice23],
       simple_regret_exact_16[slice23],
       simple_regret_exact_17[slice23],
       simple_regret_exact_18[slice23],
       simple_regret_exact_19[slice23],
       simple_regret_exact_20[slice23]]

approx23_results = pd.DataFrame(approx23).sort_values(by=[0], ascending=False)
exact23_results = pd.DataFrame(exact23).sort_values(by=[0], ascending=False)

### Best simple regret minimization IQR - approx:
lower_approx23 = np.asarray(approx23_results[4:5][0])[0]
median_approx23 = np.asarray(approx23_results[9:10][0])[0]
upper_approx23 = np.asarray(approx23_results[14:15][0])[0]

lower_exact23 = np.asarray(exact23_results[4:5][0])[0]
median_exact23 = np.asarray(exact23_results[9:10][0])[0]
upper_exact23 = np.asarray(exact23_results[14:15][0])[0]
```

```
# Iteration33 :
```

```
slice33 = 32

approx33 = [simple_regret_approx_1[slice33],
       simple_regret_approx_2[slice33],
       simple_regret_approx_3[slice33],
       simple_regret_approx_4[slice33],
       simple_regret_approx_5[slice33],
       simple_regret_approx_6[slice33],
       simple_regret_approx_7[slice33],
       simple_regret_approx_8[slice33],
       simple_regret_approx_9[slice33],
       simple_regret_approx_10[slice33],
       simple_regret_approx_11[slice33],
       simple_regret_approx_12[slice33],
       simple_regret_approx_13[slice33],
       simple_regret_approx_14[slice33],
       simple_regret_approx_15[slice33],
       simple_regret_approx_16[slice33],
       simple_regret_approx_17[slice33],
       simple_regret_approx_18[slice33],
       simple_regret_approx_19[slice33],
       simple_regret_approx_20[slice33]]

exact33 = [simple_regret_exact_1[slice33],
       simple_regret_exact_2[slice33],
       simple_regret_exact_3[slice33],
       simple_regret_exact_4[slice33],
       simple_regret_exact_5[slice33],
       simple_regret_exact_6[slice33],
       simple_regret_exact_7[slice33],
       simple_regret_exact_8[slice33],
       simple_regret_exact_9[slice33],
       simple_regret_exact_10[slice33],
       simple_regret_exact_11[slice33],
       simple_regret_exact_12[slice33],
       simple_regret_exact_13[slice33],
       simple_regret_exact_14[slice33],
       simple_regret_exact_15[slice33],
       simple_regret_exact_16[slice33],
       simple_regret_exact_17[slice33],
       simple_regret_exact_18[slice33],
       simple_regret_exact_19[slice33],
       simple_regret_exact_20[slice33]]

approx33_results = pd.DataFrame(approx33).sort_values(by=[0], ascending=False)
exact33_results = pd.DataFrame(exact33).sort_values(by=[0], ascending=False)

### Best simple regret minimization IQR - approx:
lower_approx33 = np.asarray(approx33_results[4:5][0])[0]
median_approx33 = np.asarray(approx33_results[9:10][0])[0]
upper_approx33 = np.asarray(approx33_results[14:15][0])[0]

lower_exact33 = np.asarray(exact33_results[4:5][0])[0]
median_exact33 = np.asarray(exact33_results[9:10][0])[0]
```

```
upper_exact33 = np.asarray(exact33_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]
```

```
# Iteration24 :

slice24 = 23

approx24 = [simple_regret_approx_1[slice24],
       simple_regret_approx_2[slice24],
       simple_regret_approx_3[slice24],
       simple_regret_approx_4[slice24],
       simple_regret_approx_5[slice24],
       simple_regret_approx_6[slice24],
       simple_regret_approx_7[slice24],
       simple_regret_approx_8[slice24],
       simple_regret_approx_9[slice24],
       simple_regret_approx_10[slice24],
       simple_regret_approx_11[slice24],
       simple_regret_approx_12[slice24],
       simple_regret_approx_13[slice24],
       simple_regret_approx_14[slice24],
       simple_regret_approx_15[slice24],
       simple_regret_approx_16[slice24],
       simple_regret_approx_17[slice24],
       simple_regret_approx_18[slice24],
       simple_regret_approx_19[slice24],
       simple_regret_approx_20[slice24]]

exact24 = [simple_regret_exact_1[slice24],
       simple_regret_exact_2[slice24],
       simple_regret_exact_3[slice24],
       simple_regret_exact_4[slice24],
       simple_regret_exact_5[slice24],
       simple_regret_exact_6[slice24],
       simple_regret_exact_7[slice24],
       simple_regret_exact_8[slice24],
       simple_regret_exact_9[slice24],
       simple_regret_exact_10[slice24],
       simple_regret_exact_11[slice24],
       simple_regret_exact_12[slice24],
       simple_regret_exact_13[slice24],
       simple_regret_exact_14[slice24],
       simple_regret_exact_15[slice24],
       simple_regret_exact_16[slice24],
       simple_regret_exact_17[slice24],
       simple_regret_exact_18[slice24],
       simple_regret_exact_19[slice24],
       simple_regret_exact_20[slice24]]
```

```

approx24_results = pd.DataFrame(approx24).sort_values(by=[0], ascending=False)
exact24_results = pd.DataFrame(exact24).sort_values(by=[0], ascending=False)

### Best simple regret minimization IQR - approx:
lower_approx24 = np.asarray(approx24_results[4:5][0])[0]
median_approx24 = np.asarray(approx24_results[9:10][0])[0]
upper_approx24 = np.asarray(approx24_results[14:15][0])[0]

lower_exact24 = np.asarray(exact24_results[4:5][0])[0]
median_exact24 = np.asarray(exact24_results[9:10][0])[0]
upper_exact24 = np.asarray(exact24_results[14:15][0])[0]
```

```
# Iteration34 :

slice34 = 33

approx34 = [simple_regret_approx_1[slice34],
       simple_regret_approx_2[slice34],
       simple_regret_approx_3[slice34],
       simple_regret_approx_4[slice34],
       simple_regret_approx_5[slice34],
       simple_regret_approx_6[slice34],
       simple_regret_approx_7[slice34],
       simple_regret_approx_8[slice34],
       simple_regret_approx_9[slice34],
       simple_regret_approx_10[slice34],
       simple_regret_approx_11[slice34],
       simple_regret_approx_12[slice34],
       simple_regret_approx_13[slice34],
       simple_regret_approx_14[slice34],
       simple_regret_approx_15[slice34],
       simple_regret_approx_16[slice34],
       simple_regret_approx_17[slice34],
       simple_regret_approx_18[slice34],
       simple_regret_approx_19[slice34],
       simple_regret_approx_20[slice34]]

exact34 = [simple_regret_exact_1[slice34],
       simple_regret_exact_2[slice34],
       simple_regret_exact_3[slice34],
       simple_regret_exact_4[slice34],
       simple_regret_exact_5[slice34],
       simple_regret_exact_6[slice34],
       simple_regret_exact_7[slice34],
       simple_regret_exact_8[slice34],
       simple_regret_exact_9[slice34],
       simple_regret_exact_10[slice34],
       simple_regret_exact_11[slice34],
       simple_regret_exact_12[slice34],
       simple_regret_exact_13[slice34],
       simple_regret_exact_14[slice34],
       simple_regret_exact_15[slice34],
       simple_regret_exact_16[slice34],
```

```
        simple_regret_exact_17[slice34],
        simple_regret_exact_18[slice34],
        simple_regret_exact_19[slice34],
        simple_regret_exact_20[slice34]]

approx34_results = pd.DataFrame(approx34).sort_values(by=[0], ascending=False)
exact34_results = pd.DataFrame(exact34).sort_values(by=[0], ascending=False)

### Best simple regret minimization IQR - approx:
lower_approx34 = np.asarray(approx34_results[4:5][0])[0]
median_approx34 = np.asarray(approx34_results[9:10][0])[0]
upper_approx34 = np.asarray(approx34_results[14:15][0])[0]

lower_exact34 = np.asarray(exact34_results[4:5][0])[0]
median_exact34 = np.asarray(exact34_results[9:10][0])[0]
upper_exact34 = np.asarray(exact34_results[14:15][0])[0]
```

```
# Iteration5 :

slice5 = 4

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

exact5 = [simple_regret_exact_1[slice5],
        simple_regret_exact_2[slice5],
        simple_regret_exact_3[slice5],
        simple_regret_exact_4[slice5],
        simple_regret_exact_5[slice5],
        simple_regret_exact_6[slice5],
        simple_regret_exact_7[slice5],
        simple_regret_exact_8[slice5],
        simple_regret_exact_9[slice5],
        simple_regret_exact_10[slice5],
        simple_regret_exact_11[slice5],
        simple_regret_exact_12[slice5],
        simple_regret_exact_13[slice5]
```

```
        simple_regret_exact_14[slice5],
        simple_regret_exact_15[slice5],
        simple_regret_exact_16[slice5],
        simple_regret_exact_17[slice5],
        simple_regret_exact_18[slice5],
        simple_regret_exact_19[slice5],
        simple_regret_exact_20[slice5]]

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

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

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

```
# Iteration15 :

slice15 = 14

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

exact15 = [simple_regret_exact_1[slice15],
        simple_regret_exact_2[slice15],
        simple_regret_exact_3[slice15],
        simple_regret_exact_4[slice15],
        simple_regret_exact_5[slice15],
        simple_regret_exact_6[slice15],
        simple_regret_exact_7[slice15],
        simple_regret_exact_8[slice15],
        simple_regret_exact_9[slice15],
```

```
        simple_regret_exact_10[slice15],
        simple_regret_exact_11[slice15],
        simple_regret_exact_12[slice15],
        simple_regret_exact_13[slice15],
        simple_regret_exact_14[slice15],
        simple_regret_exact_15[slice15],
        simple_regret_exact_16[slice15],
        simple_regret_exact_17[slice15],
        simple_regret_exact_18[slice15],
        simple_regret_exact_19[slice15],
        simple_regret_exact_20[slice15]]

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

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

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

```
# Iteration25 :

slice25 = 24

approx25 = [simple_regret_approx_1[slice25],
        simple_regret_approx_2[slice25],
        simple_regret_approx_3[slice25],
        simple_regret_approx_4[slice25],
        simple_regret_approx_5[slice25],
        simple_regret_approx_6[slice25],
        simple_regret_approx_7[slice25],
        simple_regret_approx_8[slice25],
        simple_regret_approx_9[slice25],
        simple_regret_approx_10[slice25],
        simple_regret_approx_11[slice25],
        simple_regret_approx_12[slice25],
        simple_regret_approx_13[slice25],
        simple_regret_approx_14[slice25],
        simple_regret_approx_15[slice25],
        simple_regret_approx_16[slice25],
        simple_regret_approx_17[slice25],
        simple_regret_approx_18[slice25],
        simple_regret_approx_19[slice25],
        simple_regret_approx_20[slice25]]

exact25 = [simple_regret_exact_1[slice25],
        simple_regret_exact_2[slice25],
        simple_regret_exact_3[slice25],
        simple_regret_exact_4[slice25],
        simple_regret_exact_5[slice25],
```

```
        simple_regret_exact_6[slice25],
        simple_regret_exact_7[slice25],
        simple_regret_exact_8[slice25],
        simple_regret_exact_9[slice25],
        simple_regret_exact_10[slice25],
        simple_regret_exact_11[slice25],
        simple_regret_exact_12[slice25],
        simple_regret_exact_13[slice25],
        simple_regret_exact_14[slice25],
        simple_regret_exact_15[slice25],
        simple_regret_exact_16[slice25],
        simple_regret_exact_17[slice25],
        simple_regret_exact_18[slice25],
        simple_regret_exact_19[slice25],
        simple_regret_exact_20[slice25]]

approx25_results = pd.DataFrame(approx25).sort_values(by=[0], ascending=False)
exact25_results = pd.DataFrame(exact25).sort_values(by=[0], ascending=False)

### Best simple regret minimization IQR - approx:
lower_approx25 = np.asarray(approx25_results[4:5][0])[0]
median_approx25 = np.asarray(approx25_results[9:10][0])[0]
upper_approx25 = np.asarray(approx25_results[14:15][0])[0]

lower_exact25 = np.asarray(exact25_results[4:5][0])[0]
median_exact25 = np.asarray(exact25_results[9:10][0])[0]
upper_exact25 = np.asarray(exact25_results[14:15][0])[0]
```

```
# Iteration35 :

slice35 = 34

approx35 = [simple_regret_approx_1[slice35],
        simple_regret_approx_2[slice35],
        simple_regret_approx_3[slice35],
        simple_regret_approx_4[slice35],
        simple_regret_approx_5[slice35],
        simple_regret_approx_6[slice35],
        simple_regret_approx_7[slice35],
        simple_regret_approx_8[slice35],
        simple_regret_approx_9[slice35],
        simple_regret_approx_10[slice35],
        simple_regret_approx_11[slice35],
        simple_regret_approx_12[slice35],
        simple_regret_approx_13[slice35],
        simple_regret_approx_14[slice35],
        simple_regret_approx_15[slice35],
        simple_regret_approx_16[slice35],
        simple_regret_approx_17[slice35],
        simple_regret_approx_18[slice35],
        simple_regret_approx_19[slice35],
        simple_regret_approx_20[slice35]]

exact35 = [simple_regret_exact_1[slice35],
        simple_regret_exact_2[slice35],
```

```
        simple_regret_exact_3[slice35],
        simple_regret_exact_4[slice35],
        simple_regret_exact_5[slice35],
        simple_regret_exact_6[slice35],
        simple_regret_exact_7[slice35],
        simple_regret_exact_8[slice35],
        simple_regret_exact_9[slice35],
        simple_regret_exact_10[slice35],
        simple_regret_exact_11[slice35],
        simple_regret_exact_12[slice35],
        simple_regret_exact_13[slice35],
        simple_regret_exact_14[slice35],
        simple_regret_exact_15[slice35],
        simple_regret_exact_16[slice35],
        simple_regret_exact_17[slice35],
        simple_regret_exact_18[slice35],
        simple_regret_exact_19[slice35],
        simple_regret_exact_20[slice35]]

approx35_results = pd.DataFrame(approx35).sort_values(by=[0], ascending=False)
exact35_results = pd.DataFrame(exact35).sort_values(by=[0], ascending=False)

### Best simple regret minimization IQR - approx:
lower_approx35 = np.asarray(approx35_results[4:5][0])[0]
median_approx35 = np.asarray(approx35_results[9:10][0])[0]
upper_approx35 = np.asarray(approx35_results[14:15][0])[0]

lower_exact35 = np.asarray(exact35_results[4:5][0])[0]
median_exact35 = np.asarray(exact35_results[9:10][0])[0]
upper_exact35 = np.asarray(exact35_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]

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_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]
```

```
# Iteration26 :

slice26 = 25

approx26 = [simple_regret_approx_1[slice26],
        simple_regret_approx_2[slice26],
        simple_regret_approx_3[slice26],
        simple_regret_approx_4[slice26],
        simple_regret_approx_5[slice26],
        simple_regret_approx_6[slice26],
        simple_regret_approx_7[slice26],
        simple_regret_approx_8[slice26],
        simple_regret_approx_9[slice26],
        simple_regret_approx_10[slice26],
        simple_regret_approx_11[slice26],
        simple_regret_approx_12[slice26],
```

```
        simple_regret_approx_13[slice26],
        simple_regret_approx_14[slice26],
        simple_regret_approx_15[slice26],
        simple_regret_approx_16[slice26],
        simple_regret_approx_17[slice26],
        simple_regret_approx_18[slice26],
        simple_regret_approx_19[slice26],
        simple_regret_approx_20[slice26]]

exact26 = [simple_regret_exact_1[slice26],
        simple_regret_exact_2[slice26],
        simple_regret_exact_3[slice26],
        simple_regret_exact_4[slice26],
        simple_regret_exact_5[slice26],
        simple_regret_exact_6[slice26],
        simple_regret_exact_7[slice26],
        simple_regret_exact_8[slice26],
        simple_regret_exact_9[slice26],
        simple_regret_exact_10[slice26],
        simple_regret_exact_11[slice26],
        simple_regret_exact_12[slice26],
        simple_regret_exact_13[slice26],
        simple_regret_exact_14[slice26],
        simple_regret_exact_15[slice26],
        simple_regret_exact_16[slice26],
        simple_regret_exact_17[slice26],
        simple_regret_exact_18[slice26],
        simple_regret_exact_19[slice26],
        simple_regret_exact_20[slice26]]

approx26_results = pd.DataFrame(approx26).sort_values(by=[0], ascending=False)
exact26_results = pd.DataFrame(exact26).sort_values(by=[0], ascending=False)

### Best simple regret minimization IQR - approx:
lower_approx26 = np.asarray(approx26_results[4:5][0])[0]
median_approx26 = np.asarray(approx26_results[9:10][0])[0]
upper_approx26 = np.asarray(approx26_results[14:15][0])[0]

lower_exact26 = np.asarray(exact26_results[4:5][0])[0]
median_exact26 = np.asarray(exact26_results[9:10][0])[0]
upper_exact26 = np.asarray(exact26_results[14:15][0])[0]
```

```
# Iteration36 :

slice36 = 35

approx36 = [simple_regret_approx_1[slice36],
        simple_regret_approx_2[slice36],
        simple_regret_approx_3[slice36],
        simple_regret_approx_4[slice36],
        simple_regret_approx_5[slice36],
        simple_regret_approx_6[slice36],
        simple_regret_approx_7[slice36],
        simple_regret_approx_8[slice36],
```

```
        simple_regret_approx_9[slice36],
        simple_regret_approx_10[slice36],
        simple_regret_approx_11[slice36],
        simple_regret_approx_12[slice36],
        simple_regret_approx_13[slice36],
        simple_regret_approx_14[slice36],
        simple_regret_approx_15[slice36],
        simple_regret_approx_16[slice36],
        simple_regret_approx_17[slice36],
        simple_regret_approx_18[slice36],
        simple_regret_approx_19[slice36],
        simple_regret_approx_20[slice36]]

exact36 = [simple_regret_exact_1[slice36],
        simple_regret_exact_2[slice36],
        simple_regret_exact_3[slice36],
        simple_regret_exact_4[slice36],
        simple_regret_exact_5[slice36],
        simple_regret_exact_6[slice36],
        simple_regret_exact_7[slice36],
        simple_regret_exact_8[slice36],
        simple_regret_exact_9[slice36],
        simple_regret_exact_10[slice36],
        simple_regret_exact_11[slice36],
        simple_regret_exact_12[slice36],
        simple_regret_exact_13[slice36],
        simple_regret_exact_14[slice36],
        simple_regret_exact_15[slice36],
        simple_regret_exact_16[slice36],
        simple_regret_exact_17[slice36],
        simple_regret_exact_18[slice36],
        simple_regret_exact_19[slice36],
        simple_regret_exact_20[slice36]]

approx36_results = pd.DataFrame(approx36).sort_values(by=[0], ascending=False)
exact36_results = pd.DataFrame(exact36).sort_values(by=[0], ascending=False)

### Best simple regret minimization IQR - approx:
lower_approx36 = np.asarray(approx36_results[4:5][0])[0]
median_approx36 = np.asarray(approx36_results[9:10][0])[0]
upper_approx36 = np.asarray(approx36_results[14:15][0])[0]

lower_exact36 = np.asarray(exact36_results[4:5][0])[0]
median_exact36 = np.asarray(exact36_results[9:10][0])[0]
upper_exact36 = np.asarray(exact36_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_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_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]
```

```
# Iteration27 :
```

```
slice27 = 26

approx27 = [simple_regret_approx_1[slice27],
       simple_regret_approx_2[slice27],
       simple_regret_approx_3[slice27],
       simple_regret_approx_4[slice27],
       simple_regret_approx_5[slice27],
       simple_regret_approx_6[slice27],
       simple_regret_approx_7[slice27],
       simple_regret_approx_8[slice27],
       simple_regret_approx_9[slice27],
       simple_regret_approx_10[slice27],
       simple_regret_approx_11[slice27],
       simple_regret_approx_12[slice27],
       simple_regret_approx_13[slice27],
       simple_regret_approx_14[slice27],
       simple_regret_approx_15[slice27],
       simple_regret_approx_16[slice27],
       simple_regret_approx_17[slice27],
       simple_regret_approx_18[slice27],
       simple_regret_approx_19[slice27],
       simple_regret_approx_20[slice27]]

exact27 = [simple_regret_exact_1[slice27],
       simple_regret_exact_2[slice27],
       simple_regret_exact_3[slice27],
       simple_regret_exact_4[slice27],
       simple_regret_exact_5[slice27],
       simple_regret_exact_6[slice27],
       simple_regret_exact_7[slice27],
       simple_regret_exact_8[slice27],
       simple_regret_exact_9[slice27],
       simple_regret_exact_10[slice27],
       simple_regret_exact_11[slice27],
       simple_regret_exact_12[slice27],
       simple_regret_exact_13[slice27],
       simple_regret_exact_14[slice27],
       simple_regret_exact_15[slice27],
       simple_regret_exact_16[slice27],
       simple_regret_exact_17[slice27],
       simple_regret_exact_18[slice27],
       simple_regret_exact_19[slice27],
       simple_regret_exact_20[slice27]]

approx27_results = pd.DataFrame(approx27).sort_values(by=[0], ascending=False)
exact27_results = pd.DataFrame(exact27).sort_values(by=[0], ascending=False)

### Best simple regret minimization IQR - approx:
lower_approx27 = np.asarray(approx27_results[4:5][0])[0]
median_approx27 = np.asarray(approx27_results[9:10][0])[0]
upper_approx27 = np.asarray(approx27_results[14:15][0])[0]

lower_exact27 = np.asarray(exact27_results[4:5][0])[0]
median_exact27 = np.asarray(exact27_results[9:10][0])[0]
```

```
upper_exact27 = np.asarray(exact27_results[14:15][0])[0]
```

```
# Iteration37 :

slice37 = 36

approx37 = [simple_regret_approx_1[slice37],
       simple_regret_approx_2[slice37],
       simple_regret_approx_3[slice37],
       simple_regret_approx_4[slice37],
       simple_regret_approx_5[slice37],
       simple_regret_approx_6[slice37],
       simple_regret_approx_7[slice37],
       simple_regret_approx_8[slice37],
       simple_regret_approx_9[slice37],
       simple_regret_approx_10[slice37],
       simple_regret_approx_11[slice37],
       simple_regret_approx_12[slice37],
       simple_regret_approx_13[slice37],
       simple_regret_approx_14[slice37],
       simple_regret_approx_15[slice37],
       simple_regret_approx_16[slice37],
       simple_regret_approx_17[slice37],
       simple_regret_approx_18[slice37],
       simple_regret_approx_19[slice37],
       simple_regret_approx_20[slice37]]

exact37 = [simple_regret_exact_1[slice37],
       simple_regret_exact_2[slice37],
       simple_regret_exact_3[slice37],
       simple_regret_exact_4[slice37],
       simple_regret_exact_5[slice37],
       simple_regret_exact_6[slice37],
       simple_regret_exact_7[slice37],
       simple_regret_exact_8[slice37],
       simple_regret_exact_9[slice37],
       simple_regret_exact_10[slice37],
       simple_regret_exact_11[slice37],
       simple_regret_exact_12[slice37],
       simple_regret_exact_13[slice37],
       simple_regret_exact_14[slice37],
       simple_regret_exact_15[slice37],
       simple_regret_exact_16[slice37],
       simple_regret_exact_17[slice37],
       simple_regret_exact_18[slice37],
       simple_regret_exact_19[slice37],
       simple_regret_exact_20[slice37]]

approx37_results = pd.DataFrame(approx37).sort_values(by=[0], ascending=False)
exact37_results = pd.DataFrame(exact37).sort_values(by=[0], ascending=False)

### Best simple regret minimization IQR - approx:
lower_approx37 = np.asarray(approx37_results[4:5][0])[0]
median_approx37 = np.asarray(approx37_results[9:10][0])[0]
upper_approx37 = np.asarray(approx37_results[14:15][0])[0]
```

```
54
55 lower_exact37 = np.asarray(exact37_results[4:5][0])[0]
56 median_exact37 = np.asarray(exact37_results[9:10][0])[0]
57 upper_exact37 = np.asarray(exact37_results[14:15][0])[0]
```

```
 1 # Iteration8 :
 2
 3 slice8 = 7
 4
 5 approx8 = [simple_regret_approx_1[slice8],
 6        simple_regret_approx_2[slice8],
 7        simple_regret_approx_3[slice8],
 8        simple_regret_approx_4[slice8],
 9        simple_regret_approx_5[slice8],
10        simple_regret_approx_6[slice8],
11        simple_regret_approx_7[slice8],
12        simple_regret_approx_8[slice8],
13        simple_regret_approx_9[slice8],
14        simple_regret_approx_10[slice8],
15        simple_regret_approx_11[slice8],
16        simple_regret_approx_12[slice8],
17        simple_regret_approx_13[slice8],
18        simple_regret_approx_14[slice8],
19        simple_regret_approx_15[slice8],
20        simple_regret_approx_16[slice8],
21        simple_regret_approx_17[slice8],
22        simple_regret_approx_18[slice8],
23        simple_regret_approx_19[slice8],
24        simple_regret_approx_20[slice8]]
25
26 exact8 = [simple_regret_exact_1[slice8],
27        simple_regret_exact_2[slice8],
28        simple_regret_exact_3[slice8],
29        simple_regret_exact_4[slice8],
30        simple_regret_exact_5[slice8],
31        simple_regret_exact_6[slice8],
32        simple_regret_exact_7[slice8],
33        simple_regret_exact_8[slice8],
34        simple_regret_exact_9[slice8],
35        simple_regret_exact_10[slice8],
36        simple_regret_exact_11[slice8],
37        simple_regret_exact_12[slice8],
38        simple_regret_exact_13[slice8],
39        simple_regret_exact_14[slice8],
40        simple_regret_exact_15[slice8],
41        simple_regret_exact_16[slice8],
42        simple_regret_exact_17[slice8],
43        simple_regret_exact_18[slice8],
44        simple_regret_exact_19[slice8],
45        simple_regret_exact_20[slice8]]
46
47 approx8_results = pd.DataFrame(approx8).sort_values(by=[0], ascending=False)
48 exact8_results = pd.DataFrame(exact8).sort_values(by=[0], ascending=False)
49
```

```
### 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]]
```

```
46
47 approx18_results = pd.DataFrame(approx18).sort_values(by=[0], ascending=False)
48 exact18_results = pd.DataFrame(exact18).sort_values(by=[0], ascending=False)
49
50 ### Best simple regret minimization IQR - approx:
51 lower_approx18 = np.asarray(approx18_results[4:5][0])[0]
52 median_approx18 = np.asarray(approx18_results[9:10][0])[0]
53 upper_approx18 = np.asarray(approx18_results[14:15][0])[0]
54
55 lower_exact18 = np.asarray(exact18_results[4:5][0])[0]
56 median_exact18 = np.asarray(exact18_results[9:10][0])[0]
57 upper_exact18 = np.asarray(exact18_results[14:15][0])[0]
```

```
 1 # Iteration28 :
 2
 3 slice28 = 27
 4
 5 approx28 = [simple_regret_approx_1[slice28],
 6        simple_regret_approx_2[slice28],
 7        simple_regret_approx_3[slice28],
 8        simple_regret_approx_4[slice28],
 9        simple_regret_approx_5[slice28],
10        simple_regret_approx_6[slice28],
11        simple_regret_approx_7[slice28],
12        simple_regret_approx_8[slice28],
13        simple_regret_approx_9[slice28],
14        simple_regret_approx_10[slice28],
15        simple_regret_approx_11[slice28],
16        simple_regret_approx_12[slice28],
17        simple_regret_approx_13[slice28],
18        simple_regret_approx_14[slice28],
19        simple_regret_approx_15[slice28],
20        simple_regret_approx_16[slice28],
21        simple_regret_approx_17[slice28],
22        simple_regret_approx_18[slice28],
23        simple_regret_approx_19[slice28],
24        simple_regret_approx_20[slice28]]
25
26 exact28 = [simple_regret_exact_1[slice28],
27        simple_regret_exact_2[slice28],
28        simple_regret_exact_3[slice28],
29        simple_regret_exact_4[slice28],
30        simple_regret_exact_5[slice28],
31        simple_regret_exact_6[slice28],
32        simple_regret_exact_7[slice28],
33        simple_regret_exact_8[slice28],
34        simple_regret_exact_9[slice28],
35        simple_regret_exact_10[slice28],
36        simple_regret_exact_11[slice28],
37        simple_regret_exact_12[slice28],
38        simple_regret_exact_13[slice28],
39        simple_regret_exact_14[slice28],
40        simple_regret_exact_15[slice28],
41        simple_regret_exact_16[slice28],
42        simple_regret_exact_17[slice28],
```

```
       simple_regret_exact_18[slice28],
       simple_regret_exact_19[slice28],
       simple_regret_exact_20[slice28]]

approx28_results = pd.DataFrame(approx28).sort_values(by=[0], ascending=False)
exact28_results = pd.DataFrame(exact28).sort_values(by=[0], ascending=False)

### Best simple regret minimization IQR - approx:
lower_approx28 = np.asarray(approx28_results[4:5][0])[0]
median_approx28 = np.asarray(approx28_results[9:10][0])[0]
upper_approx28 = np.asarray(approx28_results[14:15][0])[0]

lower_exact28 = np.asarray(exact28_results[4:5][0])[0]
median_exact28 = np.asarray(exact28_results[9:10][0])[0]
upper_exact28 = np.asarray(exact28_results[14:15][0])[0]
```

```
# Iteration38 :

slice38 = 37

approx38 = [simple_regret_approx_1[slice38],
       simple_regret_approx_2[slice38],
       simple_regret_approx_3[slice38],
       simple_regret_approx_4[slice38],
       simple_regret_approx_5[slice38],
       simple_regret_approx_6[slice38],
       simple_regret_approx_7[slice38],
       simple_regret_approx_8[slice38],
       simple_regret_approx_9[slice38],
       simple_regret_approx_10[slice38],
       simple_regret_approx_11[slice38],
       simple_regret_approx_12[slice38],
       simple_regret_approx_13[slice38],
       simple_regret_approx_14[slice38],
       simple_regret_approx_15[slice38],
       simple_regret_approx_16[slice38],
       simple_regret_approx_17[slice38],
       simple_regret_approx_18[slice38],
       simple_regret_approx_19[slice38],
       simple_regret_approx_20[slice38]]

exact38 = [simple_regret_exact_1[slice38],
       simple_regret_exact_2[slice38],
       simple_regret_exact_3[slice38],
       simple_regret_exact_4[slice38],
       simple_regret_exact_5[slice38],
       simple_regret_exact_6[slice38],
       simple_regret_exact_7[slice38],
       simple_regret_exact_8[slice38],
       simple_regret_exact_9[slice38],
       simple_regret_exact_10[slice38],
       simple_regret_exact_11[slice38],
       simple_regret_exact_12[slice38],
       simple_regret_exact_13[slice38],
```

```
        simple_regret_exact_14[slice38],
        simple_regret_exact_15[slice38],
        simple_regret_exact_16[slice38],
        simple_regret_exact_17[slice38],
        simple_regret_exact_18[slice38],
        simple_regret_exact_19[slice38],
        simple_regret_exact_20[slice38]]

approx38_results = pd.DataFrame(approx38).sort_values(by=[0], ascending=False)
exact38_results = pd.DataFrame(exact38).sort_values(by=[0], ascending=False)

### Best simple regret minimization IQR - approx:
lower_approx38 = np.asarray(approx38_results[4:5][0])[0]
median_approx38 = np.asarray(approx38_results[9:10][0])[0]
upper_approx38 = np.asarray(approx38_results[14:15][0])[0]

lower_exact38 = np.asarray(exact38_results[4:5][0])[0]
median_exact38 = np.asarray(exact38_results[9:10][0])[0]
upper_exact38 = np.asarray(exact38_results[14:15][0])[0]
```

```
# Iteration9 :

slice9 = 8

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

exact9 = [simple_regret_exact_1[slice9],
        simple_regret_exact_2[slice9],
        simple_regret_exact_3[slice9],
        simple_regret_exact_4[slice9],
        simple_regret_exact_5[slice9],
        simple_regret_exact_6[slice9],
        simple_regret_exact_7[slice9],
        simple_regret_exact_8[slice9],
        simple_regret_exact_9[slice9],
        simple_regret_exact_10[slice9]
```

```
        simple_regret_exact_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]
```

```
# Iteration29 :

slice29 = 28

approx29 = [simple_regret_approx_1[slice29],
        simple_regret_approx_2[slice29],
        simple_regret_approx_3[slice29],
        simple_regret_approx_4[slice29],
        simple_regret_approx_5[slice29],
        simple_regret_approx_6[slice29],
        simple_regret_approx_7[slice29],
        simple_regret_approx_8[slice29],
        simple_regret_approx_9[slice29],
        simple_regret_approx_10[slice29],
        simple_regret_approx_11[slice29],
        simple_regret_approx_12[slice29],
        simple_regret_approx_13[slice29],
        simple_regret_approx_14[slice29],
        simple_regret_approx_15[slice29],
        simple_regret_approx_16[slice29],
        simple_regret_approx_17[slice29],
        simple_regret_approx_18[slice29],
        simple_regret_approx_19[slice29],
        simple_regret_approx_20[slice29]]

exact29 = [simple_regret_exact_1[slice29],
        simple_regret_exact_2[slice29],
```

```
        simple_regret_exact_3[slice29],
        simple_regret_exact_4[slice29],
        simple_regret_exact_5[slice29],
        simple_regret_exact_6[slice29],
        simple_regret_exact_7[slice29],
        simple_regret_exact_8[slice29],
        simple_regret_exact_9[slice29],
        simple_regret_exact_10[slice29],
        simple_regret_exact_11[slice29],
        simple_regret_exact_12[slice29],
        simple_regret_exact_13[slice29],
        simple_regret_exact_14[slice29],
        simple_regret_exact_15[slice29],
        simple_regret_exact_16[slice29],
        simple_regret_exact_17[slice29],
        simple_regret_exact_18[slice29],
        simple_regret_exact_19[slice29],
        simple_regret_exact_20[slice29]]

approx29_results = pd.DataFrame(approx29).sort_values(by=[0], ascending=False)
exact29_results = pd.DataFrame(exact29).sort_values(by=[0], ascending=False)

### Best simple regret minimization IQR - approx:
lower_approx29 = np.asarray(approx29_results[4:5][0])[0]
median_approx29 = np.asarray(approx29_results[9:10][0])[0]
upper_approx29 = np.asarray(approx29_results[14:15][0])[0]

lower_exact29 = np.asarray(exact29_results[4:5][0])[0]
median_exact29 = np.asarray(exact29_results[9:10][0])[0]
upper_exact29 = np.asarray(exact29_results[14:15][0])[0]
```

```
# Iteration39 :

slice39 = 38

approx39 = [simple_regret_approx_1[slice39],
        simple_regret_approx_2[slice39],
        simple_regret_approx_3[slice39],
        simple_regret_approx_4[slice39],
        simple_regret_approx_5[slice39],
        simple_regret_approx_6[slice39],
        simple_regret_approx_7[slice39],
        simple_regret_approx_8[slice39],
        simple_regret_approx_9[slice39],
        simple_regret_approx_10[slice39],
        simple_regret_approx_11[slice39],
        simple_regret_approx_12[slice39],
        simple_regret_approx_13[slice39],
        simple_regret_approx_14[slice39],
        simple_regret_approx_15[slice39],
        simple_regret_approx_16[slice39],
        simple_regret_approx_17[slice39],
        simple_regret_approx_18[slice39],
        simple_regret_approx_19[slice39],
        simple_regret_approx_20[slice39]]
```

```
        simple_regret_approx_20[slice39]]

exact39 = [simple_regret_exact_1[slice39],
       simple_regret_exact_2[slice39],
       simple_regret_exact_3[slice39],
       simple_regret_exact_4[slice39],
       simple_regret_exact_5[slice39],
       simple_regret_exact_6[slice39],
       simple_regret_exact_7[slice39],
       simple_regret_exact_8[slice39],
       simple_regret_exact_9[slice39],
       simple_regret_exact_10[slice39],
       simple_regret_exact_11[slice39],
       simple_regret_exact_12[slice39],
       simple_regret_exact_13[slice39],
       simple_regret_exact_14[slice39],
       simple_regret_exact_15[slice39],
       simple_regret_exact_16[slice39],
       simple_regret_exact_17[slice39],
       simple_regret_exact_18[slice39],
       simple_regret_exact_19[slice39],
       simple_regret_exact_20[slice39]]

approx39_results = pd.DataFrame(approx39).sort_values(by=[0], ascending=False)
exact39_results = pd.DataFrame(exact39).sort_values(by=[0], ascending=False)

### Best simple regret minimization IQR - approx:
lower_approx39 = np.asarray(approx39_results[4:5][0])[0]
median_approx39 = np.asarray(approx39_results[9:10][0])[0]
upper_approx39 = np.asarray(approx39_results[14:15][0])[0]

lower_exact39 = np.asarray(exact39_results[4:5][0])[0]
median_exact39 = np.asarray(exact39_results[9:10][0])[0]
upper_exact39 = np.asarray(exact39_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]
```

```
# Iteration30 :

slice30 = 29

approx30 = [simple_regret_approx_1[slice30],
        simple_regret_approx_2[slice30],
        simple_regret_approx_3[slice30],
        simple_regret_approx_4[slice30],
        simple_regret_approx_5[slice30],
        simple_regret_approx_6[slice30],
        simple_regret_approx_7[slice30],
        simple_regret_approx_8[slice30],
        simple_regret_approx_9[slice30],
```

```
        simple_regret_approx_10[slice30],
        simple_regret_approx_11[slice30],
        simple_regret_approx_12[slice30],
        simple_regret_approx_13[slice30],
        simple_regret_approx_14[slice30],
        simple_regret_approx_15[slice30],
        simple_regret_approx_16[slice30],
        simple_regret_approx_17[slice30],
        simple_regret_approx_18[slice30],
        simple_regret_approx_19[slice30],
        simple_regret_approx_20[slice30]]

exact30 = [simple_regret_exact_1[slice30],
        simple_regret_exact_2[slice30],
        simple_regret_exact_3[slice30],
        simple_regret_exact_4[slice30],
        simple_regret_exact_5[slice30],
        simple_regret_exact_6[slice30],
        simple_regret_exact_7[slice30],
        simple_regret_exact_8[slice30],
        simple_regret_exact_9[slice30],
        simple_regret_exact_10[slice30],
        simple_regret_exact_11[slice30],
        simple_regret_exact_12[slice30],
        simple_regret_exact_13[slice30],
        simple_regret_exact_14[slice30],
        simple_regret_exact_15[slice30],
        simple_regret_exact_16[slice30],
        simple_regret_exact_17[slice30],
        simple_regret_exact_18[slice30],
        simple_regret_exact_19[slice30],
        simple_regret_exact_20[slice30]]

approx30_results = pd.DataFrame(approx30).sort_values(by=[0], ascending=False)
exact30_results = pd.DataFrame(exact30).sort_values(by=[0], ascending=False)

### Best simple regret minimization IQR - approx:
lower_approx30 = np.asarray(approx30_results[4:5][0])[0]
median_approx30 = np.asarray(approx30_results[9:10][0])[0]
upper_approx30 = np.asarray(approx30_results[14:15][0])[0]

lower_exact30 = np.asarray(exact30_results[4:5][0])[0]
median_exact30 = np.asarray(exact30_results[9:10][0])[0]
upper_exact30 = np.asarray(exact30_results[14:15][0])[0]
```

```
# Iteration40 :

slice40 = 39

approx40 = [simple_regret_approx_1[slice40],
        simple_regret_approx_2[slice40],
        simple_regret_approx_3[slice40],
        simple_regret_approx_4[slice40],
        simple_regret_approx_5[slice40],
```

```
        simple_regret_approx_6[slice40],
        simple_regret_approx_7[slice40],
        simple_regret_approx_8[slice40],
        simple_regret_approx_9[slice40],
        simple_regret_approx_10[slice40],
        simple_regret_approx_11[slice40],
        simple_regret_approx_12[slice40],
        simple_regret_approx_13[slice40],
        simple_regret_approx_14[slice40],
        simple_regret_approx_15[slice40],
        simple_regret_approx_16[slice40],
        simple_regret_approx_17[slice40],
        simple_regret_approx_18[slice40],
        simple_regret_approx_19[slice40],
        simple_regret_approx_20[slice40]]

exact40 = [simple_regret_exact_1[slice40],
        simple_regret_exact_2[slice40],
        simple_regret_exact_3[slice40],
        simple_regret_exact_4[slice40],
        simple_regret_exact_5[slice40],
        simple_regret_exact_6[slice40],
        simple_regret_exact_7[slice40],
        simple_regret_exact_8[slice40],
        simple_regret_exact_9[slice40],
        simple_regret_exact_10[slice40],
        simple_regret_exact_11[slice40],
        simple_regret_exact_12[slice40],
        simple_regret_exact_13[slice40],
        simple_regret_exact_14[slice40],
        simple_regret_exact_15[slice40],
        simple_regret_exact_16[slice40],
        simple_regret_exact_17[slice40],
        simple_regret_exact_18[slice40],
        simple_regret_exact_19[slice40],
        simple_regret_exact_20[slice40]]

approx40_results = pd.DataFrame(approx40).sort_values(by=[0], ascending=False)
exact40_results = pd.DataFrame(exact40).sort_values(by=[0], ascending=False)

### Best simple regret minimization IQR - approx:
lower_approx40 = np.asarray(approx40_results[4:5][0])[0]
median_approx40 = np.asarray(approx40_results[9:10][0])[0]
upper_approx40 = np.asarray(approx40_results[14:15][0])[0]

lower_exact40 = np.asarray(exact40_results[4:5][0])[0]
median_exact40 = np.asarray(exact40_results[9:10][0])[0]
upper_exact40 = np.asarray(exact40_results[14:15][0])[0]
```

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

lower_approx = [lower_approx1,
            lower_approx2,
            lower_approx3,
            lower_approx4
```

```
             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,
             lower_approx22,
             lower_approx23,
             lower_approx24,
             lower_approx25,
             lower_approx26,
             lower_approx27,
             lower_approx28,
             lower_approx29,
             lower_approx30,
             lower_approx31,
             lower_approx32,
             lower_approx33,
             lower_approx34,
             lower_approx35,
             lower_approx36,
             lower_approx37,
             lower_approx38,
             lower_approx39,
             lower_approx40,
             lower_approx41]

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_approx17,
              median_approx18,
              median_approx19,
              median_approx20,
              median_approx21,
              median_approx22,
              median_approx23,
              median_approx24,
              median_approx25,
              median_approx26,
              median_approx27,
              median_approx28,
              median_approx29,
              median_approx30,
              median_approx31,
              median_approx32,
              median_approx33,
              median_approx34,
              median_approx35,
              median_approx36,
              median_approx37,
              median_approx38,
              median_approx39,
              median_approx40,
              median_approx41]

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,
              upper_approx22,
              upper_approx23,
              upper_approx24,
              upper_approx25,
              upper_approx26,
              upper_approx27,
              upper_approx28,
              upper_approx29,
              upper_approx30
```

```
            upper_approx30,
            upper_approx31,
            upper_approx32,
            upper_approx33,
            upper_approx34,
            upper_approx35,
            upper_approx36,
            upper_approx37,
            upper_approx38,
            upper_approx39,
            upper_approx40,
            upper_approx41]
```

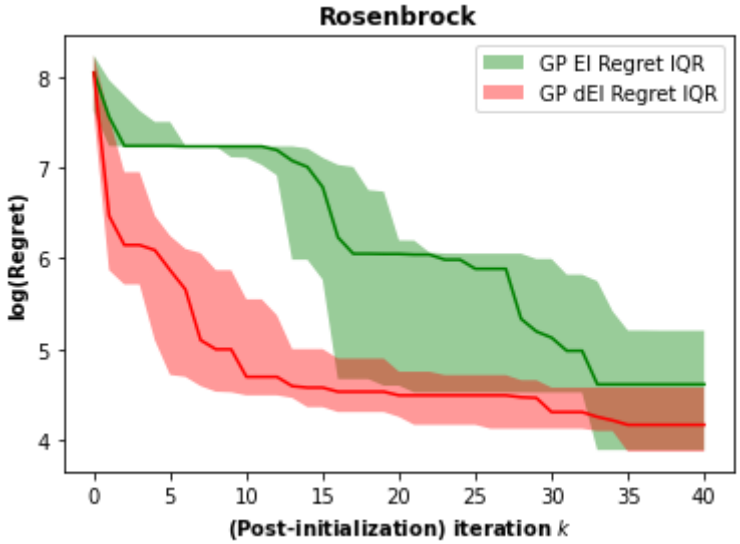
```
### Summarize arrays: 'exact'

lower_exact = [lower_exact1,
            lower_exact2,
            lower_exact3,
            lower_exact4,
            lower_exact5,
            lower_exact6,
            lower_exact7,
            lower_exact8,
            lower_exact9,
            lower_exact10,
            lower_exact11,
            lower_exact12,
            lower_exact13,
            lower_exact14,
            lower_exact15,
            lower_exact16,
            lower_exact17,
            lower_exact18,
            lower_exact19,
            lower_exact20,
            lower_exact21,
            lower_exact22,
            lower_exact23,
            lower_exact24,
            lower_exact25,
            lower_exact26,
            lower_exact27,
            lower_exact28,
            lower_exact29,
            lower_exact30,
            lower_exact31,
            lower_exact32,
            lower_exact33,
            lower_exact34,
            lower_exact35,
            lower_exact36,
            lower_exact37,
            lower_exact38,
            lower_exact39,
            lower_exact40,
```

```
            lower_exact41]

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,
            median_exact22,
            median_exact23,
            median_exact24,
            median_exact25,
            median_exact26,
            median_exact27,
            median_exact28,
            median_exact29,
            median_exact30,
            median_exact31,
            median_exact32,
            median_exact33,
            median_exact34,
            median_exact35,
            median_exact36,
            median_exact37,
            median_exact38,
            median_exact39,
            median_exact40,
            median_exact41]

upper_exact = [upper_exact1,
            upper_exact2,
            upper_exact3,
            upper_exact4,
            upper_exact5,
            upper_exact6,
            upper_exact7,
            upper_exact8,
            upper_exact9,
            upper_exact10,
            upper_exact11,
```

```
            upper_exact12,
            upper_exact13,
            upper_exact14,
            upper_exact15,
            upper_exact16,
            upper_exact17,
            upper_exact18,
            upper_exact19,
            upper_exact20,
            upper_exact21,
            upper_exact22,
            upper_exact23,
            upper_exact24,
            upper_exact25,
            upper_exact26,
            upper_exact27,
            upper_exact28,
            upper_exact29,
            upper_exact30,
            upper_exact31,
            upper_exact32,
            upper_exact33,
            upper_exact34,
            upper_exact35,
            upper_exact36,
            upper_exact37,
            upper_exact38,
            upper_exact39,
            upper_exact40,
            upper_exact41]
```

```
### Visualize!

title = 'Rosenbrock'

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(0, count, 5))
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.
```




```
1 time_approx, time_exact
```

```
(825.6076500415802, 210.44350624084473)
```

```
1
```