Dixon-Price:

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 - Dixon-Price(x) 4-D:

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

```
                            + 3 * (2 * x3_training ** 2 - x2_training)**2
                            + 4 * (2 * x4_training ** 2 - x3_training)**2
                           )


# Constraints:
lb = -10
ub = +10

# 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])
```

```
                                     for s in start_points_dict])
        x_best = np.empty((n_start, len(self.parameter_key)))
        f_best = np.empty((n_start,))
        opt = Parallel(n_jobs=self.n_jobs)(delayed(minimize)(self.acqfunc,
                                                                 x0=start_point,
                                                                 method=method,
                                                                 jac = True,
                                                                 bounds=self.parameter_
                                               start_points_arr)
        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

```

```
1623337308.882282
```

```
### ESTIMATED GP EI GRADIENTS

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

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

```

```
Evaluation       Proposed point           Current eval.          Best eval.
init     [-1.65955991  4.40648987 -9.9977125  -3.95334855].       -124757.8119225052
init     [-7.06488218 -8.1532281  -6.27479577 -3.08878546].       -64499.71723544843
init     [-2.06465052  0.77633468 -1.61610971  3.70439001].       -3468.30836846115
init     [-5.91095501  7.56234873 -9.45224814  3.4093502 ].       -121117.800613240
init     [-1.65390395  1.17379657 -7.19226123 -6.03797022].       -57099.80140311415
1        [-3.68968738  3.73001855  6.69251344 -9.63423445].       -152203.3072503334
2        [-6.24736654  2.44991805  8.11618992  9.79910357].       -186189.5031681948
3        [ 5.02242081  1.58721081  8.49408363 -8.70520033].       -142988.6701515681
4        [ 10.         -10.          -0.74888702   3.2377255 ].          -74538.173
5        [ 5.97945681  6.21258207  7.19997183 -0.82898378].       -38802.31394092165
6        [ 7.62336637 -7.47843039  2.66921668 -4.44574535].       -28622.8176318825
7        [-10.  10.  10. -10.].           -341021.0      -3468.30836846115
8        [ 3.90304129 -5.8456392  -8.34408123 -3.88978677].       -77430.77117244426
9        [ 6.88781365  2.44433273 -5.87040082  3.82976847].       -18301.7027809459
10       [-2.3791125   7.04806575  3.94693079  4.26519795].       -26661.4933943360
11       [-10. -10.  10.  10.].           -365021.0      -3468.30836846115
12       [ 10.  10.  10. -10.].           -324981.0      -3468.30836846115
13       [ 0.68336005 -8.89855382  2.4019798   1.62350406].       -51015.33952779464
14       [ 0.80119154 -9.16723906 -5.2983827  -9.59120497].       -212068.3714515436
15       [ 9.09378449  8.73744268 -1.31302496 -1.37978624].       -41491.6648447880
16       [ 1.79816375 -2.7037654   9.83736219 -0.32105306].       -116244.0603688108
```

```
17       [-7.81107405 -7.80690855  8.24125081 -3.51817053].       -96716.34725474233
18       [ 6.03011227 -4.93770333 -9.36205475  9.0561959 ].       -221388.3566155189
19       [-0.29068499 -9.04011931 -4.02183401  4.43267673].       -66267.732177043
20       [-9.45974524  0.51730768  4.69597999  0.11149562].       -6096.000605021322
21       [ 3.67388287 -2.87543511  1.9187423   6.41420845].       -26486.9050234320
22       [-10. -10. -10.  10.].           -397021.0      -3468.30836846115
23       [ 1.61122392  7.5144002  -7.47341828  9.84907412].       -219731.474787438
24       [ 10. -10. -10. -10.].           -380981.0      -3468.30836846115
25       [-4.67489351 -4.57828696  2.94378358 -9.19489192].       -116235.802363645
26       [-0.77290578 -3.36889522  7.86252834 -5.82002822].       -63841.646666585
27       [-10. -10. -10. -10.].           -397021.0      -3468.30836846115
28       [-10.           3.29618807  -2.75102817  -6.54645079].         -33857.969
29       [10. 10. 10. 10.].       -324981.0      -3468.30836846115
30       [ 10.  10. -10.  10.].           -356981.0      -3468.30836846115
31       [ 4.83821015  9.90305218 -9.36128664 -9.34724668].       -290820.3772641564
32       [-9.71362688  8.90969444  3.4563994  -2.59171866].       -57956.83084541687
33       [-4.62072823 -5.84418796  2.94898185  9.79557763].       -155109.051449009
34       [ 10. -10.  10.  10.].           -348981.0      -3468.30836846115
35       [-10.          -0.50657824  -4.45633884   1.60484049].         -5565.2911
36       [5.06118926 3.0975079  9.01609781 8.84692276].         -163768.309182576
37       [-10.  10. -10. -10.].           -373021.0      -3468.30836846115
38       [ 10. -10.  10. -10.].           -348981.0      -3468.30836846115
39       [ 8.2842522  -6.43177972 -7.3485378   3.19018699].       -53494.19368648512
40       [ 9.24124742  3.81915983 -4.04893168 -8.30867165].       -84168.85840270622
```

```
### ESTIMATED GP EI GRADIENTS

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

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

```

```
Evaluation       Proposed point          Current eval.         Best eval.
init     [-1.28010196 -9.48147536  0.99324956 -1.29355215].       -65998.62492491212
init     [-1.59264396 -3.39330358 -5.90702732  2.38541933].       -18480.2085136143
init     [-4.00690653 -4.6634545   2.42267666  0.58284189].       -5357.298687487265
init     [-7.30840109  0.27156243 -6.31120269  5.70670296].       -39506.0501987761
init     [ 7.07950585 -0.11526325  6.93122971 -8.40709046].       -100181.871642487
1        [-2.26214698  5.87274909  1.60008358 -6.75402803].       -42299.5645871526
2        [-8.77692463 -2.54457149 -1.31781306  9.47735243].       -132131.8483847979
3        [ 4.62190764 -6.83721008  8.23256169 -9.96152115].       -221384.112700435
4        [ 8.70233959  0.42166332 -8.66816053  8.53322589].       -162799.863038277
5        [-3.46661731 -0.39559918 -3.41298829 -7.58101805].       -57765.7299728270
6        [-9.3528057  -0.16784675  2.91108351 -9.91348457].       -151154.0478201447
7        [ 5.87954056 -2.52000408  8.75494609  8.0392552 ].       -131039.9203983427
8        [10. 10. 10. 10.].       -324981.0      -5357.298687487265
9        [ 9.33988639  0.28284014 -7.13390479 -8.51921815].       -123912.892183966
10       [ 3.27690332  3.28708214 -1.44062271 -2.42812913].       -1379.982986092025
11       [ 7.93183147 -5.23912097  2.68785335  6.56683347].       -33550.60090311994
12       [0.19228331 7.33765529 6.54124141 7.73210077].         -92575.14741796735
13       [ 4.17183347 -5.30217661 -2.43725911 -5.04062651].       -17658.6608942975
14       [-10. -10. -10.  10.].           -397021.0      -1379.982986092025
15       [-2.82718104  7.52894698 -6.74703384 -0.36123532].       -48139.46468298068
16       [ 10.  10. -10.  10.].           -356981.0      -1379.982986092025
17       [-10.          10.          -2.85310518 -10.          ].         -253036.86
18       [-10. -10.  10. -10.].           -365021.0      -1379.982986092025
```

```
19       [-6.69692911 -4.32856522  8.35858338  6.45628149].        -88726.52907193231
20       [ 2.6956679  -0.0955432   7.16240135 -0.63670877].        -31817.69915499351
21       [-9.46753889  7.57506663  3.40441431 -2.45231133].        -32004.14682106284
22       [  7.44166025  10.          -5.87410249 -10.        ].        -254182.10
23       [ 8.21499945 -1.44616147  5.90323425  4.12780274].        -18443.50397833349
24       [-7.34439343  4.37137915  8.9317863   2.08156362].        -76466.33625698599
25       [ 2.22110047  8.39989695 -1.66769146  5.14324042].        -50522.61342521185
26       [  3.40316599 -10.         -10.          10.        ].        -386006.40
27       [ -0.38345842  10.         -10.          10.        ].        -365008.97
28       [ 8.7657383   4.82994739 -0.03301329  7.01572752].        -41790.11034695122
29       [ 3.10174137  8.19973381  9.28968585 -8.58272206].        -191815.5957144834
30       [-10.  10. -10.  10.].          -373021.0      -1379.982986092025
31       [-10.          -1.30135562  -9.87043448 -10.        ].        -292089.00
32       [ 6.82221391 -6.94923273  4.60327609 -2.11763255].        -23524.57437851318
33       [-7.78512458  9.84189174  5.3612916   5.82187574].        -103689.1275462984
34       [-8.56227308 -7.91577403 -9.53506662  2.16480556].        -145385.8876678999
35       [-10.  10.  10. -10.].          -341021.0      -1379.982986092025
36       [10.          5.28096377 10.          4.52771423].        -121862.7026471497
37       [-9.15686549  0.80276656 -0.47030207 -0.18344141].        -322.9343567261949
38       [ -1.49914984 -10.          -1.37686679  10.        ].        -243991.24
39       [-10. -10. -10. -10.].          -397021.0      -322.9343567261949
40       [-2.45135556e+00  5.27419899e-03  9.30765213e+00 -7.78615701e+00].        -1
```

```
### ESTIMATED GP EI GRADIENTS

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

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

```

```
Evaluation       Proposed point            Current eval.         Best eval.
init     [ 1.01595805  4.16295645 -4.18190522  0.2165521 ].        -5185.478651216821
init     [ 7.85893909  7.92586178 -7.48829379 -5.85514244].        -83515.21921243734
init     [-8.97065593 -1.18380313 -9.40247578 -0.86333551].        -95899.98071510748
init     [ 2.98288095 -4.43025435  3.52509804  1.81725635].        -5245.566173274534
init     [-9.52036235  1.17708176 -4.81495106 -1.69797606].        -6987.20591009812
1        [-4.16414452 -0.84627201  7.21067826  1.72505809].        -33066.17455447326
2        [-1.42093745 -2.38317783  4.89209527 -7.84297806].        -63728.37816006544
3        [-1.98230524 -6.0350318  -5.50491833  4.81186289].        -35268.92783271308
4        [-6.95962222 -6.12078547  1.05388197 -2.78453945].        -14518.91098076006
5        [0.63810404 8.56152948 2.44970438 5.87960129].        -60435.21909838218
6        [ 5.86849691 -6.28138395 -5.51873265 -6.54080399].        -57423.89305426612
7        [-10.  10. -10.  10.].          -373021.0      -5185.478651216821
8        [-5.69652173  2.33901134  0.4494956   0.55949224].        -609.8771978113376
9        [-10.  10.  10.  10.].          -341021.0      -609.8771978113376
10       [-3.88225406  5.58733297  9.07159878  6.32749837].        -104829.2971770740
11       [-5.56655434 -5.05469054  1.85578009  7.1880508 ].        -48086.20229320106
12       [ 0.76213235  6.97711173  8.30170294 -8.73896127].        -153483.395027611
13       [ 8.30702859  4.35962691  4.99472689 -0.47784886].        -8120.920764787085
14       [-0.77767833 -1.76973304 -8.48492547 -9.95433933].        -234676.0428512438
15       [-5.87917964  9.0108278   8.03503233 -0.77077353].        -100145.0520964455
16       [-2.25892289  0.28501765  8.21214282  0.29800465].        -54626.84217539074
17       [-10.          -0.16358411  10.         -10.        ].        -264919.52
18       [-10. -10. -10. -10.].          -397021.0      -609.8771978113376
19       [ 9.04342598 -7.52262686 -3.84028198  2.09841499].        -26504.2937293117
20       [ 8.2215293   7.78343336 -6.67089118  3.55133193].        -49422.28098726387
```

```
21       [-10. -10. -10.  10.].           -397021.0      -609.8771978113376
22       [ 6.00557472 -9.33087149  6.90123489  9.7261517 ].       -222296.5634073108
23       [10. 10. 10. 10.].      -324981.0      -609.8771978113376
24       [7.88158233 1.12253401 0.53834524 6.85949842].         -35124.93281450683
25       [ 9.85580767 -7.52663578  8.20821445 -3.07379111].       -82664.53413280251
26       [ 10. -10. -10.  10.].           -380981.0      -609.8771978113376
27       [ 5.71924422  2.07614097  0.7997372  -8.17755021].       -70738.41859742705
28       [ 4.8092043   5.72160034 -1.0963792  -9.62895308].       -146581.3676586469
29       [-7.87970602  3.81911201 -1.34505225 -9.31070001].       -124937.4707069073
30       [-10. -10.  10. -10.].           -365021.0      -609.8771978113376
31       [-7.4127131   8.16208771 -8.89228429 -5.05995323].       -121569.1349316843
32       [-10.  10.  10. -10.].           -341021.0      -609.8771978113376
33       [ 10.  10.  10. -10.].           -324981.0      -609.8771978113376
34       [-10. -10.  10.  10.].           -365021.0      -609.8771978113376
35       [-7.29001145  4.72509972  0.9480778   8.87212374].       -103435.9294256939
36       [-3.97957448  5.36380316 -7.66030075  7.28691783].       -97079.18765095875
37       [-10.          -9.47355807   7.95468146  -0.29994404].          -127691.13
38       [  2.35947303  10.          -10.          10.        ].          -362825.40
39       [-9.51579139  9.81021872  0.52860391 -0.74326529].       -81973.90595630789
40       [ 5.12390521 -1.21623476 -7.41214585  8.27970981].       -120596.7323907361
```

```
### ESTIMATED GP EI GRADIENTS

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

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

```

```
Evaluation       Proposed point          Current eval.         Best eval.
init     [9.34059678 0.94464498 9.4536872  4.29631987].         -98038.88529472692
init     [ 3.95457649 -5.67821009  9.5254891  -9.8753949 ].       -250082.3135303139
init     [-4.94035275 -1.30416935  5.58765844 -6.04629851].       -30605.91051461647
init     [ 7.25986471  9.66801354 -6.72315517  1.94667888].       -84981.75728489687
init     [-9.82027805 -2.26857435 -9.11679884  9.13305935].       -209926.3636588899
1        [ 0.90405303  0.4880816   2.75220488 -1.97009113].       -745.6386114710477
2        [ 5.30486793 -6.47973031  3.69857697  5.99163089].       -34382.12799826672
3        [ 2.44668677  7.07389915 -9.38608839  0.44996751].       -105258.3008366837
4        [ 4.73707556 -0.72479098 -9.3121442  -0.43521843].       -91408.63072502242
5        [-7.88752674  6.9996629   9.5911258   7.81351701].       -167099.6706018694
6        [ 6.97297538 -6.40571993 -2.05106734 -5.92730455].       -32891.56312061896
7        [0.46199955 7.11757846 8.44850995 7.74973553].         -125416.3674114957
8        [ 6.465232   -8.2562289  -5.78689094  5.87653584].       -73152.30153078973
9        [-6.21902391 -6.61688609 -0.04378302 -3.21504386].       -19491.75644008322
10       [-6.38615157  7.9844663   9.46020353 -2.30152042].       -123642.1261867514
11       [ 2.70946926  9.36046965  8.3372206  -9.2753314 ].       -217193.1187977168
12       [-4.57747469  6.76415735 -4.11298047  8.65053766].       -115282.9636007825
13       [2.35747262 4.35064884 6.12515272 9.24202752].         -126021.9424107598
14       [-8.19755724  1.6166877  -9.27463953 -1.1878353 ].       -88160.50255348616
15       [ 7.99683297  3.755866   -3.73641094 -8.81543277].       -103946.0620075571
16       [-9.01084566  5.87964272 -2.96427056 -7.97390816].       -80461.69979034693
17       [ 10.  10. -10. -10.].           -356981.0      -745.6386114710477
18       [-7.07788877 -1.21727011  4.87587121  3.08177701].       -8198.50117296675
19       [ 2.12735998  5.39297028 -6.17417945  6.93749789].       -63309.96222694547
20       [10.          10.           4.89395292  4.99469583].       -84690.58972027371
21       [-6.91618391 -9.19178706  4.7577589   8.1363842 ].       -136010.9053978401
22       [ 2.53687866 -7.51555741 -8.87336299  8.92430991].       -219167.0015024325
```

```
23       [-8.24208797 -9.25718255  2.59909975 -9.02269984].       -168857.199441884
24       [9.4019532  2.20661934 0.75567978 8.69750771].          -90720.49609506624
25       [ -9.59891868  10.          -10.           1.93321675].          -197497.20
26       [ 10.  10. -10.  10.].           -356981.0      -745.6386114710477
27       [  8.2767657   -8.39034252 -10.          -10.        ].          -341855.08
28       [-8.96018513  2.79650004  1.77717835  8.72422378].       -91883.97912615376
29       [-2.17475388 -2.43104151 -6.91798907 -8.86765723].       -137132.669750074
30       [ 10. -10.  10.  10.].           -348981.0      -745.6386114710477
31       [ 9.78624228  6.18125461  8.04643129 -3.59879505].       -55846.80063846495
32       [-3.71704512  0.63300159 -3.02912155 -4.65155591].       -9580.777575844764
33       [ 0.0851234  -9.86477687 -7.48975636 -1.85511777].       -121214.948303217
34       [-3.19455143 -5.06844861 -2.27724596  5.36985965].       -21064.2560416366
35       [ 2.63221979  3.9114145  -0.64914921  5.04538811].       -12229.2796807114
36       [ 2.94320937  7.31830872  1.30397082 -5.33484823].       -34126.57198613119
37       [-10.         -10.         -10.          -1.52272857].          -221478.01
38       [ -5.6707746  10.        -10.        -10.       ].       -369345.4342838263
39       [-10.         -10.          -6.70127059 -10.        ].          -289111.19
40       [ 0.01296168 -7.12732782  7.82016317 -1.55057487].       -70938.12309347525
```

```
### ESTIMATED GP EI GRADIENTS

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

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

```

```
Evaluation       Proposed point           Current eval.          Best eval.
init     [-5.56013658  7.41464612 -5.86561689  8.37221816].       -123365.276489362
init     [-0.23177622  2.23487726  5.31815713  0.36835976].       -9167.819359373088
init     [-4.06398997 -6.24557543 -8.38517462  4.76880592].       -89817.03050330501
init     [-1.17381554 -6.83380265  7.59874062 -4.51827076].       -67194.19259941578
init     [-1.71529962 -4.07840135  2.57575818  1.5967562 ].       -3383.131100651736
1        [ 1.55325716 -9.96715655  0.30945224  2.79590352].       -78973.83235307777
2        [ 4.56139684 -4.673529    0.23379905  2.52038143].       -3764.515116656609
3        [-2.15134925  9.53418759  4.82671391  6.46019907].       -96545.59507289826
4        [ 4.17941015  0.37676612 -9.58435076 -5.8403882 ].       -125098.433085905
5        [-4.20489896 -6.1571556  -6.8351617  -9.35801008].       -175060.215088253
6        [-3.76365226  7.78239697 -8.18300187 -5.22131556].       -94683.43682804274
7        [ 0.41211986  0.47256529 -3.34919958  6.13953029].       -26245.2731263362
8        [-0.25183718 -3.30684237  8.97845951  1.46012155].       -82281.90882897378
9        [-8.43262179 -0.81279137 -3.32426401 -3.2850448 ].       -4335.934742847627
10       [-4.45934361  6.06017597  8.73696795 -2.92485867].       -76933.0298360569
11       [ 5.70578654  1.32017865  2.21996359 -6.47121455].       -26841.30454703817
12       [-10.         -10.          -9.45509917  -9.59674333].          -345256.21
13       [-5.79261282  9.88159419  1.54685764 -4.35636722].       -86296.40204165212
14       [-9.58242141  6.39325499  3.81306329  5.48600232].       -31052.7207547133
15       [ 8.81346656  0.26074237 -7.90133971  4.47083934].       -55957.58957399476
16       [ 10. -10. -10.  10.].           -380981.0      -3383.131100651736
17       [-9.81344616  7.08235098  9.35724572  9.34014123].       -218138.872769031
18       [ 9.83597477 -2.84190849  8.94930056 -9.75329078].       -211370.9218805128
19       [ 2.74326124 -7.27587891 -4.48922446  9.17807962].       -147733.798337997
20       [10. 10. 10. 10.].       -324981.0      -3383.131100651736
21       [ 3.2666596   8.62078011  9.59676632 -9.95043148].       -276765.8023028767
22       [ 8.41974157  8.57362385 -4.47123065 -2.49610691].       -42578.43219101316
23       [-6.71412576 -6.47032038  5.49564249 -9.1363839 ].       -134102.605403016
24       [ 5.76122226 -5.97967704 -2.13038836 -8.57292222].       -98296.95177806134
```

```
25       [-9.5137551  -9.52877563 -2.42470317  1.12786197].        -74613.93662651363
26       [ 5.63902137 -4.07372558  5.15047326  1.24959081].        -11347.11766688968
27       [ 10. -10. -10. -10.].            -380981.0      -3383.131100651736
28       [-3.51452738  2.62244882 -9.37027444  2.4460871 ].        -92205.82153243455
29       [ 10. -10.  10.  10.].            -348981.0      -3383.131100651736
30       [-5.87723567  1.52217007  0.14509913  6.77549639].        -33888.10530914785
31       [-7.85956398 -6.05646207  2.03337719  8.46405003].        -93690.63779473328
32       [-0.66025888 -9.79876343  4.17691916  1.06487357].        -80269.72780342265
33       [-10.  10.  10. -10.].            -341021.0      -3383.131100651736
34       [6.44652978 7.77340351 4.97530518 2.3664283 ].            -31586.85254624169
35       [-10. -10.  10.  10.].            -365021.0      -3383.131100651736
36       [-1.53275275  3.86512508  6.75586768  7.15377169].        -61460.81820364318
37       [6.21477535 1.27955868 4.70987785 6.52962258].            -31574.76671054321
38       [ 5.99299568  5.49723006  7.66029555 -4.08517444].        -46139.14838000956
39       [-0.19518197  9.15097708  6.59601634 -4.81104503].        -80723.41013644819
40       [-9.79702142 -1.03775527  9.09024991  4.83976418].        -89074.50715232249
```

```
### 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     [ 7.85720303 -3.36040389  6.42458246 -9.16606749].        -127092.7561755139
init     [-7.8468664   1.90104128  0.59634724 -1.62385143].        -624.5256755977402
init     [-3.29184301  2.45038864 -1.23717148  4.71764213].        -8859.81358392905
init     [0.36072824 1.577172   2.90710192 9.80448543].            -144159.4084820958
init     [ 6.39716394 -1.7359813   7.5253531   6.47518865].        -63008.48019540512
1        [ 4.34664291  8.74699069 -2.96380463 -4.92731805].        -55069.53937594843
2        [-5.57681425  2.83874786  4.53696952  0.08226863].        -5473.77883338242
3        [ 8.56608527  1.00066366 -5.6557535   4.78400723].        -22620.59205919053
4        [-3.88060276 -3.60562409  2.63626196 -8.23698378].        -73548.32801665392
5        [-5.8247851  -6.52272924 -8.70125002  9.33818614].        -225528.6475274047
6        [-4.22599656 -6.273846   -3.61341454  0.26400804].        -16991.30735279416
7        [8.53192076 6.17342963 2.07530059 5.72450643].            -25349.66111346605
8        [-10.          6.98409391   5.63877945 -10.        ].            -183975.60
9        [ 6.30471624 -8.43012351 -5.35385847 -9.5705537 ].        -192096.1937162695
10       [-3.69260396  5.83022375 -9.57366863  1.14098381].        -105387.4543345151
11       [ 5.16565529  3.58828793  9.82334313 -3.91747081].        -110233.116869007
12       [-7.664189    9.57492033 -1.81280471  3.6531581 ].        -76331.11332073945
13       [-10.  10.  10.  10.].            -341021.0      -624.5256755977402
14       [-9.29604707 -8.38751066  6.35426496  9.78206391].        -205876.7887789898
15       [ 1.34667705  1.66784567 -7.21098005 -3.98954394].        -37546.82697659439
16       [10. 10. 10. 10.].        -324981.0      -624.5256755977402
17       [-9.10121252  5.55793128  3.33297346  6.97861209].        -46379.25067804724
18       [ 6.4837722  -2.51724083  1.49155872 -2.11632026].        -475.2595795537462
19       [ 4.60029783 -7.6833382  -3.64473115  5.6386107 ].        -47362.89117482109
20       [-10.          6.45895016 -10.          8.03296471].            -207303.46
21       [-1.92509357 -4.01673092  9.89260541  8.76451337].        -204685.5921988336
22       [-0.07427914  7.26951153  5.44189591  7.34136235].        -72374.71439239231
23       [-3.30368771 -6.47927689  4.84493609 -0.0774172 ].        -23905.61284536291
24       [-9.11446523 -2.03954712 -8.55029407 -2.95608945].        -69358.13765365355
25       [ 0.77200041  9.62659846 -2.18138662 -6.42799569].        -96910.333085085
26       [-10. -10.  10. -10.].            -365021.0      -475.2595795537462
```

```
27       [-7.64605755  8.06941535 -6.32181203 -9.57376147].       -197433.6645985006
28       [ 0.86575562  4.43818713 -7.09708215  9.85575157].       -192987.0062133993
29       [ 10. -10. -10.  10.].           -380981.0      -475.2595795537462
30       [-10.         -10.          -1.91102782 -10.        ].          -252291.54
31       [ 10.  10. -10. -10.].           -356981.0      -475.2595795537462
32       [ 10.  10.  10. -10.].           -324981.0      -475.2595795537462
33       [ 10. -10.  10.  10.].           -348981.0      -475.2595795537462
34       [-3.95425029 -7.44845879 -8.80473726 -9.44785354].       -246016.8469944214
35       [10.         10.         -6.17969262 10.        ].       -255539.0613635251
36       [-9.6115913  -0.89584212 10.         -7.6423252 ].       -167075.3849129440
37       [ 10.         -10.          2.75613529   2.89237383].          -74966.256
38       [-10.         -10.         -10.          2.84292408].          -223359.31
39       [ 0.90848658 10.         10.          0.53515499].       -187930.3595840614
40       [ 9.07118018 -7.24665332 -8.6698848  -1.14240882].       -93484.3177546862
```

```
### 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     [-8.47383421  5.59837584 -1.23181537  4.46930356].       -17019.79107531184
init     [ 9.55979024  0.76991741  0.02240927 -8.55897733].       -86052.31833948194
init     [-4.63122040e+00 -2.34998349e-03  3.58459992e+00  6.07478072e+00].        -2
init     [-2.38117734 -8.68127306 -4.23708801  8.19187055].       -129535.1629563858
init     [-5.73229293 -0.95752076  8.62412039 -9.50201545].       -185668.2647263406
1        [ 6.75835989  5.37295013 -3.72010646  1.45250665].       -6975.589978136528
2        [-2.26867274 -4.96775263 -3.10681926 -5.91972827].       -28537.54981454135
3        [-7.1620489   2.23165943  9.48914494 -4.53159483].       -99540.92877448558
4        [ 4.29877809  2.09137364 -5.85501544 -4.79573932].       -24060.70100027514
5        [ 5.54571144 -1.93184305 -4.33889721  9.71942618].       -154147.0884339929
6        [-8.29791508  2.55444133  2.18471017 -8.00103752].       -64495.96200318241
7        [-4.37541545 -0.5346361  -5.18090179  4.60429146].       -17952.04503879531
8        [ 1.07740606  7.90693058 -1.60531584 -3.10335   ].       -32497.45135574664
9        [ 4.32564604  9.62119313 -9.80342151  4.4727457 ].       -175341.3282758457
10       [ 10.         -10.          -1.76857889   2.88827014].          -74435.768
11       [ 10. -10. -10.  10.].           -380981.0      -6975.589978136528
12       [-7.17360778 -4.84967241  6.40731588  0.96189955].       -28712.39669282479
13       [ 10.  10.  10. -10.].           -324981.0      -6975.589978136528
14       [ -6.5528384   -1.39813508 -10.          10.        ].          -298359.59
15       [ 10. -10. -10. -10.].           -380981.0      -6975.589978136528
16       [-7.9829215   9.96558946 -9.45385173 -6.10303813].       -199109.3733995131
17       [ 7.66294306 -2.84213803 -8.47618925 -8.69671584].       -166674.9497706916
18       [ 2.98610483 -8.04222501  6.13596627  8.13477803].       -116499.0321392406
19       [-1.85882377  5.50532386 -7.48281258  2.03048285].       -42817.96641194064
20       [8.55716728 9.13690962 1.75279783 8.46693102].            -130501.4500009226
21       [-1.57366787 -9.43769678 -8.83544712 -7.55230358].       -207266.5363637169
22       [-8.33660472 -7.44132911 -3.97589571 -1.8706741 ].       -33506.95236753423
23       [ 0.91280031 -4.58400286  8.74425647 -3.60044314].       -78987.87988216867
24       [-10.  10.  10. -10.].           -341021.0      -6975.589978136528
25       [-2.08000587  6.91069233  5.94203563  1.06121231].       -31288.57977997730
26       [ 3.72013071 -5.34851399  1.54251013  0.57722557].       -6039.95309003843
27       [ 8.25477835  2.45208354  3.14975175 -8.1648879 ].       -68776.69144110718
28       [-9.22461904 -8.50807928  2.67217363 -9.06121187].       -153473.5569856350
```

```
29       [8.50304142 3.73774679 8.76724186 6.66865119].          -94016.03530372416
30       [ 10. -10.  10. -10.].          -348981.0      -6039.95309003843
31       [ 8.12214103  9.99850144  5.40115294 -1.18595284].       -80677.87391528238
32       [-9.85036512 -6.59222902  1.73615674  9.00261156].       -122181.2260239403
33       [-0.56037139  1.61542605 10.         10.        ].       -262538.5592601795
34       [-4.91115304  1.5427759  -6.68424648 -9.77374709].       -179755.6270391527
35       [-10.  10. -10.  10.].          -373021.0      -6039.95309003843
36       [ -5.45447602 -10.          10.          10.        ].          -361164.74
37       [-10. -10. -10.  10.].          -397021.0      -6039.95309003843
38       [-10.         10.         10.          3.661088].          -197750.9185587367
39       [ 10.  10. -10. -10.].          -356981.0      -6039.95309003843
40       [-10. -10. -10. -10.].          -397021.0      -6039.95309003843
```

```
### 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     [7.46858806 9.37081326 7.3838908  0.61711383].          -86573.97855073148
init     [-5.34543344 -9.77202391 -1.39062363 -1.9529728 ].       -78014.94707602388
init     [ 0.45349343 -0.43216408  1.10712948  0.86772035].       -25.89351629774407
init     [ 5.21791151  4.24749148  2.39364192 -1.47816459].       -2094.662571070279
init     [-4.21849944  9.47710482 -3.32451909 -5.62397878].       -85839.98682911636
1        [-3.60620407 -1.35048134 -4.59708517  6.02111773].       -29614.34759252149
2        [7.02116965 7.81234303 5.75333375 8.88043753].          -129115.3070880865
3        [ 2.38733468 -2.49139217 -6.58364982  0.33931039].       -24248.0812794335
4        [ 7.56352013 -7.67982822 -8.37749723  2.84161744].       -92575.686823112
5        [-5.28183607  6.44896586 -9.02454394 -6.64184392].       -126938.7289250310
6        [-8.91376283 -9.57999256 -6.95098062  9.64939713].       -257290.7332043392
7        [-9.6826463  -0.3889901   5.14264133 -8.41133424].       -83205.11096307916
8        [-0.44591076  5.04602604 -4.13162746  0.15920843].       -7889.447613833232
9        [-0.81981182 -3.75576157  2.39337966 -7.50364036].       -50973.32732738554
10       [ 6.98608516  8.9084123  -7.19395552 -9.08060074].       -191413.5170304686
11       [ 6.61186169 -9.02334481  1.34103459  6.04361295].       -69893.69310739316
12       [-4.95566202 -0.69807525 -2.84596135 -4.0933849 ].       -6249.828029368294
13       [-8.62371369 -2.43071156 -9.9135532  -8.61774405].       -220135.3921522664
14       [ 7.25721416 -1.35983896  9.25077409  2.93432361].       -89601.25460647872
15       [ 9.5056863   3.58618034  2.73863527 -1.09937986].       -989.4998714779908
16       [ 8.63675564 -9.78898035  4.97635133 -5.02173071].       -85866.47285240376
17       [-8.87173722  9.60530842  4.18381046 -3.54573537].       -78594.56686815829
18       [ 7.65858112  1.1873229  -3.41218412 -4.96882152].       -12703.59774024508
19       [-3.51296977 -2.14237485  8.01964739  7.68652932].       -100174.8125148277
20       [ 3.88729122  2.89386017 -8.55538582  8.36656893].       -150385.5235925565
21       [ 8.14290147 -0.90943489  9.4506528  -6.76425   ].       -123773.1552840801
22       [-9.31540474  9.38993355 -6.57652645  3.14559475].       -89662.74292924849
23       [-1.31717943  4.6340913   9.30919578 -8.69237554].       -169726.852751085
24       [ 6.49597242 -8.88231691  9.8207384   9.40242253].       -279494.5738611478
25       [ 3.49120939  0.47218526  8.86783187 -1.27059239].       -73915.08882010846
26       [-10.         10.          7.85957702   7.2351591 ].          -164507.45
27       [-7.83064872  2.15675112  8.42197817  0.68410525].       -59439.79169643539
28       [ 6.22824474 -6.53188051 -3.40849765 -4.72519097].       -24440.4769001340
29       [-0.69636553  6.93093178  9.44036372  4.76930131].       -111972.8843779919
30       [ 10.        -10.         -10.          -9.94977397].          -377630.35
```

```
31       [-7.949679   -9.34510791  9.14340971 -1.5517733 ].       -160357.7522994395
32       [ 0.76354257 -5.34208385 -8.78129711 -9.98537045].       -256107.8141899402
33       [-3.82169833  7.79709999 -3.42272713  9.03621934].       -143406.9458619359
34       [-6.12416735 -5.39688856  2.41170879  5.10668644].       -19107.73733976571
35       [-10.  10.  10. -10.].           -341021.0      -25.89351629774407
36       [ 6.77387947  8.29094342  7.88717361 -9.5531293 ].       -196648.5142504547
37       [-3.18294357 -7.91863135 -7.28881591 -4.14702255].       -79145.8077242034
38       [-10.         -10.         10.          9.18615659].         -321453.85
39       [ 8.37519617  9.65355887 -5.54273128  0.08164679].       -71597.6689957656
40       [-4.16342788 -4.96923312 -9.40204518 -4.25782433].       -113218.0602519102
```

```
### 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     [-9.79251692  0.03749184 -0.08453414 -7.32340942].       -46403.7672209163
init     [-7.15777829 -5.62882649 -1.62983639 -5.03797663].       -21353.0900000741
init     [-8.31880698 -3.0900272  -6.66447307  7.57118171].       -85802.65922587285
init     [ 9.01928063 -9.22503248  3.98214783  1.45519631].       -57052.85171456846
init     [7.96014236 3.33797946 0.95675566 4.04854848].           -4516.871493114117
1        [ 7.46801686  8.12554554  5.16555974 -3.06196092].       -37961.04796864094
2        [ 1.79155607  6.63416988 -1.83334365 -8.8517705 ].       -115413.8905207449
3        [ 9.33019641 -7.52767745 -5.63058289 -0.50920484].       -36948.48564229587
4        [ 9.24871681 -5.28271506  6.79982967  9.30415074].       -143743.462714014
5        [-3.79629409 -6.6853837   6.54189652  8.77158881].       -129771.5779963778
6        [ 2.12600285  4.95023923 -9.32534107 -9.64414998].       -242691.8532438531
7        [ 1.28476433  2.5242373  -3.51043207  6.58802611].       -34357.7548633644
8        [ 2.77700112 -8.36901764 -0.12784879 -1.65296659].       -38044.75381098915
9        [-7.46662777  0.446501   -9.80889853 -0.24867195].       -111161.8531809383
10       [ 5.90229198 -0.97787337 -7.39679132  1.70640703].       -37321.40235687539
11       [1.45553679 9.66867785 2.5326465  3.88938752].           -71932.96085986265
12       [-2.80325151 -5.66028587 -7.21789028 -7.12490995].       -92468.98160414296
13       [-7.10958489 -6.43009064 -2.06472288  6.66090206].       -49844.05786060399
14       [ 2.88579401 -2.77392671 -3.18766244 -7.65057666].       -59757.0993733727
15       [10.          2.70093503 10.         10.        ].       -261303.9011520012
16       [-6.00453231  6.54755256 -3.53825723  9.75186528].       -168043.8232803098
17       [-8.05861704  3.05396769 -0.03663706  1.43109669].       -1605.374475503401
18       [ 7.16153312 -6.98673646  8.87492942 -7.80656261].       -148687.6690932148
19       [-4.01060296 -0.34982372  8.60881423  0.96692656].       -66464.95145720668
20       [ 10.  10. -10.  10.].           -356981.0      -1605.3744755034013
21       [-1.91232744 -4.09825371  2.92332128 -6.9653553 ].       -39302.57410984345
22       [ 10. -10. -10. -10.].           -380981.0      -1605.3744755034013
23       [ 3.33271293 -7.85136664  0.86644336  4.68323042].       -36441.97328817176
24       [-10. -10. -10. -10.].           -397021.0      -1605.3744755034013
25       [-5.38540934 -6.04055244 -8.33599896 -4.84152513].       -87608.44575302256
26       [-10.         -10.         -10.          2.43246565].         -222527.85
27       [-10.  10.  10. -10.].           -341021.0      -1605.3744755034013
28       [-9.94128745  8.70452119  9.5162855   3.90820484].       -143220.4347754996
29       [-6.21765874  9.31634603 -7.93239744 -3.26898575].       -108885.1788291416
30       [ 10. -10. -10.  10.].           -380981.0      -1605.3744755034013
31       [-6.1692829   8.70589311 -5.26969916  0.31795138].       -56524.2353441076
32       [ 10.  10.  10. -10.].           -324981.0      -1605.3744755034013
```

```
33       [ 7.64937266 -1.85456578  9.71079193 -0.13843564].       -109237.2642438487
34       [-10. -10.  10. -10.].          -365021.0      -1605.3744755034013
35       [ 3.06205081 -6.73516315  9.16175403  3.84221381].       -108499.088600745
36       [-4.06768138  6.31875927  5.56497672 -5.99182406].       -40942.05250288964
37       [10.         10.         3.17963682 10.        ].       -227547.3777268466
38       [ 10.  10. -10. -10.].          -356981.0      -1605.3744755034013
39       [-10.         -10.          10.          3.31529771].          -221195.31
40       [ -4.36163706 -10.         -10.          10.        ].          -392256.10
```

```
### 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     [ 5.42641287 -9.58496101  2.6729647   4.97607765].       -74102.72845843069
init     [-0.02985975 -5.50406709 -6.0387427   5.21061424].       -40371.05435055173
init     [-6.61778327 -8.23320372  3.70719637  9.06786692].       -147677.318290659
init     [-9.92103467  0.24384527  6.25241923  2.25052134].       -18605.7045481122
init     [ 4.43510635 -4.16247864  8.35548245  4.29151567].       -67109.3842877247
1        [-3.98599887 -7.72031276  6.57362653 -9.06207361].       -156405.430572730
2        [ 5.24133604  6.79774962 -7.90085176  0.21262617].       -57280.06880918705
3        [-6.98285725 -7.53553636 -0.44064091 -4.85155595].       -38348.42146299815
4        [-1.78826799  3.25598727 -1.45510672  1.58046094].       -1234.288419082589
5        [ 4.63220256 -9.79505919 -9.12123737  0.49337993].       -163638.392675186
6        [-7.83635489 -1.28730572 -7.74844317 -2.63232565].       -46381.8543295735
7        [ 6.92515619 -6.60878201  5.11920166 -9.02608502].       -123052.729551494
8        [ 8.75235832  9.38171401  9.18458426 -0.57986069].       -132475.170696769
9        [0.4502773  8.55166428 3.92918152 5.72526907].        -59209.78291322129
10       [ 1.23952645 -2.7410793  -5.02186756 -4.62861357].       -18030.53509888344
11       [ 1.99559358 -0.41077916  2.20886236  5.83753048].       -17711.50470929883
12       [-6.52612311  7.27650284 -4.77362568 -5.95519154].       -52657.29570167327
13       [-4.63875915  8.6124843   7.12702008 -7.73195445].       -123346.8915830602
14       [4.4189373  5.3779954  6.27947744 2.68037042].        -22182.70197600124
15       [-8.76637523  9.03915533 -9.83105068 -1.79825538].       -162304.383360464
16       [-1.72806915 -4.12214739 -2.17527422  6.47352369].       -32687.84052614772
17       [-4.86937269  4.69882169 -9.05647957  8.60545103].       -179812.440102977
18       [ -7.43018216  -0.3648024   -8.57110538 -10.        ].          -239282.39
19       [ 6.43486771  9.20981525 -3.38150431 10.        ].       -219318.155120282
20       [ 9.21445384  0.88200448  5.69431392 -3.45005854].       -13772.76544465842
21       [ 5.52855938 -4.30300172 -3.84014298 -9.60506256].       -147341.802689123
22       [ 10. -10.  10.  10.].          -348981.0      -1234.2884190825898
23       [ 0.06951243 -3.70532939 -9.7893336  -7.65624326].       -180548.394433312
24       [ 10.  10. -10. -10.].          -356981.0      -1234.2884190825898
25       [-10. -10. -10.  10.].          -397021.0      -1234.2884190825898
26       [-10.  10.  10.  10.].          -341021.0      -1234.2884190825898
27       [ 0.14344824  1.66407921  3.95524263 -4.55031754].       -8303.322150673222
28       [-9.0177282   4.17602767  0.45261751  6.82006506].       -38276.40299654401
29       [ 8.81065532 -2.0428769  -2.61424238  2.35505314].       -1553.486136509709
30       [ 9.21235497 -0.61948647  1.97276737  9.10396676].       -107732.7337650096
31       [   5.71056569  10.          10.         -10.        ].          -328218.95
32       [ 10. -10. -10.  10.].          -380981.0      -1234.2884190825898
33       [-4.41668352 -7.65512696  8.81717256  1.83811107].       -109472.656208088
34       [ 8.16731183  9.29909245 -0.28328168 -7.77519069].       -113354.615637680
```

```
35      [10. 10. 10. 10.].       -324981.0      -1234.2884190825898
36      [ 2.86039405  4.90144469 -7.02565409 -9.39688178].       -165370.5136728394
37      [-10. -10. -10. -10.].           -397021.0      -1234.2884190825898
38      [-8.92471105 -0.19134021  7.430891   -6.47226304].       -60292.8143574527
39      [ 10. -10. -10. -10.].           -380981.0      -1234.2884190825898
40      [-6.42077873  8.72393559  2.25703754  1.73041764].       -50447.2554778789
```

```
### 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    [-6.39460622 -9.61049517 -0.73562947  4.49867858].       -80243.41172762495
init    [-1.59592791 -0.29145804 -9.74438371 -0.25256785].       -108928.0486060041
init    [ 8.83613305  7.01590179  4.5992894  -7.82527856].       -75431.53172537561
init    [ 7.87808341  7.14308494 -6.69826765  2.64668028].       -39961.8709790843
init    [-9.59032774 -7.66525462 -3.67265377 -6.84175387].       -73885.31196002741
1       [ 8.98204795  9.73346661 -3.23891901 -5.20250642].       -78768.56237678342
2       [-2.46813998 -7.3003779  -2.91425156  2.82909317].       -27001.415791210
3       [-2.7011834   8.03388106 -4.64351945  0.87154575].       -38595.660189414
4       [ 8.76589534 -3.7692322   0.38597262 -4.05301551].       -5098.7058724192575
5       [ 0.40212877  5.87654958 -6.74210512 -8.93336634].       -141815.517547776
6       [-10.  10. -10.  10.].           -373021.0      -5098.7058724192575
7       [-3.08409357 -3.60866716  5.26352461 -6.93028397].       -45137.382284599
8       [-9.90602549 -4.748682    9.51818141  7.44931505].       -151073.3013262496
9       [ 1.38380213 -6.05647422 -9.74783179 -9.20728444].       -254312.040468482
10      [-0.66080332 -7.11489274 -6.24181798  9.58065679].       -186590.993155904
11      [  3.1851585  10.        -10.         10.        ].       -362176.938583523
12      [ 3.41314276  7.39306433 -0.25317011 -0.71476575].       -22600.982647305
13      [ 10. -10.  10. -10.].           -348981.0      -5098.7058724192575
14      [-8.33083797 -2.90069237  2.58400674  2.53507465].       -2567.49074682586
15      [-2.57164218  8.12805756  6.98338235 -0.57169137].       -60443.36981524969
16      [ -1.41391623   9.22169745   8.03274203 -10.        ].        -249307.84
17      [-2.1973427   9.33772956 -3.2166429   7.28382045].       -110568.297016009
18      [ 0.61155877 -9.53496613  8.85943923  8.7384669 ].       -231647.5606548612
19      [ 5.39782382 -1.2844643   8.43553257 -3.8545582 ].       -63703.09002460646
20      [ 1.76289051  1.78627203  1.36617813 -1.10226957].       -59.14051345519054
21      [-9.44572799  0.38465493 -4.5694765  -7.65384804].       -64709.69624131256
22      [-4.77974029  4.4732243  -0.45397362 -2.86226686].       -5231.043647009441
23      [ 5.18121212 -1.99532779 -7.30143683  5.2012205 ].       -50509.29405586867
24      [-9.61971011  3.1176179   9.12836755 -1.67386312].       -82083.92958097468
25      [1.10334859 1.21591187 8.47774174 7.54136028].       -105273.963188660
26      [-6.93748111 -6.73987294 -6.40149091  7.86706383].       -110580.836561804
27      [-4.32346474  1.55695409  1.34266742 -0.49860569].       -212.0275085815722
28      [-10. -10.  10. -10.].           -365021.0      -59.14051345519054
29      [ 5.12071964 -9.99142491 -3.10371566 -1.00945639].       -78379.5784854561
30      [-10.          7.18758757  10.          10.        ].        -281735.00
31      [9.27706847 6.91645074 8.95699922 9.69899375].       -214148.0128752347
32      [-10.         10.         -7.41854521 -10.        ].        -290452.60
33      [-10.          6.87079663   3.49327571 -10.        ].        -177308.28
34      [ 8.55239177 -8.42492067  0.36601974  7.11796586].       -76653.76529178502
35      [-5.90876122  2.65012068  1.31773895  9.65084872].       -137687.0095101019
36      [ 10. -10. -10. -10.].           -380981.0      -59.14051345519054
```

```
37       [ 3.3949602  -1.74065319 -0.11285713 -9.65175421].       -139047.277062059
38       [-9.61711857  5.0757532  -5.65530706  5.18194171].       -32088.249316447391
39       [ 8.1258056  -4.10654926  8.48234431  7.03237438].       -99787.39434305123
40       [-10.         -10.         -10.          -0.30965293].           -221036.488
```

```
### 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     [-6.91674315  4.80099393 -4.7336997   0.67478787].       -10615.027220241049
init     [-9.70850075  8.37494016  8.01429708 -9.33157145].       -198779.9445700966
init     [ 9.13898673 -7.25581357 -4.32343294  2.12166369].       -25246.229651049434
init     [ 8.88450272  7.05471082 -9.95481533  0.42452054].       -126529.632126287906
init     [ 1.04075267 -0.29245173  5.36268308 -6.78566494].       -40114.0772599786
1        [ 9.00627049  5.34951301  6.50018506 -1.86719396].       -23513.617348450025
2        [ 8.88428387 -2.74934901 -7.2021893  -8.7943139 ].       -138985.0382288978
3        [7.07957877 4.20923579 8.31123577 5.90287485].           -70536.44119328947
4        [ 3.65158786 -4.292633    3.33049006  6.75429274].       -35227.82899489502
5        [ 3.07120909 -3.55916591 -5.97833992 -2.63162   ].       -19461.58116596912
6        [-10.         -2.87258748  10.          -4.76015699].           -129987.231
7        [-6.77288607  5.95533828  4.43070962  2.50634443].       -15729.19686434966
8        [ 9.11568286 -9.45162454  9.96288854  5.4127943 ].       -196776.2123688205
9        [-0.54343888  6.9130662  -3.08529747  0.79842711].       -18999.284222345348
10       [-2.82350692 -6.04303308 -5.89223718  4.65288339].       -38294.741478352604
11       [-8.01556949  4.33863986 -9.09107963 -5.10325364].       -96943.55441519922
12       [-7.70853825  9.04000734 -1.47997073  6.83741383].       -94812.1104929485
13       [ 5.27841301  8.33268337 -0.85821955 -5.81325175].       -54590.834140882194
14       [-5.69729876  6.22751761  8.3350766   9.36945888].       -178627.8986028522
15       [-7.09189339 -6.84298482 -8.67391943 -7.30482965].       -147874.186359747288
16       [ 4.71575825  4.71416309 -2.11686206  5.24056754].       -16241.087545771734
17       [-7.8093924   0.25400768 -8.20800109 -6.68611527].       -92580.90723825619
18       [ 10.  10. -10.  10.].           -356981.0       -10615.027220241049
19       [-6.1194662   5.36901759 -1.57323996 -7.39976686].       -57545.69035912643
20       [-6.70156564 -8.80716834  2.02538422  1.11522447].       -53308.827806509675
21       [ 10. -10.  10. -10.].           -348981.0       -10615.027220241049
22       [-10.         -0.41342984   1.74032343  -2.48889043].           -914.114694
23       [-6.34363966 -3.26687378  8.41196286 -0.3285002 ].       -64747.61027266566
24       [ 4.65005132  2.01886662 -9.59457072 10.        ].       -275230.6443043745
25       [ 0.51881787  7.98996703 -8.64243835 -7.78587054].       -159793.441604006455
26       [-6.7368404  -7.4664175   6.39994154  9.35043234].       -165503.025051163331
27       [1.25134656 9.64918569 6.20390452 1.16976574].           -82069.24464947048
28       [ -2.04380425 -10.          10.          -10.        ].           -358352.662
29       [ 10. -10. -10.  10.].           -380981.0       -914.1146944958429
30       [-8.25635404  8.72047473  9.41845964  5.43226615].       -146723.897449135988
31       [ -9.20935458 -10.         -10.           9.81994689].           -384554.461
32       [ 10.  10. -10. -10.].           -356981.0       -914.1146944958429
33       [ 10.  10.  10. -10.].           -324981.0       -914.1146944958429
34       [ 9.48871887 -9.90924884 -0.40737412 -6.97413465].       -108417.239630489931
35       [-10.  10. -10.  10.].           -373021.0       -914.1146944958429
36       [ 0.40718932 -9.84720785  1.53979956 -5.27681426].       -87273.69065909751
37       [ 10.          1.16456784   4.57476593 -10.        ].           -157918.919
38       [10.         10.          1.42232846 10.        ].       -230119.71557819826
```

```
39       [-3.65265375  1.74553188  0.17691954  8.81247621].       -96496.97808738812
40       [-10.          -3.0292551  -10.          2.99456833].       -128512.82
```

```
### 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     [ 5.55404821 -5.2491756   6.48557065  9.31498396].       -140521.0909730103
init     [ 9.45202228 -0.93101505  2.18084926  5.51053029].       -14230.66474647201
init     [ 2.8322669   4.44036459 -9.29926952 -4.03101058].       -94860.11143837457
init     [-8.82975016  7.14121885 -2.54291944  3.59695903].       -27991.65897596937
init     [-4.87440101 -3.0483757  -9.8117446  -2.83332435].       -118576.8785832125
1        [ 9.1114829  -9.99975933 -5.06042598  4.24465356].       -90932.40249777345
2        [-9.72135464 -7.69244795 -6.22844507  7.60723681].       -114241.3048574365
3        [-8.34192968  0.21366179 -9.02437947 -5.9293723 ].       -104788.2931595007
4        [ 0.57275335 -7.71738011 -5.09743991 -0.17013756].       -38898.41073512287
5        [3.60696134 2.18226175 0.08284273 4.5407231 ].       -6865.378070859648
6        [-3.20440472  8.67469235  4.12830206 -1.70990841].       -49217.10845788232
7        [ 10.  10. -10. -10.].           -356981.0       -6865.378070859648
8        [ 0.65604441 -6.68523471  7.23179469 -8.11619975].       -114911.9051730723
9        [-5.8544789  -4.61653174  0.32592983  0.79886988].       -4821.022362497320
10       [-0.67697589 -0.58702519 -7.04742331  4.68276363].       -40323.04821787296
11       [-5.02166508 -3.62694175 -6.53971562 -2.35341132].       -27090.90316698181
12       [ -1.57683176 -10.          -4.63399817 -10.        ].       -257183.80
13       [ 7.2681675   7.28099793  2.98739184 -5.83963596].       -36892.64728027531
14       [ 4.93721914 -8.17097589  3.20600942  1.12318112].       -35565.30512561760
15       [10.          9.80393383 10.          10.        ].       -319423.2651946959
16       [-9.47000929  7.85217408 -2.22992097 -9.61629645].       -175525.3366554377
17       [ 7.08271592 -1.41666864  9.88865573 -6.13938682].       -133626.9225457206
18       [-10. -10. -10. -10.].           -397021.0       -4821.0223624973205
19       [-10.  10.  10.  10.].           -341021.0       -4821.0223624973205
20       [-5.02171266  8.52779293 -9.72542944 -6.23642938].       -173843.3079099923
21       [ 8.93458891 -9.49016566  4.77058082 -8.20222919].       -135127.6544321828
22       [ 0.71134297 -9.39245611 -7.81811845  9.90861339].       -280501.5702086523
23       [-3.64167041  0.3273323   8.92202364  2.23313088].       -75782.05741843494
24       [ 10.  10.  10. -10.].           -324981.0       -4821.0223624973205
25       [-10.  10.  10. -10.].           -341021.0       -4821.0223624973205
26       [6.59092781 5.95539015 9.06337472 2.02599095].       -83523.16203461745
27       [ 10.         -10.          -7.14775678 -10.        ].       -281675.40
28       [-6.18889295  0.52207805  6.45694382 -9.35075775].       -134197.1226513083
29       [-10.  10. -10.  10.].           -373021.0       -4821.0223624973205
30       [ -8.65802875 -10.          10.          -10.        ].       -363869.62
31       [ 8.88831075 -1.71588696 -4.56090352 -7.81783805].       -70020.99745189029
32       [ 0.39718666 10.          10.          10.        ].       -332382.9295732697
33       [10.          10.          -2.17509088 10.        ].       -235780.9376924654
34       [-7.09330624 -4.73947338  5.56631532  8.31026191].       -89109.78683642866
35       [ 0.24211269  0.27141474  2.710911   -4.33124387].       -5471.486939449708
36       [ 9.59338719  2.55754291 -9.41153485  5.1553867 ].       -107208.7295595644
37       [-1.39950129  9.80866879  1.05065622  9.78159231].       -220180.2613611463
38       [ 0.57896953  8.0496214  -3.56336142  0.80079954].       -34285.84649325997
39       [-1.07167921  7.40842816 -9.34890912  8.21125902].       -191812.6955804864
40       [-8.60763284 -5.54465216 -0.95193328 -8.27109768].       -86007.76396937127
```

```
### 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     [ 0.27886688  5.46330104  7.40855371 -9.83906103]. 	 -178393.1355710015
init     [-3.80528149  9.15207479  0.26233425 -3.6343115 ]. 	 -61708.37227853236
init     [ 0.78399875 -5.57490115  6.12962716 -3.15490749]. 	 -27840.0322802925
init     [ 0.77777698 -9.88252429  3.46304956 -5.79951476]. 	 -95425.8534148436
init     [ 8.65115186 -2.51510501  5.04837844  5.26278004]. 	 -18811.8457713201
1        [-6.20242914 -2.96747056  4.29623333 -2.85129273]. 	 -6530.500693022571
2        [ 3.07044426 -6.31045846  1.63194229  8.38251886]. 	 -89311.7774639001
3        [2.67305604 5.51907321 6.71543751 9.91762785]. 	 -172702.858532045
4        [0.19437843 2.05845053 1.65792191 5.18484457]. 	 -11033.9318563383
5        [-1.31829204 -2.32031879 -2.5251031  -6.82350009]. 	 -37571.24827805165
6        [ 0.24276126 -4.12052041  2.21586778  7.93881754]. 	 -64196.1703940990
7        [ 4.79563633  4.13838411 -7.01002178  7.65374311]. 	 -90010.55097432152
8        [-8.61484726  8.56315525  8.59846951 -0.87156311]. 	 -106727.497541840
9        [-10.  10.  10.  10.]. 	 -341021.0 	 -6530.500693022571
10       [ 10.  10.  10. -10.]. 	 -324981.0 	 -6530.500693022571
11       [-10.  10.  10. -10.]. 	 -341021.0 	 -6530.500693022571
12       [ 6.82279714 -0.09161239 -2.028231    8.81142257]. 	 -99320.62563521019
13       [-8.96893842 -9.80111106 -6.18093935 -0.65507665]. 	 -103470.007468530
14       [-7.86622501  8.36454916 -9.66739114 -0.29939709]. 	 -139797.522002299
15       [ 8.17029418 -7.04437314 -7.57697513 -9.26345515]. 	 -189645.520659918
16       [ 7.58127047 -7.28224206 10.         -9.59132251]. 	 -269423.8582247585
17       [ 7.82123082  6.77924015  4.68125101 -3.26858149]. 	 -19422.0262598941
18       [-10. -10.  10.  10.]. 	 -365021.0 	 -6530.500693022571
19       [ 4.70632974  6.57054462 -5.97157681 -3.82679991]. 	 -30893.633433224
20       [ 10.  10. -10. -10.]. 	 -356981.0 	 -6530.500693022571
21       [10. 10. 10. 10.]. 	 -324981.0 	 -6530.500693022571
22       [-6.57180368 -3.23798554 -6.61948905  6.73352215]. 	 -64217.4423529888
23       [ 5.22498155  9.63662115  0.71281827 -6.55975151]. 	 -94541.1682905092
24       [ 6.53295252 -8.00803896 -9.95913138  5.7696372 ]. 	 -180869.609203696
25       [-0.36579001  0.76849399 -8.32825279  0.42263085]. 	 -57399.91613724862
26       [-8.5289272   0.57296063 -7.17479097 -6.36514597]. 	 -62826.40535662578
27       [-8.01189683 -0.39981096  2.62732101  7.36608046]. 	 -45677.01194384305
28       [ 4.35966764 -9.87212395  9.12501977  5.28483058]. 	 -174727.411606503
29       [1.349273   6.70863794 1.93835752 6.58155122]. 	 -44417.2615828796
30       [-1.6443209   0.20666129 10.          1.64778014]. 	 -119848.637485724
31       [ 8.48637076 -6.74776524 -0.95263072 -1.01656878]. 	 -13950.8290677969
32       [ 9.96814488  0.36226106 -0.55139331 -6.60568064]. 	 -31119.42122046227
33       [-10.  10. -10. -10.]. 	 -373021.0 	 -6530.500693022571
34       [ -2.49234001  10.         -10.          10.        ]. 	 -366718.49
35       [-10. -10. -10.  10.]. 	 -397021.0 	 -6530.500693022571
36       [-10. -10.  10. -10.]. 	 -365021.0 	 -6530.500693022571
37       [-9.95223123  8.68923607  1.06013654  3.90062293]. 	 -55509.59506149538
38       [-10.         -10.          -6.3088306 -10.        ]. 	 -282660.258979979
39       [ 10.  10. -10.  10.]. 	 -356981.0 	 -6530.500693022571
40       [2.55484322 9.95245564 9.5986323  1.83943949]. 	 -167669.640615908
```

```
### ESTIMATED GP EI GRADIENTS
```

```
### 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     [ 6.97635395 -6.4220815  -8.91273571 -2.76923108].       -95759.56429474254
init     [-4.49198143  0.6000045  -3.88162169 -3.91051282].       -7452.834860002497
init     [-7.76517448 -5.00201972  8.35259796 -4.71706293].       -74656.79367914848
init     [ 4.35547375  7.31430068  6.14158964 -5.78898835].       -49831.89480053248
init     [-6.65513937 -9.06587217 -9.21155376 -5.9953838 ].       -180751.6072602438
1        [ 3.91355621 -9.41682078 -0.01015161 -7.34731365].       -107072.458255194
2        [4.92526955 5.96067446 4.10838652 1.54460964].       -11082.60249594799
3        [ 7.60284646 -9.83670929  6.07418637  6.51677002].       -115033.369674890
4        [-3.39221728 -3.69203211  3.60318038  4.66933918].       -10938.171069717
5        [ 9.63144016  3.76660236 -9.40395266  2.32534909].       -92305.21346503733
6        [-0.46099561  9.58086516 -0.66748888 -7.87674906].       -130229.313894340
7        [3.72122725 0.49817372 5.20397899 9.4368106 ].       -128249.405728734
8        [-7.21002719  4.48364211 -4.95008418 -7.56274535].       -67479.32735563519
9        [-0.8289478  -8.11611777 -5.43376832  1.07172118].       -48927.45485006536
10       [-9.35592252  5.23319628 -9.05013592  1.59286285].       -84569.94721781593
11       [-8.83646821 -9.52000508 -3.03305299  9.46202552].       -207340.535118876
12       [ 1.57690606 -4.26679043  6.91876346 -3.82537765].       -34428.16075055872
13       [ 3.63484615 -6.9061913   1.72145752  0.75302069].       -17340.749513044
14       [-5.29773051  4.13262973  2.40625612 -8.86801578].       -99267.1777902554
15       [-7.20628386  8.66608008  6.24591119  2.13266135].       -64085.5805962747
16       [-6.14779635 -0.43746793  3.56596824 -2.19771089].       -2292.657920210773
17       [ 1.30223912 -8.21616194  4.9325588   9.93275327].       -193511.188084685
18       [-10.  10.  10. -10.].            -341021.0       -2292.657920210773
19       [ 1.14120333  4.18919902 -5.55151656 -7.17257482].       -59247.26899808023
20       [  8.97273996 -10.          10.          -10.        ].          -349746.39
21       [ 2.37201846  9.77107763 -3.75463112 -4.47165852].       -79796.77105889232
22       [-10.           7.06202255  10.          10.        ].          -280283.82
23       [-9.21357548 -6.74167545  1.0076465  -2.03849531].       -20594.139037128658
24       [ 10. -10. -10.  10.].            -380981.0       -2292.657920210773
25       [-1.31632644  3.88365909 -4.41034801  5.96888866].       -28567.653718024530
26       [-9.3265994   0.78368818  6.22686599  6.08844431].       -36455.45765738404
27       [ 5.60066467 -3.87589042 -5.27284941 -7.26059882].       -60853.24367998289
28       [10. 10. 10. 10.].       -324981.0       -2292.657920210773
29       [  9.53515943  -0.72551563   3.94693025 -10.        ].          -157013.36
30       [8.36219672 9.80306436 9.96457014 1.990924  ].       -174579.7954762038
31       [ 6.2784228  -7.31871577  0.31170191  8.68142997].       -111046.1667329579
32       [ 3.97139572 -0.78817608 -6.48761029  2.35482403].       -22917.5104954821
33       [ 10.  10. -10. -10.].            -356981.0       -2292.657920210773
34       [ 3.73345413  9.62274081 -7.405405    9.4741026 ].       -235658.229726594
35       [-5.15268108  8.42996177 -8.74823726 -2.99493539].       -109026.664587861
36       [-9.47155828 -7.66943862  5.8027646   6.8674039 ].       -80648.71564628197
37       [-1.93342917 -4.84402422 -8.9163958   9.21804539].       -213287.653506188
38       [-9.98712406  7.81056755 -6.70440634  6.67862108].       -91978.99245763329
39       [ 9.44215564 -2.57690299 10.          0.65841078].       -123546.592197031
40       [ 9.4625835   7.46115737 -0.75066995  9.52610204].       -153800.4453569196
```

```
### 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     [-5.53417842  0.46326683  1.01402913 -9.087961  ].       -107926.0181001286
init     [-2.78542329 -5.53838117  3.77452324 -6.7253715 ].       -41773.31935362981
init     [-8.59350266  8.8202172   1.2736276  -8.44015321].       -133847.7142986104
init     [ 4.45281022 -6.83095653 -4.99437387 -4.13025488].       -31578.1535212759
init     [ 3.93221428 -0.71471824 -5.69875713 -1.06347476].       -13215.2544831498
1        [-0.64007705  3.2211867   3.79933264 -4.73341031].       -9619.11510308440
2        [ 5.98502725 -7.94806028 -2.90882686  9.11344119].       -145121.673621640
3        [ 0.47384394 -7.09622115 -6.44235327  3.14296454].       -47197.66759876199
4        [ 5.52616367  4.65190076  3.38866017 -9.07435877].       -107947.342272070
5        [ 4.28544997 -2.78264932  9.08402216  2.69441349].       -84872.13346649229
6        [1.87133514 5.63089801 3.59434817 4.26424392].           -13097.1592284471
7        [7.45907335 5.14332641 9.11613095 8.2671896 ].           -147101.318418398
8        [ 1.24349389  9.55943585 -9.57929245  8.27200487].       -242462.044581461
9        [-6.58978683 -7.60537449 -1.72785531 -2.22440249].       -31052.4910471898
10       [-3.41259436 -0.53328708 -1.59942228  9.36100657].       -125259.545280376
11       [ 8.89995329 -8.45639035  4.20210818  2.73243867].       -42247.8362385096
12       [-3.05457595  8.6513061  -8.42526724 -0.14730159].       -100287.069581637
13       [-3.45060725 -2.87892412 -7.28800971 -6.14683127].       -63996.2111066633
14       [2.62583757 0.46449641 0.44128596 2.38948669].           -494.3569330727978
15       [ 10.  10.  10. -10.].           -324981.0       -494.3569330727978
16       [-10.          0.35454522 -10.           1.74117232].        -120938.23
17       [ 8.37017332  6.7425363  -6.78563457 -5.15425157].       -49897.64263964637
18       [-4.12908744 -6.44608013 -9.57829988  9.62751237].       -275501.78703328
19       [ 4.39317174 -9.17548236 -7.80685399  8.14414445].       -184248.9712183503
20       [-8.09573386 -0.93369103  8.83705044  0.93227014].       -74538.58726244236
21       [ 10.          0.10546533  10.         -10.        ].        -264553.58
22       [-9.22904309  8.72149497  4.93469127  1.94582391].       -57000.67547234631
23       [ 7.11441315  6.24817716 -3.01719087  7.26579957].       -57715.08912167868
24       [ 9.48449776  6.83252536  5.34568415 -0.39785071].       -21841.9911189496
25       [-10. -10. -10. -10.].           -397021.0       -494.3569330727978
26       [-10.  10. -10.  10.].           -373021.0       -494.3569330727978
27       [ 10.         -10.          1.74464626 -10.        ].        -230278.17
28       [ 10. -10.  10.  10.].           -348981.0       -494.3569330727978
29       [-10. -10.  10.  10.].           -365021.0       -494.3569330727978
30       [-2.89541268  8.35462631  9.63290282 -9.0193402 ].       -228571.765127278
31       [-0.79327379 -7.57161458  9.67752913 -2.41264056].       -140612.4870975568
32       [ 10. -10. -10. -10.].           -380981.0       -494.3569330727978
33       [ 5.71022868 -8.15408013  5.60144666 -7.0087603 ].       -81831.21082517967
34       [-10.         10.        -10.         -8.5477213].       -294123.6588286265
35       [-6.26576795 -3.18520554  7.2262102  10.        ].       -184857.377258646
36       [-2.92772479 -8.45467826  1.20067393  9.69578106].       -182569.751208846
37       [ 9.01567238 -0.68638457 -2.64806786 -7.18554316].       -45713.36037414307
38       [ 5.45605189 -3.49800546 -1.82105288  4.02505857].       -5735.877064089329
39       [ 10.        -10.        -10.          0.37152958].        -205003.39
40       [-8.93738848  4.07242298  5.7081894   7.4415521 ].       -58980.26003404534
```

```
### 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     [-4.10669995  0.61173511 -6.16958426 -8.64199284].       -113948.993932591
init     [5.7397092  3.12667044 2.75041792 1.51205788].           -849.5238302802878
init     [-9.21874168 -2.84372791  8.91366374 -8.79910639].       -165071.6247490734
init     [ 7.28084207  7.54581052 -8.97612669  3.04837231].       -96579.16352863169
init     [ 1.03502737  1.95026506 -0.32942751 -4.34023678].       -5872.837304694351
1        [-6.82082511  3.53524771 -7.6305547  -1.10007868].       -40739.58096344288
2        [ 6.36132887 -5.79483735 -1.80465394 -5.23795116].       -20725.35045584762
3        [-9.93358214e+00  6.76426027e-04 -8.86041968e-01 -1.86772363e+00].       -5
4        [-3.57261465 -5.71224609 -1.88427713  4.42421021].       -16723.5217284345
5        [ 7.32299713 -3.1980267   7.52939765  8.3238999 ].       -109850.133841723
6        [-9.15358294  6.24405427 -7.34906493 -3.27748646].       -49685.22640168171
7        [-8.32522503  1.10793827 -8.79126558  8.08570378].       -148868.7982529325
8        [ 9.48510639 -1.89215997  8.72310011 -8.54004817].       -146533.506053131
9        [ 10.         -10.          4.5407681    0.06947864].           -80238.860
10       [10.         10.          3.79244055 10.        ].       -227327.024934022
11       [-3.55419883 -1.3121061   6.26693875  4.48480965].       -23865.23094014127
12       [ 2.54749487 -9.29639909  6.99658479 -6.71299771].       -120125.416478096
13       [ 3.37722984  0.46032172 -5.15280505  5.104616  ].       -21454.83186816024
14       [-9.67252227 -8.84826579  4.65918096  0.87645251].       -63629.75766545245
15       [ 4.34389407  6.23064858  6.70660755 -5.49281777].       -43293.8482378960
16       [0.31334945 6.26826312 1.77763055 9.3516953 ].           -132149.586903221
17       [-10.  10.  10.  10.].           -341021.0      -571.5810915631586
18       [ 5.73058679  6.05017845 -8.13181    -6.04335437].       -83268.56175905156
19       [ 10. -10. -10.  10.].           -380981.0      -571.5810915631586
20       [-4.9003479  -9.03322285  6.16587778  1.19907682].       -78302.75020896044
21       [-3.9804138   5.24617004  2.12715187  4.24226727].       -11623.8685201275
22       [-6.17626749 -8.46700681  8.99586277 -2.9797324 ].       -132117.736762757
23       [-1.57575393  9.85203222  3.11647346 -4.88636175].       -84848.88023404627
24       [-10.          -6.37390263  10.          10.        ].           -288945.88
25       [ -0.78316717 -10.         -10.          10.        ].           -389330.94
26       [-10.  10.  10. -10.].           -341021.0      -571.5810915631586
27       [ 10. -10. -10. -10.].           -380981.0      -571.5810915631586
28       [-10.  10. -10.  10.].           -373021.0      -571.5810915631586
29       [-0.17344573 -4.6667978  -4.02317816 -1.15895835].       -8121.929303327734
30       [-3.85451606  2.65749216  2.16933315 -9.66637665].       -137275.615645221
31       [-10. -10. -10. -10.].           -397021.0      -571.5810915631586
32       [-3.51762081 -5.5603206   0.04416167 -4.59494337].       -15772.61607592267
33       [  0.33244897 -10.          10.          10.        ].           -356434.70
34       [-10.         -10.          -6.53648227   4.37662049].           -123698.48
35       [-6.84485236  3.70314401  9.9884421  -2.66542633].       -117535.6670283434
36       [ 2.56432668  6.10803492 -0.24395397  0.04946162].       -10493.2389260836
37       [ 3.96142624 -7.37930587 -0.01775028  7.07304132].       -62258.7407587610
38       [10.         10.         10.          0.92594656].       -180855.581201202
39       [ 5.09178292 -3.65066379  9.79297187 -1.34047755].       -115708.657622996
40       [-0.7429384  10.         10.          4.62494734].       -193196.678375200
```

```
### 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)
```

8

```
Evaluation       Proposed point          Current eval.          Best eval.
init     [ 3.00748483  0.10906747  7.57202942 -6.36319549].       -60950.54433734527
init     [7.04466137 5.00272572 3.32203335 9.75790897].        -144653.2121556773
init     [-4.86063155 -9.43388149  2.71438231  6.94624775].       -103843.635466677
init     [ 4.7234925  -9.58385776 -7.76793739 -4.04552516].       -121548.022510702
init     [ 3.73940383  7.23252112 -6.02731282  3.14378061].       -35863.1928389845
1        [-6.1664519   4.28513481  2.79740714  0.66407663].       -4132.892348949624
2        [ 5.61099687  3.24280582 -9.3946128  -2.53775904].       -92553.92031901842
3        [ 3.30844576  7.22382341 -9.24562554  8.22016932].       -184254.3962213914
4        [ 8.03656335 -7.89614249  7.70734605  1.23688198].       -75516.33414231456
5        [ 9.562822   -7.89156436  1.83282657  5.62935728].       -42311.3181835149
6        [-5.31480829 -4.60774758 -2.97712464 -7.93522102].       -72575.53761772846
7        [ 4.54666061  8.69770438  0.01712558 -6.54510361].       -72662.9494514385
8        [ 2.31421647 -2.51006802  8.67058127  9.73721889].       -201299.963647480
9        [ 8.39884901 -4.60470488 -7.7957923   9.97387584].       -221097.759264194
10       [ 8.47467859 -1.0716028   7.04016464 -4.60023784].       -35231.89169888245
11       [-4.41945067  2.73837136  0.20384538  5.36041706].       -13921.3451769947
12       [-3.93005265 -7.62357754 -2.48223981 -1.02660088].       -30182.8401449935
13       [-8.79601456 -6.39611748  6.71849908 -7.75307177].       -96086.06277058301
14       [-5.37332728  7.94915331  0.37919817 -7.51567844].       -85641.09865300173
15       [-2.80981525 -1.87682768 -5.05682874  3.23441465].       -11341.8454624224
16       [ 6.58567988 -3.25808927 -0.548504   -0.50180546].       -509.251179041512
17       [-9.953162   -4.99542568 -2.13845079 -6.90241283].       -45853.3237027980
18       [-10.          10.          6.37642542 -10.        ].          -253539.95
19       [-5.24271212  1.15655578  6.68333114 -8.837557  ].       -112916.766169817
20       [0.98616771 7.97275138 2.77260242 6.55930039].        -59728.3888002034
21       [ 2.30528361 -9.9326     -9.84502306  5.1831096 ].       -216805.312547680
22       [-10.  10.  10.  10.].          -341021.0      -509.25117904151205
23       [-10.          -10.          -10.          6.93681481].          -265767.73
24       [-6.69867140e+00 -9.34293864e+00  9.22919429e+00 -1.76365303e-04].        -1
25       [ 9.59876928 -6.95266335  1.85259451 -7.10765861].       -55163.28799388627
26       [-9.72541821  2.45726486 -7.47816001 -4.84939727].       -48849.1899642669
27       [1.66207421 1.59849313 1.05654863 0.1874204 ].        -29.31705439832732
28       [-3.83822248 -8.47380926  1.10967586  1.83939291].       -43992.72279207087
29       [-9.69577207  7.7269764  -6.0387706   7.13214099].       -92668.5059306543
30       [ 10. -10.  10.  10.].          -348981.0      -29.317054398327326
31       [ 2.22467047 -6.27762742  3.37822164  3.65453913].       -16452.9145886564
32       [ 8.43667194  0.37351363 -4.30241107 -9.74147844].       -154909.4298591131
33       [9.7167587  8.21173065 5.04176705 0.80664879].        -36907.33794073301
34       [ 10.          6.2609192  10.         -10.        ].       -266442.126852759
35       [-9.41925546 -6.25583456  7.52919496  9.6768977 ].       -187672.305289533
36       [-10. -10. -10. -10.].          -397021.0      -29.317054398327326
37       [ 0.18809976  6.78655743  9.41844766 -0.24741502].       -104588.823822441
38       [ 0.82468521  6.99127321 -9.44308176 -9.79827769].       -269213.5509537866
39       [ -0.28042347 -10.          10.         -10.        ].          -356926.13
40       [ 10.  10. -10. -10.].          -356981.0      -29.317054398327326
```

```
### 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     [-8.04932797  5.22499433 -5.06124054 -7.23736625].       -62523.95884648901
init     [-3.37106873 -8.3400087   3.43954163  6.13187596].       -64291.98059909556
init     [ 9.65483829  2.7132147  -5.68153488  0.98054864].       -11832.54772826069
init     [ 9.11199160e-01 -5.31847854e+00 -7.72548314e+00 -6.81465902e-03].       -5
init     [-6.95783155  0.65372161 -2.25986463  3.76654767].       -4213.270400555922
1        [-6.5477408   0.83865275 -8.96461914  6.51115227].       -112037.9546475548
2        [-4.76936448 -2.97550534  8.89602795 -8.21242057].       -142547.9773890224
3        [-2.32173541  7.26125369  5.5769216  -4.96985133].       -39978.81776250941
4        [-9.13977787 -5.66464753  4.94279629  8.45310839].       -95912.86613447932
5        [-1.55246852 -8.63332631 -1.23512782 -5.59496547].       -62092.94247875736
6        [-4.83369566 -2.4386249   4.54159233  6.11885823].       -26110.73895282561
7        [-1.74139105 -2.25943387 -4.66754333 -7.46934292].       -60650.71086015194
8        [ 3.38990203 -3.49236324 -8.09259704  9.98332142].       -227238.8418316005
9        [  6.67023876   3.57477814  -8.82832409 -10.        ].       -244772.10
10       [ 10.  10.  10. -10.].           -324981.0      -4213.270400555922
11       [ 1.66645895 -8.36624855 -7.84433287  7.99584163].       -163760.6843129029
12       [ 9.63400574  7.61710851  2.75684065 -5.16449309].       -33128.05839023063
13       [ 5.21228094 -8.67935387 -9.05580191 -9.13323666].       -255545.4270122839
14       [-10.          5.61972235   0.6755425   10.        ].       -169813.94
15       [ 2.34954077 -3.13748972  3.15811446 -8.40909356].       -78673.48727660593
16       [-10.  10. -10.  10.].           -373021.0      -4213.270400555922
17       [2.41764449 1.87228964 5.94389184 3.41628568].           -15450.10891765153
18       [ 9.11708637 -4.55121628  1.89198796  7.70152498].       -57073.3981043833
19       [-8.09435026 -4.81876152  2.44801843 -2.8413461 ].       -7628.651360237111
20       [ 9.51568311 -8.82578504 -2.38869497 -2.78357922].       -45372.47140173492
21       [-10.         -10.          10.          -6.19493038].       -238445.55
22       [ 9.47335062 -2.82596526 -9.98254377 -9.14656194].       -248467.4666931006
23       [ 2.51530768 -4.62186548  9.97645257  4.84711249].       -133173.3165602473
24       [-10.          10.          10.           7.86195831].       -248259.72
25       [-3.57674008  9.13103764 -6.48638635 -0.12139103].       -75096.2457779599
26       [-9.42856911  8.98664152  3.80558549 -5.85683224].       -76548.44636499413
27       [ 7.78673479  3.29014441 -2.52376044  9.91619556].       -159397.9328236306
28       [0.49969388 9.76369343 7.17560466 9.63248979].           -225686.5916492223
29       [-10. -10. -10.  10.].           -397021.0      -4213.270400555922
30       [-10. -10. -10. -10.].           -397021.0      -4213.270400555922
31       [ 10.         -10.          10.          -7.20817038].       -239861.43
32       [-0.48600753 -1.21446282 -1.18333432  9.57555526].       -136332.3932864208
33       [-2.27169776  9.64730011 -3.13354755  9.00021215].       -180395.1423090053
34       [ 7.2052829   0.12871334 -1.41519373 -3.82625005].       -3955.347207256568
35       [-9.87217663 -3.42357815  6.86830815 -0.41111743].       -31185.86267222706
36       [ 10.  10. -10.  10.].           -356981.0      -3955.3472072565687
37       [9.2656487  2.76184778 5.54584805 1.48958409].           -10500.03663922194
38       [ 10. -10.  10.  10.].           -348981.0      -3955.3472072565687
39       [ 10. -10. -10.  10.].           -380981.0      -3955.3472072565687
40       [ 3.4885124   9.13405872  0.1327617  -3.54242654].       -56129.35503796725
```

```
### 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     [1.76261602 7.95427456 7.83061459 6.31674955].           -91316.16764460134
init     [-9.28220829  3.83515164 -2.42638116  0.37021891].       -3319.224464998000
```

```
init     [ 3.15902931 -6.12299564 -4.55367196  4.37211867].       -24439.65199379687
init     [ 5.66007219  7.0065528   5.50489788 -9.26671387].       -136303.9099361228
init     [-7.6661253   5.02561399 -5.21563568 -4.90387972].       -25528.5956441878З
1        [-0.38031735 -3.41587185  0.21282112 -4.72742343].       -9079.144305704322
2        [ 2.2515589  -8.62950692  7.2874785   9.15451767].       -185415.9807778578
3        [-2.00279263 -5.09620423  0.51102349  6.68881567].       -37586.2072143727
4        [ 0.69284317  3.15192848 -8.93401192  2.30035677].       -75718.37358614859
5        [-3.75442852  7.02008144  4.58355261 -6.10855163].       -44260.2464515866
6        [ 4.26761042  1.24919415 -4.83066711 -6.15878884].       -32247.4565093441
7        [8.1991784  6.51057117 8.40600493 7.35687831].           -106174.9468725487
8        [ 7.25959653  3.0895583  -0.80389347  9.92972863].       -157149.4398753884
9        [ 5.87064721 -4.24128552  3.56822046 -0.20436374].       -4532.363410064421
10       [-3.49997926  0.57776952  2.16068958 -2.13900545].       -480.6088906930014
11       [ 4.5495202  -8.7912022  -5.43120396  4.47127488].       -67060.84679719032
12       [-2.50232563 -8.25576548 -9.16975052 -0.52445896].       -132307.5732365573
13       [ -7.19696294 -10.          -10.           10.        ].          -394628.35
14       [-5.18813627  2.95972213 -5.66334369  8.80164762].       -115472.3931840499
15       [ 4.75754346  6.89202164 -0.21987262  0.41733899].       -16441.31494689493
16       [-10.  10.  10.  10.].           -341021.0      -480.60889069300146
17       [-9.50037711 -9.91987748  2.91255554  7.95992373].       -148718.9421409524
18       [-7.44381676  2.82917673 10.          0.65771705].       -118134.0988112058
19       [-6.47540494 -2.09537926  8.57287319 -8.06099953].       -126138.0318844791
20       [ 10.  10. -10. -10.].           -356981.0      -480.60889069300146
21       [ 10. -10. -10. -10.].           -380981.0      -480.60889069300146
22       [ 10. -10.  10. -10.].           -348981.0      -480.60889069300146
23       [ 7.85939026 -1.78725481  3.46352795 -8.99642842].       -102417.3729816852
24       [-10. -10. -10. -10.].           -397021.0      -480.60889069300146
25       [-10.  10. -10.  10.].           -373021.0      -480.60889069300146
26       [-10.          10.          10.          -9.38597884].          -307101.71
27       [-9.097715   -9.95838996 -3.21884215 -2.81167224].       -90434.37458027249
28       [ 10. -10. -10.  10.].           -380981.0      -480.60889069300146
29       [-6.65831638  3.00753142  3.58232452  9.98884726].       -156443.6970564247
30       [ 10.  10. -10.  10.].           -356981.0      -480.60889069300146
31       [-0.23811104  1.99927023  6.38129647  0.93804175].       -19155.89405781995
32       [ 9.90314811 -6.95624333  1.94751497  7.86459784].       -75106.73762381607
33       [-10.         -10.          10.          -2.58536706].          -220666.38
34       [-2.27573402  9.29093008 -8.81850473 -9.77130898].       -285004.0317102952
35       [ -1.55817331 -10.          10.          -10.        ].          -357957.93
36       [-10.          10.          -0.70989091  10.        ].          -249701.41
37       [10.          2.54991108 -8.1094797   2.28446869].       -51380.54267496981
38       [-3.04267003 -6.68627557  7.31523894 -0.25861506].       -56109.64118801696
39       [-9.85682293 -3.71243416 -8.67655993  1.78249843].       -75227.2359084002
40       [ 7.19496551 -6.80891187 -4.43319748 -4.77076971].       -31029.6874903111
```

```
end_approx = time.time()
end_approx

time_approx = end_approx - start_approx
time_approx

start_exact = time.time()
start_exact
```

```
1623338621.6591516
```

```
### 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     [-1.65955991  4.40648987 -9.9977125  -3.95334855].       -124757.8119225052
init     [-7.06488218 -8.1532281  -6.27479577 -3.08878546].       -64499.71723544843
init     [-2.06465052  0.77633468 -1.61610971  3.70439001].       -3468.30836846115
init     [-5.91095501  7.56234873 -9.45224814  3.4093502 ].       -121117.80061324051
init     [-1.65390395  1.17379657 -7.19226123 -6.03797022].       -57099.80140311415
1        [-3.68968738  3.73001855  6.69251344 -9.63423445].       -152203.3072503334
2        [-6.24736654  2.44991805  8.11618992  9.79910357].       -186189.5031681948
3        [ 5.02242081  1.58721081  8.49408363 -8.70520033].       -142988.6701515681
4        [ 4.834096    2.3003145   3.44822198 -9.83233841].       -145715.3611470581
5        [ 5.97945681  6.21258207  7.19997183 -0.82898378].       -38802.31394092165
6        [ 7.62336637 -7.47843039  2.66921668 -4.44574535].       -28622.81763188250
7        [ 8.34520318 -6.98520682 -5.12472045  0.93369144].       -26795.14592457248
8        [-3.48919405 -8.28301398  0.72048033 -3.27749114].       -41601.54892194371
9        [ 6.88781365  2.44433273 -5.87040082  3.82976847].       -18301.70278094595
10       [-2.33870576  9.62033584 -7.70378864 -6.00184274].       -131410.9354322270
11       [-3.91489669  0.50941142 -6.18065382 -5.71796552].       -37831.64052139965
12       [-5.92284857 -3.00788369  3.80958755  8.45413048].       -81714.36275571153
13       [ 9.44061342 -4.90799622  5.02017652 -4.37529581].       -16677.17535187311
14       [ 0.80119154 -9.16723906 -5.2983827  -9.59120497].       -212068.3714515436
15       [ 5.95349222 -6.56308056 -1.24122127 -4.6635801 ].       -21172.28908882003
16       [ 3.51356171 -0.42084275  9.12268125 -4.78166983].       -88920.5498486188
17       [-4.17792502 -0.4653672   2.61268639  4.2979804 ].       -5382.162305521617
18       [7.24062114 7.93125641 9.15218487 2.550167  ].          -104626.0726496303
19       [-0.29068499 -9.04011931 -4.02183401  4.43267673].       -66267.732177043
20       [ 9.25882725 -0.72458929 -7.51576865  3.34962275].       -42574.16034242221
21       [ 3.67388287 -2.87543511  1.9187423   6.41420845].       -26486.90502343201
22       [ 0.40329128  9.51082022  6.60849111 -5.6804143 ].       -96762.42587676083
23       [ 8.93914697  2.22501625 -5.96404096 -4.66426372].       -24103.53345906661
24       [ 3.93143455  7.07805892 -0.25860009 -8.16404801].       -90042.89252068444
25       [ 3.89328185 -4.07537543 -8.8093261  -7.83912687].       -147235.600848856
26       [-0.77290578 -3.36889522  7.86252834 -5.82002822].       -63841.64666658512
27       [-1.11943725  7.1330845   6.26202574 -5.04594866].       -44400.0764607695
28       [ 9.79638625 -7.33868908  5.30707856 -1.44242443].       -31419.01284011945
29       [-3.88904088  2.54869972 -2.56765033 -8.54195486].       -89139.45022593862
30       [ 3.22849664 -0.94636522  1.73151996  2.78573568].       -914.2605580537679
31       [ 7.60039592 -2.79296515  0.85767676  6.35227795].       -25727.16401188736
32       [-9.71362688  8.90969444  3.4563994  -2.59171866].       -57956.83084541687
33       [-0.34017093  4.07119008 -4.52754702  8.38197452].       -90484.85026690093
34       [ 2.77285071 -3.44005595  3.15412156 -9.81294563].       -146050.6849055735
35       [2.42697826 4.5464692  7.89956641 0.77869529].          -46596.73668690369
36       [ 4.28299283 -2.88572558 -7.91460873 -9.74866917].       -206394.3811396367
37       [-5.29793953  6.45935575 -8.17585408  9.52412026].       -208136.0323567456
38       [-8.48055013  1.62480641  2.59877833 -3.00115072].       -1842.659616352281
39       [ 8.2842522  -6.43177972 -7.3485378   3.19018699].       -53494.19368648512
40       [-1.85555351 -8.45085429  8.40440563 -7.38203213].       -149594.3623172437
```

```
### EXACT GP EI GRADIENTS

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

```

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

```

```
Evaluation 	 Proposed point 	 	 Current eval. 	 	 Best eval.
init 	 [-1.28010196 -9.48147536  0.99324956 -1.29355215]. 	 -65998.62492491212
init 	 [-1.59264396 -3.39330358 -5.90702732  2.38541933]. 	 -18480.20851361437
init 	 [-4.00690653 -4.6634545   2.42267666  0.58284189]. 	 -5357.298687487265
init 	 [-7.30840109  0.27156243 -6.31120269  5.70670296]. 	 -39506.05019877611
init 	 [ 7.07950585 -0.11526325  6.93122971 -8.40709046]. 	 -100181.8716424874
1 	 [-2.26214698  5.87274909  1.60008358 -6.75402803]. 	 -42299.56458715266
2 	 [-8.77692463 -2.54457149 -1.31781306  9.47735243]. 	 -132131.8483847979
3 	 [ 4.62190764 -6.83721008  8.23256169 -9.96152115]. 	 -221384.1127004357
4 	 [-1.79607725 -9.21541389  5.09748227  5.91370687]. 	 -86981.67408544771
5 	 [ 2.40271381 -0.94556933  5.9547712  -8.79894592]. 	 -104166.7973122442
6 	 [-9.3528057  -0.16784675  2.91108351 -9.91348457]. 	 -151154.0478201447
7 	 [ 5.87954056 -2.52000408  8.75494609  8.0392552 ]. 	 -131039.9203983427
8 	 [ 4.63750927 -4.88709839  9.82790351 -9.28611417]. 	 -227221.6289480885
9 	 [ 9.33988639  0.28284014 -7.13390479 -8.51921815]. 	 -123912.8921839668
10 	 [ 3.27690332  3.28708214 -1.44062271 -2.42812913]. 	 -1379.982986092025
11 	 [ 7.93183147 -5.23912097  2.68785335  6.56683347]. 	 -33550.60090311994
12 	 [ 3.43398606  3.86980772 -5.45233614  2.66724497]. 	 -12230.98770306435
13 	 [ 4.17183347 -5.30217661 -2.43725911 -5.04062651]. 	 -17658.66089429753
14 	 [-4.72742551  4.89518394  1.14984476 -5.58625981]. 	 -20605.19547957942
15 	 [-5.88451653 -2.90349692  0.10790176 -9.73341335]. 	 -144552.9273555162
16 	 [-8.75223676  2.75928373  0.07619989 -2.96137219]. 	 -2487.651424363742
17 	 [4.31659868 2.15880409 6.0999252  0.39576396]. 	 -15859.27733421717
18 	 [-4.85408782 -4.80236503  3.8199279   0.05267402]. 	 -8755.446175846268
19 	 [-4.9308896  -9.98716007 -7.22985281  2.73591316]. 	 -124930.1878355095
20 	 [ 2.6956679  -0.0955432   7.16240135 -0.63670877]. 	 -31817.69915499351
21 	 [-9.46753889  7.57506663  3.40441431 -2.45231133]. 	 -32004.14682106284
22 	 [ 4.29264853  5.29216254 -2.52190965 -4.29746325]. 	 -11754.37898908226
23 	 [ 8.21499945 -1.44616147  5.90323425  4.12780274]. 	 -18443.50397833349
24 	 [-7.34439343  4.37137915  8.9317863   2.08156362]. 	 -76466.33625698599
25 	 [ 5.30366211 -5.60567839  4.68352264  5.46326371]. 	 -26089.64022131350
26 	 [9.39325451 9.28524423 5.63715791 4.99259105]. 	 -69888.83518514827
27 	 [ 4.34821886  7.76812586 -4.17336063 -3.55419976]. 	 -32744.95039590380
28 	 [ 8.7657383   4.82994739 -0.03301329  7.01572752]. 	 -41790.11034695122
29 	 [ 3.10174137  8.19973381  9.28968585 -8.58272206]. 	 -191815.5957144834
30 	 [ 8.89371155  3.57965324 -9.99644447 -9.20542318]. 	 -245044.3578313064
31 	 [-2.31767317  2.06463895 -4.17948996 -8.2218637 ]. 	 -81192.22142715866
32 	 [-3.34605771 -2.85953726  7.38718484  2.85957379]. 	 -38749.06466331854
33 	 [2.69648123 2.25330911 6.21283671 1.03305146]. 	 -17031.09193502638
34 	 [ 6.45380163  9.97749924 -1.44184381 -9.79838237]. 	 -224061.8907623874
35 	 [-1.70037873 -0.26704715  2.11534591  9.18425958]. 	 -111272.3641833730
36 	 [2.50955865 6.96221021 8.45332699 7.14714992]. 	 -108416.1769691969
37 	 [ 6.2942239   1.56763564 -7.75477741 -9.45778093]. 	 -181663.7169076208
38 	 [-9.14232831 -9.09059733  6.87151726 -3.23202446]. 	 -93886.96222572782
39 	 [-8.59317205  2.85629998  8.59001874  9.77178052]. 	 -197222.8526831886
40 	 [-0.2464776   1.83345636  4.71569644  9.43313357]. 	 -125619.2257112983
```

```
### EXACT GP EI GRADIENTS

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

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

```
Evaluation       Proposed point            Current eval.         Best eval.
init      [ 1.01595805  4.16295645 -4.18190522  0.2165521 ].       -5185.478651216821
init      [ 7.85893909  7.92586178 -7.48829379 -5.85514244].       -83515.21921243734
init      [-8.97065593 -1.18380313 -9.40247578 -0.86333551].       -95899.98071510748
init      [ 2.98288095 -4.43025435  3.52509804  1.81725635].       -5245.566173274534
init      [-9.52036235  1.17708176 -4.81495106 -1.69797606].       -6987.20591009812
1         [-8.15565982  3.06821805  1.15681525 -2.76870474].       -2344.195455162361
2         [-1.42093745 -2.38317783  4.89209527 -7.84297806].       -63728.37816006544
3         [ 8.45325406 -3.16605859 -0.0933454   6.57617463].       -30343.01324038464
4         [ 1.13840429  0.92550401  7.42668624 -9.88496764].       -177270.1816038173
5         [0.63810404 8.56152948 2.44970438 5.87960129].          -60435.21909838218
6         [ 5.86849691 -6.28138395 -5.51873265 -6.54080399].       -57423.8930542661
7         [8.56580786 8.01799207 3.28161133 3.48133984].          -31167.61503743036
8         [-5.69652173  2.33901134  0.4494956   0.55949224].       -609.8771978113376
9         [ 9.96659594  7.93342819 -6.55928132 -5.07182197].       -58716.1904146615
10        [-3.88225406  5.58733297  9.07159878  6.32749837].       -104829.297177074
11        [-2.40285477  0.65691205 -0.26902882  5.49784057].       -14782.1159688005
12        [ 0.76213235  6.97711173  8.30170294 -8.73896127].       -153483.395027611
13        [ 8.30702859  4.35962691  4.99472689 -0.47784886].       -8120.920764787085
14        [-0.77767833 -1.76973304 -8.48492547 -9.95433933].       -234676.042851243
15        [ 7.27670478  4.74345362 -2.16801039  3.01652036].       -4609.888898484995
16        [-2.25892289  0.28501765  8.21214282  0.29800465].       -54626.84217539074
17        [ 0.79104414 -3.46101982  3.28279319  5.69558101].       -18127.0807021340
18        [-9.93423934 -6.52454398 -8.30416003  0.2549217 ].       -81073.1434509744
19        [ 6.52814059 -8.00196858  1.95175632  1.51111704].       -30331.37890947134
20        [ 7.28710188  2.40569536  7.8323326  -3.26322345].       -44207.08000729988
21        [ 6.34004657 -8.81902158  3.34124191  2.81273361].       -48089.44942390537
22        [ 6.00557472 -9.33087149  6.90123489  9.7261517 ].       -222296.563407310
23        [ 0.0268974  -4.25081081  5.42138196  7.38921469].       -57609.6048962670
24        [ 5.06938905 -8.85553535 -5.65175444  5.33826212].       -77657.2226813831
25        [-9.38113247  2.5918443  -9.46685486  8.43471911].       -186884.828946646
26        [7.13144725 9.3222     6.90505094 2.72277157].          -78057.21413414551
27        [ 4.90616113 -3.08882279 -9.59176825 -9.06186807].       -226291.2110509072
28        [ 4.8092043   5.72160034 -1.0963792  -9.62895308].       -146581.367658646
29        [ 5.40108179  4.81561102  9.56340016 -3.46184916].       -99368.5874814143
30        [ 4.61023726 -9.93444922  1.0541817  -6.45500409].       -101861.814921637
31        [ 8.29982721  7.15908754 -2.82215582  4.95650246].       -28830.98250177468
32        [-3.83311915  6.5274824   8.88792247  3.53862368].       -85749.89190879729
33        [-4.41232158 -6.8789768  -0.05876633 -6.4866039 ].       -48160.37249436202
34        [ 6.02219464 -2.2056801   3.19763028 -7.01160877].       -37789.62034303983
35        [3.63888383 5.04663199 6.90315115 3.13107829].          -29567.5645882436
36        [-8.59859583  3.02955784 -0.48297113  7.98756816].       -67188.53033624896
37        [ 3.63408602 -0.32876955 -1.35724627 -5.25019471].       -12841.4394728517
38        [6.65113509 8.12716792 6.32446538 3.56973434].          -48472.42156072955
39        [-1.42802071 -4.78782448 -0.61918123 -0.3976933 ].       -4571.722162803517
40        [ 4.59836289  6.63636051  3.11441338 -5.05268474].       -23635.6648138057
```

```
### EXACT GP EI GRADIENTS

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

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

```

```
Evaluation       Proposed point           Current eval.          Best eval.
init     [9.34059678 0.94464498 9.4536872  4.29631987].          -98038.88529472692
init     [ 3.95457649 -5.67821009  9.5254891  -9.8753949 ].      -250082.3135303139
init     [-4.94035275 -1.30416935  5.58765844 -6.04629851].      -30605.91051461647
init     [ 7.25986471  9.66801354 -6.72315517  1.94667888].      -84981.75728489687
init     [-9.82027805 -2.26857435 -9.11679884  9.13305935].      -209926.3636588899
1        [ 0.90405303  0.4880816   2.75220488 -1.97009113].      -745.6386114710477
2        [ 5.30486793 -6.47973031  3.69857697  5.99163089].      -34382.12799826672
3        [ 2.44668677  7.07389915 -9.38608839  0.44996751].      -105258.3008366837
4        [ 4.73707556 -0.72479098 -9.3121442  -0.43521843].      -91408.63072502242
5        [ 2.79273046  2.57666508  0.0799114  -4.11678866].      -4816.91703378007
6        [ 6.97297538 -6.40571993 -2.05106734 -5.92730455].      -32891.5631206189
7        [ 6.1769514  -2.79926084  0.04009536 -9.1639363 ].      -113012.938389662
8        [ 6.465232   -8.2562289  -5.78689094  5.87653584].      -73152.30153078973
9        [-6.21902391 -6.61688609 -0.04378302 -3.21504386].      -19491.7564400832
10       [-6.38615157  7.9844663   9.46020353 -2.30152042].      -123642.126186751
11       [ 3.60801592 -1.49238283  9.19474904 -9.70837594].      -215908.873845362
12       [ 4.11943982 -3.21456897 -0.24437003 -2.82458509].      -1640.59575442509
13       [2.35747262 4.35064884 6.12515272 9.24202752].          -126021.942410759
14       [-3.04455609  1.1937293   1.58678127 -4.47668773].      -6057.495942968814
15       [ 7.99683297  3.755866   -3.73641094 -8.81543277].      -103946.062007557
16       [-9.10473965 -4.5410053  -6.12870369 -7.58472702].      -82953.26934576029
17       [-7.30897249  8.05551792 -4.97066563 -9.81135946].      -198808.160504695
18       [-7.07788877 -1.21727011  4.87587121  3.08177701].      -8198.50117296675
19       [ 2.12735998  5.39297028 -6.17417945  6.93749789].      -63309.96222694547
20       [9.56336066 5.19000472 8.3192805  4.47006518].          -61256.56530619037
21       [-3.12601948 -1.58928398  5.38862215  3.37290353].      -12036.1533312127
22       [ 2.53687866 -7.51555741 -8.87336299  8.92430991].      -219167.001502432
23       [-8.24208797 -9.25718255  2.59909975 -9.02269984].      -168857.199441884
24       [0.93058337 3.00935793 7.4418012  0.28260586].          -35633.6797991140
25       [ 9.54173367 -9.81008555  0.86052579  2.50072544].      -67927.58047957321
26       [1.4296318  2.36539809 4.36157558 3.39953596].          -5416.752009334068
27       [-8.21767572  5.23712262 -5.71315147 -7.84922309].      -85352.43573873308
28       [-8.96018513  2.79650004  1.77717835  8.72422378].      -91883.97912615376
29       [-8.91707679  6.6642134   6.77411094  4.06821176].      -43709.77745315251
30       [-5.9705453   3.68337157 -1.81727078 -5.99138024].      -23940.1265141130
31       [ 6.47649147  6.79223988  6.68707488 -8.95483858].      -129723.600192451
32       [-3.71704512  0.63300159 -3.02912155 -4.65155591].      -9580.777575844764
33       [-4.25401474  9.99476907  2.24016058 -4.03644858].      -86979.63668770296
34       [-1.62200141  3.68679899  0.78104455 -6.12826713].      -23784.76990158775
35       [ 2.63221979  3.9114145  -0.64914921  5.04538811].      -12229.2796807114
36       [ 2.94320937  7.31830872  1.30397082 -5.33484823].      -34126.57198613119
37       [5.74745778 7.10563151 4.07291212 1.43798663].          -20200.2117527759
38       [-0.43817221  8.29492841 -1.88251509 -0.75725474].      -38158.66949280496
39       [ 3.65340937  8.70637033 -8.71244257  4.47020964].      -114701.451409868
40       [ 4.96764821 -8.24675188  7.9264222  -8.5024187 ].      -162853.4148627654
```

```
### 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     [-5.56013658  7.41464612 -5.86561689  8.37221816].      -123365.276489362
```

```
init     [-0.23177622  2.23487726  5.31815713  0.36835976].       -9167.819359373088
init     [-4.06398997 -6.24557543 -8.38517462  4.76880592].       -89817.03050330501
init     [-1.17381554 -6.83380265  7.59874062 -4.51827076].       -67194.19259941578
init     [-1.71529962 -4.07840135  2.57575818  1.5967562 ].       -3383.131100651736
1        [ 1.55325716 -9.96715655  0.30945224  2.79590352].       -78973.83235307777
2        [ 4.56139684 -4.673529    0.23379905  2.52038143].       -3764.515116656097
3        [-2.15134925  9.53418759  4.82671391  6.46019907].       -96545.59507289826
4        [-3.38088417 -8.29206158  5.15473834  1.65891555].       -51046.5385766993
5        [-3.02773187 -2.8997851  -5.591614    3.48603667].       -17223.16765746139
6        [-3.76365226  7.78239697 -8.18300187 -5.22131556].       -94683.43682804274
7        [ 0.41211986  0.47256529 -3.34919958  6.13953029].       -26245.27312633629
8        [-0.25183718 -3.30684237  8.97845951  1.46012155].       -82281.90882897378
9        [ 2.48093212  5.13883859  2.67965072 -5.30386824].       -16808.73137046458
10       [-4.45934361  6.06017597  8.73696795 -2.92485867].       -76933.0298360569
11       [ 5.70578654  1.32017865  2.21996359 -6.47121455].       -26841.30454703817
12       [-7.95698611  7.84422949 -3.18165461  6.42226612].       -64233.77703910508
13       [-5.79261282  9.88159419  1.54685764 -4.35636722].       -86296.40204165212
14       [-9.58242141  6.39325499  3.81306329  5.48600232].       -31052.72075471336
15       [ 8.81346656  0.26074237 -7.90133971  4.47083934].       -55957.58957399476
16       [-7.88879274  5.49063761  1.77286879  5.52254046].       -23408.69727556262
17       [-9.81344616  7.08235098  9.35724572  9.34014123].       -218138.8727690319
18       [-0.08236537 -8.08957624  3.29476233  2.13206532].       -37103.3694018523
19       [ 3.38262059  4.78268054 -6.48529893  8.6960044 ].       -121988.0377694543
20       [ 4.24154943  2.47872609 -8.90447672 -3.83782359].       -79128.92635198032
21       [ 9.52121411 -1.72632669 -7.50690066 -7.2709549 ].       -90676.6246520302
22       [ 0.71939891 -8.42139745 -6.9306136   2.91365032].       -74870.03397220623
23       [-6.71412576 -6.47032038  5.49564249 -9.1363839 ].       -134102.6054030161
24       [3.68663816 3.62581784 8.99921246 2.29340655].           -76258.77495566735
25       [ 6.46854411 -4.35327928 -1.02970826 -3.81613679].       -5769.191423731238
26       [ 5.63902137 -4.07372558  5.15047326  1.24959081].       -11347.11766688968
27       [-1.51733203  8.15335737 -4.12851163  2.03908977].       -38809.1169144927
28       [-5.23708913 -0.55987899 -9.39960273  5.03858434].       -108860.0233416873
29       [ 6.26331658 -6.83420093 -0.62129925 -5.88826337].       -34971.43896351736
30       [-5.87723567  1.52217007  0.14509913  6.77549639].       -33888.10530914785
31       [-7.86616406  4.19461375  2.22892018 -8.6880036 ].       -92372.18114221275
32       [-0.66025888 -9.79876343  4.17691916  1.06487357].       -80269.72780342265
33       [-6.83456102  2.26706667 -4.70519547  6.68073144].       -41262.9262105125
34       [-2.9787225   5.21658538  4.27673789 -9.91195279].       -157346.8138477022
35       [-5.51319281  5.35686139  6.09740957 -9.24808436].       -131082.3842347402
36       [-1.53275275  3.86512508  6.75586768  7.15377169].       -61460.81820364318
37       [-6.77904168  6.57025153  8.02232788  4.05841054].       -64643.70473120608
38       [-4.90625507 -7.98234335  2.23697977 -8.24396814].       -107525.6365292444
39       [-0.19518197  9.15097708  6.59601634 -4.81104503].       -80723.41013644819
40       [-2.41654775 -3.00554909 -1.86072836  5.49196288].       -16614.04018806096
```

```
### 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     [ 7.85720303 -3.36040389  6.42458246 -9.16606749].       -127092.7561755139
init     [-7.8468664   1.90104128  0.59634724 -1.62385143].       -624.5256755977402
init     [-3.29184301  2.45038864 -1.23717148  4.71764213].       -8859.81358392905
```

```
init     [0.36072824 1.577172   2.90710192 9.80448543].         -144159.4084820958
init     [ 6.39716394 -1.7359813   7.5253531   6.47518865].       -63008.48019540512
1        [ 4.34664291  8.74699069 -2.96380463 -4.92731805].       -55069.53937594843
2        [-5.57681425  2.83874786  4.53696952  0.08226863].       -5473.77883338242
3        [ 8.56608527  1.00066366 -5.6557535   4.78400723].       -22620.59205919053
4        [-3.88060276 -3.60562409  2.63626196 -8.23698378].       -73548.32801665392
5        [-4.92678916 -8.12144034 -3.72029518 -9.97541476].       -205743.2236843433
6        [ 0.22761873  0.23504219  8.807863   -7.52808231].       -115714.2258046076
7        [-8.50059073 -1.38311772  0.26491341 -2.68077816].       -1197.283861430405
8        [-3.05696573 -2.87038333  2.44675902  5.15470832].       -11720.74995014418
9        [-2.16296848  1.0549018   5.36997986 -5.80365616].       -25038.9315654272
10       [-1.9469586  -7.49626221 -0.88842205 -4.94026588].       -36281.36864476779
11       [ 5.16565529  3.58828793  9.82334313 -3.91747081].       -110233.116869007
12       [-4.21628332 -0.5792177   4.06056059  4.91923215].       -11316.0237781142
13       [-9.87266828  6.19478468 -3.35719509  5.95853309].       -38047.95048968376
14       [-9.29604707 -8.38751066  6.35426496  9.78206391].       -205876.7887789898
15       [-6.53620178  6.01567506 -4.70249423 -1.18971461].       -17118.5828519783
16       [-5.88055486 -8.69471242 -5.34294918  3.25915367].       -65205.50674296516
17       [-9.10121252  5.55793128  3.33297346  6.97861209].       -46379.25067804724
18       [-3.49163905 -5.19036252  7.22794293 -3.69916265].       -44312.46925460934
19       [ 4.60029783 -7.6833382  -3.64473115  5.6386107 ].       -47362.8911748210
20       [9.23295363 1.41311142 4.18371188 5.17919173].           -13295.19009100279
21       [ 9.23386226 -3.14703755  0.93975378  4.99615186].       -9961.289920070994
22       [-7.24071559  9.57212304  4.29760153  5.1102351 ].       -84078.63324907442
23       [1.65398501 5.21942386 4.30203553 5.07407906].           -17523.2723359009
24       [-7.90638723  6.02936142  9.88101478  9.32707469].       -228236.2384016493
25       [ 0.77200041  9.62659846 -2.18138662 -6.42799569].       -96910.333085085
26       [-0.70291207  4.05680639  2.39108638 -6.14689486].       -23846.3840080097
27       [-5.11903377 -4.99683891  3.04621959  8.47640559].       -86897.06141964762
28       [-4.65927739 -2.67274277 -2.19018009 -5.80529003].       -20574.08393667018
29       [ 9.5791857  -7.62340548  2.29835671 -2.13668967].       -24002.68921035135
30       [ 6.54205018  2.40126832 -8.32963234 -8.02728349].       -131166.1309941800
31       [-9.99270743  7.04938997 -9.69486923  9.40023195].       -261272.7694853841
32       [-5.17905235  6.05000114 -5.09537955 -1.91364807].       -19257.04511561468
33       [-1.74330707  5.34344546 -2.52507635 -7.61939236].       -63395.78154380738
34       [ 9.06078812 -3.36823727  1.41571386 -8.15029929].       -69704.62612614207
35       [ 3.03015401  1.48505336  7.76143542 -6.05794541].       -59719.47598587538
36       [ 3.82371652  6.64522721 -8.34300974  6.76242933].       -106851.4017539855
37       [-0.84589908  6.56578119 -2.97015905  4.94894767].       -26329.1562864958
38       [-0.38903965 -2.73642296 -2.59390311 -0.40842046].       -1295.029516175667
39       [ 0.92127522  9.6840203  -5.88009706  7.0686776 ].       -125062.543057291
40       [ 6.91593389 -2.78410453 -6.23227478 -1.02922871].       -19886.0279604183
```

```
### EXACT GP EI GRADIENTS

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

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

```

```
Evaluation       Proposed point            Current eval.          Best eval.
init     [-8.47383421  5.59837584 -1.23181537  4.46930356].       -17019.79107531184
init     [ 9.55979024  0.76991741  0.02240927 -8.55897733].       -86052.31833948194
init     [-4.63122040e+00 -2.34998349e-03  3.58459992e+00  6.07478072e+00].       -2
init     [-2.38117734 -8.68127306 -4.23708801  8.19187055].       -129535.162956385
init     [-5.73229293 -0.95752076  8.62412039 -9.50201545].       -185668.2647263406
```

```
1         [ 6.75835989  5.37295013 -3.72010646  1.45250665].        -6975.589978136528
2         [-2.26867274 -4.96775263 -3.10681926 -5.91972827].        -28537.54981454135
3         [-7.1620489   2.23165943  9.48914494 -4.53159483].        -99540.92877448558
4         [ 4.29877809  2.09137364 -5.85501544 -4.79573932].        -24060.70100027514
5         [ 5.54571144 -1.93184305 -4.33889721  9.71942618].        -154147.0884339929
6         [-8.29791508  2.55444133  2.18471017 -8.00103752].        -64495.96200318241
7         [-4.37541545 -0.5346361  -5.18090179  4.60429146].        -17952.04503879531
8         [ 1.07740606  7.90693058 -1.60531584 -3.10335   ].        -32497.45135574664
9         [ 4.32564604  9.62119313 -9.80342151  4.4727457 ].        -175341.3282758457
10        [ 9.91679152 -2.93825443 -0.25081413 -6.81258225].        -34866.32361728202
11        [ 4.86271426  9.61287061 -5.26853416 -9.32001869].        -199256.6986385452
12        [-7.17360778 -4.84967241  6.40731588  0.96189955].        -28712.39669282479
13        [-1.95354323 -4.49660935 -0.56509588 -6.77693153].        -37846.953342321
14        [-3.96682274 -5.05776767  7.16060915 -3.46972848].        -41985.26763565934
15        [ 2.19617536  8.32020227  3.48291605 -7.99462824].        -99741.80156219704
16        [ 1.41863484  1.05470351 -4.69092279  5.37098748].        -21104.84207714151
17        [-7.34903531 -2.78590343  1.95047524 -5.36666749].        -13828.54148107305
18        [-6.08936989  4.67588389 -8.44980741 -7.34664296].        -116439.5586432195
19        [ 7.70140256 -7.00291695  4.88315627  8.18434349].        -92006.69879158467
20        [ 6.36008573  4.22154581 -9.3808628   8.21213064].        -173511.1033765117
21        [-0.48423379 -1.91873994 -2.3040535  -5.17695284].        -13098.62862982250
22        [ 0.8663081   8.49846794  2.380057   -2.97697714].        -42197.25780012144
23        [-1.53060732 -4.40544996 -8.1972214   5.76814545].        -83398.09905191144
24        [5.35915716 0.292985   1.96191354 3.5557524 ].          -2413.526174359907
25        [-2.08000587  6.91069233  5.94203563  1.06121231].        -31288.57977997730
26        [ 1.7834508   8.18962124  7.52724359 -2.99170249].        -68623.86706774902
27        [ 8.25477835  2.45208354  3.14975175 -8.1648879 ].        -68776.69144110718
28        [-8.53174551 -7.24135683 -8.67740592  4.51547898].        -110333.2973871063
29        [ 6.33950265 -8.4958139  -6.70519534 -9.77466497].        -223671.9881930372
30        [-6.72396884 -8.73495195 -8.73106899  4.65265186].        -139608.1688287641
31        [-6.55621397  3.13032354 -2.64641281 -4.06209353].        -6863.084339853854
32        [ 5.39547929  4.90665832  7.72371295 -2.08746083].        -42944.64207707912
33        [ 1.37330385  7.30169397 -5.16091107  1.80765069].        -29044.31360697798
34        [-6.85227879  3.44695012  2.62600612 -2.7844801 ].        -2920.933953033779
35        [5.77785207 9.69501445 1.503218   9.14237305].          -176279.8437645275
36        [-1.3949032  -0.75431933 -7.96830523  5.46245939].        -67276.20454852493
37        [ 2.95590367  5.52224194  4.13929932 -6.38631837].        -33200.82794128400
38        [-2.94506043  1.50124625 -8.69803676 -2.58980888].        -69411.95622164142
39        [-4.32988338 -6.09934985 -9.42005188 -1.27633322].        -114167.7928186582
40        [ 1.8230842   3.85897299  5.62694179 -1.50053216].        -12177.88760762808
```

```
### EXACT GP EI GRADIENTS

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

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

```
Evaluation       Proposed point           Current eval.          Best eval.
init      [7.46858806 9.37081326 7.3838908  0.61711383].          -86573.97855073148
init      [-5.34543344 -9.77202391 -1.39062363 -1.9529728 ].        -78014.94707602388
init      [ 0.45349343 -0.43216408  1.10712948  0.86772035].        -25.89351629774407
init      [ 5.21791151  4.24749148  2.39364192 -1.47816459].        -2094.662571070279
init      [-4.21849944  9.47710482 -3.32451909 -5.62397878].        -85839.98682911636
1         [-3.60620407 -1.35048134 -4.59708517  6.02111773].        -29614.34759252149
2         [7.02116965 7.81234303 5.75333375 8.88043753].          -129115.3070880865
```

```
3        [ 2.38733468 -2.49139217 -6.58364982  0.33931039].       -24248.08127943354
4        [ 7.56352013 -7.67982822 -8.37749723  2.84161744].       -92575.686823112
5        [-5.28183607  6.44896586 -9.02454394 -6.64184392].       -126938.728925031
6        [-8.91376283 -9.57999256 -6.95098062  9.64939713].       -257290.733204339
7        [-9.6826463  -0.3889901   5.14264133 -8.41133424].       -83205.11096307916
8        [-0.44591076  5.04602604 -4.13162746  0.15920843].       -7889.447613833232
9        [-0.81981182 -3.75576157  2.39337966 -7.50364036].       -50973.32732738554
10       [ 6.98608516  8.9084123  -7.19395552 -9.08060074].       -191413.517030468
11       [ 6.61186169 -9.02334481  1.34103459  6.04361295].       -69893.69310739316
12       [-4.95566202 -0.69807525 -2.84596135 -4.0933849 ].       -6249.828029368294
13       [-8.62371369 -2.43071156 -9.9135532  -8.61774405].       -220135.392152266
14       [ 7.25721416 -1.35983896  9.25077409  2.93432361].       -89601.25460647872
15       [ 9.5056863   3.58618034  2.73863527 -1.09937986].       -989.4998714779908
16       [ 8.63675564 -9.78898035  4.97635133 -5.02173071].       -85866.47285240376
17       [-8.87173722  9.60530842  4.18381046 -3.54573537].       -78594.56686815829
18       [ 7.65858112  1.1873229  -3.41218412 -4.96882152].       -12703.59774024508
19       [-3.51296977 -2.14237485  8.01964739  7.68652932].       -100174.812514827
20       [ 3.88729122  2.89386017 -8.55538582  8.36656893].       -150385.5235925565
21       [ 8.14290147 -0.90943489  9.4506528  -6.76425   ].       -123773.1552840801
22       [-9.31540474  9.38993355 -6.57652645  3.14559475].       -89662.74292924849
23       [-1.31717943  4.6340913   9.30919578 -8.69237554].       -169726.852751085
24       [ 6.49597242 -8.88231691  9.8207384   9.40242253].       -279494.573861147
25       [ 3.49120939  0.47218526  8.86783187 -1.27059239].       -73915.08882010846
26       [ 4.70604571 -9.24361287 -3.09457652 -8.48137246].       -144057.432789729
27       [-7.83064872  2.15675112  8.42197817  0.68410525].       -59439.79169643539
28       [ 6.22824474 -6.53188051 -3.40849765 -4.72519097].       -24440.47690013403
29       [-3.33713093  8.41075107 -0.91495656  7.81727567].       -102748.276973589
30       [-8.73569396  9.75149541  9.21231033  9.65324137].       -281554.475751926
31       [-7.949679   -9.34510791  9.14340971 -1.5517733 ].       -160357.752299439
32       [-9.83001322 -9.60151204  4.93364162  8.43985391].       -161398.1157902435
33       [ 0.46483799  6.04685576 -1.54766819 -8.53591252].       -97320.47350202779
34       [-6.12416735 -5.39688856  2.41170879  5.10668644].       -19107.73733976571
35       [ 0.84836326 -3.15814819 -7.48375775 -9.06258209].       -158507.7161246386
36       [ 6.77387947  8.29094342  7.88717361 -9.5531293 ].       -196648.514250454
37       [-3.18294357 -7.91863135 -7.28881591 -4.14702255].       -79145.8077242034
38       [-9.90294061  0.67931568  0.49901318 -0.27352452].       -353.860344687546
39       [ 8.37519617  9.65355887 -5.54273128  0.08164679].       -71597.6689957656
40       [-4.16342788 -4.96923312 -9.40204518 -4.25782433].       -113218.060251910
```

```
### EXACT GP EI GRADIENTS

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

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

```

```
Evaluation       Proposed point           Current eval.          Best eval.
init     [-9.79251692  0.03749184 -0.08453414 -7.32340942].       -46403.7672209163
init     [-7.15777829 -5.62882649 -1.62983639 -5.03797663].       -21353.0900000741
init     [-8.31880698 -3.0900272  -6.66447307  7.57118171].       -85802.65922587285
init     [ 9.01928063 -9.22503248  3.98214783  1.45519631].       -57052.85171456846
init     [7.96014236 3.33797946 0.95675566 4.04854848].           -4516.871493114117
1        [ 7.46801686  8.12554554  5.16555974 -3.06196092].       -37961.04796864094
2        [ 1.79155607  6.63416988 -1.83334365 -8.8517705 ].       -115413.890520744
3        [ 9.33019641 -7.52767745 -5.63058289 -0.50920484].       -36948.48564229587
4        [ 9.24871681 -5.28271506  6.79982967  9.30415074].       -143743.462714014
```

```
5        [-3.79629409 -6.6853837   6.54189652  8.77158881].        -129771.5779963778
6        [ 2.12600285  4.95023923 -9.32534107 -9.64414998].        -242691.8532438531
7        [-2.63291278  2.0648516  -8.06169706  7.76658863].        -115606.9228359033
8        [-7.33859604 -6.03880343  3.43779957  7.6829753 ].        -68148.48186924399
9        [ 4.5742908  -2.26634671  4.22642647 -4.96365815].        -12525.61373048868
10       [ 5.90229198 -0.97787337 -7.39679132  1.70640703].        -37321.40235687539
11       [ 3.76231019  5.61395399 -7.52330761 -1.7438099 ].        -42498.51492727356
12       [-2.80325151 -5.66028587 -7.21789028 -7.12490995].        -92468.98160414296
13       [ 4.59063459 -6.63437507  2.59596454 -2.48616035].        -15532.13435579707
14       [-6.7116605  -4.04781318  5.68123009  4.08874643].        -20376.36019655792
15       [ 8.77910755  7.96563606  5.34858784 -0.95326829].        -35293.19974799102
16       [-6.00453231  6.54755256 -3.53825723  9.75186528].        -168043.8232803098
17       [ 4.61339941 -7.99195479 -3.91090888  2.54264043].        -35934.96597058500
18       [-2.21057806  1.20088845 -6.72351962 -8.31126179].        -107895.9368687414
19       [-4.01060296 -0.34982372  8.60881423  0.96692656].        -66464.95145720668
20       [ 0.76122047  4.08559236  4.4817093  -3.30378374].        -7238.979040113192
21       [-1.91232744 -4.09825371  2.92332128 -6.9653553 ].        -39302.57410984345
22       [-1.49303262  7.7941114  -0.46493609 -0.37276328].        -30423.78391428087
23       [ 5.02159077 -5.17831978 -6.91006052 -0.58676192].        -35379.77097239700
24       [ 5.319375    7.52329844 -3.17784069 -8.25196861].        -101470.5753208458
25       [-5.38540934 -6.04055244 -8.33599896 -4.84152513].        -87608.44575302256
26       [ 8.62767184 -3.2594571   7.51514305  3.31157527].        -41725.4643291665
27       [ 7.59743528 -0.24000876  9.10743465  9.12356887].        -182017.0232794030
28       [-0.78769787  9.1489014   2.08254341  9.14955056].        -165938.5096383604
29       [-9.6547581  -1.49679449  1.00515371  3.34367319].        -2374.438015296140
30       [ 2.60369403 -1.68571753 -4.43767666  7.25026529].        -53104.84891982783
31       [-6.1692829   8.70589311 -5.26969916  0.31795138].        -56524.23534410765
32       [-1.18542896  0.40789086 -6.59527104  7.70779768].        -85417.72425695068
33       [ 3.8845512  -5.42246098 -7.03261448 -8.49555409].        -130365.6395442401
34       [ 5.31529745  7.16306101  4.53196711 -9.55319334].        -149134.0914835391
35       [ 3.06205081 -6.73516315  9.16175403  3.84221381].        -108499.0886007453
36       [-5.99732874  8.49617413  3.35263634  5.54524537].        -59380.49767926420
37       [ 1.08245553 -8.94111158  1.08361444 -1.57389027].        -50880.03298459585
38       [ 1.21816957  5.83576602 -6.28781207 -6.81494174].        -64383.31707706594
39       [-5.08588753 -2.04676062  9.62851081 -5.07771755].        -112862.1276441603
40       [ 6.31302522 -3.63294911  7.2082138   9.81186853].        -172935.3875377381
```

```
### EXACT GP EI GRADIENTS

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

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

```

```
Evaluation        Proposed point              Current eval.          Best eval.
init     [ 5.42641287 -9.58496101  2.6729647   4.97607765].        -74102.72845843069
init     [-0.02985975 -5.50406709 -6.0387427   5.21061424].        -40371.05435055173
init     [-6.61778327 -8.23320372  3.70719637  9.06786692].        -147677.3182906593
init     [-9.92103467  0.24384527  6.25241923  2.25052134].        -18605.70454811222
init     [ 4.43510635 -4.16247864  8.35548245  4.29151567].        -67109.3842877247
1        [-3.98599887 -7.72031276  6.57362653 -9.06207361].        -156405.4305727304
2        [ 5.24133604  6.79774962 -7.90085176  0.21262617].        -57280.06880918705
3        [-6.98285725 -7.53553636 -0.44064091 -4.85155595].        -38348.42146299815
4        [-1.78826799  3.25598727 -1.45510672  1.58046094].        -1234.288419082589
5        [ 4.63220256 -9.79505919 -9.12123737  0.49337993].        -163638.3926751861
6        [-7.83635489 -1.28730572 -7.74844317 -2.63232565].        -46381.85432957351
```

```
7        [-8.78408217 -4.4360313   3.61659233  9.00687312].        -108194.1564884725
8        [ 8.75235832  9.38171401  9.18458426 -0.57986069].        -132475.1706967695
9        [0.4502773  8.55166428 3.92918152 5.72526907].            -59209.78291322129
10       [ 1.23952645 -2.7410793  -5.02186756 -4.62861357].        -18030.53509888344
11       [-5.31806714 -4.24445138  1.33120268 -2.17377886].        -3905.054505195477
12       [-7.17834919 -3.57387423  4.11540377 -2.61834418].        -6783.704136380395
13       [-4.63875915  8.6124843   7.12702008 -7.73195445].        -123346.8915830602
14       [4.4189373  5.3779954  6.27947744 2.68037042].            -22182.70197600124
15       [-8.76637523  9.03915533 -9.83105068 -1.79825538].        -162304.3833604644
16       [-1.72806915 -4.12214739 -2.17527422  6.47352369].        -32687.84052614772
17       [ 4.04923794 -1.19125696  8.88072996  1.0047913 ].        -75972.96016838716
18       [-3.15658879  8.4378996  -6.47515603 -5.59613802].        -78555.94986395625
19       [ 8.67011191 -4.44329106 -2.6700974   7.10050894].        -45860.10611431824
20       [ 7.26842072  4.30150932 -4.14983461 -3.13661989].        -6804.157833513567
21       [-1.88045224  9.9970487   1.83389128 -6.26444981].        -104959.1260880316
22       [-3.4220057  -9.61484884 -3.35490766  7.58797804].        -130217.254194649
23       [ 0.06951243 -3.70532939 -9.7893336  -7.65624326].        -180548.3944333123
24       [-6.32728027  5.50781547 -3.21169796  9.88098785].        -167294.0655340885
25       [-8.52234944 -3.55025167 -9.89809408 -2.34727393].        -123510.8970101143
26       [ 1.35249603 -2.58501891  4.8062863   6.46961256].        -32333.17259547553
27       [-5.32719949 -3.84380917 -0.88484076 -5.01546417].        -13044.17864582639
28       [-9.0177282   4.17602767  0.45261751  6.82006506].        -38276.40299654401
29       [-5.40929599  5.3065897   8.96440883  3.90370863].        -81974.5082563838
30       [ 9.21235497 -0.61948647  1.97276737  9.10396676].        -107732.7337650096
31       [ 0.39304073 -7.6415586   1.70522151 -3.96620909].        -31180.45677758349
32       [ 6.5529982   9.84465494  7.62453119 -8.15052117].        -166894.3295125242
33       [4.22395324 5.29166215 3.44422297 0.56791109].            -6423.354644085020
34       [ 7.09018384  5.80556675  4.97544214 -5.62140512].        -26604.68435614685
35       [-2.57459635 -3.70262064  4.83174198 -7.26181384].        -49941.20527412557
36       [-3.66972549  6.64597511  5.60457936  6.42645268].        -50132.39410201717
37       [9.89814898 3.60176919 5.79727835 9.39887058].            -129535.1053390210
38       [ 0.77397722 -3.73697848 -0.31484555 -1.13828133].        -1555.194529467911
39       [-6.94667037  5.78064742 -2.99714345  8.53924505].        -100001.9948580765
40       [-4.02048281 -6.951201    7.06956642  1.94156942].        -54578.94172287029
```

```
### 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     [-6.39460622 -9.61049517 -0.73562947  4.49867858].        -80243.41172762495
init     [-1.59592791 -0.29145804 -9.74438371 -0.25256785].        -108928.0486060041
init     [ 8.83613305  7.01590179  4.5992894  -7.82527856].        -75431.53172537561
init     [ 7.87808341  7.14308494 -6.69826765  2.64668028].        -39961.87097908433
init     [-9.59032774 -7.66525462 -3.67265377 -6.84175387].        -73885.31196002741
1        [ 8.98204795  9.73346661 -3.23891901 -5.20250642].        -78768.56237678342
2        [-2.46813998 -7.3003779  -2.91425156  2.82909317].        -27001.41579121021
3        [ 8.5668058   2.92682213  8.40866209 -9.14225066].        -158547.6141739462
4        [ 8.76589534 -3.7692322   0.38597262 -4.05301551].        -5098.705872419257
5        [ 0.40212877  5.87654958 -6.74210512 -8.93336634].        -141815.5175477769
6        [-9.57187238  4.8761458  -5.96218891  3.90472017].        -25109.49012053475
7        [ 6.45130653 -7.70340012  7.43044673 -4.70207094].        -72497.57459422454
8        [ 6.63688003 -6.08240957 -3.75262294 -5.63404841].        -30707.15696293844
```

```
9          [ 1.38380213 -6.05647422 -9.74783179 -9.20728444].        -254312.0404684823
10         [ 0.01626441 -5.441819    4.18137887 -1.59234324].        -11914.73258542305
11         [ 7.11903192 -7.10840682 -3.70390674 -0.4859192 ].        -21336.93053970189
12         [ 3.41314276  7.39306433 -0.25317011 -0.71476575].        -22600.98264730552
13         [ 4.20641113 -2.1133109   2.2765559   1.21529123].        -523.9372697703078
14         [-8.33083797 -2.90069237  2.58400674  2.53507465].        -2567.49074682586
15         [-1.84717545  7.23531104 -8.11301938 -5.04046295].        -83032.55435278178
16         [-8.69423942 -2.75516543  9.30621651  3.78992547].        -95635.41800358966
17         [-2.1973427   9.33772956 -3.2166429   7.28382045].        -110568.2970160097
18         [ 0.61155877 -9.53496613  8.85943923  8.7384669 ].        -231647.5606548612
19         [ 5.36664831 -2.11942177  3.02047124  1.99056466].        -1385.753710236697
20         [ 1.76289051  1.78627203  1.36617813 -1.10226957].        -59.14051345519054
21         [5.63201496 3.07625892 1.29545321 9.62756645].            -135923.776808196
22         [1.64271671 9.5204946  6.16736798 5.92850153].            -94275.94175554602
23         [-8.98209171  2.94475055  0.39880701 -9.50217262].        -131371.1252137147
24         [-2.28632874  9.98430773  6.8094362  -5.18605818].        -110716.3994632515
25         [-6.31817008 -7.90763218  7.25451881 -7.85973695].        -127092.1847245969
26         [ 3.60790617 -1.08269505 -9.80543179  7.63242395].        -176012.5755772683
27         [9.61664775 9.79324053 9.25525413 9.88924306].            -283627.8652284671
28         [-5.67449855 -2.20755798 -7.43358352  7.37321022].        -92614.67391777792
29         [-7.40043441  8.29073044  3.78341128  2.22255965].        -43436.37056286866
30         [-2.58274536  2.3969424   9.40397512  8.94997705].        -182693.7790784494
31         [-2.81062329  2.64868747  9.45106608 -8.50015051].        -166464.6173721468
32         [-2.5896196  -3.43395846  2.79604208  8.9737073 ].        -102657.3680065169
33         [ 7.90813127 -7.00473826 -3.23898373  8.7030729 ].        -114438.9168006048
34         [ 9.42753538 -4.45432105  7.90433013  7.33298397].        -91856.74916855985
35         [ 9.26787693  1.46443124 -9.51187767 -9.61152017].        -247735.2542922033
36         [-3.18762103 -2.67621939 -1.33710706 -5.53711239].        -16451.39569891577
37         [-9.75952228  8.75485968 -6.60363198 -8.59274501].        -166959.8927635811
38         [ 6.8026804   0.88063016 -5.80139913 -2.22954404].        -14319.78142365637
39         [7.33669893 2.81720503 8.89758199 5.04348725].            -79790.17619431006
40         [ 1.76632599 -9.14784715  2.51238446 -9.75983822].        -197640.029210196
```

```
### EXACT GP EI GRADIENTS

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

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

```

```
Evaluation       Proposed point           Current eval.         Best eval.
init       [-6.91674315  4.80099393 -4.7336997   0.67478787].        -10615.02722024104
init       [-9.70850075  8.37494016  8.01429708 -9.33157145].        -198779.9445700966
init       [ 9.13898673 -7.25581357 -4.32343294  2.12166369].        -25246.22965104943
init       [ 8.88450272  7.05471082 -9.95481533  0.42452054].        -126529.6321262879
init       [ 1.04075267 -0.29245173  5.36268308 -6.78566494].        -40114.0772599786
1          [ 9.00627049  5.34951301  6.50018506 -1.86719396].        -23513.61734845002
2          [8.11179655 4.25164412 2.83994041 0.51622188].            -2067.802572957712
3          [7.07957877 4.20923579 8.31123577 5.90287485].            -70536.44119328947
4          [ 3.65158786 -4.292633    3.33049006  6.75429274].        -35227.82899489502
5          [ 3.07120909 -3.55916591 -5.97833992 -2.63162   ].        -19461.58116596912
6          [-8.34831975  3.35126666 -1.83244524 -0.35201807].        -2037.208480552417
7          [-6.77288607  5.95533828  4.43070962  2.50634443].        -15729.19686434966
8          [ 9.11568286 -9.45162454  9.96288854  5.4127943 ].        -196776.2123688205
9          [-0.54343888  6.9130662  -3.08529747  0.79842711].        -18999.28422234534
10         [ 9.87704008 -9.57194896 -7.30925724 -2.85091279].        -103075.0157507258
```

```
11       [-8.01556949  4.33863986 -9.09107963 -5.10325364].        -96943.55441519922
12       [-7.70853825  9.04000734 -1.47997073  6.83741383].        -94812.1104929485
13       [ 8.2042246  -8.88295657 -0.07055749  6.74550618].        -78233.31027220882
14       [-5.69729876  6.22751761  8.3350766   9.36945888].        -178627.8986028522
15       [-5.77909101 -5.04328215 -7.7933351  -1.05081511].        -54882.65678468381
16       [ 4.71575825  4.71416309 -2.11686206  5.24056754].        -16241.08754577173
17       [-7.8093924   0.25400768 -8.20800109 -6.68611527].        -92580.90723825619
18       [-9.43342891  5.47058984 -1.80600297  0.74344179].        -9747.777715504932
19       [ 3.45355371  3.20265039 -5.41025361  2.04079201].        -10530.50206993445
20       [-8.69701704 -3.37114475 -9.15259224 -0.82149058].        -90142.1875695829
21       [ 9.94633892  4.14929651  9.20784367 -1.98509533].        -83377.10843544235
22       [-9.44413253  7.85647531 -5.46813037 -2.76954875].        -45256.65091147188
23       [-6.34363966 -3.26687378  8.41196286 -0.3285002 ].        -64747.61027266566
24       [ 4.20796658  8.79614205 -5.71197443  7.9768651 ].        -125621.8429352873
25       [ 2.30229124 -0.93404272  8.38361118 -2.73516381].        -60245.50527258935
26       [-4.51712007  1.04990949  6.57777777  4.88865804].        -28839.96783413804
27       [ 3.59342012 -4.54813843  8.78430934 -4.99047946].        -85318.44477426192
28       [ 2.8327017  -6.12881065  4.97078983  3.7126317 ].        -21754.18976933502
29       [ 2.40471286 -1.77558974  0.60610937 -2.25237082].        -415.3744113897338
30       [-8.25635404  8.72047473  9.41845964  5.43226615].        -146723.8974491359
31       [-8.96587415  9.17498919 -3.32823657 -6.17903405].        -88895.69305612215
32       [ 9.3826277   6.88204754  7.40125444 -8.34942264].        -115985.3182359701
33       [ 0.44916896 -0.41837093 -5.42637997 -7.92033387].        -79081.71266163523
34       [ 4.36794641 -5.148217   -6.70140915 -2.75489941].        -33713.72102852014
35       [-9.25013146  6.25126713  6.92764134  4.34208954].        -43330.68275712555
36       [-7.74821935  4.55726898  0.61079043 -0.96146709].        -4984.381501419528
37       [-6.85775607 -0.31294798 -7.19574807  5.45422751].        -50320.30662778412
38       [-1.07792303  1.04005608 -0.20712497  7.76819596].        -58492.26726927265
39       [3.8468842  2.76795099 7.84856776 2.62267418].            -43922.80089101722
40       [-4.43765623 -5.50523994 -7.91455609 -6.54045094].        -94754.2282893823
```

```
### EXACT GP EI GRADIENTS

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

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

```

```
Evaluation       Proposed point            Current eval.         Best eval.
init     [ 5.55404821 -5.2491756   6.48557065  9.31498396].        -140521.0909730103
init     [ 9.45202228 -0.93101505  2.18084926  5.51053029].        -14230.66474647201
init     [ 2.8322669   4.44036459 -9.29926952 -4.03101058].        -94860.11143837457
init     [-8.82975016  7.14121885 -2.54291944  3.59695903].        -27991.65897596937
init     [-4.87440101 -3.0483757  -9.8117446  -2.83332435].        -118576.8785832125
1        [ 9.1114829  -9.99975933 -5.06042598  4.24465356].        -90932.40249777345
2        [-9.72135464 -7.69244795 -6.22844507  7.60723681].        -114241.3048574365
3        [-8.34192968  0.21366179 -9.02437947 -5.9293723 ].        -104788.2931595007
4        [ 5.90709134  3.33477276 -6.9423112  -1.38736973].        -27002.15125281818
5        [3.60696134 2.18226175 0.08284273 4.5407231 ].            -6865.378070859648
6        [-8.12476166 -3.94190857 -9.00862896  0.63619536].        -86462.32514334322
7        [-8.87693875 -6.00070012 -7.77180414  6.2427699 ].        -90810.86918314439
8        [ 0.65604441 -6.68523471  7.23179469 -8.11619975].        -114911.9051730723
9        [ 7.28906482 -5.39350739  9.45774561  2.8390882 ].        -107286.8557419656
10       [-8.6173938   7.4519909  -9.59943625 -2.01230926].        -123816.7415286000
11       [-5.02166508 -3.62694175 -6.53971562 -2.35341132].        -27090.90316698181
12       [ 9.32398288 -7.33660406  5.53755865  8.82281307].        -123726.7272112767
```

```
     13       [-0.75891568  3.78629    -1.56788422 -2.47440102].       -2502.500301936376
     14       [ 2.48982037  1.10355433 -7.1409562   7.77830704].       -96218.98123823383
     15       [ 4.82494175 -7.76326919  2.73425909  5.27736897].       -39563.05535409677
     16       [-1.15603278  0.66129511 -0.04265958  3.97446009].       -4017.368442314087
     17       [ 8.87703854  6.66597873 -9.57363046 -2.06637977].       -107780.4218532048
     18       [ 6.09113644  9.30429862  4.22979639 -0.01419842].       -58011.34247774954
     19       [-0.65258029  1.63108103 -1.68979163  5.05365461].       -11262.1449875199
     20       [-6.45270271  8.83022358  0.4849949   9.50856377].       -183102.573222953
     21       [ 7.492922    9.63861344 -0.96382108 -8.02431679].       -131147.827886515
     22       [-2.66640016  7.36309694 -2.52238466  6.03529864].       -47508.46268182049
     23       [-8.28356595 -0.67622285 -4.17175474 -8.6825744 ].       -100065.579863086
     24       [-1.77315193  7.23300882 -3.9636076   7.83188382].       -88558.42289498379
     25       [ 5.93344405  1.47006601 -7.34947524  5.96355466].       -58729.15307304359
     26       [-8.53456332  4.52175024  2.18682304  8.16928641].       -73999.04842053082
     27       [-3.7316997  -2.69333488  4.81507064  2.14991621].       -7987.818259805882
     28       [-2.8223465  -4.36264408 -2.36201774  1.63406436].       -4318.213755288604
     29       [-7.57969205 -4.07295239 -4.47347671  8.24655099].       -88173.376195915
     30       [ 6.13050214  4.79629782 -0.05834002  2.94825545].       -4492.719786855489
     31       [ 9.12435703  8.82496516  4.30754928 -6.42608441].       -69982.18196934072
     32       [-2.96954382 -7.24459566  5.84777328 -1.65496192].       -40480.56945429444
     33       [ 4.57662502 -2.14577864  7.49768929  3.67887838].       -40971.0402451526
     34       [-3.2474913   3.25385506  4.29490232 -1.09832791].       -4619.792502024318
     35       [-0.36547047  0.13954019 -6.58617163  4.378791  ].       -30585.2511636327
     36       [4.7079286  2.50009076 7.71307617 9.70272847].           -171266.224816538
     37       [-0.17463732  5.08167154  0.24746715 -3.0858656 ].       -6859.486256061779
     38       [-8.29169788  6.80690525  0.93514183 -5.7185067 ].       -37173.0037398132
     39       [-1.07167921  7.40842816 -9.34890912  8.21125902].       -191812.695580486
     40       [-1.15604885  5.18351496 -4.25344506 -6.05659408].       -33012.58299521856
```

```
### 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      [ 0.27886688  5.46330104  7.40855371 -9.83906103].       -178393.1355710015
init      [-3.80528149  9.15207479  0.26233425 -3.6343115 ].       -61708.37227853236
init      [ 0.78399875 -5.57490115  6.12962716 -3.15490749].       -27840.0322802925
init      [ 0.77777698 -9.88252429  3.46304956 -5.79951476].       -95425.8534148436
init      [ 8.65115186 -2.51510501  5.04837844  5.26278004].       -18811.8457713201
1         [ 4.8502271  -1.84168373  9.82765998 -0.95725402].       -114361.737155912
2         [-4.77473587 -9.74252485  0.96090492 -9.55982645].       -208414.623132540
3         [-9.92742519 -4.56358619 -0.03785913 -6.46617548].       -33499.4292741171
4         [0.19437843 2.05845053 1.65792191 5.18484457].           -11033.9318563383
5         [-1.31829204 -2.32031879 -2.5251031  -6.82350009].       -37571.24827805165
6         [ 0.24276126 -4.12052041  2.21586778  7.93881754].       -64196.1703940990
7         [ 4.79563633  4.13838411 -7.01002178  7.65374311].       -90010.55097432152
8         [-8.61484726  8.56315525  8.59846951 -0.87156311].       -106727.497541840
9         [-1.22137626  2.83846894  3.6206131   5.92588181].       -19994.1019636413
10        [ 2.54244816 -2.26775385  6.69847887 -4.22542869].       -28884.4702345699
11        [ 5.03921566 -0.10483683 -6.92496957  4.51444489].       -36819.0292724918
12        [ 6.82279714 -0.09161239 -2.028231    8.81142257].       -99320.62563521019
13        [-8.96893842 -9.80111106 -6.18093935 -0.65507665].       -103470.007468530
14        [-6.79730927  6.67022322 -4.17180278 -1.34367333].       -21026.2883415379
```

```
15       [-6.15510499  2.90554086  9.18832867 -8.99547985].       -176933.1346399491
16       [-3.45035456 -3.84880861  6.14592988 -4.05004414].       -23961.0462346427
17       [ 6.31982958 -8.10015369  6.28733636  3.72133219].       -55855.62754418988
18       [-4.1962459   8.74327257  3.07792362 -3.43828096].       -51383.08845376526
19       [ 7.56622239  1.19315477  7.79884993 -2.17352434].       -43623.84267009443
20       [-7.77223206  2.88456515 -5.76179988  4.45343241].       -21625.15435909677
21       [-0.12388008  8.42440271  0.8938253  -5.69235191].       -56844.93004190398
22       [-6.17602376  5.74038283  2.81136066 -3.07356876].       -11781.15259317922
23       [ 5.22498155  9.63662115  0.71281827 -6.55975151].       -94541.1682905092
24       [ 3.16163099 -7.41787886  9.3383186  -7.47424887].       -163972.467945802
25       [-5.41504132 -6.2939085   8.79453795 -5.38270416].       -101778.745451419
26       [-2.19162723 -1.00437391 -6.54627731 -8.64645046].       -120031.956694991
27       [-8.01189683 -0.39981096  2.62732101  7.36608046].       -45677.01194384305
28       [-1.57727485  2.09123977 -0.53319287  7.30830626].       -46327.8955500913
29       [1.349273   6.70863794 1.93835752 6.58155122].           -44417.2615828796
30       [ 6.15573528 -8.96314526 -0.77934299  2.38573627].       -48682.07798919664
31       [ 2.0109895   0.71797124 -0.34174846 -6.35836892].       -26377.0555005158
32       [ 3.66886588 -6.97835601  4.98345433 -8.59838295].       -108863.082329054
33       [ 3.30130157 -6.57503668 -9.3979205   1.10333169].       -115102.061918727
34       [ 8.58796337 -3.84470179 -9.43423321 -1.58187498].       -100984.312188933
35       [-8.21361433  2.64733509 -3.12534377  0.08413083].       -1968.331302837032
36       [ 0.67700616 -9.84109187 -5.33539649 -4.94960676].       -99695.7947886156
37       [ 8.89834037 -5.46605614 -6.42257075 -5.19338335].       -43024.5051668800
38       [ 9.49126672  7.54900651 -5.0903024   1.16455439].       -28029.7177755545
39       [ 0.62477561 -8.91363205 -2.55768355  8.92968243].       -156580.288537099
40       [ 4.24459215 -8.48876313 -3.33299209  0.63828868].       -42037.25228493964
```

```
### 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     [ 6.97635395 -6.4220815  -8.91273571 -2.76923108].       -95759.56429474254
init     [-4.49198143  0.6000045  -3.88162169 -3.91051282].       -7452.834860002497
init     [-7.76517448 -5.00201972  8.35259796 -4.71706293].       -74656.79367914848
init     [ 4.35547375  7.31430068  6.14158964 -5.78898835].       -49831.8948005324
init     [-6.65513937 -9.06587217 -9.21155376 -5.9953838 ].       -180751.6072602438
1        [ 2.8544977  -1.6003636  -1.93515132 -2.19159348].       -794.416570303183
2        [4.92526955 5.96067446 4.10838652 1.54460964].           -11082.60249594799
3        [ 7.60284646 -9.83670929  6.07418637  6.51677002].       -115033.369674890
4        [-3.39221728 -3.69203211  3.60318038  4.66933918].       -10938.1710697176
5        [ 9.63144016  3.76660236 -9.40395266  2.32534909].       -92305.21346503733
6        [-0.46099561  9.58086516 -0.66748888 -7.87674906].       -130229.313894340
7        [3.72122725 0.49817372 5.20397899 9.4368106 ].           -128249.405728734
8        [-7.21002719  4.48364211 -4.95008418 -7.56274535].       -67479.32735563519
9        [-0.12944698 -4.53817618  4.90890952  4.67662515].       -17790.1045786644
10       [-0.90690653  4.09601032  6.89554636 -6.11770942].       -45695.15188495538
11       [ 2.80129854 -5.37735112 -4.98205843  0.05319293].       -15240.79344667239
12       [-5.20285545  0.11177617 -7.97281939  3.50515359].       -52732.03172874354
13       [-2.43889404  8.05457271  2.80173527 -8.85012489].       -129812.6145843062
14       [-5.29773051  4.13262973  2.40625612 -8.86801578].       -99267.17779025543
15       [ 5.0359643  -1.51971436  2.2276439  -6.68211888].       -30736.9407608207
16       [-6.14779635 -0.43746793  3.56596824 -2.19771089].       -2292.657920210773
```

```
17       [ 1.30223912 -8.21616194  4.9325588   9.93275327].        -193511.1880846855
18       [-3.38741217 -3.11731418  9.47034559 -7.63348082].        -146826.9102801833
19       [ 1.14120333  4.18919902 -5.55151656 -7.17257482].        -59247.26899808023
20       [-9.82307368  4.09585192 -0.07682717 -0.82564631].        -3938.269130835627
21       [ 2.37201846  9.77107763 -3.75463112 -4.47165852].        -79796.77105889232
22       [-9.97494156  6.50361239 -9.02849993  6.38393647].        -124294.6613468436
23       [-7.06812114 -9.87155199  9.61231195 -4.09498276].        -197616.0306430343
24       [-8.30626002  7.94244324 -7.62887706  7.94118769].        -143100.6796446362
25       [-1.17724002  6.52233722  8.70454021 -9.0579384 ].        -174556.3023549198
26       [ 1.90180821 -6.73740977 -2.2149103  -3.83996585].        -20643.98356805410
27       [ 5.60066467 -3.87589042 -5.27284941 -7.26059882].        -60853.24367998289
28       [-6.99724941 -9.4660758  -9.53790685 -5.4842791 ].        -198753.4934411679
29       [ 5.06996404 -5.67116173  8.30667438  4.29567671].        -72235.94884794042
30       [8.36219672 9.80306436 9.96457014 1.990924  ].            -174579.7954762038
31       [ 6.2784228  -7.31871577  0.31170191  8.68142997].        -111046.1667329579
32       [ 3.97139572 -0.78817608 -6.48761029  2.35482403].        -22917.5104954821
33       [-3.65326945  4.62365068 -0.23931856 -5.75842037].        -22110.28176039971
34       [-9.4004366  -8.92783557 -3.57895325  4.60637687].        -69154.01180968466
35       [5.2149277  1.50219974 7.87183203 2.58286734].            -45105.24456577214
36       [-6.92125004  8.09480936 -7.27610869 -0.63501244].        -67085.12006740912
37       [-2.36535354  6.69953474 -9.97868456  3.88382463].        -134544.9193827431
38       [-9.98712406  7.81056755 -6.70440634  6.67862108].        -91978.99245763329
39       [ 8.55061527 -5.65265524 -0.11705234 -5.94541258].        -26339.89687181307
40       [-6.14289032 -9.00474832 -6.14296401 -3.02096022].        -80499.6473401903
```

```
### 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     [-5.53417842  0.46326683  1.01402913 -9.087961  ].        -107926.0181001286
init     [-2.78542329 -5.53838117  3.77452324 -6.7253715 ].        -41773.31935362981
init     [-8.59350266  8.8202172   1.2736276  -8.44015321].        -133847.7142986104
init     [ 4.45281022 -6.83095653 -4.99437387 -4.13025488].        -31578.15352127590
init     [ 3.93221428 -0.71471824 -5.69875713 -1.06347476].        -13215.25448314987
1        [-0.64007705  3.2211867   3.79933264 -4.73341031].        -9619.115103084405
2        [ 5.98502725 -7.94806028 -2.90882686  9.11344119].        -145121.6736216408
3        [ 0.47384394 -7.09622115 -6.44235327  3.14296454].        -47197.66759876199
4        [ 5.52616367  4.65190076  3.38866017 -9.07435877].        -107947.3422720706
5        [ 4.28544997 -2.78264932  9.08402216  2.69441349].        -84872.13346649229
6        [1.87133514 5.63089801 3.59434817 4.26424392].            -13097.15922844718
7        [7.45907335 5.14332641 9.11613095 8.2671896 ].            -147101.3184183988
8        [ 1.24349389  9.55943585 -9.57929245  8.27200487].        -242462.0445814614
9        [-3.08885414  6.03421433  8.15398859  0.37041672].        -60132.37051845459
10       [-1.22767225  3.39992625 -0.74204049 -9.66914731].        -142171.8305547916
11       [ 8.89995329 -8.45639035  4.20210818  2.73243867].        -42247.83623850960
12       [-3.05457595  8.6513061  -8.42526724 -0.14730159].        -100287.0695816375
13       [ 6.62003549 -4.05866358 -4.45226831 -1.87309798].        -7673.952612365588
14       [2.62583757 0.46449641 0.44128596 2.38948669].            -494.3569330727978
15       [ 9.53262449 -4.26198748 -2.34360629  3.98226383].        -6846.774114023887
16       [ 1.37015504 -5.39192653 -6.37692601 -2.95442377].        -31281.61212273921
17       [ 9.57679297  4.69037918  0.54002543 -2.39908886].        -2975.458131358822
18       [-4.12908744 -6.44608013 -9.57829988  9.62751237].        -275501.78703328
```

```
19       [ 4.39317174 -9.17548236 -7.80685399  8.14414445].       -184248.9712183503
20       [-8.09573386 -0.93369103  8.83705044  0.93227014].       -74538.58726244236
21       [ 6.90566575 -9.11487258 -7.09427085  5.69761637].       -107657.2244470814
22       [ 7.51003771  0.82133786 -8.23718568 -8.4246618 ].       -144921.6269479681
23       [-7.84947382  1.82830735 -5.97591025  9.61433411].       -160721.0352024896
24       [ 9.48449776  6.83252536  5.34568415 -0.39785071].       -21841.9911189496
25       [-1.84073522 -5.78419579 -3.31676836  9.58628876].       -151820.2110251825
26       [ 3.71569124 -3.68849756 -8.24504839  9.60750967].       -208387.7705449479
27       [ 2.76396001  3.74026705 -3.78256436 -9.60942869].       -145207.001623198
28       [-1.65137032  7.33727682  7.08169934  6.97408568].       -82376.49029248668
29       [ 5.50320058 -9.13160875 -0.88628097 -1.61042153].       -52527.06079935218
30       [ 3.69840767  4.90939092  3.68410322 -7.00644089].       -41170.3387936889
31       [-7.32395661  0.55758749  9.64375706  4.66874035].       -107977.336183553
32       [-7.7492907  -0.54323383 -1.51122391 -7.4057133 ].       -49756.1246962349
33       [ 5.71022868 -8.15408013  5.60144666 -7.0087603 ].       -81831.21082517967
34       [ 8.11852922 -9.32709753 -2.42268411  4.28621517].       -62544.23612123542
35       [ 5.48023944 -0.42882921 -4.05769248 -6.47321623].       -34290.17931873099
36       [ 2.98657565  6.317717   -9.76871974 -5.96968596].       -140247.6421202972
37       [-4.32766021 -8.23561819 -7.16295742 -8.59650544].       -172134.3585690773
38       [ 5.45605189 -3.49800546 -1.82105288  4.02505857].       -5735.877064089329
39       [-3.88161067  4.74292144 -7.74311013 -7.05113894].       -90542.63574517332
40       [-2.15864174  9.72631791  0.77984478  4.7821444 ].       -81550.30066083447
```

```
### 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     [-4.10669995  0.61173511 -6.16958426 -8.64199284].       -113948.9939325919
init     [5.7397092  3.12667044 2.75041792 1.51205788].           -849.5238302802878
init     [-9.21874168 -2.84372791  8.91366374 -8.79910639].       -165071.6247490734
init     [ 7.28084207  7.54581052 -8.97612669  3.04837231].       -96579.16352863169
init     [ 1.03502737  1.95026506 -0.32942751 -4.34023678].       -5872.837304694351
1        [-6.82082511  3.53524771 -7.6305547  -1.10007868].       -40739.58096344288
2        [ 6.36132887 -5.79483735 -1.80465394 -5.23795116].       -20725.35045584762
3        [-9.93358214e+00  6.76426027e-04 -8.86041968e-01 -1.86772363e+00].       -5
4        [-3.57261465 -5.71224609 -1.88427713  4.42421021].       -16723.5217284345
5        [-6.06192646 -5.22204103 -5.99548203 -7.48165956].       -80879.44433220002
6        [-9.15358294  6.24405427 -7.34906493 -3.27748646].       -49685.22640168171
7        [-8.32522503  1.10793827 -8.79126558  8.08570378].       -148868.7982529325
8        [ 9.08506073 -7.59845958  3.69354279 -6.90608989].       -59984.3042023997
9        [-1.16236813 -7.73189354  3.93657692  3.54340093].       -35446.964834388
10       [-9.19308579 -8.75896264 -4.61339917  6.673686  ].       -96016.16894455833
11       [-3.55419883 -1.3121061   6.26693875  4.48480965].       -23865.23094014127
12       [ 2.54749487 -9.29639909  6.99658479 -6.71299771].       -120125.4164780967
13       [ 3.37722984  0.46032172 -5.15280505  5.104616  ].       -21454.83186816024
14       [-9.67252227 -8.84826579  4.65918096  0.87645251].       -63629.75766545245
15       [ 4.34389407  6.23064858  6.70660755 -5.49281777].       -43293.8482378960
16       [-3.31878925 -7.0607148   8.23210147  8.97015571].       -175511.4432515234
17       [-4.02064857 -8.99836262  1.60115438  8.52912304].       -138528.5580699571
18       [ 8.85510862  4.75071066  1.0351052  -1.66500369].       -2796.41214289730
19       [6.20008629 7.73399648 6.15686017 0.04789374].           -39815.46374019333
20       [-4.9003479  -9.03322285  6.16587778  1.19907682].       -78302.75020896044
```

```
21       [-3.9804138   5.24617004  2.12715187  4.24226727].       -11623.86852012758
22       [-6.17626749 -8.46700681  8.99586277 -2.9797324 ].       -132117.7367627571
23       [ 2.2532422  -6.07277145 -1.50076647 -5.72195561].       -28509.34209412592
24       [ 4.47269207 -2.43605748  1.21501839 -0.91049035].       -209.3588064773532
25       [-3.67672456  3.25386039  0.52443989  6.72958781].       -33715.24135027962
26       [-0.3244946   6.36986955 -8.03561878  5.93095993].       -83076.16665179531
27       [ 6.35510707  3.32105732 -8.75281378 -2.10839642].       -69179.28723819098
28       [-2.91977083  6.85050341 -0.67534542 -7.76919063].       -77801.28372380593
29       [-0.17344573 -4.6667978  -4.02317816 -1.15895835].       -8121.929303327734
30       [-3.85451606  2.65749216  2.16933315 -9.66637665].       -137275.6156452218
31       [-7.5840042  -5.22426666  0.16639832 -5.57743579].       -23287.91277881447
32       [-3.51762081 -5.5603206   0.04416167 -4.59494337].       -15772.61607592267
33       [-4.83988631  3.65652134  2.55147026 -5.27952282].       -13610.68384395733
34       [3.6433799  0.50243261 0.83772047 3.3394581 ].           -1872.321792921562
35       [ 8.84769313  0.53124068 -9.84379667 -7.08877703].       -160962.3785205168
36       [ 2.56432668  6.10803492 -0.24395397  0.04946162].       -10493.23892608360
37       [ 3.96142624 -7.37930587 -0.01775028  7.07304132].       -62258.74075876109
38       [-2.13418732 -1.3388094  -4.55084815 -1.1407694 ].       -5764.990442989307
39       [-4.30988812 -2.17672448 -8.53201855  1.05532615].       -66376.99165116323
40       [ 8.72108438  4.65068098 -6.50045303  3.7493288 ].       -26371.45099567085
```

```
### 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     [ 3.00748483  0.10906747  7.57202942 -6.36319549].       -60950.54433734527
init     [7.04466137 5.00272572 3.32203335 9.75790897].           -144653.2121556773
init     [-4.86063155 -9.43388149  2.71438231  6.94624775].       -103843.6354666777
init     [ 4.7234925  -9.58385776 -7.76793739 -4.04552516].       -121548.0225107028
init     [ 3.73940383  7.23252112 -6.02731282  3.14378061].       -35863.1928389845
1        [-6.1664519   4.28513481  2.79740714  0.66407663].       -4132.892348949624
2        [ 5.61099687  3.24280582 -9.3946128  -2.53775904].       -92553.92031901842
3        [ 3.30844576  7.22382341 -9.24562554  8.22016932].       -184254.3962213914
4        [ 8.03656335 -7.89614249  7.70734605  1.23688198].       -75516.33414231456
5        [ 9.562822   -7.89156436  1.83282657  5.62935728].       -42311.3181835149
6        [-5.31480829 -4.60774758 -2.97712464 -7.93522102].       -72575.53761772846
7        [ 4.54666061  8.69770438  0.01712558 -6.54510361].       -72662.9494514385
8        [ 2.31421647 -2.51006802  8.67058127  9.73721889].       -201299.9636474805
9        [ 8.39884901 -4.60470488 -7.7957923   9.97387584].       -221097.7592641949
10       [ 8.47467859 -1.0716028   7.04016464 -4.60023784].       -35231.89169888245
11       [-4.41945067  2.73837136  0.20384538  5.36041706].       -13921.34517699476
12       [-0.50618251  2.84808342  2.56143498 -7.45357454].       -48011.17126053040
13       [-8.79601456 -6.39611748  6.71849908 -7.75307177].       -96086.06277058301
14       [-5.37332728  7.94915331  0.37919817 -7.51567844].       -85641.09865300173
15       [-6.68510816 -6.15755534 -5.39539851 -2.06645774].       -26887.36562033140
16       [ 6.58567988 -3.25808927 -0.548504   -0.50180546].       -509.2511790415120
17       [-9.953162   -4.99542568 -2.13845079 -6.90241283].       -45853.32370279803
18       [ 4.51256909 -6.23379347  7.47441333  1.86633096].       -52481.12747656245
19       [-4.43773509 -5.27206334  8.92089553 -3.48332795].       -89296.490364151
20       [9.38012521 4.17126213 8.55885628 8.58114875].           -139107.3242913438
21       [-3.95267274  7.48115078 -5.84508201 -7.53019976].       -94877.29388338875
22       [-3.79269379 -4.68807322 -6.02444214 -3.60712623].       -26605.62398079398
```

```
23       [-1.48557132 -6.91241224  5.24004516 -6.8972013 ].       -62641.29701724463
24       [-6.69867140e+00 -9.34293864e+00  9.22919429e+00 -1.76365303e-04].       -1
25       [ 5.06070396  1.82378818 -6.22573519 -3.71986079].       -21808.07055295088
26       [2.36700321 7.98380377 2.34188126 1.3901976 ].           -31345.53039273366
27       [-5.43407955 -0.05634127  2.95176781  1.99797455].       -1118.762230872074
28       [-3.83822248 -8.47380926  1.10967586  1.83939291].       -43992.72279207087
29       [-1.37315623 -5.1962267  -3.57504256  5.0488736 ].       -20882.51853552796
30       [ 8.07717593  2.28889096 -5.63953652 -6.0170719 ].       -35709.1012182352
31       [ 2.22467047 -6.27762742  3.37822164  3.65453913].       -16452.91458865641
32       [-6.30146814 -4.97194609 -5.29613136  3.6201607 ].       -21427.09418541821
33       [-0.21700463 -8.29617986 -3.95647696  1.04002189].       -42873.00957153866
34       [ 6.61137674 -9.22930759 -9.25673748 -0.33719932].       -151871.7352455236
35       [-9.55306477  4.89288691  5.82252732  4.20207291].       -22061.1467115303
36       [-6.36992416 -8.43466409 -8.46599827 -3.31032615].       -117056.8774180467
37       [7.36117624 9.27028018 0.60302011 9.85396042].           -204310.6154617963
38       [-1.23888407  0.65768954  9.33361935  7.44558348].       -131640.2575456100
39       [-4.93743541  7.64415695 -5.25569901 -2.10011618].       -37297.5977569819
40       [-3.1771503   5.37157347  0.27879879 -7.8322229 ].       -67448.47153656623
```

```
### 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     [-8.04932797  5.22499433 -5.06124054 -7.23736625].       -62523.95884648901
init     [-3.37106873 -8.3400087   3.43954163  6.13187596].       -64291.98059909556
init     [ 9.65483829  2.7132147  -5.68153488  0.98054864].       -11832.54772826069
init     [ 9.11199160e-01 -5.31847854e+00 -7.72548314e+00 -6.81465902e-03].       -5
init     [-6.95783155  0.65372161 -2.25986463  3.76654767].       -4213.270400555922
1        [-6.5477408   0.83865275 -8.96461914  6.51115227].       -112037.9546475548
2        [-2.30389925 -5.27595204 -9.70176571 -1.31398656].       -119780.5356061062
3        [ 4.66969615 -0.45575046 -6.64393505 -5.40706888].       -40634.50938526745
4        [-9.13977787 -5.66464753  4.94279629  8.45310839].       -95912.86613447932
5        [-4.03600104  9.52409788  4.77638865 -9.1959919 ].       -180773.1012779256
6        [-4.83369566 -2.4386249   4.54159233  6.11885823].       -26110.73895282561
7        [-1.74139105 -2.25943387 -4.66754333 -7.46934292].       -60650.71086015194
8        [ 3.38990203 -3.49236324 -8.09259704  9.98332142].       -227238.8418316005
9        [ 0.48549986 -6.62909349  6.4319956  -6.53556085].       -64201.3007874265
10       [-0.99362165  7.76510241 -3.3125504  -8.02417682].       -99962.4878729243
11       [ 1.66645895 -8.36624855 -7.84433287  7.99584163].       -163760.6843129029
12       [-5.10416772  3.72102212 -2.58654699  5.58265898].       -19326.10864838541
13       [ 2.86565519 -0.33761145  0.96097259 -4.59742072].       -6858.299462594941
14       [-9.56069024  1.08179853 -2.00683023  4.38882877].       -7111.548749259283
15       [ 2.34954077 -3.13748972  3.15811446 -8.40909356].       -78673.48727660593
16       [-0.58070913  1.68475454  1.86922719  4.30763163].       -5133.222751366833
17       [2.41764449 1.87228964 5.94389184 3.41628568].           -15450.10891765153
18       [ 9.11708637 -4.55121628  1.89198796  7.70152498].       -57073.3981043833
19       [-2.30605317 -5.20595438 -0.16009014  4.3386911 ].       -12198.55156824516
20       [ 9.51568311 -8.82578504 -2.38869497 -2.78357922].       -45372.47140173492
21       [ 1.40856514 -0.68378535 -2.05805581  4.56833676].       -7924.924850958267
22       [ 9.47335062 -2.82596526 -9.98254377 -9.14656194].       -248467.4666931006
23       [ 2.51530768 -4.62186548  9.97645257  4.84711249].       -133173.3165602473
24       [-6.97129621  0.24238742  5.42427763  6.34874102].       -33080.43574516348
```

```
25       [-4.0793351   1.80214266 -3.34734211 -6.24787259].        -28039.75056109067
26       [-9.42856911  8.98664152  3.80558549 -5.85683224].        -76548.44636499413
27       [ 4.54762806 -0.20468487 -1.40689689 -3.80297829].        -3784.604786435664
28       [ 8.01410845  2.35469457 -6.26141828  6.51240303].        -50606.99507745763
29       [-1.35226567 -4.29413716  6.11966642 -7.64814281].        -70911.56918684061
30       [-3.1156522   0.15294593 -2.91082318 -3.69003264].        -4517.462324540883
31       [-0.25259458  3.44934982  5.3577388  -1.83043753].        -9900.95065364107
32       [ 3.48506606  9.86842239  6.83677971 -2.25957315].        -94206.99163811968
33       [5.71950689 3.41725053 1.95649279 2.58774267].            -1221.362042840305
34       [ 7.2052829   0.12871334 -1.41519373 -3.82625005].        -3955.347207256568
35       [-9.87217663 -3.42357815  6.86830815 -0.41111743].        -31185.86267222706
36       [ 7.7670299   5.6968523  -1.97128227 -9.1879163 ].        -123288.9292800704
37       [ 1.74657984 -8.41100693  0.75411934 -2.6998782 ].        -40095.03661508263
38       [ 1.81559099 -7.1068488   3.58237457 -7.22161856].        -63482.91101321683
39       [-6.0350497  -0.46255327  3.66257593  1.55122853].        -2372.796398631036
40       [ 5.01697203 -9.11312334 -3.14545699 -9.55509343].        -192421.0792605565
```

```
### EXACT GP EI GRADIENTS

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

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

```

```
Evaluation       Proposed point            Current eval.          Best eval.
init     [1.76261602 7.95427456 7.83061459 6.31674955].            -91316.16764460134
init     [-9.28220829  3.83515164 -2.42638116  0.37021891].        -3319.224464998000
init     [ 3.15902931 -6.12299564 -4.55367196  4.37211867].        -24439.65199379687
init     [ 5.66007219  7.0065528   5.50489788 -9.26671387].        -136303.9099361228
init     [-7.6661253   5.02561399 -5.21563568 -4.90387972].        -25528.59564418783
1        [-0.38031735 -3.41587185  0.21282112 -4.72742343].        -9079.144305704322
2        [ 2.2515589  -8.62950692  7.2874785   9.15451767].        -185415.9807778578
3        [-2.00279263 -5.09620423  0.51102349  6.68881567].        -37586.20721437270
4        [-3.33679846  5.09475498 -0.14129379 -8.38664837].        -85513.79864179283
5        [-3.75442852  7.02008144  4.58355261 -6.10855163].        -44260.24645158661
6        [ 4.26761042  1.24919415 -4.83066711 -6.15878884].        -32247.45650934417
7        [ 2.6185745   0.28739637 -8.78495862 -6.04256041].        -97992.96104999761
8        [ 7.25959653  3.0895583  -0.80389347  9.92972863].        -157149.4398753884
9        [-0.16988855 -8.40538175  0.41458404 -7.27792385].        -84798.34386339581
10       [-3.49997926  0.57776952  2.16068958 -2.13900545].        -480.6088906930014
11       [ 4.5495202  -8.7912022  -5.43120396  4.47127488].        -67060.84679719032
12       [-0.70786657 -9.22478058 -6.0640182   9.9771379 ].        -247316.6028277325
13       [ 8.67220664  4.88808428  2.02305877 -4.03008287].        -6862.640262287456
14       [ 9.52899347  6.79711983  6.56641001 -4.90357522].        -39636.73363245442
15       [ 1.71798379  1.88011029  1.20564189 -7.2953826 ].        -44362.43065982318
16       [ 9.55666698  7.58178032  2.06918445 -5.98411054].        -41647.48666171433
17       [-9.50037711 -9.91987748  2.91255554  7.95992373].        -148718.9421409524
18       [-2.25218145  6.59461508 -1.33537048  0.21013843].        -15970.20748835903
19       [-1.46332207 -1.76668806 -2.76606375  9.63548465].        -143054.2742251565
20       [-1.63212907 -1.90877137  7.15298389 -3.13853811].        -33393.20604746404
21       [4.54765595 2.9735209  2.57874967 2.57833034].            -1136.998357015914
22       [ 6.06549104  8.60733958 -3.56990525 -2.40980586].        -42191.6730902873
23       [ 1.22203319  5.55870232 -3.75390246  0.12880885].        -8932.049557196993
24       [-5.60837187  6.92296066 -0.01608257  6.72439001].        -53502.44803785327
25       [6.91222192 4.2742851  8.67083525 4.66313836].            -70668.90861369018
26       [ 1.83295647  4.07927113 -9.98664886 -7.75277291].        -184312.6124842821
```

```
27      [-2.2425647  -9.23164718  2.8552998  -4.13191458].       -65526.34699296131
28      [ 5.57263302 -8.85636713 -2.01901426  5.58267701].       -63235.42678761073
29      [3.11294633 6.7952711  0.15491125 1.93098803].           -16281.35870452535
30      [-9.14044254 -3.87106781 -4.47306772 -1.71281667].       -9368.24229317686
31      [-0.23811104  1.99927023  6.38129647  0.93804175].       -19155.89405781995
32      [-5.10325813 -0.90245399  4.7361361  -5.68753312].       -20791.81255127767
33      [ 1.17277156 -0.55696559 -0.11362563  0.94183043].       -15.91282206586385
34      [-3.64074181  6.79317563 -9.91667555  5.01593159].       -141114.1325824055
35      [-7.87299021  1.54599425  8.40784034  6.40641952].       -80775.5242096358
36      [ 8.45138073 -4.14458111  9.01765328 -3.81340398].       -86455.60628745613
37      [-4.82118187 -9.5810585  -9.04143076 -6.29416261].       -192069.3490494220
38      [-8.22281743 -8.12367492  9.37205505  1.0375603 ].       -140953.0301692480
39      [ 8.59741241 -4.97476183 -9.43840164 -1.25458632].       -104659.4115477620
40      [ 6.73402828  8.01852006 -9.08410083  6.16042668].       -132591.5202178472
```

```
end_exact = time.time()
end_exact

time_exact = end_exact - start_exact
time_exact
```

```
224.3106837272644
```

```
### 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
```

```
(8.624304580680482, 6.818115605285224)
```

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

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

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

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

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

```

min_simple_regret_approx_2, min_simple_regret_exact_2
```

(5.777449072618791, 7.229826449156426)

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

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

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

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

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

min_simple_regret_approx_3, min_simple_regret_exact_3
```

(6.413257621837379, 6.413257621837379)

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

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

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

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

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

min_simple_regret_approx_4, min_simple_regret_exact_4
```

(6.6142410478018, 6.6142410478018)

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

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

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

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

min_simple_regret_approx_5 = min(simple_regret_approx_5)
```

```
min_simple_regret_exact_5 = min(simple_regret_exact_5)

min_simple_regret_approx_5, min_simple_regret_exact_5
```

    (8.233374344863629, 8.233374344863629)

```
### 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
```

    (6.163861138038082, 7.087810821495902)

```
### 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
```

    (8.706151524348773, 7.788844100022022)

```
### 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
```

```
(6.8971996351568015, 5.868902328763247)
```

```
### 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
```

```
(7.381112326420343, 7.772516063330129)
```

```
### 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
```

```
(7.118249904126745, 7.118249904126745)
```

```
### 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
```

```
(4.079916196415566, 4.079916196415566)
```

```
### 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
```

```
(6.817956049900704, 6.029180309616237)
```

```
### 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
```

```
(8.480741292954773, 7.825045631843265)
```

```
### 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
```

    (8.784238895138476, 7.5849414083812965)

```
### 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
```

    (7.73746708734624, 6.677607971404637)

```
### 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
```

    (6.2032577928686425, 6.2032577928686425)

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

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

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

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

```

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

min_simple_regret_approx_17, min_simple_regret_exact_17
```

    (6.348406365668042, 5.344049557390553)

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

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

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

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

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

min_simple_regret_approx_18, min_simple_regret_exact_18
```

    (3.3781694080673623, 6.2329413703551175)

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

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

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

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

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

min_simple_regret_approx_19, min_simple_regret_exact_19
```

    (8.28282366579389, 7.107721943542023)

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

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

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

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

```

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

min_simple_regret_approx_20, min_simple_regret_exact_20
```

```
(6.175053822284406, 2.767125203481616)
```

```
# Iteration1 :

slice1 = 0

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

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

```

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

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

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

```
# Iteration11 :

slice11 = 10

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

exact11 = [simple_regret_exact_1[slice11],
       simple_regret_exact_2[slice11],
       simple_regret_exact_3[slice11],
       simple_regret_exact_4[slice11],
       simple_regret_exact_5[slice11],
       simple_regret_exact_6[slice11],
       simple_regret_exact_7[slice11],
       simple_regret_exact_8[slice11],
       simple_regret_exact_9[slice11],
       simple_regret_exact_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_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 :
```

```
# iterations33 :

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_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_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_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]
```

```
upper_approx37 = np.asarray(approx37_results[14:15][0])[0]

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

```
# Iteration8 :

slice8 = 7

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

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

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

```

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

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

```
# Iteration18 :

slice18 = 17

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

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

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

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

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

```
# Iteration28 :

slice28 = 27

approx28 = [simple_regret_approx_1[slice28],
       simple_regret_approx_2[slice28],
       simple_regret_approx_3[slice28],
       simple_regret_approx_4[slice28],
       simple_regret_approx_5[slice28],
       simple_regret_approx_6[slice28],
       simple_regret_approx_7[slice28],
       simple_regret_approx_8[slice28],
       simple_regret_approx_9[slice28],
       simple_regret_approx_10[slice28],
       simple_regret_approx_11[slice28],
       simple_regret_approx_12[slice28],
       simple_regret_approx_13[slice28],
       simple_regret_approx_14[slice28],
       simple_regret_approx_15[slice28],
       simple_regret_approx_16[slice28],
       simple_regret_approx_17[slice28],
       simple_regret_approx_18[slice28],
       simple_regret_approx_19[slice28],
       simple_regret_approx_20[slice28]]

exact28 = [simple_regret_exact_1[slice28],
       simple_regret_exact_2[slice28],
       simple_regret_exact_3[slice28],
       simple_regret_exact_4[slice28],
       simple_regret_exact_5[slice28],
       simple_regret_exact_6[slice28],
       simple_regret_exact_7[slice28],
       simple_regret_exact_8[slice28],
       simple_regret_exact_9[slice28],
       simple_regret_exact_10[slice28],
       simple_regret_exact_11[slice28],
       simple_regret_exact_12[slice28],
       simple_regret_exact_13[slice28],
       simple_regret_exact_14[slice28],
       simple_regret_exact_15[slice28],
       simple_regret_exact_16[slice28],
       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]
```

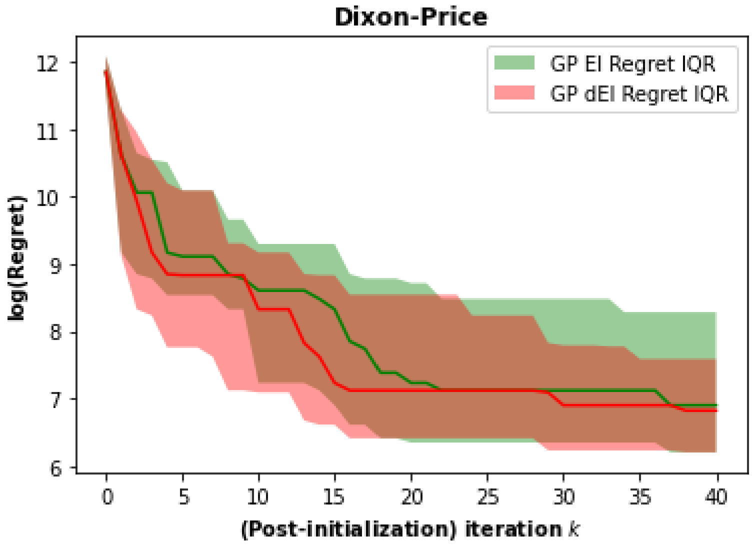
```
### Summarize arrays: 'exact'

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 = 'Dixon-Price'

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
```

```
(1312.7767605781555, 224.3106837272644)
```

```
1
```