Sum Squares:

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 - Sum Squares(x) 4-D:

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

```
                            + 3 * x3_training ** 2
                            + 4 * x4_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
```

```
14
15         v = solve(self.L, Kstar.T)
16         dv = solve(self.L, dKstar.T)
17
18         ds = -2 * np.diag(np.dot(dv.T, v))
19         dm = np.dot(dKstar, self.alpha)
20         return ds, dm
21
```

```
 1 class Acquisition_new(Acquisition):
 2     def __init__(self, mode, eps=1e-08, **params):
 3
 4         self.params = params
 5         self.eps = eps
 6
 7         mode_dict = {
 8             'dEI_GP': self.dEI_GP
 9         }
10
11         self.f = mode_dict[mode]
12
13     def dEI_GP(self, tau, mean, std, ds, dm):
14         gamma = (mean - tau - self.eps) / (std + self.eps)
15         gamma_h = (mean - tau) / (std + self.eps)
16         dsdx = ds / (2 * (std + self.eps))
17         dmdx = (dm - gamma * dsdx) / (std + self.eps)
18
19         f = (std + self.eps) * (gamma * norm.cdf(gamma) + norm.pdf(gamma))
20         df1 = f / (std + self.eps) * dsdx
21         df2 = (std + self.eps) * norm.cdf(gamma) * dmdx
22         df = df1 + df2
23
24         df_arr = []
25
26         for j in range(0, dim):
27           df_arr.append([df])
28         return f, np.asarray(df_arr).transpose()
29
30     def d_eval(self, tau, mean, std, ds, dm):
31
32         return self.f(tau, mean, std, ds, dm, **self.params)
33
```

```
 1 ## dGPGO:
 2
 3 class dGPGO(GPGO):
 4     n_start = n_start_AcqFunc
 5     eps = 1e-08
 6
 7     def d_optimizeAcq(self, method='L-BFGS-B', n_start=n_start_AcqFunc):
 8         start_points_dict = [self._sampleParam() for i in range(n_start)]
 9         start_points_arr = np.array([list(s.values())
10                                      for s in start_points_dict])
11         x_best = np.empty((n_start, len(self.parameter_key)))
```

```
        x_best = np.empty((n_start, len(self.parameter_key)))
        f_best = np.empty((n_start,))
        opt = Parallel(n_jobs=self.n_jobs)(delayed(minimize)(self.acqfunc,
                                                             x0=start_point,
                                                             method=method,
                                                             jac = True,
                                                             bounds=self.parameter_
                                           start_points_arr)
        x_best = np.array([res.x for res in opt])
        f_best = np.array([np.atleast_1d(res.fun)[0] for res in opt])

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

        return x_best, f_best

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

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

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

        return -f, df

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

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

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

```

```
###Reproducible set-seeds:

run_num_1 = 1
run_num_2 = 2
run_num_3 = 3
run_num_4 = 4
```

```
run_num_5 = 5
run_num_6 = 6
run_num_7 = 7
run_num_8 = 8
run_num_9 = 9
run_num_10 = 10
run_num_11 = 11
run_num_12 = 12
run_num_13 = 13
run_num_14 = 14
run_num_15 = 15
run_num_16 = 16
run_num_17 = 17
run_num_18 = 18
run_num_19 = 19
run_num_20 = 20

```

```
start_approx = time.time()
start_approx

```

```
1623403889.2482922
```

```
### 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].      -403.9670698747264
init     [-7.06488218 -8.1532281  -6.27479577 -3.08878546].      -339.1443857291230
init     [-2.06465052  0.77633468 -1.61610971  3.70439001].      -68.19362594821958
init     [-5.91095501  7.56234873 -9.45224814  3.4093502 ].      -463.8472853559696
init     [-1.65390395  1.17379657 -7.19226123 -6.03797022].      -306.5051972029273
1        [-3.68968738  3.73001855  6.69251344 -9.63423445].      -547.0829721238734
2        [-6.24736654  2.44991805  8.11618992  9.79910357].      -632.7411249157192
3        [ 5.02242081  1.58721081  8.49408363 -8.70520033].      -549.8336084521746
4        [3.79278798 1.97641781 4.93314944 0.83942124].          -98.02409771907602
5        [ 8.61288784 -4.84214031  8.68890985  2.38239267].      -370.2691250098800
6        [ 7.62336637 -7.47843039  2.66921668 -4.44574535].      -270.4023170641875
7        [ 8.34520318 -6.98520682 -5.12472045  0.93369144].      -249.5040426908201
8        [ 5.11485712  8.8169484   2.24644677 -3.69207642].      -251.3042040385079
9        [ 6.88781365  2.44433273 -5.87040082  3.82976847].      -221.4448252686258
10       [-2.3791125   7.04806575  3.94693079  4.26519795].      -224.5130803139234
11       [ 0.21574837 -9.72295851 -1.09025783  9.93401343].      -587.4228691963074
12       [-8.46531076 -4.36418907  0.71237859  0.71571197].      -113.3252029655746
13       [ 0.68336005 -8.89855382  2.4019798   1.62350406].      -186.6870836820792
14       [ 0.80119154 -9.16723906 -5.2983827  -9.59120497].      -620.9018807473929
15       [-9.61154737 -0.8945556  -5.6169918   6.14430992].      -339.6442704931166
16       [ -5.31158516  10.         -10.         -10.        ].          -928.212936
17       [-7.81107405 -7.80690855  8.24125081 -3.51817053].      -436.1732598542958
```

```
18      [ 6.03011227 -4.93770333 -9.36205475  9.0561959 ].      -676.1270265902582
19      [-0.29068499 -9.04011931 -4.02183401  4.43267673].      -290.65195017530436
20      [-10. -10.  10.  10.].          -1000.0        -68.19362594821958
21      [ 3.67388287 -2.87543511  1.9187423   6.41420845].      -205.64666573416594
22      [ 3.448531   -6.48781134  9.99507595  9.2886456 ].      -740.8961362069978
23      [ 8.93914697  2.22501625 -5.96404096 -4.66426372].      -283.5405209992468
24      [ 9.23917112 10.          8.32776287  7.64014102].      -726.9042053363082
25      [-4.67489351 -4.57828696  2.94378358 -9.19489192].      -427.95778737876265
26      [-0.77290578 -3.36889522  7.86252834 -5.82002822].      -344.2452631104227
27      [ 3.58017019 -3.83060401 -9.33431749 -1.38245855].      -311.19788820183504
28      [ 10.         -10.          10.          -8.41625213].          -883.333195
29      [-9.45430148  6.57554623  2.12369659 -3.97863695].      -252.70790245553422
30      [ 4.26165255  9.82255076 -5.30046477  5.42295688].      -413.0453148762932
31      [ 4.83821015  9.90305218 -9.36128664 -9.34724668].      -831.934306733533
32      [-8.3338311  -7.55164056  6.9573681   4.55310386].      -411.6452231486886
33      [8.4505224  1.05689488 9.5999994  9.38096037].          -702.1350176248097
34      [-7.7198349   5.41016528 -0.49950198  9.58059297].      -486.0351810567215
35      [ 8.70670289  5.22400553  9.63731482 -2.01696272].      -425.29320799922755
36      [ 4.28299283 -2.88572558 -7.91460873 -9.74866917].      -603.068147908977
37      [-10.  10.  10.  10.].          -1000.0        -68.19362594821958
38      [-7.48258291 -9.50258848 -9.10755479  9.14610542].      -820.0350630872958
39      [-5.24950925 -3.23166559 -3.31699955  8.97607806].      -403.73203930668006
40      [-0.10709283  6.73080844  8.27788037 -3.28854698].      -339.44710850060244
```

```
### 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].      -191.08815384130935
init     [-1.59264396 -3.39330358 -5.90702732  2.38541933].      -153.0053498623895
init     [-4.00690653 -4.6634545   2.42267666  0.58284189].      -78.51782080476644
init     [-7.30840109  0.27156243 -6.31120269  5.70670296].      -303.3198914381554
init     [ 7.07950585 -0.11526325  6.93122971 -8.40709046].      -476.9884901124878
1        [-2.26214698  5.87274909  1.60008358 -6.75402803].      -264.24405355223536
2        [-8.77692463 -2.54457149 -1.31781306  9.47735243].      -454.4748241053045
3        [ 4.62190764 -6.83721008  8.23256169 -9.96152115].      -715.1097438917498
4        [ 8.70233959  0.42166332 -8.66816053  8.53322589].      -592.7611112023912
5        [-3.46661731 -0.39559918 -3.41298829 -7.58101805].      -277.163239027877
6        [-9.3528057  -0.16784675  2.91108351 -9.91348457].      -506.06324643051016
7        [ 5.87954056 -2.52000408  8.75494609  8.0392552 ].      -535.7355782296529
8        [ 4.7244456   5.97941072 10.         -5.5225782 ].      -515.8225713617458
9        [ 9.33988639  0.28284014 -7.13390479 -8.51921815].      -530.3795790959404
10       [ 3.27690332  3.28708214 -1.44062271 -2.42812913].      -62.15733903644859
11       [ 7.93183147 -5.23912097  2.68785335  6.56683347].      -311.9776014590769
12       [0.19228331 7.33765529 6.54124141 7.73210077].          -475.2243903479375
13       [ 4.17183347 -5.30217661 -2.43725911 -5.04062651].      -193.08270621975657
14       [-8.05438678  2.46491222  8.14573136 -1.42407893].      -284.19555217143665
15       [-2.82718104  7.52894698 -6.74703384 -0.36123532].      -258.45239846251934
16       [  6.25333676  10.         -10.          10.        ].          -939.1042206
17       [9.80424847 2.68419823 1.52454554 9.29803994].          -463.32003253331396
18       [-6.61534183 -6.24706505 -5.2993531   5.07806808].      -309.21092247405176
19       [-6.69692911 -4.32856522  8.35858338  6.45628149].      -458.65384396032124
```

```
20       [ 2.6956679  -0.0955432   7.16240135 -0.63670877].       -162.8064539127141
21       [-9.46753889  7.57506663  3.40441431 -2.45231133].       -263.2229953067182
22       [-6.92481486 -6.05204209  8.22183433 -7.26328795].       -535.0245740300666
23       [ 8.21499945 -1.44616147  5.90323425  4.12780274].       -244.3685276491935
24       [ 9.08768254  9.47052549 -1.28440169  9.47550915].       -626.0578381338491
25       [-8.3153683   7.84850355  6.26717182  8.61646094].       -607.1492904634072
26       [  8.13151305 -10.         -10.           7.53827941].         -793.424136
27       [-9.72223078 -7.55183843 -2.77560218 -5.80278316].       -366.3833709394279
28       [2.20427625 9.4191281  0.81949044 2.28341955].       -205.1694949759091
29       [-10.         -10.         -10.          -3.48227388].         -648.504925
30       [-10.           8.16671794  -9.07081276   6.16070885].         -632.046836
31       [ 8.83382471  9.44259501  0.10144192 -7.94157322].       -508.6668724391631
32       [ 6.82221391 -6.94923273  4.60327609 -2.11763255].       -224.6341966637042
33       [7.69547463 6.60571897 8.09091716 1.46377282].       -351.4507211831887
34       [ 3.28862204  3.30580034 -5.60210305  4.7997572 ].       -218.9730192656345
35       [ 3.8676788   9.51050452 -9.7880626  -7.8834117 ].       -731.8695605128078
36       [ 9.08194027  6.18795668 -7.19398665 -2.39358803].       -337.240641132387
37       [-9.15686549  0.80276656 -0.47030207 -0.18344141].       -85.93520897719323
38       [-10.          10.         -10.          -2.50104353].         -625.020874
39       [-3.34012576  8.8147458  -3.48058828  7.95492405].       -456.0226783132896
40       [ 9.74568636 -8.44738753 -9.22358342 -4.76901703].       -583.8926816678596
```

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

```
Evaluation       Proposed point           Current eval.          Best eval.
init     [ 1.01595805  4.16295645 -4.18190522  0.2165521 ].       -88.34515671669995
init     [ 7.85893909  7.92586178 -7.48829379 -5.85514244].       -492.7558969714181
init     [-8.97065593 -1.18380313 -9.40247578 -0.86333551].       -351.4764929161654
init     [ 2.98288095 -4.43025435  3.52509804  1.81725635].       -98.64051704307819
init     [-9.52036235  1.17708176 -4.81495106 -1.69797606].       -174.4920942433137
1        [-4.16414452 -0.84627201  7.21067826  1.72505809].       -186.6573966961027
2        [-1.42093745 -2.38317783  4.89209527 -7.84297806].       -331.2251441029536
3        [-1.98230524 -6.0350318  -5.50491833  4.81186289].       -260.3012268281973
4        [-6.95962222 -6.12078547  1.05388197 -2.78453945].       -157.7110124486851
5        [0.63810404 8.56152948 2.44970438 5.87960129].       -303.2887506246535
6        [ 5.86849691 -6.28138395 -5.51873265 -6.54080399].       -375.8485219824623
7        [-2.40374089  0.96559291 -8.83808773 -9.60595066].       -611.0752459364239
8        [-5.69652173  2.33901134  0.4494956   0.55949224].       -45.25057308251345
9        [10.          8.03008047  5.30780555  4.93421421].       -410.8686635455698
10       [ 8.1180514  -0.53355909 -7.24048118  8.43264977].       -508.184161051329
11       [-5.56655434 -5.05469054  1.85578009  7.1880508 ].       -299.0903765519785
12       [ 0.76213235  6.97711173  8.30170294 -8.73896127].       -610.1736129974997
13       [ 3.67118459  1.71144371  9.88206907 -0.72605793].       -314.4101830216651
14       [-0.62701324 -5.16336383 -3.65845081 -8.79638686].       -403.3722720050600
15       [-5.87917964  9.0108278   8.03503233 -0.77077353].       -393.0163896290459
16       [-2.25892289  0.28501765  8.21214282  0.29800465].       -207.9382990379678
17       [-1.84189639  8.25161501 -1.80201859 -7.43879364].       -370.6552986898451
18       [ 9.11906792 -1.31777968  5.50936209 -7.02941839].       -375.3405898275435
19       [ 9.04342598 -7.52262686 -3.84028198  2.09841499].       -256.8200622081252
20       [ 8.2215293   7.78343336 -6.67089118  3.55133193].       -372.7074150323998
21       [-7.28954964  7.36324705 -9.29157054  5.48379747].       -540.86033595497
```

22	[6.00557472 -9.33087149 6.90123489 9.7261517].	-731.4704897916151
23	[-7.61707939 -9.16553986 -8.60430175 -7.68253114].	-684.2213051973378
24	[9.74299344 7.12197974 6.72226466 -5.47596591].	-451.8824490012956
25	[-6.95400694 6.25970572 7.54096982 8.42899977].	-581.5168700233202
26	[-9.70150953 -0.26222488 9.40969028 -6.71739223].	-540.3770582359848
27	[0.43830467 -9.26112347 9.37038161 -3.55612677].	-485.7252314280399
28	[4.8092043 5.72160034 -1.0963792 -9.62895308].	-463.0749585433426
29	[-5.13673807 9.0547271 -9.708375 -3.88389544].	-533.4584546359836
30	[3.80793903 3.6708587 8.43447663 9.38135723].	-606.9114487071026
31	[-7.15771281 -10. 7.58946545 2.38460784].	-446.778228
32	[-10. 0.35707499 0.171531 -10.].	-500.343273
33	[8.76561414 -10. 4.80654157 -10.].	-746.144516
34	[7.90503153 -0.64035752 2.38167539 7.36332189].	-297.2008089562036
35	[1.03309652 7.452861 -5.03799012 9.03578044].	-514.8829087026113
36	[-8.88729783 8.15406653 -1.07063663 9.18066587].	-552.538956638318
37	[1.80597178 -2.31601717 -9.56815682 0.32529802].	-289.06155514047856
38	[-10. -7.8452911 2.30126748 -10.].	-638.984680
39	[-9.51579139 9.81021872 0.52860391 -0.74326529].	-286.07910792952964
40	[-7.75542884 -3.54792255 -7.28474302 9.700393].	-620.9151254999871

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

Evaluation	Proposed point	Current eval.	Best eval.
init	[9.34059678 0.94464498 9.4536872 4.29631987].		-430.9815193091966
init	[3.95457649 -5.67821009 9.5254891 -9.8753949].		-742.4213397738982
init	[-4.94035275 -1.30416935 5.58765844 -6.04629851].		-267.7054836935381
init	[7.25986471 9.66801354 -6.72315517 1.94667888].		-390.4072884141779
init	[-9.82027805 -2.26857435 -9.11679884 9.13305935].		-689.7298760679505
1	[0.90405303 0.4880816 2.75220488 -1.97009113].		-39.54269054725363
2	[5.30486793 -6.47973031 3.69857697 5.99163089].		-296.7524113440073
3	[2.44668677 7.07389915 -9.38608839 0.44996751].		-371.17222324388183
4	[4.73707556 -0.72479098 -9.3121442 -0.43521843].		-284.39627806246625
5	[-7.88752674 6.9996629 9.5911258 7.81351701].		-680.3769142637927
6	[6.97297538 -6.40571993 -2.05106734 -5.92730455].		-283.84127012666863
7	[0.46199955 7.11757846 8.44850995 7.74973553].		-555.8988542896269
8	[6.465232 -8.2562289 -5.78689094 5.87653584].		-416.72887032150777
9	[-6.21902391 -6.61688609 -0.04378302 -3.21504386].		-167.59440037861728
10	[-6.38615157 7.9844663 9.46020353 -2.30152042].		-457.9606734243433
11	[-5.59342186 -8.95203702 0.35851975 5.34445778].		-306.2028268087292
12	[-4.57747469 6.76415735 -4.11298047 8.65053766].		-462.5379563592547
13	[2.35747262 4.35064884 6.12515272 9.24202752].		-497.6267463804779
14	[-8.19755724 1.6166877 -9.27463953 -1.1878353].		-336.1279290628333
15	[7.99683297 3.755866 -3.73641094 -8.81543277].		-444.8921163772508
16	[-9.01084566 5.87964272 -2.96427056 -7.97390816].		-431.02928199826374
17	[6.90236259 7.30024494 7.61968438 -3.28837303].		-371.66212056260554
18	[-7.07788877 -1.21727011 4.87587121 3.08177701].		-162.37176096021716
19	[2.12735998 5.39297028 -6.17417945 6.93749789].		-369.5709010638187
20	[1.07471056 9.97468132 0.14850747 -1.47257909].		-208.88365771699065
21	[-1.50568443 -2.30964713 -7.09911011 -8.48537587].		-452.1345330132159
22	[2.53687866 -7.51555741 -8.87336299 8.92430991].		-674.1859016743401
23	[-8.24208797 -9.25718255 2.59909975 -9.02269984].		-585.2252793459877

```
24      [9.4019532  2.20661934 0.75567978 8.69750771].          -402.4347793626921
25      [ 1.95749234  9.59160176  4.03193371 -9.29334154].      -582.0636810101753
26      [ -5.56550739 -10.          10.          5.82558038].          -666.724419
27      [ 3.59432101 -9.79143691 -9.66209508 -8.17160021].      -751.8320608147276
28      [-8.96018513  2.79650004  1.77717835  8.72422378].      -409.84915344511105
29      [ -7.39754728  -8.09044391 -10.          -4.7966712 ].          -577.666489
30      [ 10. -10.  10.  10.].          -1000.0        -39.54269054725363
31      [ 10.          3.6398269 -10.         10.       ].      -826.4966797663338
32      [-3.71704512  0.63300159 -3.02912155 -4.65155591].      -128.6924283100281
33      [-10.          10.          8.80393277 -10.        ].          -932.527696
34      [ 9.12536468  2.62439166  0.58549488 -0.29671977].      -98.42772700934157
35      [ 2.63221979  3.9114145  -0.64914921  5.04538811].      -140.61485650740008
36      [-1.39477641  7.72463019 -8.7378325  -9.72371559].      -728.536954256011
37      [-9.47248982 -4.29578099 -4.59695165  2.46352973].      -214.3073402513634
38      [-1.06513664 -0.80313724 10.          1.54426552].      -311.9635989102733
39      [ 0.78586014 -3.52650845  2.1016074  -9.56713317].      -404.86050897554617
40      [ 8.25946959 -1.6795748   7.30914567 -3.87243287].      -294.1145575948101
```

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

```
Evaluation       Proposed point          Current eval.          Best eval.
init     [-5.56013658  7.41464612 -5.86561689  8.37221816].       -524.4616052483334
init     [-0.23177622  2.23487726  5.31815713  0.36835976].       -95.43421434439179
init     [-4.06398997 -6.24557543 -8.38517462  4.76880592].       -396.4299395095289
init     [-1.17381554 -6.83380265  7.59874062 -4.51827076].       -349.66121999082225
init     [-1.71529962 -4.07840135  2.57575818  1.5967562 ].       -66.31107991358093
1        [ 1.55325716 -9.96715655  0.30945224  2.79590352].       -232.65661511944194
2        [ 4.56139684 -4.673529    0.23379905  2.52038143].       -90.06336406337539
3        [-2.15134925  9.53418759  4.82671391  6.46019907].       -423.2579596168082
4        [ 4.17941015  0.37676612 -9.58435076 -5.8403882 ].       -429.77125084788634
5        [-4.20489896 -6.1571556  -6.8351617  -9.35801008].       -583.9500222830909
6        [-3.76365226  7.78239697 -8.18300187 -5.22131556].       -445.2295866800745
7        [ 0.41211986  0.47256529 -3.34919958  6.13953029].       -185.04322093078548
8        [-0.25183718 -3.30684237  8.97845951  1.46012155].       -272.29986042933757
9        [-8.43262179 -0.81279137 -3.32426401 -3.2850448 ].       -148.74864077353737
10       [-4.45934361  6.06017597  8.73696795 -2.92485867].       -356.5602305084666
11       [ 5.70578654  1.32017865  2.21996359 -6.47121455].       -218.33292961577024
12       [5.9157065  9.20555265 3.92858506 1.34998596].          -258.07117287445936
13       [-5.79261282  9.88159419  1.54685764 -4.35636722].       -311.93621768939175
14       [-9.58242141  6.39325499  3.81306329  5.48600232].       -337.5734596849989
15       [ 8.81346656  0.26074237 -7.90133971  4.47083934].       -345.06029108836725
16       [-6.17059134 -9.38746474  8.58746863  7.33289857].       -650.6446439355226
17       [-9.81344616  7.08235098  9.35724572  9.34014123].       -808.2502114393861
18       [ 9.7156969  -5.89977377 -3.89397489  8.8167595 ].       -520.4395408412056
19       [ 2.74326124 -7.27587891 -4.48922446  9.17807962].       -510.810301099718
20       [ 4.35598802  6.34978281 -7.65239241 -0.14981832].       -275.38122619690256
21       [ 3.2666596   8.62078011  9.59676632 -9.95043148].       -831.6448825126107
22       [-8.94088297 -9.96667826  6.85803065 -1.44447318].       -428.0525034625426
23       [-6.71412576 -6.47032038  5.49564249 -9.1363839 ].       -553.3098785928355
24       [ 5.76122226 -5.97967704 -2.13038836 -8.57292222].       -412.3004021515765
25       [-9.5137551  -9.52877563 -2.42470317  1.12786197].       -294.8325131409716
```

```
26      [ 5.63902137 -4.07372558  5.15047326  1.24959081].       -150.8170754227820
27      [7.18690545 1.48116608 2.75088061 8.99150002].          -402.1296386680617
28      [-3.51452738  2.62244882 -9.37027444  2.4460871 ].      -313.4458756741793
29      [-1.75324557  5.72458797  1.1956778  -9.98715416].      -471.8776141259798
30      [-5.87723567  1.52217007  0.14509913  6.77549639].      -222.8684692354771
31      [ 7.10449555  9.72511591 -4.2661579  -9.79460285].      -677.9669052974186
32      [ 1.40844728  4.2793359  -8.13817505  5.29587464].      -349.4839874679371
33      [ 3.47693238  0.38314793  9.88420956 -6.09536587].      -454.0893994782494
34      [6.59172263 4.78312379 9.0481541  1.10643587].          -339.7114329273215
35      [-10.          7.60996207  10.         -10.        ].          -915.82304
36      [-1.53275275  3.86512508  6.75586768  7.15377169].      -373.8587569563232
37      [ 9.95008312  0.47304221 -2.88574361 -2.56098577].      -150.6688331464956
38      [ 9.60918915 -5.3145805   5.79589968  7.49734303].      -474.4440168894315
39      [-0.19518197  9.15097708  6.59601634 -4.81104503].      -390.6257706179896
40      [-9.79702142 -1.03775527  9.09024991  4.83976418].      -439.7267000766916
```

```
### 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].       -544.2132200910305
init     [-7.8468664   1.90104128  0.59634724 -1.62385143].       -80.41569220533873
init     [-3.29184301  2.45038864 -1.23717148  4.71764213].       -116.4614081942586
init     [0.36072824 1.577172   2.90710192 9.80448543].          -414.9705305999620
init     [ 6.39716394 -1.7359813   7.5253531   6.47518865].       -384.5560589397142
1        [ 4.34664291  8.74699069 -2.96380463 -4.92731805].       -295.3792632862785
2        [-5.57681425  2.83874786  4.53696952  0.08226863].       -108.9971859349811
3        [ 8.56608527  1.00066366 -5.6557535   4.78400723].       -262.8900162206665
4        [-3.88060276 -3.60562409  2.63626196 -8.23698378].       -333.3013666694193
5        [-5.8247851  -6.52272924 -8.70125002  9.33818614].       -694.9622520060385
6        [-4.22599656 -6.273846   -3.61341454  0.26400804].       -136.0304290975082
7        [8.53192076 6.17342963 2.07530059 5.72450643].           -293.016652032392
8        [10. 10. 10. 10.].       -1000.0        -80.41569220533873
9        [ 6.30471624 -8.43012351 -5.35385847 -9.5705537 ].       -634.2568059910016
10       [-3.69260396  5.83022375 -9.57366863  1.14098381].       -361.7911115062092
11       [ 5.16565529  3.58828793  9.82334313 -3.91747081].       -403.3161362642976
12       [  9.60336478  10.         -10.         -10.        ].          -992.22461
13       [-9.04988797  5.58244396 -7.40585203 -6.38326187].       -471.7518947971859
14       [-9.29604707 -8.38751066  6.35426496  9.78206391].       -731.0023081636889
15       [ 1.34667705  1.66784567 -7.21098005 -3.98954394].       -227.0375006504399
16       [-9.83578154 -4.31734713 -5.94863658 -7.45471468].       -462.4714862773936
17       [-9.10121252  5.55793128  3.33297346  6.97861209].       -372.7435125298843
18       [1.23390261 8.80744396 5.27933725 7.8968852 ].           -489.7220432653936
19       [ 4.60029783 -7.6833382  -3.64473115  5.6386107 ].       -306.2580300685843
20       [ 0.06453549 -7.7967855   6.9049098   3.12194752].       -303.6034563600570
21       [-6.22558255  3.84150491  5.39828881 -9.10226956].       -487.1020090194400
22       [-9.4350203  -3.73521946  6.6050899   3.36629923].       -293.1328564562584
23       [ 4.77823775  7.5917621  -6.41237773  7.5815048 ].       -491.3738841649841
24       [-7.90638723  6.02936142  9.88101478  9.32707469].       -776.0980053940647
25       [ 0.77200041  9.62659846 -2.18138662 -6.42799569].       -365.4906376493122
26       [-5.79950773 10.         -5.35543478 10.        ].       -719.67633510379
27       [-7.27926549 -9.74422149  7.88236493 -5.94339879].       -570.5783982561215
```

```
28       [ 8.36509188 -7.29232874  1.13811484 -2.82161515].       -212.0628435950243
29       [ 10. -10.  10.  10.].           -1000.0        -80.41569220533873
30       [ 8.08345178  2.08679927  1.56696175 -4.33527253].       -156.5961139436581
31       [-8.92132407  9.40956057  1.22340745 -0.16940941].       -261.2746588397886
32       [ 10. -10. -10.  10.].           -1000.0        -80.41569220533873
33       [ 9.40870759  9.42696533  6.05032309 -6.25673327].       -532.6652023672408
34       [ 6.65824193 -6.56874738 -6.82923631 -1.86056356].       -284.3912625545989
35       [ 2.81064774 -1.42429224  4.93656056 -5.71164422].       -215.5573668319805
36       [-3.03139038 -7.16170787 -9.53470606 -7.094842  ].       -585.8484378997774
37       [-8.35321984  3.29137006 -9.31499484  8.00911363].       -608.333506253447
38       [ 2.03162479 -1.02863257 -8.55919583  9.47754861].       -585.3188791662384
39       [ 9.0636722   1.00028557 -8.00613524 -9.87243744].       -666.3059844288002
40       [-2.36116824  8.24932043  7.93515436 -4.1497598 ].       -399.4597404941487
```

```
1 ### ESTIMATED GP EI GRADIENTS
2
3 np.random.seed(run_num_7)
4 surrogate_approx_7 = GaussianProcess(cov_func, optimize=opt)
5
6 approx_7 = GPGO(surrogate_approx_7, Acquisition(util_grad_approx), objfunc, param)
7 approx_7.run(init_evals=n_init, max_iter=iters)
8
```

```
Evaluation       Proposed point           Current eval.          Best eval.
init     [-8.47383421  5.59837584 -1.23181537  4.46930356].       -218.9402949114849
init     [ 9.55979024  0.76991741  0.02240927 -8.55897733].       -385.6010135228649
init     [-4.63122040e+00 -2.34998349e-03  3.58459992e+00  6.07478072e+00].       -20
init     [-2.38117734 -8.68127306 -4.23708801  8.19187055].       -478.684726691054
init     [-5.73229293 -0.95752076  8.62412039 -9.50201545].       -618.9724223135001
1        [ 6.75835989  5.37295013 -3.72010646  1.45250665].       -153.36929299141528
2        [-2.26867274 -4.96775263 -3.10681926 -5.91972827].       -223.63371729470776
3        [-7.1620489   2.23165943  9.48914494 -4.53159483].       -413.52857382347213
4        [ 4.29877809  2.09137364 -5.85501544 -4.79573932].       -222.06726053075232
5        [ 5.54571144 -1.93184305 -4.33889721  9.71942618].       -472.56601814369407
6        [-8.29791508  2.55444133  2.18471017 -8.00103752].       -352.29101715701233
7        [-4.37541545 -0.5346361  -5.18090179  4.60429146].       -185.03916123697073
8        [ 1.07740606  7.90693058 -1.60531584 -3.10335   ].       -172.45414805073628
9        [ 4.32564604  9.62119313 -9.80342151  4.4727457 ].       -572.1889646870435
10       [-1.05786283 -9.49375755  8.62025049  9.63510673].       -775.6492210333183
11       [ 9.55989635 -2.53974033 -1.5412528   1.20869694].       -117.26235379790558
12       [-7.17360778 -4.84967241  6.40731588  0.96189955].       -225.361386687391
13       [ 3.19083533 -3.29321979  8.95741732  2.24125439].       -292.6708834181588
14       [ 5.03801863 -7.21025243 -0.6201886  -5.32713967].       -244.02468194075794
15       [ 7.74721754  1.25369836  8.59242013 -3.588522  ].       -336.1619103268165
16       [-7.9829215   9.96558946 -9.45385173 -6.10303813].       -679.4672172306633
17       [ 0.80947682  9.04408849  5.10436576 -8.20692078].       -511.8241702434949
18       [  7.14875229  10.         -10.         -10.        ].           -951.104659
19       [ 7.70140256 -7.00291695  4.88315627  8.18434349].       -496.8628515670783
20       [8.55716728 9.13690962 1.75279783 8.46693102].       -536.1639305576136
21       [-1.57366787 -9.43769678 -8.83544712 -7.55230358].       -642.9612065829849
22       [-8.33660472 -7.44132911 -3.97589571 -1.8706741 ].       -241.66666251299313
23       [ 1.27717217 -7.67877202  7.69464844 -6.92481786].       -488.9935018952279
24       [-1.30465393  9.10240953 -3.61979728  6.19560824].       -360.26088308742817
25       [-2.08000587  6.91069233  5.94203563  1.06121231].       -210.26980979847923
26       [ 9.4609801  -7.96418498  6.23210666 -0.33356069].       -333.32914058674754
27       [ 8.25477835  2.45208354  3.14975175 -8.1648879 ].       -376.5911791348866
28       [-9.22461904 -8.50807928  2.67217363 -9.06121187].       -579.7122005728986
29       [8.50304142 3.73774679 8.76724186 6.66865119].       -508.72044015241454
```

```
30      [-10.          -7.53632619 -10.          10.        ].          -913.592424
31      [ 8.12214103  9.99850144  5.40115294 -1.18595284].       -359.0525327741966
32      [-9.85036512 -6.59222902  1.73615674  9.00261156].       -517.1754400527367
33      [ 8.38403241 -7.27722642 -8.85640552 -5.67569856].       -540.3700210808483
34      [-4.91115304  1.5427759  -6.68424648 -9.77374709].       -545.021720919354
35      [ 4.36462018 -6.62940457 -6.98915348  2.56911614].       -279.8941494492369
36      [  8.63929868  10.          10.         -10.        ].          -974.637483
37      [ 10. -10.  10. -10.].           -1000.0        -117.26235379790558
38      [-1.78814767 -0.5809474  -9.9654654  -1.9493752 ].       -317.0042283787086
39      [-10.           5.51005024  10.          10.        ].          -860.721307
40      [-10.         10.         10.          0.4249071].       -600.7221841773804
```

```
### 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].          -396.4929378718437
init     [-5.34543344 -9.77202391 -1.39062363 -1.9529728 ].       -240.6164747296011
init     [ 0.45349343 -0.43216408  1.10712948  0.86772035].       -7.268149341268288
init     [ 5.21791151  4.24749148  2.39364192 -1.47816459].       -89.2374155003189
init     [-4.21849944  9.47710482 -3.32451909 -5.62397878].       -357.1006001814876
1        [-3.60620407 -1.35048134 -4.59708517  6.02111773].       -225.0673183583388
2        [7.02116965 7.81234303 5.75333375 8.88043753].           -586.1134614902139
3        [-6.66057257  8.24583762 -1.16469123  7.80712487].       -428.2252152756431
4        [ 7.56352013 -7.67982822 -8.37749723  2.84161744].       -418.0128981331368
5        [-5.28183607  6.44896586 -9.02454394 -6.64184392].       -531.8596567468478
6        [-8.91376283 -9.57999256 -6.95098062  9.64939713].       -780.399537642458
7        [-9.6826463  -0.3889901   5.14264133 -8.41133424].       -456.3987200853980
8        [-0.44591076  5.04602604 -4.13162746  0.15920843].       -102.4360195138091
9        [-0.81981182 -3.75576157  2.39337966 -7.50364036].       -271.2868542882077
10       [ 6.98608516  8.9084123  -7.19395552 -9.08060074].       -692.6132327878008
11       [ 6.61186169 -9.02334481  1.34103459  6.04361295].       -358.0543691391289
12       [-4.95566202 -0.69807525 -2.84596135 -4.0933849 ].       -116.8548918636406
13       [-8.62371369 -2.43071156 -9.9135532  -8.61774405].       -678.0828161743012
14       [ 7.25721416 -1.35983896  9.25077409  2.93432361].       -347.5369652539947
15       [ 9.5056863   3.58618034  2.73863527 -1.09937986].       -143.4143645140530
16       [ 8.63675564 -9.78898035  4.97635133 -5.02173071].       -441.4051554941350
17       [-8.87173722  9.60530842  4.18381046 -3.54573537].       -366.0333881846408
18       [ 7.65858112  1.1873229  -3.41218412 -4.96882152].       -195.1590866468364
19       [-3.51296977 -2.14237485  8.01964739  7.68652932].       -450.7956609764924
20       [ 3.88729122  2.89386017 -8.55538582  8.36656893].       -531.4416684731586
21       [ 8.14290147 -0.90943489  9.4506528  -6.76425   ].       -518.9258156611875
22       [-4.98980983 -9.72316766  2.42842233  6.45619727].       -398.3998187124128
23       [ 5.71554054 -8.97318576 -3.75596631 -4.65975223].       -322.8785413433199
24       [ 6.49597242 -8.88231691  9.8207384   9.40242253].       -842.9516706692345
25       [ 3.49120939  0.47218526  8.86783187 -1.27059239].       -255.0074072343243
26       [-2.65701843  7.36182894  9.93375458 -6.19230739].       -564.8699204894327
27       [ 4.44778296  1.88262846 -7.75003774  1.47885616].       -215.8086700171643
28       [ 6.22824474 -6.53188051 -3.40849765 -4.72519097].       -248.2852463067154
29       [-0.69636553  6.93093178  9.44036372  4.76930131].       -454.9068968396014
30       [-8.73569396  9.75149541  9.21231033  9.65324137].       -893.8359353668989
31       [-7.949679   -9.34510791  9.14340971 -1.5517733 ].       -498.2973049662701
```

```
32       [ 0.76354257 -5.34208385 -8.78129711 -9.98537045].       -687.8227460899298
33       [-9.68589847  2.33775791  0.56999271  2.04645734].       -122.4734789052245
34       [ 9.79209434  8.82224434 -7.45862516  5.47372843].       -538.2891817040652
35       [-10.          4.23528701  10.         -0.58153048].          -437.228022
36       [ 6.77387947  8.29094342  7.88717361 -9.5531293 ].       -735.0365695021964
37       [-8.27889813  4.97508899 -8.23376812  4.42310931].       -399.6835714138959
38       [-9.24737554 -8.93907261  9.94556598  9.97466233].       -940.0463943398267
39       [-4.20839573 -9.05466236 -9.81290419 -4.33256618].       -545.6482004818091
40       [ 2.5863293  -7.14175826  6.07563387  1.88704149].       -233.6822043865636
```

```
### 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].       -310.44693919044926
init     [-7.15777829 -5.62882649 -1.62983639 -5.03797663].       -224.09509945393205
init     [-8.31880698 -3.0900272  -6.66447307  7.57118171].       -450.8358595558308
init     [ 9.01928063 -9.22503248  3.98214783  1.45519631].       -307.59276095652154
init     [7.96014236 3.33797946 0.95675566 4.04854848].          -153.95720360034693
1        [ 7.46801686  8.12554554  5.16555974 -3.06196092].       -305.3716974158376
2        [ 1.79155607  6.63416988 -1.83334365 -8.8517705 ].       -414.7329035737106
3        [ 9.33019641 -7.52767745 -5.63058289 -0.50920484].       -296.5319700941453
4        [ 9.24871681 -5.28271506  6.79982967  9.30415074].       -626.3348541920698
5        [-3.79629409 -6.6853837   6.54189652  8.77158881].       -539.9528703527935
6        [ 2.12600285  4.95023923 -9.32534107 -9.64414998].       -686.4540986420868
7        [ 1.28476433  2.5242373  -3.51043207  6.58802611].       -224.97191943921874
8        [ 2.77700112 -8.36901764 -0.12784879 -1.65296659].       -158.77087782876356
9        [ 4.5742908  -2.26634671  4.22642647 -4.96365815].       -183.3364420801886
10       [ 5.90229198 -0.97787337 -7.39679132  1.70640703].       -212.53438872188386
11       [1.45553679 9.66867785 2.5326465  3.88938752].          -268.8374859100074
12       [-2.80325151 -5.66028587 -7.21789028 -7.12490995].       -431.2870786551177
13       [-7.10958489 -6.43009064 -2.06472288  6.66090206].       -323.4980354582162
14       [ 2.88579401 -2.77392671 -3.18766244 -7.65057666].       -288.3260143685056
15       [4.13407718 2.80066864 9.99697821 6.6531594 ].          -509.65492380429157
16       [-6.00453231  6.54755256 -3.53825723  9.75186528].       -539.7485958744875
17       [-8.05861704  3.05396769 -0.03663706  1.43109669].       -91.79092371187993
18       [ 7.16153312 -6.98673646  8.87492942 -7.80656261].       -628.9793248365662
19       [-4.01060296 -0.34982372  8.60881423  0.96692656].       -242.4045244504999
20       [ 4.17786482  9.47787302 -9.66536225 -2.21135987].       -496.9328405677091
21       [ 8.81854895 -2.08902609  7.42386265  1.27961156].       -258.38569858206495
22       [-6.35451721 -9.81793612  6.43303349 -5.11766372].       -462.0773156544369
23       [ 3.33271293 -7.85136664  0.86644336  4.68323042].       -224.3776525780219
24       [ 10.          1.74355487  10.         -10.        ].          -806.079967
25       [-5.38540934 -6.04055244 -8.33599896 -4.84152513].       -404.2072796206927
26       [  9.95876677  10.         -10.         -10.        ].          -999.177035
27       [-10.          1.02178117   8.8378448   -6.60727142].          -511.034717
28       [-9.94128745  8.70452119  9.5162855   3.90820484].       -583.1419040323834
29       [ 10.         -1.12249134 -10.         10.        ].          -802.519973
30       [-10.  10.  10. -10.].          -1000.0        -91.79092371187993
31       [-6.1692829   8.70589311 -5.26969916  0.31795138].       -273.3587612579158
32       [ 6.74008081 -8.88765608 -7.80847043  7.72967928].       -625.3179491961921
33       [ 6.29253375  8.28665611 -8.4124348   8.36618625].       -669.2127874890457
```

```
34      [-6.19124126  3.7611987  -7.85922135 -6.42044595].       -416.8152851277189
35      [ 3.06205081 -6.73516315  9.16175403  3.84221381].       -410.9646387419819
36      [-1.69401184  5.11511387  6.886713   -6.82399272].       -383.7464101615669
37      [ 8.80322501  4.03136267 -5.83288165 -5.9488901 ].       -353.6252390245419
38      [10. 10. 10. 10.].       -1000.0        -91.79092371187993
39      [ 10.          10.           9.05992236 -10.        ].           -946.24657
40      [-10.         -10.          -10.           2.78176773].           -630.95292
```

```
### 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].       -333.66852780407373
init     [-0.02985975 -5.50406709 -6.0387427   5.21061424].       -278.59164421659773
init     [-6.61778327 -8.23320372  3.70719637  9.06786692].       -549.5010991841266
init     [-9.92103467  0.24384527  6.25241923  2.25052134].       -236.08347396370496
init     [ 4.43510635 -4.16247864  8.35548245  4.29151567].       -337.43331296672727
1        [-3.98599887 -7.72031276  6.57362653 -9.06207361].       -593.2170550074999
2        [ 5.24133604  6.79774962 -7.90085176  0.21262617].       -307.3416184837157
3        [-6.98285725 -7.53553636 -0.44064091 -4.85155595].       -257.0617857366076
4        [-1.78826799  3.25598727 -1.45510672  1.58046094].       -40.744242419126515
5        [ 4.63220256 -9.79505919 -9.12123737  0.49337993].       -463.9082783243551
6        [-7.83635489 -1.28730572 -7.74844317 -2.63232565].       -272.5544380760249
7        [ 6.92515619 -6.60878201  5.11920166 -9.02608502].       -539.809307689828
8        [ 8.75235832  9.38171401  9.18458426 -0.57986069].       -507.0516092544958
9        [0.4502773  8.55166428 3.92918152 5.72526907].       -323.8948991980415
10       [ 1.23952645 -2.7410793  -5.02186756 -4.62861357].       -177.917172953133
11       [ 1.99559358 -0.41077916  2.20886236  5.83753048].       -155.26413998063086
12       [-6.52612311  7.27650284 -4.77362568 -5.95519154].       -358.7050015084682
13       [-4.63875915  8.6124843   7.12702008 -7.73195445].       -561.3835822655649
14       [4.4189373  5.3779954  6.27947744 2.68037042].       -224.40572911328755
15       [-8.76637523  9.03915533 -9.83105068 -1.79825538].       -543.1455551204746
16       [-1.72806915 -4.12214739 -2.17527422  6.47352369].       -218.79191091835867
17       [-4.86937269  4.69882169 -9.05647957  8.60545103].       -610.1432572383127
18       [ 8.01708502 -8.18565856 -9.9066972  -9.21975056].       -832.7268139183689
19       [ 9.49546442  5.26511905 -5.65588327  8.33754155].       -519.6322449518909
20       [-3.58608995e+00 -9.52603561e+00  8.08057615e+00 -3.61266107e-03].       -39
21       [ 5.52855938 -4.30300172 -3.84014298 -9.60506256].       -480.86561786014903
22       [-8.18817025  9.94113172  4.25384459  1.09316473].       -323.7639494970617
23       [ 0.06951243 -3.70532939 -9.7893336  -7.65624326].       -549.4291642409903
24       [ 8.6903743   3.38596008 -0.62995648 -1.58863793].       -109.73767412227923
25       [-1.43315326  0.86980528  4.14345257 -8.25404774].       -327.58886456679573
26       [ 8.07136813  3.92807734 -7.01900891 -8.38076196].       -524.7547092612123
27       [-8.76993163 -9.29399334 -6.39604219  1.47070195].       -381.048249051727
28       [-9.0177282   4.17602767  0.45261751  6.82006506].       -302.8655738264081
29       [ 3.4501367   8.92902793 -1.97206855 -8.11369106].       -446.35361649463425
30       [ 9.21235497 -0.61948647  1.97276737  9.10396676].       -428.8392875581332
31       [ 10.           2.04907673  10.          -10.        ].           -808.397436
32       [-10.         -10.          -8.16579781  10.        ].           -900.040761
33       [-2.20161943  0.97131884  9.36126949  6.53202581].       -440.30359268890663
34       [ 7.6664045  10.          9.85246722 10.        ].       -949.9870888035418
35       [ 7.56256959 -9.64367941 -9.19534672  8.03957583].       -755.3958856610491
```

```
36       [-0.7298507  -9.42107843  0.09262759 -0.25432344].       -178.3305807575113
37       [ -0.39274505  10.         -10.         -10.        ].          -900.154248
38       [ 4.35248423  1.26737828 -5.59159894  5.17708748].       -223.16348979181913
39       [ 9.35010821 -2.82927717  4.58923203 -1.80520422].       -179.6523432185739
40       [-1.10075059  3.7166708  -3.80698923 -5.91781882].       -212.40075477803543
```

```
### 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].       -308.1901117295372
init     [-1.59592791 -0.29145804 -9.74438371 -0.25256785].       -287.8310851024486
init     [ 8.83613305  7.01590179  4.5992894  -7.82527856].       -484.9233304137926
init     [ 7.87808341  7.14308494 -6.69826765  2.64668028].       -326.7315575089055
init     [-9.59032774 -7.66525462 -3.67265377 -6.84175387].       -437.19018412688746
1        [ 8.98204795  9.73346661 -3.23891901 -5.20250642].       -409.8940110092612
2        [-2.46813998 -7.3003779  -2.91425156  2.82909317].       -170.1764091752233
3        [-2.7011834   8.03388106 -4.64351945  0.87154575].       -204.10806795611475
4        [ 8.76589534 -3.7692322   0.38597262 -4.05301551].       -171.40980738291742
5        [ 0.40212877  5.87654958 -6.74210512 -8.93336634].       -524.8174583999808
6        [ -9.25339454  10.         -10.          7.84640083].          -831.889334
7        [-3.08409357 -3.60866716  5.26352461 -6.93028397].       -310.7860080203654
8        [-9.90602549 -4.748682    9.51818141  7.44931505].       -636.9858133184193
9        [ 1.38380213 -6.05647422 -9.74783179 -9.20728444].       -699.4336893349999
10       [-0.66080332 -7.11489274 -6.24181798  9.58065679].       -585.7168713525834
11       [0.47931285 3.42089165 1.4955809  6.56081621].       -202.52226415289624
12       [ 3.41314276  7.39306433 -0.25317011 -0.71476575].       -123.20018963353341
13       [-5.13468268  2.50992841  8.47930131  9.3389305 ].       -603.5225915248926
14       [-8.33083797 -2.90069237  2.58400674  2.53507465].       -131.9685802935269
15       [-2.57164218  8.12805756  6.98338235 -0.57169137].       -286.35419388641105
16       [ 7.14813561 -4.63106617  1.0316659   8.89685877].       -413.79877772048144
17       [-2.1973427   9.33772956 -3.2166429   7.28382045].       -422.4712377697771
18       [ 0.61155877 -9.53496613  8.85943923  8.7384669 ].       -723.1173679369668
19       [ 5.39782382 -1.2844643   8.43553257 -3.8545582 ].       -305.34130395597913
20       [ 1.76289051  1.78627203  1.36617813 -1.10226957].       -19.948639322038893
21       [-9.44572799  0.38465493 -4.5694765  -7.65384804].       -386.48360188487584
22       [-4.77974029  4.4732243  -0.45397362 -2.86226686].       -96.25395078503722
23       [ 5.18121212 -1.99532779 -7.30143683  5.2012205 ].       -302.95134299644883
24       [-9.61971011  3.1176179   9.12836755 -1.67386312].       -373.1664586261421
25       [ 8.2768875  -9.94795023 -5.92111287  2.98014894].       -407.13417814806763
26       [-6.93748111 -6.73987294 -6.40149091  7.86706383].       -509.48044981061787
27       [-4.32346474  1.55695409  1.34266742 -0.49860569].       -29.94325738858083
28       [-7.61201419  6.73443611  5.25467801 -8.86396157].       -545.762201314582
29       [ 8.23610841 -9.41107658 -3.36881055 -9.9391775 ].       -674.1658577495111
30       [9.42572279 5.99651616 4.39003384 9.20003049].       -557.1400978967978
31       [  5.09767769 -10.          6.10153625 -10.        ].          -737.672553
32       [ 3.85643653 10.         10.          6.49267521].       -683.4914281030631
33       [-7.50907996  9.79968712 -7.89677045 -9.28885043].       -780.6619367533552
34       [ 1.56093955 -9.95977    -8.48212704 -1.71366569].       -428.4166070719457
35       [ 9.26787693  1.46443124 -9.51187767 -9.61152017].       -731.1353910679393
36       [-9.82066768 -0.950261   -9.6834058   2.92002989].       -413.6628477314643
37       [-2.29865052 -4.54219424  2.84952395  9.40503458].       -424.7249133328572
```

```
38      [-4.46892425 -9.8170717   9.44949192 -1.40793229].      -488.5288632488556
39      [ 10. -10.  10.  10.].           -1000.0        -19.948639322038893
40      [-9.63435922  8.48491508  4.43619797  9.32516508].      -643.6828175562798
```

```
### 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].      -162.9855143833216
init     [-9.70850075  8.37494016  8.01429708 -9.33157145].      -775.5340081417003
init     [ 9.13898673 -7.25581357 -4.32343294  2.12166369].      -262.896783965427
init     [ 8.88450272  7.05471082 -9.95481533  0.42452054].      -476.4881936814493
init     [ 1.04075267 -0.29245173  5.36268308 -6.78566494].      -271.7103262017248
1        [ 9.00627049  5.34951301  6.50018506 -1.86719396].      -279.0503579558785
2        [ 8.88428387 -2.74934901 -7.2021893  -8.7943139 ].      -559.0227598865044
3        [7.07957877 4.20923579 8.31123577 5.90287485].          -432.1614135764528
4        [ 3.65158786 -4.292633    3.33049006  6.75429274].      -265.9458642217923
5        [ 3.07120909 -3.55916591 -5.97833992 -2.63162   ].      -169.6909890964862
6        [-1.65285505  6.06514615  0.23991581 -4.68469538].      -164.2620874180567
7        [-6.77288607  5.95533828  4.43070962  2.50634443].      -200.8247064280947
8        [ 9.11568286 -9.45162454  9.96288854  5.4127943 ].      -676.7328994714871
9        [-0.54343888  6.9130662  -3.08529747  0.79842711].      -126.9834191405119
10       [-2.82350692 -6.04303308 -5.89223718  4.65288339].      -271.7613613951719
11       [ 9.57564946  9.09185994  0.40699926 -8.19654001].      -526.2469144713443
12       [-7.70853825  9.04000734 -1.47997073  6.83741383].      -416.4368787746895
13       [-7.51472777 -2.86866187 -2.69659984  9.1633357 ].      -430.6114122434688
14       [-5.69729876  6.22751761  8.3350766   9.36945888].      -669.5907092612638
15       [-7.09189339 -6.84298482 -8.67391943 -7.30482965].      -583.1006139759294
16       [ 0.32849255  7.22400743  8.16912739 -5.81811846].      -440.0864109134870
17       [-7.8093924   0.25400768 -8.20800109 -6.68611527].      -442.0460447682908
18       [-2.20704908 -1.48961893  8.03710516  8.4355361 ].      -487.7272502636997
19       [ 3.45355371  3.20265039 -5.41025361  2.04079201].      -136.9128325139473
20       [ 9.23927444  8.83645046 -3.1931206   2.71000944].      -301.4945678833609
21       [ 9.40854602 -1.63495674 -6.22777515  7.27708637].      -422.0463992480705
22       [-0.33850406  9.87135488  8.01814511  3.0043928 ].      -423.9793368383325
23       [-6.34363966 -3.26687378  8.41196286 -0.3285002 ].      -274.301699618427
24       [ 4.07591095  4.5574632  -7.917332   -6.6119907 ].      -421.0801132673950
25       [ 2.1929946  -9.25644022  9.18015489 -3.38417908].      -474.8090002196256
26       [-8.10752289 -1.57339231  8.58331018 -9.25565541].      -634.3713234024771
27       [1.89896828 9.38445191 1.28394687 8.89610969].          -501.2505848367997
28       [-7.52353818 -6.94794261 -1.00786132 -1.33236509].      -163.2995799131125
29       [ 2.40471286 -1.77558974  0.60610937 -2.25237082].      -33.48288474225401
30       [-8.25635404  8.72047473  9.41845964  5.43226615].      -604.4209490093992
31       [-7.2146841  -8.72640886  6.64000657  6.64927951].      -513.4728241727345
32       [ 8.73799259 -5.21790059  3.56099694 -7.14884124].      -373.2713093912246
33       [ 2.17030082 -8.76142379 -2.25624259 -9.35864802].      -523.8443626376102
34       [ 10.        -10.        -10.         -3.7304149].      -655.6639813866544
35       [-7.08535682 -9.48461069  7.3913004  -5.84596029].      -530.7129329366356
36       [-4.69103101  6.73657336 -8.76681722  8.61296539].      -640.0725571553805
37       [-5.93967174  1.79432319  2.35507141  7.08984956].      -259.4218430268972
38       [ 3.19730535 -7.92525441 -2.35894485  8.81422764].      -463.2983745862259
```

```
39      [10.          4.08568743  1.89990658 10.        ].      -544.214618581755
40      [-9.28072647  1.80933599  0.81593588 -8.55403877].      -387.3628486026613
```

```
### 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].      -559.2187253383422
init    [ 9.45202228 -0.93101505  2.18084926  5.51053029].      -226.80639002099542
init    [ 2.8322669   4.44036459 -9.29926952 -4.03101058].      -371.880837115183
init    [-8.82975016  7.14121885 -2.54291944  3.59695903].      -251.1102762788094
init    [-4.87440101 -3.0483757  -9.8117446  -2.83332435].      -363.2668776074504
1       [ 9.1114829  -9.99975933 -5.06042598  4.24465356].      -431.9015624918471
2       [-9.72135464 -7.69244795 -6.22844507  7.60723681].      -560.7130380018729
3       [-9.63074404  1.6639904   1.75101086 -5.56192983].      -231.22732967764995
4       [ 0.57275335 -7.71738011 -5.09743991 -0.17013756].      -197.51142602710658
5       [3.60696134 2.18226175 0.08284273 4.5407231 ].          -105.02795653920944
6       [-3.20440472  8.67469235  4.12830206 -1.70990841].      -223.5925649645513
7       [-8.87693875 -6.00070012 -7.77180414  6.2427699 ].      -487.90836843374564
8       [ 0.65604441 -6.68523471  7.23179469 -8.11619975].      -510.202477173446
9       [-5.8544789  -4.61653174  0.32592983  0.79886988].      -79.7711169699346
10      [-8.6173938   7.4519909  -9.59943625 -2.01230926].      -477.9688957311798
11      [-1.49330781  2.57117389  4.71236205 -7.69030542].      -318.6340964742683
12      [-5.30538029  6.95799883  6.90644832  6.01970967].      -413.01925828964454
13      [ 7.2681675   7.28099793  2.98739184 -5.83963596].      -322.03104311105756
14      [-1.49428528 -5.65598761  8.84676953  3.5970451 ].      -352.7642073554095
15      [ 5.65679901  9.5654791  -5.75009086  1.45833834].      -322.69379340075477
16      [4.90678758 6.08014104 9.9141949  3.21154933].          -434.1427724240993
17      [ 7.08271592 -1.41666864  9.88865573 -6.13938682].      -498.3035834641204
18      [ 0.48041618 -7.17050782 -6.11767413  8.4939688 ].      -503.9309991535369
19      [ 10.         -5.0363443  -10.         -3.27173471].          -493.546519
20      [-8.13883095  3.00580382 -8.65994083  9.43255566].      -665.1864327750363
21      [ 8.93458891 -9.49016566  4.77058082 -8.20222919].      -597.3349464061171
22      [ 9.51932974 -9.42611124  4.3520765   1.87769368].      -339.2454291055169
23      [ 1.91227999 -3.59178921 -1.76186535 -8.40306683].      -321.21735109429136
24      [-4.72872862  4.78804377 -4.77881532 -7.87064177].      -384.5108359711843
25      [ 8.37504685  2.72018974 -6.83424791  8.64419372].      -523.9494477640571
26      [ 10.          3.71169799  -6.23894611 -10.        ].          -644.326749
27      [-10.         -4.51306038  -9.57252845 -10.        ].          -815.635336
28      [ 7.80523392  9.96347751 -0.53570157  8.93068156].      -579.3526655763468
29      [-10.         -1.66281198  10.          3.1451065 ].          -445.096666
30      [-9.30946088 -9.80242291  7.46915009 -9.02972561].      -772.3494390532742
31      [-5.20904937 -4.95723463  3.41952388  9.19786713].      -449.765015136702
32      [-10.  10.  10. -10.].          -1000.0        -79.7711169699346
33      [-10.  10. -10. -10.].          -1000.0        -79.7711169699346
34      [10. 10. 10. 10.].       -1000.0        -79.7711169699346
35      [ 10.  10. -10.  10.].          -1000.0        -79.7711169699346
36      [ 0.20045141  0.43042958  9.93737957 -2.41817443].      -320.0555283706186
37      [ 10.          8.53599884  10.         -10.        ].          -945.726552
38      [-1.38924419  9.89895651 -6.08754122  6.25885054].      -465.77599401198086
39      [ 1.37149923 -4.02128522 -9.91081836 -9.35542541].      -678.9913805061929
40      [  7.38548532 -10.         -8.27225936 -10.        ].          -859.836217
```

```
### 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].       -611.6615754592598
init     [-3.80528149  9.15207479  0.26233425 -3.6343115 ].       -235.04045120202608
init     [ 0.78399875 -5.57490115  6.12962716 -3.15490749].       -215.30445206449784
init     [ 0.77777698 -9.88252429  3.46304956 -5.79951476].       -366.4491321572651
init     [ 8.65115186 -2.51510501  5.04837844  5.26278004].       -274.73972442923775
1        [-6.20242914 -2.96747056  4.29623333 -2.85129273].       -143.97423371105057
2        [1.00854423 7.16161365 0.70898236 3.83323949].           -163.8774495060855
3        [2.67305604 5.51907321 6.71543751 9.91762785].           -596.7942385381396
4        [-5.7724008  -5.62578169 -7.22205566  2.80544114].       -284.57571408043947
5        [-1.31829204 -2.32031879 -2.5251031  -6.82350009].       -217.87470331061357
6        [ 0.24276126 -4.12052041  2.21586778  7.93881754].       -300.8458156729556
7        [ 4.79563633  4.13838411 -7.01002178  7.65374311].       -438.99092445744697
8        [-8.61484726  8.56315525  8.59846951 -0.87156311].       -445.71037165155775
9        [-9.06063141  7.52304771 -1.55477555  8.90428917].       -519.6849787906098
10       [-7.70141397 -8.88627386  0.42465228  3.83571444].       -276.63531297625127
11       [ 7.71405527  0.8769244  -3.27790749 -7.73627   ].       -332.6781682024899
12       [ 6.82279714 -0.09161239 -2.028231    8.81142257].       -369.47318014324986
13       [-8.83738244  4.0407375  -9.99557193 -2.68265241].       -439.2753179802774
14       [-9.85481133  2.18770252  5.41117638 -7.79034754].       -437.2899393957444
15       [ 8.17029418 -7.04437314 -7.57697513 -9.26345515].       -681.4781544321248
16       [-1.72940849  3.0518742   7.43147169  1.85385719].       -201.04618652613695
17       [-7.40602326 -9.16073537 -8.1385771  -6.13555201].       -571.9766307870241
18       [ 7.34655904 -3.81093188 -9.64319433  9.64563555].       -734.1450644476762
19       [ 7.56622239  1.19315477  7.79884993 -2.17352434].       -261.4579708896355
20       [  7.44601737  -6.03715981 -10.          -0.32879653].          -428.770200
21       [-6.316551    3.89166133 -4.71810535 -8.70962295].       -440.40055409042446
22       [ 9.65221726  8.97277123  6.94150178 -3.74665223].       -454.8894976126941
23       [ 5.22498155  9.63662115  0.71281827 -6.55975151].       -386.67505567565115
24       [ 2.20669363  8.74237933 -9.75975673  0.83537193].       -446.27782896046733
25       [-0.55443781 -1.44037781 -6.47875427  0.2204847 ].       -130.57400255903315
26       [ 8.89205552 -9.66595279 -3.21552805  9.28652324].       -641.9068555813558
27       [-8.01189683 -0.39981096  2.62732101  7.36608046].       -302.2552006610292
28       [-1.57727485  2.09123977 -0.53319287  7.30830626].       -225.73260879918254
29       [1.349273   6.70863794 1.93835752 6.58155122].           -276.3711392771723
30       [ 8.55403072 -7.94155738  8.07976264 -8.05251959].       -654.5280888649233
31       [-10. -10.  10. -10.].           -1000.0        -130.57400255903315
32       [9.93637329 9.49260215 9.71522224 9.87516166].           -952.1824052431184
33       [-6.88415677  0.98201804 -8.25417685  8.82276035].       -565.079039948062
34       [-6.64906649 -5.41282173  9.91982187  9.3338334 ].       -746.4977448586994
35       [ 8.16011255  8.92056461 -3.04860577  1.01501804].       -257.7434206770887
36       [ 0.67700616 -9.84109187 -5.33539649 -4.94960676].       -377.5463111769999
37       [ 7.25471633 -7.141688   -0.51594006 -0.71281971].       -157.4693541324512
38       [ 10.           4.04905205 -10.           1.54614142].          -442.351858
39       [-9.91563139 -7.9921648  -4.10602285  9.76219956].       -657.8495743221604
40       [ 10.  10. -10.  10.].           -1000.0        -130.57400255903315
```

```
### 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].		-400.1409127715913
init	[-4.49198143 0.6000045 -3.88162169 -3.91051282].		-127.26731063140548
init	[-7.76517448 -5.00201972 8.35259796 -4.71706293].		-408.6387460656439
init	[4.35547375 7.31430068 6.14158964 -5.78898835].		-373.17505498643254
init	[-6.65513937 -9.06587217 -9.21155376 -5.9953838].		-607.007632459987
1	[3.91355621 -9.41682078 -0.01015161 -7.34731365].		-408.60132987115037
2	[4.92526955 5.96067446 4.10838652 1.54460964].		-155.49735537615913
3	[7.60284646 -9.83670929 6.07418637 6.51677002].		-531.8853593071874
4	[-3.39221728 -3.69203211 3.60318038 4.66933918].		-164.92898037857765
5	[9.63144016 3.76660236 -9.40395266 2.32534909].		-408.0711964371901
6	[-0.46099561 9.58086516 -0.66748888 -7.87674906].		-433.30779867587194
7	[3.72122725 0.49817372 5.20397899 9.4368106].		-451.8016557305748
8	[8.49239079 2.67562175 -2.00937567 -1.52509795].		-107.85507155339401
9	[-0.8289478 -8.11611777 -5.43376832 1.07172118].		-225.6017494454485
10	[-0.90690653 4.09601032 6.89554636 -6.11770942].		-326.7282335927336
11	[-8.83646821 -9.52000508 -3.03305299 9.46202552].		-645.0621025739723
12	[-5.20285545 0.11177617 -7.97281939 3.50515359].		-266.93664629346597
13	[3.63484615 -6.9061913 1.72145752 0.75302069].		-119.76147168170785
14	[0.93691127 -4.95007973 -0.40679652 9.18253617].		-387.6567136784933
15	[-7.20628386 8.66608008 6.24591119 2.13266135].		-337.35961251591385
16	[-6.14779635 -0.43746793 3.56596824 -2.19771089].		-95.64627741914462
17	[1.30223912 -8.21616194 4.9325588 9.93275327].		-604.335219875349
18	[-2.00153572 8.29433076 3.25430495 9.69475726].		-549.3227664094798
19	[1.14120333 4.18919902 -5.55151656 -7.17257482].		-334.64244889261784
20	[1.70134615 4.96133179 -9.08286378 5.10911973].		-404.0318659073334
21	[2.37201846 9.77107763 -3.75463112 -4.47165852].		-318.84907150899494
22	[-8.39053544 9.20499859 -6.65884284 8.09859265].		-635.234458453372
23	[-9.21357548 -6.74167545 1.0076465 -2.03849531].		-195.45825573001872
24	[-5.45975356 7.48081612 -2.98212522 -2.50654362].		-193.54438471635504
25	[8.44725336 -1.16828432 7.91853452 -4.41015655].		-339.9933560045092
26	[1.48543842 -2.35317467 -4.1430873 -1.67478431].		-75.99651646117569
27	[5.60066467 -3.87589042 -5.27284941 -7.26059882].		-355.6865014047286
28	[-10. 6.48275172 10. -10.].		-884.052139
29	[9.98850983 -4.80991884 -2.64374788 5.39267067].		-283.33276339270355
30	[8.36219672 9.80306436 9.96457014 1.990924].		-575.8595635674714
31	[6.2784228 -7.31871577 0.31170191 8.68142997].		-448.3061736856189
32	[-9.71396708 -2.10459634 6.69994244 7.91517354].		-488.4873827048736
33	[9.57582248 9.17312205 -1.86804383 -9.36794429].		-621.4909965238489
34	[10. -10. 10. -10.].	-1000.0	-75.99651646117569
35	[-8.57773271 4.63013958 -7.94362611 -4.39284626].		-382.94586390580287
36	[4.99716754 9.4594823 -9.31957327 -7.1633234].		-669.751440773997
37	[-1.04315432 -10. -10. 9.78057615].		-883.726856
38	[-9.98712406 7.81056755 -6.70440634 6.67862108].		-535.0156893708369
39	[0.33296563 5.73801053 9.98513878 5.30488345].		-477.6365391125071
40	[9.4625835 7.46115737 -0.75066995 9.52610204].		-565.5552219708684

```
### 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].		-364.5052686545986
init	[-2.78542329 -5.53838117 3.77452324 -6.7253715].		-292.7694790431682
init	[-8.59350266 8.8202172 1.2736276 -8.44015321].		-519.2518779260945
init	[4.45281022 -6.83095653 -4.99437387 -4.13025488].		-256.2187854356974
init	[3.93221428 -0.71471824 -5.69875713 -1.06347476].		-118.4353660802023
1	[-0.64007705 3.2211867 3.79933264 -4.73341031].		-154.0872642720883
2	[5.98502725 -7.94806028 -2.90882686 9.11344119].		-519.7669380533303
3	[0.47384394 -7.09622115 -6.44235327 3.14296454].		-264.9618889185273
4	[5.52616367 4.65190076 3.38866017 -9.07435877].		-437.6438475693719
5	[4.28544997 -2.78264932 9.08402216 2.69441349].		-310.4491882302683
6	[1.87133514 5.63089801 3.59434817 4.26424392].		-178.4090409147218
7	[7.45907335 5.14332641 9.11613095 8.2671896].		-631.2426140402113
8	[1.24349389 9.55943585 -9.57929245 8.27200487].		-733.3046944193816
9	[-6.58978683 -7.60537449 -1.72785531 -2.22440249].		-187.85705046016238
10	[-3.41259436 -0.53328708 -1.59942228 9.36100657].		-370.4028212436752
11	[8.89995329 -8.45639035 4.20210818 2.73243867].		-305.0682678921419
12	[-3.05457595 8.6513061 -8.42526724 -0.14730159].		-372.06280377102144
13	[-3.45060725 -2.87892412 -7.28800971 -6.14683127].		-338.96249394589125
14	[2.62583757 0.46449641 0.44128596 2.38948669].		-30.7493233028567
15	[-9.57158135 -8.22028104 7.2184175 3.81965746].		-441.4369963411156
16	[7.96462988 4.97612078 -6.1641524 4.64151999].		-313.1240408731446
17	[8.37017332 6.7425363 -6.78563457 -5.15425157].		-405.38313951635405
18	[-4.12908744 -6.44608013 -9.57829988 9.62751237].		-746.1407249039194
19	[4.39317174 -9.17548236 -7.80685399 8.14414445].		-635.8281736052663
20	[-8.7687876 7.06373411 2.00583578 9.41241664].		-543.1287949362841
21	[-8.65310965 4.93087707 -8.68746282 -7.9504542].		-602.7583229578638
22	[-9.22904309 8.72149497 4.93469127 1.94582391].		-325.50264160836826
23	[5.32114119 -7.48188522 8.86730528 -9.96863312].		-773.6536503113464
24	[9.48449776 6.83252536 5.34568415 -0.39785071].		-269.6846613147235
25	[-2.96913047 1.32765639 8.54306292 7.91176568].		-481.6769955880185
26	[-9.53324189 5.05618557 -7.07152101 4.53301251].		-374.2247636937185
27	[10. -10. -10. -10.].	-1000.0	-30.7493233028567
28	[-10. -9.24748792 -10. -10.].		-971.032065
29	[-0.33483549 -9.49402643 9.99682498 7.11449175].		-682.6586909037793
30	[-2.89541268 8.35462631 9.63290282 -9.0193402].		-751.7554172693181
31	[-0.79327379 -7.57161458 9.67752913 -2.41264056].		-419.53502625404093
32	[-8.45006419 0.50230015 8.37739395 -3.43626462].		-329.68204182867333
33	[-4.0000179 -0.5564691 -3.01592062 0.52101113].		-44.99260095461599
34	[1.61080585 4.63614811 -5.15881738 -9.82035341].		-511.17998897186004
35	[-7.65657066 -10. -0.48355379 10.].		-659.324547
36	[6.85895173 -2.37520122 -9.29506748 8.96495049].		-639.0045681483889
37	[9.01567238 -0.68638457 -2.64806786 -7.18554316].		-309.7895084469941
38	[5.45605189 -3.49800546 -1.82105288 4.02505857].		-128.9936736073418
39	[8.82399561 -6.28660037 2.64461737 -5.0640013].		-280.4640265206088
40	[8.37035142 -8.82828923 -6.3218622 1.82878166].		-359.2157588479259

```
### 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].       -430.5408947278945
init     [5.7397092  3.12667044 2.75041792 1.51205788].           -84.33607001372373
init     [-9.21874168 -2.84372791  8.91366374 -8.79910639].       -649.2160718740936
init     [ 7.28084207  7.54581052 -8.97612669  3.04837231].       -445.7720200383797
init     [ 1.03502737  1.95026506 -0.32942751 -4.34023678].       -84.3545380937588
1        [-6.82082511  3.53524771 -7.6305547  -1.10007868].       -251.0363955498036
2        [ 6.36132887 -5.79483735 -1.80465394 -5.23795116].       -227.14164161943438
3        [-9.93358214e+00  6.76426027e-04 -8.86041968e-01 -1.86772363e+00].       -1
4        [-3.57261465 -5.71224609 -1.88427713  4.42421021].       -166.96913100401667
5        [ 7.32299713 -3.1980267   7.52939765  8.3238999 ].       -521.3057613159242
6        [-9.15358294  6.24405427 -7.34906493 -3.27748646].       -366.7584440566941
7        [-8.32522503  1.10793827 -8.79126558  8.08570378].       -565.1378998269158
8        [ 9.48510639 -1.89215997  8.72310011 -8.54004817].       -617.1348995086978
9        [ 1.0372515  -9.73544251 -7.97334565 -0.40984316].       -382.02818070028843
10       [ 9.1855761  -9.95264225 -0.41398097  9.23568727].       -624.1908016035516
11       [-3.55419883 -1.3121061   6.26693875  4.48480965].       -214.3532082540939
12       [ 2.54749487 -9.29639909  6.99658479 -6.71299771].       -506.4497511543947
13       [ 3.37722984  0.46032172 -5.15280505  5.104616  ].       -195.7120911255773
14       [-9.67252227 -8.84826579  4.65918096  0.87645251].       -318.3378799110803
15       [ 4.34389407  6.23064858  6.70660755 -5.49281777].       -352.13132184462563
16       [0.31334945 6.26826312 1.77763055 9.3516953 ].           -437.97716439741885
17       [-9.30935589  2.86017104  6.58259319  9.55371198].       -598.110513248124
18       [9.55069994 7.35606602 4.58315479 8.92855136].           -581.3313249716628
19       [-8.55606558  9.23104048  1.47366339 -1.29796204].       -256.88434797773255
20       [-4.9003479  -9.03322285  6.16587778  1.19907682].       -307.016926806265
21       [-6.7341369   9.61539255  8.20758674  4.15148688].       -501.2929609040184
22       [-6.17626749 -8.46700681  8.99586277 -2.9797324 ].       -459.8185502973863
23       [-1.57575393  9.85203222  3.11647346 -4.88636175].       -321.25142319522627
24       [-7.58786513 -8.19846824 -4.31973255 -8.94272425].       -567.874996683665
25       [ 3.40727255 -7.86151092 -6.92538287  7.8882845 ].       -527.9991273958915
26       [ 10.         -10.         -10.           3.35249994].          -644.957023
27       [-7.62900322  7.16838361 -1.09951782 -9.53917183].       -528.5831524964061
28       [-1.17026134  8.45485557 -6.6619293   8.49876673].       -566.398726789921
29       [-0.17344573 -4.6667978  -4.02317816 -1.15895835].       -97.51871213021056
30       [-3.85451606  2.65749216  2.16933315 -9.66637665].       -416.85519263096563
31       [ 9.76774387 -3.50335172 -9.68143559 -8.30741039].       -677.1986219733903
32       [-3.51762081 -5.5603206   0.04416167 -4.59494337].       -158.6678555193422
33       [-0.33584057  9.5576468  -9.38205416 -4.71504013].       -535.8052478292103
34       [-7.88630797  8.82046909 -0.94744659  8.63888828].       -519.0097314064002
35       [ 9.8231555  -9.92528962  5.24677685 -0.79795826].       -378.6500834052633
36       [ 2.56432668  6.10803492 -0.24395397  0.04946162].       -81.38027887864618
37       [ 3.96142624 -7.37930587 -0.01775028  7.07304132].       -324.7138071803278
38       [-9.77378055 -8.74370198 -2.80647787  9.9213683 ].       -665.7945847679596
39       [-10.         -10.           6.33837827  10.        ].          -820.525117
40       [-10.         -10.         -10.          -0.93028828].          -603.461745
```

```
### ESTIMATED GP EI GRADIENTS

np.random.seed(run_num_18)
surrogate_approx_18 = GaussianProcess(cov_func, optimize=opt)

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

```
Evaluation       Proposed point           Current eval.          Best eval.
init     [ 3.00748483  0.10906747  7.57202942 -6.36319549].       -343.0366725069492
init     [7.04466137 5.00272572 3.32203335 9.75790897].          -513.6566494158249
init     [-4.86063155 -9.43388149  2.71438231  6.94624775].      -416.7270243155722
init     [ 4.7234925  -9.58385776 -7.76793739 -4.04552516].      -452.4996897110763
init     [ 3.73940383  7.23252112 -6.02731282  3.14378061].      -267.1207899158314
1        [-6.1664519   4.28513481  2.79740714  0.66407663].      -99.9903410594818
2        [ 5.61099687  3.24280582 -9.3946128  -2.53775904].      -343.0519975384772
3        [ 3.30844576  7.22382341 -9.24562554  8.22016932].      -642.0425720582784
4        [ 8.03656335 -7.89614249  7.70734605  1.23688198].      -373.6135402668539
5        [ 9.562822   -7.89156436  1.83282657  5.62935728].      -352.8375538330246
6        [-5.31480829 -4.60774758 -2.97712464 -7.93522102].      -349.1706066518353
7        [ 4.54666061  8.69770438  0.01712558 -6.54510361].      -343.3266506149231
8        [ 2.31421647 -2.51006802  8.67058127  9.73721889].      -622.7471464704078
9        [ 8.39884901 -4.60470488 -7.7957923   9.97387584].      -693.1832086099516
10       [ 8.47467859 -1.0716028   7.04016464 -4.60023784].      -307.4573495762565
11       [ 0.14066288  5.10392834 -2.9351928  -9.28691931].      -422.9535060537450
12       [-3.93005265 -7.62357754 -2.48223981 -1.02660088].      -154.3833638230351
13       [-8.79601456 -6.39611748  6.71849908 -7.75307177].      -535.0456864449842
14       [-8.05883063  7.37144065  6.236766   -9.81425211].      -675.5909538391936
15       [-2.80981525 -1.87682768 -5.05682874  3.23441465].      -133.5003292210129
16       [ 6.58567988 -3.25808927 -0.548504   -0.50180546].      -66.51127573185336
17       [-9.953162   -4.99542568 -2.13845079 -6.90241283].      -353.2661160857886
18       [-7.48394579  2.61975937 -2.44662114 -5.23530564].      -197.3272885382654
19       [-9.43772874 -0.83234697  6.66346072  2.91251469].      -257.5924201878812
20       [-8.76219046 -0.51368957 -0.40828288  7.68983987].      -314.3383692611986
21       [-5.5082365   9.11461961  2.87454124  8.85506424].      -534.9308638940636
22       [-10.         -10.          10.           6.53860464].          -771.013402
23       [ -5.12443701 -10.         -10.         -10.        ].          -926.259854
24       [-6.69867140e+00 -9.34293864e+00  9.22919429e+00 -1.76365303e-04].       -47
25       [ 9.59876928 -6.95266335  1.85259451 -7.10765861].      -401.1869896779841
26       [-10.          -9.98151397 -10.           1.34265309].          -606.472111
27       [3.47426713 5.81482046 6.59216091 1.73752403].          -222.1405215365365
28       [-3.83822248 -8.47380926  1.10967586  1.83939291].      -175.5704449659059
29       [-9.69577207  7.7269764  -6.0387706   7.13214099].      -526.2903161230462
30       [ -6.81500531  -1.54551249 -10.           9.03677848].          -677.874976
31       [ 2.22467047 -6.27762742  3.37822164  3.65453913].      -171.4261403037070
32       [-7.07091215  8.67818641 -8.75973987 -9.56261259].      -796.5930035886437
33       [ 7.97475265  9.50332613  9.57456992 -4.90365982].      -615.4237811171081
34       [ 9.81767754  1.62533553 -5.3312624   6.22021796].      -341.7017459185144
35       [-0.8295843   2.88245159 -2.1343876   9.28963231].      -376.1611693850035
36       [ 5.64146375 -2.61248194 -6.31462374 -5.99601424].      -308.9084028171971
37       [-10.          -0.84745603 -10.         -10.        ].          -801.436363
38       [ 0.79508317 -8.87292864 -3.97405035  8.73739679].      -510.8375216809741
39       [ 10. -10.  10. -10.].          -1000.0        -66.51127573185336
40       [ 10.  10. -10. -10.].          -1000.0        -66.51127573185336
```

```
### 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].       -405.7591605426394
init     [-3.37106873 -8.3400087   3.43954163  6.13187596].       -336.3665457376282
init     [ 9.65483829  2.7132147  -5.68153488  0.98054864].       -208.6243886962717
init     [ 9.11199160e-01 -5.31847854e+00 -7.72548314e+00 -6.81465902e-03].       -2
init     [-6.95783155  0.65372161 -2.25986463  3.76654767].       -121.3346135965923
1        [-6.5477408   0.83865275 -8.96461914  6.51115227].       -454.9531908278426
2        [-4.76936448 -2.97550534  8.89602795 -8.21242057].       -547.6474479851142
3        [-2.32173541  7.26125369  5.5769216  -4.96985133].       -302.945918194523
4        [-9.13977787 -5.66464753  4.94279629  8.45310839].       -506.8258744362434
5        [-1.55246852 -8.63332631 -1.23512782 -5.59496547].       -281.2699816452975
6        [-1.56376705  9.78608394 -1.82255739 -9.88981742].       -595.1793462637571
7        [-1.74139105 -2.25943387 -4.66754333 -7.46934292].       -301.764742107778
8        [ 3.38990203 -3.49236324 -8.09259704  9.98332142].       -631.0218446479364
9        [ 9.65341158 -3.79534468  9.58489508  3.02722194].       -434.2645691657497
10       [ 6.69276372 -9.56702686 -4.00113681 -5.37275537].       -391.3423807033970
11       [ 1.66645895 -8.36624855 -7.84433287  7.99584163].       -583.0999231748925
12       [ 9.63400574  7.61710851  2.75684065 -5.16449309].       -338.3432172087955
13       [2.61303639 6.56017039 6.36236978 5.08945476].           -317.9490770285931
14       [-6.41646303  6.43565965  4.11382594  9.720744  ].       -552.748575512695
15       [ 2.34954077 -3.13748972  3.15811446 -8.40909356].       -337.9805041983478
16       [ 8.68185637  0.99664649 -7.85471635 -9.94231562].       -657.84950498431
17       [-6.03612505 -9.67480441  4.33765388 -8.025091  ].       -537.6925518985452
18       [ 9.11708637 -4.55121628  1.89198796  7.70152498].       -372.5412063784399
19       [-10.          7.08537403  10.         -10.        ].           -900.405056
20       [-5.74949647  6.76752381  8.75305237  3.31669901].       -398.5052132890754
21       [ 1.40856514 -0.68378535 -2.05805581  4.56833676].       -99.10476462698388
22       [ 9.47335062 -2.82596526 -9.98254377 -9.14656194].       -739.3084528981631
23       [ 2.51530768 -4.62186548  9.97645257  4.84711249].       -441.6168693747741
24       [ 3.60353212  6.45652058 -3.97851869  8.29411939].       -419.0142581173057
25       [-3.57674008  9.13103764 -6.48638635 -0.12139103].       -305.823332965609
26       [ 4.61297469  9.30853289 -7.11465111 -3.32354209].       -390.6156141478248
27       [ 10.          8.20499148  10.         -10.        ].           -934.643776
28       [ 8.01410845  2.35469457 -6.26141828  6.51240303].       -362.5767565111792
29       [-10. -10. -10. -10.].          -1000.0        -99.10476462698388
30       [ 3.3786164  -8.90123527  4.19750545 -0.68899999].       -224.6350672543548
31       [ 9.49871881 -9.38584751  3.89522858 -8.56743145].       -605.5358693173803
32       [-8.59193625 -4.98422145 -8.77638346 -0.95433499].       -358.2240363268813
33       [10. 10. 10. 10.].       -1000.0        -99.10476462698388
34       [ 7.2052829   0.12871334 -1.41519373 -3.82625005].       -116.5183137120562
35       [-9.87217663 -3.42357815  6.86830815 -0.41111743].       -263.0986868654941
36       [-9.45083441  9.85873571 -6.29219981  5.88664854].       -541.0934703574821
37       [-1.3512532   6.30091737 -9.33657544 -9.71056899].       -719.9245280097841
38       [9.20140794 1.7795054  7.7472568  8.90074731].           -587.9523618432378
39       [  3.04056371 -10.          10.         -10.        ].           -909.245027
40       [-2.43385008  2.0090913  -8.95119579  0.54424733].       -255.5530607705036
```

```
### 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].          -473.2086565964213
init     [-9.28220829  3.83515164 -2.42638116  0.37021891].       -133.7863915866155
```

```
init     [ 3.15902931 -6.12299564 -4.55367196  4.37211867].       -223.63108911163158
init     [ 5.66007219  7.0065528   5.50489788 -9.26671387].       -564.6196272537535
init     [-7.6661253   5.02561399 -5.21563568 -4.90387972].       -287.0837808154094
1        [-0.38031735 -3.41587185  0.21282112 -4.72742343].       -113.01100973396937
2        [ 2.2515589  -8.62950692  7.2874785   9.15451767].       -648.549100750603
3        [-2.00279263 -5.09620423  0.51102349  6.68881567].       -235.6982290011646
4        [ 0.69284317  3.15192848 -8.93401192  2.30035677].       -280.9656097504359
5        [-3.75442852  7.02008144  4.58355261 -6.10855163].       -324.943296034215
6        [ 4.26761042  1.24919415 -4.83066711 -6.15878884].       -243.06222494906058
7        [8.1991784  6.51057117 8.40600493 7.35687831].           -580.478990957242
8        [ 7.25959653  3.0895583  -0.80389347  9.92972863].       -468.1292594707154
9        [ 5.87064721 -4.24128552  3.56822046 -0.20436374].       -108.80515424337919
10       [-3.49997926  0.57776952  2.16068958 -2.13900545].       -45.22460573983692
11       [ 4.5495202  -8.7912022  -5.43120396  4.47127488].       -343.7317320352756
12       [-2.50232563 -8.25576548 -9.16975052 -0.52445896].       -395.93016354238904
13       [-7.9326964  -8.04446097  8.74972574  5.65037046].       -549.7342240062541
14       [-5.18813627  2.95972213 -5.66334369  8.80164762].       -450.53305687060646
15       [ 4.75754346  6.89202164 -0.21987262  0.41733899].       -118.47586344087527
16       [-8.10580282 -9.69922266  5.29323984 -8.366023  ].       -617.8704067656233
17       [-9.71299857 -8.09319586 -8.5000152   5.4297126 ].       -560.0198700971824
18       [ 8.32521458 -3.43393064  6.7347488  -9.28298885].       -573.6590094219368
19       [-7.65449559 -5.71668941 -5.03336124 -6.1314202 ].       -350.33380944234796
20       [ 8.29123381  9.55745115 -4.69441223 -6.74923504].       -499.7555162343599
21       [ 9.94326022  5.89059493 -7.07853366  1.04477436].       -322.94977116904477
22       [ 8.99279405 -7.7213658  -5.32675342 -8.43383711].       -569.7506640120123
23       [ 10.  10. -10.  10.].           -1000.0        -45.22460573983692
24       [-7.98806158  6.10949412  8.75356783  1.32356094].       -375.34306789674616
25       [ 10.        -10.         10.         -4.0447058].       -665.4385799151439
26       [-3.43708279  9.82442968 -0.04543597  2.6091698 ].       -232.08963649678637
27       [  9.1257942   -3.89348241 -10.          10.        ].           -813.598536
28       [-10.         -10.          -0.87433276  10.        ].           -702.293373
29       [-6.65831638  3.00753142  3.58232452  9.98884726].       -500.0310924922751
30       [ 0.27964777 -8.02330517 -7.70099623 -9.36846377].       -657.813537361623
31       [-6.32885238 -1.33673671  7.25620452 -9.21312566].       -541.1123525427233
32       [ 9.90314811 -6.95624333  1.94751497  7.86459784].       -453.6370258076464
33       [ 1.82513814 -0.17452724  9.71446858 -1.05729301].       -290.9762223132123
34       [-2.27573402  9.29093008 -8.81850473 -9.77130898].       -793.033723234532
35       [ 2.20862081  1.02179984  9.94977153 -9.58444806].       -671.4065945027759
36       [ 1.33714879  7.92249666 -4.81885468  9.30691688].       -543.4587621889207
37       [-10.  10.  10. -10.].           -1000.0        -45.22460573983692
38       [ 0.61332381 -9.10358491  7.2885009  -7.27839155].       -537.3933528276375
39       [ 8.59741241 -4.97476183 -9.43840164 -1.25458632].       -396.9582345727328
40       [-9.62278682 -8.04740633  1.73736184  1.2127856 ].       -237.0581977973891
```

```
1 end_approx = time.time()
2 end_approx
3
4 time_approx = end_approx - start_approx
5 time_approx
6
7 start_exact = time.time()
8 start_exact
```

```
1623405106.7887886
```

```
1 ### EXACT GP EI GRADIENTS
2
3 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].       -403.9670698747264
init     [-7.06488218 -8.1532281  -6.27479577 -3.08878546].       -339.1443857291230
init     [-2.06465052  0.77633468 -1.61610971  3.70439001].       -68.19362594821958
init     [-5.91095501  7.56234873 -9.45224814  3.4093502 ].       -463.8472853559696
init     [-1.65390395  1.17379657 -7.19226123 -6.03797022].       -306.5051972029273
1        [-8.66550005 -0.82832373 -7.73316154 -9.44433302].       -612.6501988839965
2        [-9.45425553  0.44102495 -3.48020376  7.18978642].       -332.8795232598801
3        [ 1.05535193  7.78262178 -2.90094298 -5.0896264 ].       -251.1157691882347
4        [ 0.36065123 -2.54502396  4.20709157 -0.88713151].       -69.33123079316337
5        [4.31653906 0.48371229 3.15042815 6.26587691].           -205.9209108850890
6        [-2.18453853 -0.65522867  7.90275422 -7.78885119].       -435.6562421463968
7        [-7.49436108  5.707379    2.39984407  2.30443383].       -159.8332137207093
8        [9.95524731 0.51195576 6.31813346 1.94032376].           -234.4470026797404
9        [ 3.72279404  3.73873796  2.8148554  -8.58902967].       -360.6714737255779
10       [-8.0672788   8.31968062 -3.08213183  1.31932513].       -238.9762437111158
11       [-9.07094298 -0.72335704  3.0860416  -5.74907367].       -244.1068479205673
12       [ 9.50816228 -2.28968358 -1.66144831  0.39469839].       -109.7948303890625
13       [-0.9997413   5.38898234  1.90675627  9.44528666].       -426.8426629535013
14       [ 9.95448444 -6.77382559  0.6907992  -2.18042584].       -211.3098248435756
15       [ 9.47479366 -7.15746716  8.48081097 -5.61952091].       -534.3189123436204
16       [-0.57863171 -1.9526447   7.93107874  9.46945546].       -555.3488339586509
17       [ 1.41163983  7.78798458 -5.4504285   5.879598  ].       -350.6983377737508
18       [7.24062114 7.93125641 9.15218487 2.550167  ].           -455.5371215285705
19       [ 1.73340809 -0.79712138  4.12404223 -8.81370517].       -366.0242764664785
20       [ 9.27773282  3.95870839 -6.87632371  2.93517947].       -293.7316677911381
21       [ 3.67388287 -2.87543511  1.9187423   6.41420845].       -205.6466657341659
22       [-0.65340689 -9.89374011 -4.27583131 -7.13102488].       -454.4533908639449
23       [ 1.46622394 -8.48969563  8.49397553  3.2260202 ].       -404.3713625443585
24       [-6.37078152 -4.70876505  1.96500217  7.16093445].       -301.6314234715553
25       [ 5.63092799 -6.47097389 -5.18951977  8.86446367].       -510.5625669249395
26       [ 0.85123861 -6.41314817 -2.8317578  -9.95014277].       -503.0594672468104
27       [7.48228808 7.27455485 4.76900475 9.59774227].           -598.5197772506983
28       [ 7.2600034  -5.058828   -9.38822712  1.42622368].       -376.4440121089604
29       [-8.14556936  9.02614777  7.93274952 -3.02434334].       -454.6651426182432
30       [-6.794959   -6.35219734  8.60754909  3.28590136].       -392.330584807425
31       [ 7.67671057 -8.55914263  6.8929731   8.43475844].       -632.5695644702222
32       [-2.99226276  6.13926076  9.01406243  2.73386497].       -357.9907169589165
33       [ 0.28295684 -9.94416127 -8.08787596  1.017732  ].       -398.2370779078278
34       [-9.80124982 -9.64866069 -9.24896237  8.45579856].       -824.8898358530407
35       [-6.08792809  7.67902993 -8.94758926 -9.98693715].       -794.1315845126153
36       [-9.22428747  6.49527367  4.73427234  9.62226052].       -607.0562333131068
37       [ 8.59534303  9.94948517  6.22941747 -7.95335309].       -641.3046592741798
38       [-4.59424901 -7.22979121  2.7104647  -4.13747317].       -216.1614793821492
39       [-4.54159544  6.88626923  1.97738434 -9.20653185].       -466.2385579394507
40       [ 9.24124742  3.81915983 -4.04893168 -8.30867165].       -439.8902594616322
```

```
### 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].       -191.0881538413093
init     [-1.59264396 -3.39330358 -5.90702732  2.38541933].       -153.0053498623895
init     [-4.00690653 -4.6634545   2.42267666  0.58284189].       -78.51782080476644
init     [-7.30840109  0.27156243 -6.31120269  5.70670296].       -303.3198914381554
init     [ 7.07950585 -0.11526325  6.93122971 -8.40709046].       -476.9884901124878
1        [-2.26214698  5.87274909  1.60008358 -6.75402803].       -264.2440535522353
2        [-8.77692463 -2.54457149 -1.31781306  9.47735243].       -454.4748241053045
3        [ 4.62190764 -6.83721008  8.23256169 -9.96152115].       -715.1097438917498
4        [-1.79607725 -9.21541389  5.09748227  5.91370687].       -390.9142921783268
5        [ 2.40271381 -0.94556933  5.9547712  -8.79894592].       -423.62493364241
6        [-9.3528057  -0.16784675  2.91108351 -9.91348457].       -506.0632464305101
7        [ 5.87954056 -2.52000408  8.75494609  8.0392552 ].       -535.7355782296529
8        [ 4.63750927 -4.88709839  9.82790351 -9.28611417].       -703.9646815716937
9        [ 9.33988639  0.28284014 -7.13390479 -8.51921815].       -530.3795790959404
10       [ 3.27690332  3.28708214 -1.44062271 -2.42812913].       -62.15733903644859
11       [-9.54784817 -9.77123516 -1.33662368 -7.87639568].       -535.6256021440508
12       [-2.96123479 -3.33304178 -1.61052438 -7.91919815].       -289.6234098277667
13       [ 8.24410843 -4.50629319 -9.14716774 -0.7552634 ].       -361.8724045914075
14       [-8.05438678  2.46491222  8.14573136 -1.42407893].       -284.1955521714366
15       [ 3.44386072 -8.43602737  9.56543893 -1.5863816 ].       -438.7525841798323
16       [-1.34793439  0.93092109 -9.43976601 -7.53345388].       -497.8894114784636
17       [ 9.03586561  8.65203831 -3.55223781 -3.74321591].       -325.264243144737
18       [-4.83483999 -0.67671656  7.12137888  6.05966042].       -323.3116172875468
19       [ 5.96901911 -6.4980248   1.15627057 -4.29021072].       -197.7123586466667
20       [ 2.7175175   8.75716136 -7.73831698  8.94877073].       -660.727290252873
21       [-0.97064662  5.71904986  2.24241061  5.71414455].       -212.0482253413894
22       [ 7.2947619  -0.41197328 -1.91453856  6.24062424].       -220.3309326341394
23       [ 4.3246802   5.93710713  9.94852436 -3.42911748].       -433.1561386676652
24       [ 9.08768254  9.47052549 -1.28440169  9.47550915].       -626.0578381338491
25       [-8.3153683   7.84850355  6.26717182  8.61646094].       -607.1492904634072
26       [-1.98821907 -8.3662106  -9.04514766  9.34465119].       -738.6740869975897
27       [ 4.65489512 -6.89649322 -4.68436379  9.75064323].       -562.921252129937
28       [-2.06548982  6.29520283 -8.54907419  2.19592912].       -322.0738326582007
29       [-8.28655489 -4.3134418  -7.01772085 -4.32878793].       -328.5773898493526
30       [-8.89644946  9.3286275  -0.25458627  1.65018447].       -264.2802726180422
31       [-4.97130687  9.70356423  8.09713188 -7.65325255].       -644.0119421829494
32       [-9.59619427  5.31379981 -7.60600463 -8.79621865].       -631.6076507802376
33       [-5.91395541 -9.95702215  8.27387062 -1.60869564].       -448.9818604716679
34       [6.57844776 6.4036562  2.95666365 3.41494265].           -198.1625133766961
35       [-1.86778618  8.66304444  9.55841408  1.68621651].       -439.0484469717455
36       [ 5.93945595 -8.97558297 -7.58415343 -7.63236592].       -601.9695042185256
37       [ 9.39742381  9.21967728  2.44657325 -9.8274745 ].       -662.590654907447
38       [-9.88857333 -7.5048766   4.42401129  4.2555812 ].       -341.5857414386333
39       [1.12823537 0.09292935 9.75939677 1.39090662].           -294.7661473975298
40       [ 9.04056741e+00 -3.83617873e-03 -8.57635936e+00  9.68329656e+00].       -67
```

```
### EXACT GP EI GRADIENTS

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

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

```
8
```

```
Evaluation       Proposed point           Current eval.          Best eval.
init     [ 1.01595805  4.16295645 -4.18190522  0.2165521 ].       -88.34515671669995
init     [ 7.85893909  7.92586178 -7.48829379 -5.85514244].       -492.7558969714181
init     [-8.97065593 -1.18380313 -9.40247578 -0.86333551].       -351.4764929161654
init     [ 2.98288095 -4.43025435  3.52509804  1.81725635].       -98.64051704307819
init     [-9.52036235  1.17708176 -4.81495106 -1.69797606].       -174.4920942433137
1        [-8.15565982  3.06821805  1.15681525 -2.76870474].       -120.020279425763
2        [-1.42093745 -2.38317783  4.89209527 -7.84297806].       -331.2251441029536
3        [ 8.45325406 -3.16605859 -0.0933454   6.57617463].       -264.5157893840233
4        [ 1.13840429  0.92550401  7.42668624 -9.88496764].       -559.3264262752866
5        [0.63810404 8.56152948 2.44970438 5.87960129].           -303.2887506246535
6        [ 5.86849691 -6.28138395 -5.51873265 -6.54080399].       -375.8485219824623
7        [8.56580786 8.01799207 3.28161133 3.48133984].           -282.7352850094372
8        [-5.69652173  2.33901134  0.4494956   0.55949224].       -45.25057308251345
9        [ 3.96837274 -2.2216255  -7.45265132  7.31215743].       -406.1158420982276
10       [-6.42084406  3.24403587  8.67983875 -6.55430112].       -460.1290305129863
11       [ 2.18515843  8.99251209  7.28350237 -6.70324395].       -505.3876026931724
12       [-2.72651171  9.00355969 -8.67976885 -2.48355723].       -420.2494284755112
13       [ 9.269165   -6.13947207  3.652865   -3.76806157].       -258.1270742670227
14       [-9.29661104 -4.95456057 -3.38567493 -7.53109359].       -396.7801842780212
15       [-7.84219765  3.23461023 -8.04906866  6.47813264].       -444.6527994770182
16       [ 1.42193987 -5.88473194  4.2213673   6.35798213].       -286.4376255012328
17       [-6.15176384  9.71905132  5.92710716 -1.02032366].       -336.3201549675979
18       [-2.81265395 -5.74190397 -4.52594611  9.34249244].       -484.4311692585337
19       [8.12779714 2.75640128 6.53799595 9.64216013].           -581.3777632061915
20       [ 1.80142901 -0.19124188  6.21119276  1.82345834].       -132.3550412005999
21       [-3.62352307 -0.2756247   9.36336721  9.20561639].       -615.2732860783813
22       [ 4.32784604 -9.39015626 -8.35844439  0.9398328 ].       -408.204241279043
23       [-7.3054603   9.22700221 -1.69156137 -7.95574223].       -485.404367270767
24       [-7.64233022 -3.66459122  5.83243181 -1.81051622].       -200.4273270621861
25       [-1.43206454  0.98025432 -5.12464534 -3.81519958].       -140.9815670330446
26       [ 3.22940946 -8.54389906  2.29605083 -9.65190689].       -544.8782824456842
27       [-9.64633453 -8.64000915  2.55656499  0.0891741 ].       -261.9911680680803
28       [ 4.8092043   5.72160034 -1.0963792  -9.62895308].       -463.0749585433426
29       [-4.57367082  4.70330442  0.96100676  5.21021252].       -176.5164697485229
30       [-0.03900672  5.44993303 -8.40538771  9.47023976].       -630.0984538708466
31       [ 9.81044492  5.32205618 -7.69733457  7.79502216].       -573.6897537549203
32       [ 7.84315129 -7.85495676  9.20373129 -9.44655234].       -795.9911274427625
33       [-8.95431705  7.72705238  9.34552684  9.87068298].       -851.332616514976
34       [-1.50729579 -6.91789741 -3.42581996 -3.06901958].       -170.8708016876460
35       [-9.18667067 -7.59742778  2.61529509  8.71026956].       -523.8312242014836
36       [ 6.59731785 -9.97414254  9.44593105  6.33125206].       -670.5074923271802
37       [ 7.67214643  1.22340892  0.50474236 -2.87021239].       -95.57206088088981
38       [0.72359474 5.81307087 9.95354979 8.72935896].           -670.1334669229193
39       [-7.54755393  5.55091734  3.50874872 -6.30871298].       -314.7243274181966
40       [ 9.48861272  9.59091787  8.21232001 -9.74938237].       -856.5336085083729
```

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

Evaluation	Proposed point	Current eval.	Best eval.
init	[9.34059678 0.94464498 9.4536872 4.29631987].		-430.9815193091966
init	[3.95457649 -5.67821009 9.5254891 -9.8753949].		-742.4213397738982
init	[-4.94035275 -1.30416935 5.58765844 -6.04629851].		-267.7054836935381
init	[7.25986471 9.66801354 -6.72315517 1.94667888].		-390.4072884141779
init	[-9.82027805 -2.26857435 -9.11679884 9.13305935].		-689.7298760679505
1	[0.90405303 0.4880816 2.75220488 -1.97009113].		-39.54269054725363
2	[3.73198569 5.2237111 7.59236872 3.1779813].		-281.8324812545453
3	[2.44668677 7.07389915 -9.38608839 0.44996751].		-371.1722232438818
4	[-1.82164623 7.78625266 -3.67169291 -1.97886623].		-180.6774886890901
5	[-4.94394655 -5.09126603 6.79562789 0.72571849].		-216.9329316343741
6	[-0.64992377 -2.93345677 7.73048423 3.76868319].		-253.7257894445501
7	[4.76684188 7.76155203 8.77574573 -2.83923925].		-406.4924189273480
8	[6.465232 -8.2562289 -5.78689094 5.87653584].		-416.7288703215077
9	[8.79925179 3.1604314 9.38133139 4.96545247].		-460.054494247633
10	[4.72420198 -0.65184323 -7.9968434 7.39016899].		-433.47478782701046
11	[-8.04993793 2.0155275 8.08009116 3.81517369].		-327.0120236508857
12	[-9.17413496 6.49412876 -0.0216068 -3.40881276].		-214.99358734087312
13	[7.4669678 -0.49105387 -2.31165822 -3.48505386].		-120.85156878113192
14	[-2.97227241 -9.48175302 2.9340437 -5.46357932].		-333.8703171303171
15	[9.07920677 -8.80464059 1.21913664 2.21450647].		-261.5504256317644
16	[5.47690173 -0.73324521 0.58878483 4.45399807].		-111.46414758276835
17	[-7.30897249 8.05551792 -4.97066563 -9.81135946].		-642.3774645771106
18	[8.47299609 1.21768899 -9.54223458 -7.21631805].		-556.2209026611812
19	[-7.26713292 9.68917818 6.60979943 7.07148879].		-571.6637290788216
20	[-1.42143776 -5.08019617 -3.66374362 8.36161306].		-373.57261510920404
21	[1.81166733 6.20019232 -8.88610416 -8.09196541].		-578.9750666061898
22	[7.22240637 6.56280788 1.20376972 -4.76882648].		-233.61805691548042
23	[8.15789053 -7.1406334 -8.84919575 -8.91334439].		-721.2440977004662
24	[-5.7252179 7.83796592 -1.45789227 5.34998214].		-276.5111245817975
25	[-0.21495674 -0.63438816 -8.39611869 -1.79253525].		-225.1882609462966
26	[-1.63940945 -5.39486772 -5.08790678 -7.82823115].		-383.6820567654115
27	[9.75534259 -3.4929218 4.78235981 -3.0503703].		-225.39964653535037
28	[-7.04265413 6.13439386 5.68122084 -9.968166].		-619.1466975551934
29	[-3.2352429 2.63088959 -8.61302665 6.58224183].		-420.16627142783
30	[-8.21195759 1.47768352 -8.26625511 -0.93381519].		-280.2843083371928
31	[0.03428082 -9.14036167 2.41295918 2.91511377].		-218.55226745193056
32	[3.12487814 8.35783444 -9.80687408 8.85890148].		-751.9165353335245
33	[1.11931221 4.75302372 4.31366368 -9.95682394].		-498.81178393345857
34	[7.14353471 -7.79955026 7.22323479 8.55306386].		-621.8410248557129
35	[-4.36915981 -5.79418209 -9.92842561 4.31344674].		-456.3788462546732
36	[-3.83163138 2.24931365 3.25809077 9.71770794].		-434.3810799169872
37	[-9.93132703 -5.93815207 1.75753497 6.55121158].		-350.09483681574534
38	[6.265255 8.55385114 6.64643969 9.79775764].		-702.0998596843979
39	[-9.45101788 -9.58186954 -1.8463332 -7.40723668].		-502.6416463197836
40	[9.6924145 -9.42297893 3.11916894 -8.68401978].		-602.3644056711958

```
### 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].		-524.4616052483334

init	[-0.23177622 2.23487726 5.31815713 0.36835976].	-95.43421434439179
init	[-4.06398997 -6.24557543 -8.38517462 4.76880592].	-396.4299395095289
init	[-1.17381554 -6.83380265 7.59874062 -4.51827076].	-349.6612199908222
init	[-1.71529962 -4.07840135 2.57575818 1.5967562].	-66.31107991358093
1	[1.55325716 -9.96715655 0.30945224 2.79590352].	-232.6566151194419
2	[4.56139684 -4.673529 0.23379905 2.52038143].	-90.06336406337539
3	[-2.15134925 9.53418759 4.82671391 6.46019907].	-423.2579596168082
4	[4.17941015 0.37676612 -9.58435076 -5.8403882].	-429.7712508478863
5	[-1.64281038 -0.49239586 -3.85335981 -6.28438287].	-205.7027509068079
6	[5.99523581 -2.91629472 -9.16699548 -0.36155287].	-305.5767027508958
7	[0.41211986 0.47256529 -3.34919958 6.13953029].	-185.0432209307854
8	[-0.25183718 -3.30684237 8.97845951 1.46012155].	-272.2998604293375
9	[5.5415181 3.91207905 9.37361692 -5.36801521].	-440.1735793671933
10	[-4.45934361 6.06017597 8.73696795 -2.92485867].	-356.5602305084666
11	[5.70578654 1.32017865 2.21996359 -6.47121455].	-218.3329296157702
12	[-5.87467781 5.32583313 -8.31959504 -5.69114571].	-428.4443791395114
13	[2.9815035 -8.48201937 0.93978558 5.70416668].	-285.5783291527575
14	[-9.58242141 6.39325499 3.81306329 5.48600232].	-337.5734596849989
15	[6.62625087 -6.52286908 9.86229052 -7.95949353].	-674.2113141583889
16	[9.65440402 5.95164116 7.12483663 3.08692663].	-354.4579372997239
17	[3.03726586 -8.01693119 -1.04416929 -9.07540311].	-470.4899902916590
18	[9.7156969 -5.89977377 -3.89397489 8.8167595].	-520.4395408412056
19	[7.15391269 7.21660365 -2.49156889 1.4360498].	-182.2099061328707
20	[-9.73827868 -7.65361264 7.60275994 -2.93918901].	-419.9508486654189
21	[-7.16849145 -2.61242905 8.92563371 -9.57092226].	-670.4478635548404
22	[-7.39893672 -6.99718047 -3.94407948 -8.91890481].	-517.5200743046123
23	[-0.11167655 9.5823139 -0.84390716 -8.63983797].	-484.3776893738321
24	[7.95822051 9.29053159 -1.57180752 9.83446986].	-630.2401547822719
25	[-9.5137551 -9.52877563 -2.42470317 1.12786197].	-294.8325131409716
26	[8.20310112 -3.59994201 7.94375735 3.50410474].	-331.6348752953627
27	[7.18690545 1.48116608 2.75088061 8.99150002].	-402.1296386680617
28	[-4.33158294 -1.37708037 3.5076389 8.60673283].	-355.7693034886027
29	[-8.81527451 -0.35720276 -0.94654011 1.43124026].	-88.8458616054862
30	[-6.27164306 -8.78410127 9.40092224 5.45388376].	-577.7657861928184
31	[-7.86616406 4.19461375 2.22892018 -8.6880036].	-413.8959877241560
32	[8.65155997 9.21819838 -9.28742892 -8.3325903].	-781.2971049019651
33	[-5.92385062 -9.68619544 3.03044051 -7.85977736].	-497.3918798725256
34	[9.75254154 -9.86413725 -8.03493271 -8.37145655].	-763.7200440750528
35	[9.16072849 -4.22061729 -3.04911592 -4.06652096].	-213.5838616416364
36	[-1.53275275 3.86512508 6.75586768 7.15377169].	-373.8587569563232
37	[5.82348485 5.47992375 -9.89375574 7.34217845].	-603.2616497133873
38	[9.5734532 9.71059272 3.34689136 -7.08737201].	-514.770641680687
39	[-9.37467564 7.27428589 -4.94384797 6.17759581].	-419.6906719067352
40	[-9.79702142 -1.03775527 9.09024991 4.83976418].	-439.7267000766916

```
### 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].	-544.2132200910305	
init	[-7.8468664 1.90104128 0.59634724 -1.62385143].	-80.41569220533873	
init	[-3.29184301 2.45038864 -1.23717148 4.71764213].	-116.4614081942586	

```
init     [0.36072824 1.577172   2.90710192 9.80448543].          -414.9705305999620
init     [ 6.39716394 -1.7359813   7.5253531   6.47518865].      -384.5560589397142
1        [ 4.34664291  8.74699069 -2.96380463 -4.92731805].      -295.3792632862785
2        [-5.57681425  2.83874786  4.53696952  0.08226863].      -108.9971859349811
3        [ 8.56608527  1.00066366 -5.6557535   4.78400723].      -262.8900162206665
4        [-3.88060276 -3.60562409  2.63626196 -8.23698378].      -333.3013666694193
5        [-4.92678916 -8.12144034 -3.72029518 -9.97541476].      -595.7462250530609
6        [ 0.22761873  0.23504219  8.807863   -7.52808231].      -459.5857446178621
7        [-8.50059073 -1.38311772  0.26491341 -2.68077816].      -105.0428955131633
8        [-3.05696573 -2.87038333  2.44675902  5.15470832].      -150.0672009474836
9        [-2.16296848  1.0549018   5.36997986 -5.80365616].      -228.1438188160918
10       [-1.9469586  -7.49626221 -0.88842205 -4.94026588].      -216.1713309721182
11       [ 5.16565529  3.58828793  9.82334313 -3.91747081].      -403.3161362642976
12       [-4.21628332 -0.5792177   4.06056059  4.91923215].      -164.7078679424255
13       [-9.87266828  6.19478468 -3.35719509  5.95853309].      -350.0490362308179
14       [-9.29604707 -8.38751066  6.35426496  9.78206391].      -731.0023081636889
15       [-6.53620178  6.01567506 -4.70249423 -1.18971461].      -187.1006659690734
16       [-5.88055486 -8.69471242 -5.34294918  3.25915367].      -313.9066219423437
17       [-9.10121252  5.55793128  3.33297346  6.97861209].      -372.7435125298843
18       [-3.49163905 -5.19036252  7.22794293 -3.69916265].      -277.5359638964721
19       [ 4.60029783 -7.6833382  -3.64473115  5.6386107 ].      -306.2580300685843
20       [9.23295363 1.41311142 4.18371188 5.17919173].          -249.0476435619113
21       [ 9.23386226 -3.14703755  0.93975378  4.99615186].      -207.5674480812264
22       [-7.24071559  9.57212304  4.29760153  5.1102351 ].      -395.5451895517752
23       [1.65398501 5.21942386 4.30203553 5.07407906].          -215.7280794670408
24       [-7.90638723  6.02936142  9.88101478  9.32707469].      -776.0980053940647
25       [ 0.77200041  9.62659846 -2.18138662 -6.42799569].      -365.4906376493122
26       [-0.70291207  4.05680639  2.39108638 -6.14689486].      -201.6985893474340
27       [-5.11903377 -4.99683891  3.04621959  8.47640559].      -391.3774734122860
28       [-4.65927739 -2.67274277 -2.19018009 -5.80529003].      -185.1922092780074
29       [ 9.5791857  -7.62340548  2.29835671 -2.13668967].      -242.1025224311059
30       [ 6.54205018  2.40126832 -8.32963234 -8.02728349].      -520.2280452584203
31       [-9.99270743  7.04938997 -9.69486923  9.40023195].      -834.6709106068297
32       [-5.17905235  6.05000114 -5.09537955 -1.91364807].      -192.5644848160365
33       [-1.74330707  5.34344546 -2.52507635 -7.61939236].      -311.4925297533266
34       [ 9.06078812 -3.36823727  1.41571386 -8.15029929].      -376.5101770039557
35       [ 3.03015401  1.48505336  7.76143542 -6.05794541].      -341.1070495987792
36       [ 3.82371652  6.64522721 -8.34300974  6.76242933].      -494.678133910072
37       [-0.84589908  6.56578119 -2.97015905  4.94894767].      -211.3683770069463
38       [-0.38903965 -2.73642296 -2.59390311 -0.40842046].      -35.979602158865184
39       [-7.71728666 -1.70003119 -9.82757704  2.93172058].      -389.4604794630251
40       [ 8.4887509   7.53494062  3.7895797  -1.06406981].      -233.2212730940249
```

```
### 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].       -218.9402949114849
init     [ 9.55979024  0.76991741  0.02240927 -8.55897733].       -385.6010135228649
init     [-4.63122040e+00 -2.34998349e-03  3.58459992e+00  6.07478072e+00].       -20
init     [-2.38117734 -8.68127306 -4.23708801  8.19187055].       -478.684726691054
init     [-5.73229293 -0.95752076  8.62412039 -9.50201545].       -618.9724223135001
```

1	[6.75835989 5.37295013 -3.72010646 1.45250665].	-153.36929299141528
2	[-2.26867274 -4.96775263 -3.10681926 -5.91972827].	-223.6337172947077
3	[-7.1620489 2.23165943 9.48914494 -4.53159483].	-413.528573823472
4	[4.29877809 2.09137364 -5.85501544 -4.79573932].	-222.0672605307523
5	[5.54571144 -1.93184305 -4.33889721 9.71942618].	-472.56601814369407
6	[-8.29791508 2.55444133 2.18471017 -8.00103752].	-352.291017157012
7	[-4.37541545 -0.5346361 -5.18090179 4.60429146].	-185.039161236970
8	[1.07740606 7.90693058 -1.60531584 -3.10335].	-172.45414805073628
9	[4.32564604 9.62119313 -9.80342151 4.4727457].	-572.1889646870435
10	[9.91679152 -2.93825443 -0.25081413 -6.81258225].	-301.443262781508
11	[4.86271426 9.61287061 -5.26853416 -9.32001869].	-639.1839028193142
12	[-7.17360778 -4.84967241 6.40731588 0.96189955].	-225.361386687391
13	[-1.95354323 -4.49660935 -0.56509588 -6.77693153].	-228.92052621341074
14	[-3.96682274 -5.05776767 7.16060915 -3.46972848].	-268.8767433914991
15	[2.19617536 8.32020227 3.48291605 -7.99462824].	-435.3231535215285
16	[1.41863484 1.05470351 -4.69092279 5.37098748].	-185.64161983295463
17	[-7.34903531 -2.78590343 1.95047524 -5.36666749].	-196.14837635124087
18	[-6.08936989 4.67588389 -8.44980741 -7.34664296].	-510.8985930649146
19	[7.70140256 -7.00291695 4.88315627 8.18434349].	-496.8628515670783
20	[6.36008573 4.22154581 -9.3808628 8.21213064].	-609.8517078778882
21	[-0.48423379 -1.91873994 -2.3040535 -5.17695284].	-130.72695870445762
22	[0.8663081 8.49846794 2.380057 -2.97697714].	-197.64199016762936
23	[-1.53060732 -4.40544996 -8.1972214 5.76814545].	-375.8280615241591
24	[5.35915716 0.292985 1.96191354 3.5557524].	-91.01306073127876
25	[-2.08000587 6.91069233 5.94203563 1.06121231].	-210.26980979847923
26	[1.7834508 8.18962124 7.52724359 -2.99170249].	-343.09981218345575
27	[8.25477835 2.45208354 3.14975175 -8.1648879].	-376.5911791348866
28	[-8.53174551 -7.24135683 -8.67740592 4.51547898].	-485.1155012345409
29	[6.33950265 -8.4958139 -6.70519534 -9.77466497].	-701.6022360798938
30	[-6.72396884 -8.73495195 -8.73106899 4.65265186].	-513.0939020983822
31	[-6.55621397 3.13032354 -2.64641281 -4.06209353].	-149.5947101475803
32	[5.39547929 4.90665832 7.72371295 -2.08746083].	-273.65898458471764
33	[1.37330385 7.30169397 -5.16091107 1.80765069].	-201.49084623048654
34	[-6.85227879 3.44695012 2.62600612 -2.7844801].	-122.41769688807648
35	[5.77785207 9.69501445 1.503218 9.14237305].	-562.481117799075
36	[-1.3949032 -0.75431933 -7.96830523 5.46245939].	-312.91926513092915
37	[2.95590367 5.52224194 4.13929932 -6.38631837].	-284.2693244832293
38	[-2.94506043 1.50124625 -8.69803676 -2.58980888].	-266.97683202735817
39	[-4.32988338 -6.09934985 -9.42005188 -1.27633322].	-365.88026551138324
40	[1.8230842 3.85897299 5.62694179 -1.50053216].	-137.10079003760225

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

Evaluation	Proposed point	Current eval.	Best eval.
init	[7.46858806 9.37081326 7.3838908 0.61711383].		-396.4929378718437
init	[-5.34543344 -9.77202391 -1.39062363 -1.9529728].		-240.6164747296011
init	[0.45349343 -0.43216408 1.10712948 0.86772035].		-7.268149341268288
init	[5.21791151 4.24749148 2.39364192 -1.47816459].		-89.2374155003189
init	[-4.21849944 9.47710482 -3.32451909 -5.62397878].		-357.10060018148766
1	[0.60170966 0.88443101 -4.01894503 7.30848062].		-264.0378044941037
2	[7.02116965 7.81234303 5.75333375 8.88043753].		-586.1134614902139

3	[-5.28657279 3.22224241 1.64030978 8.03391987].	-314.960866553580
4	[2.01798769 6.23998049 3.20776054 -3.6794717].	-166.970218271677
5	[6.18864373 3.76083688 2.39483588 6.95973427].	-277.5444200427654
6	[5.31957203 0.17720235 3.02971668 -9.32403424].	-403.648655441147
7	[-5.91002641 5.80568061 6.24381112 -1.33276742].	-226.400874666027
8	[-9.27759228 9.22697757 2.87405495 2.86852945].	-314.0423691104986
9	[2.42569589 3.21956414 -4.76633104 -5.95362288].	-236.551423750049
10	[2.20520572 -7.37651559 -3.37397445 1.7426148].	-159.986832927392
11	[-6.34898306 9.83341554 6.32597455 -8.95513905].	-674.5336320624151
12	[8.00107334 -0.44199164 -1.12484661 -1.72916891].	-80.16382793057868
13	[-8.61525402 -2.74447457 -4.68872489 -4.17595369].	-224.993663277944
14	[-6.34006473 1.24595555 9.03530235 -9.91281867].	-681.2671928338723
15	[-8.89405762 0.18184498 3.68315517 -6.1682503].	-272.056539285059
16	[-8.65538444 -9.95000572 -0.8056557 -5.4445582].	-393.441006799035
17	[0.77394804 -4.05218442 9.15227305 -7.22765206].	-493.687515580146
18	[7.65858112 1.1873229 -3.41218412 -4.96882152].	-195.159086646836
19	[8.99471149 -7.06943968 7.98129242 -8.91056538].	-689.554577618884
20	[-3.7303124 -7.91719667 7.6848428 1.59571254].	-326.6348576691213
21	[1.33996667 -3.8816386 6.10050835 7.5768102].	-373.210564461254
22	[8.10196309 -4.71207807 6.96912675 -0.51371413].	-256.810957059475
23	[-1.96559981 -7.6076111 -8.63132834 7.20736715].	-550.899128092505
24	[0.9064848 4.3933078 9.12760828 3.74901493].	-345.5841717460037
25	[3.49120939 0.47218526 8.86783187 -1.27059239].	-255.0074072343243
26	[4.70604571 -9.24361287 -3.09457652 -8.48137246].	-509.499550751348
27	[-4.67750826 8.1568971 -0.7182425 7.08842384].	-357.4796511247296
28	[6.22824474 -6.53188051 -3.40849765 -4.72519097].	-248.285246306715
29	[-9.65934023 9.87152341 -6.81756352 3.32424398].	-471.836711551673
30	[-8.83638633 6.44073684 9.43819723 8.2225006].	-698.7246712984665
31	[1.32232428 5.97115076 -9.97797227 1.77052509].	-384.2766525110257
32	[-9.83001322 -9.60151204 4.93364162 8.43985391].	-638.9542222121304
33	[-3.74524206 6.40648873 2.768778 3.71779783].	-174.399511363025
34	[-8.59751154 -0.89623581 4.58815738 6.69905726].	-318.186719324231
35	[-9.64043577 5.63573891 -9.59696525 -8.34346678].	-711.2200853655197
36	[6.77387947 8.29094342 7.88717361 -9.5531293].	-735.0365695021964
37	[-4.23722401 -4.65556036 5.34390872 -7.32401606].	-361.539477907490
38	[-9.64948571 -5.75933148 -4.25572867 8.0285645].	-471.617443690235
39	[-4.20839573 -9.05466236 -9.81290419 -4.33256618].	-545.6482004818091
40	[5.25390249 -2.43746082 8.87729151 -5.83889565].	-412.275644885104

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

Evaluation	Proposed point	Current eval.	Best eval.
init	[-9.79251692 0.03749184 -0.08453414 -7.32340942].		-310.446939190449
init	[-7.15777829 -5.62882649 -1.62983639 -5.03797663].		-224.095099453932
init	[-8.31880698 -3.0900272 -6.66447307 7.57118171].		-450.8358595558308
init	[9.01928063 -9.22503248 3.98214783 1.45519631].		-307.592760956521
init	[7.96014236 3.33797946 0.95675566 4.04854848].		-153.957203600346
1	[7.46801686 8.12554554 5.16555974 -3.06196092].		-305.3716974158376
2	[1.79155607 6.63416988 -1.83334365 -8.8517705].		-414.7329035737106
3	[9.33019641 -7.52767745 -5.63058289 -0.50920484].		-296.5319700941453
4	[9.24871681 -5.28271506 6.79982967 9.30415074].		-626.3348541920698

```
5       [-3.79629409 -6.6853837   6.54189652  8.77158881].       -539.9528703527935
6       [ 2.12600285  4.95023923 -9.32534107 -9.64414998].       -686.4540986420868
7       [-2.63291278  2.0648516  -8.06169706  7.76658863].       -451.7119279431956
8       [-7.33859604 -6.03880343  3.43779957  7.6829753 ].       -398.3571211179544
9       [ 4.5742908  -2.26634671  4.22642647 -4.96365815].       -183.3364420801886
10      [ 5.90229198 -0.97787337 -7.39679132  1.70640703].       -212.5343887218838
11      [ 3.76231019  5.61395399 -7.52330761 -1.7438099 ].       -259.1519008011605
12      [-2.80325151 -5.66028587 -7.21789028 -7.12490995].       -431.2870786551177
13      [ 4.59063459 -6.63437507  2.59596454 -2.48616035].       -154.0448598387984
14      [-6.7116605  -4.04781318  5.68123009  4.08874643].       -241.5164852169624
15      [ 8.77910755  7.96563606  5.34858784 -0.95326829].       -293.4325022962699
16      [-6.00453231  6.54755256 -3.53825723  9.75186528].       -539.7485958744875
17      [ 4.61339941 -7.99195479 -3.91090888  2.54264043].       -220.7718431688619
18      [-2.21057806  1.20088845 -6.72351962 -8.31126179].       -419.6963598143668
19      [-4.01060296 -0.34982372  8.60881423  0.96692656].       -242.4045244504999
20      [ 0.76122047  4.08559236  4.4817093  -3.30378374].       -137.8806891853268
21      [-1.91232744 -4.09825371  2.92332128 -6.9653553 ].       -256.9504827802194
22      [-1.49303262  7.7941114  -0.46493609 -0.37276328].       -124.92979795666568
23      [ 5.02159077 -5.17831978 -6.91006052 -0.58676192].       -223.4703326794803
24      [ 5.319375    7.52329844 -3.17784069 -8.25196861].       -444.1717475845218
25      [-5.38540934 -6.04055244 -8.33599896 -4.84152513].       -404.2072796206927
26      [ 8.62767184 -3.2594571   7.51514305  3.31157527].       -308.983091083153
27      [ 7.59743528 -0.24000876  9.10743465  9.12356887].       -639.63036464057
28      [-0.78769787  9.1489014   2.08254341  9.14955056].       -515.8933245010221
29      [-9.6547581  -1.49679449  1.00515371  3.34367319].       -145.4467450012392
30      [ 2.60369403 -1.68571753 -4.43767666  7.25026529].       -281.8068192321635
31      [-6.1692829   8.70589311 -5.26969916  0.31795138].       -273.3587612579158
32      [-1.18542896  0.40789086 -6.59527104  7.70779768].       -369.8713724176066
33      [ 3.8845512  -5.42246098 -7.03261448 -8.49555409].       -510.9666034454216
34      [ 5.31529745  7.16306101  4.53196711 -9.55319334].       -557.5414624297882
35      [ 3.06205081 -6.73516315  9.16175403  3.84221381].       -410.9646387419819
36      [-5.99732874  8.49617413  3.35263634  5.54524537].       -337.0573978164796
37      [ 1.08245553 -8.94111158  1.08361444 -1.57389027].       -174.4898456401515
38      [ 1.21816957  5.83576602 -6.28781207 -6.81494174].       -373.9797329812589
39      [-5.08588753 -2.04676062  9.62851081 -5.07771755].       -415.5022333930536
40      [ 6.31302522 -3.63294911  7.2082138   9.81186853].       -607.2170203390922
```

```
### 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].       -333.6685278040737
init     [-0.02985975 -5.50406709 -6.0387427   5.21061424].       -278.5916442165977
init     [-6.61778327 -8.23320372  3.70719637  9.06786692].       -549.5010991841266
init     [-9.92103467  0.24384527  6.25241923  2.25052134].       -236.0834739637049
init     [ 4.43510635 -4.16247864  8.35548245  4.29151567].       -337.4333129667272
1        [-3.98599887 -7.72031276  6.57362653 -9.06207361].       -593.2170550074999
2        [ 5.24133604  6.79774962 -7.90085176  0.21262617].       -307.3416184837157
3        [-6.98285725 -7.53553636 -0.44064091 -4.85155595].       -257.0617857366076
4        [-1.78826799  3.25598727 -1.45510672  1.58046094].       -40.74424241912651
5        [-1.40205459 -1.08756953 -5.49198735 -3.68161027].       -149.0341638516993
6        [-7.83635489 -1.28730572 -7.74844317 -2.63232565].       -272.5544380760249
```

7	[-5.20012595 -8.20174682 -6.41522866 -9.36379388].	-635.7666315037931
8	[8.75235832 9.38171401 9.18458426 -0.57986069].	-507.0516092544958
9	[0.09080076 1.54250591 0.74156137 -6.26579398].	-163.4573303748934
10	[7.61866635 3.1582889 -9.92206389 -2.11139281].	-391.1676285103485
11	[-5.27427457 -6.40291003 2.75132161 1.08965876].	-137.2712224299336
12	[-6.52612311 7.27650284 -4.77362568 -5.95519154].	-358.7050015084682
13	[-4.63875915 8.6124843 7.12702008 -7.73195445].	-561.3835822655649
14	[1.97874821 -3.6500026 -0.91985358 1.11445038].	-38.066872921725896
15	[-0.68593392 -5.63554024 -8.94330918 -1.26699516].	-310.3585769502361
16	[-1.72806915 -4.12214739 -2.17527422 6.47352369].	-218.7919109183586
17	[9.77980853 4.77521454 2.07686121 -1.93264192].	-169.1304793162748
18	[6.72458671 7.60340645 -6.58863506 -4.60938833].	-376.0598243526946
19	[9.49546442 5.26511905 -5.65588327 8.33754155].	-519.6322449518909
20	[8.44493706 -6.68369259 6.16068353 -8.53242366].	-565.7315340365443
21	[5.92388227 4.62357418 -2.61549919 2.52758312].	-123.92447143460122
22	[-0.225636 3.78441799 -1.70036618 9.35169568].	-387.1851346713122
23	[0.06951243 -3.70532939 -9.7893336 -7.65624326].	-549.4291642409903
24	[6.86181738 8.02777689 4.32733578 -9.39873447].	-585.4972848497147
25	[-1.28331277 8.23518986 -8.29753762 5.6102354].	-469.7299524281028
26	[-4.83093129 5.08423132 7.84230321 6.7211185].	-440.2356079427068
27	[5.35675599 -9.63550681 -5.43181407 -9.73961761].	-682.3352346696515
28	[-9.60014845 2.679852 -4.36546846 -4.53937851].	-246.1218373334794
29	[-8.34805762 7.39914299 1.52335969 -1.54052628].	-195.63945921068625
30	[8.09691188 -5.29235202 -2.08697079 -6.19971526].	-288.3901801368731
31	[9.69021811 -5.08822733 -6.00588414 1.29865971].	-260.6384426685979
32	[0.48985277 5.60659539 8.92177354 -0.54899036].	-303.1074706825371
33	[2.01855246 2.65802605 9.14801825 -9.0221554].	-594.8606244069149
34	[-9.97222211 -8.7542074 -9.22111918 2.24905585].	-528.0376340046972
35	[9.24425402 5.72914572 7.9608583 8.31970138].	-618.0979726361129
36	[-1.19008575 9.11454743 3.96726914 2.83945942].	-247.03404641138022
37	[-9.64107024 -4.0751099 -3.48110988 6.50535551].	-331.7962561372619
38	[4.91751685 1.96038685 3.35896035 2.19411135].	-84.97254763886573
39	[-9.85819989 6.65223147 -1.60745661 9.49758397].	-554.2566273197592
40	[4.93882826 9.82928676 -1.70148576 -7.44832242].	-448.2169701835680

```
1 ### EXACT GP EI GRADIENTS
2
3 np.random.seed(run_num_11)
4 surrogate_exact_11 = dGaussianProcess(cov_func, optimize=opt)
5
6 exact_11 = dGPGO(surrogate_exact_11, Acquisition_new(util_grad_exact), objfunc, param)
7 exact_11.run(init_evals=n_init, max_iter=iters)
8
```

Evaluation	Proposed point	Current eval.	Best eval.
init	[-6.39460622 -9.61049517 -0.73562947 4.49867858].	-308.1901117295372
init	[-1.59592791 -0.29145804 -9.74438371 -0.25256785].	-287.8310851024486
init	[8.83613305 7.01590179 4.5992894 -7.82527856].	-484.9233304137926
init	[7.87808341 7.14308494 -6.69826765 2.64668028].	-326.7315575089055
init	[-9.59032774 -7.66525462 -3.67265377 -6.84175387].	-437.1901841268746
1	[8.98204795 9.73346661 -3.23891901 -5.20250642].	-409.8940110092612
2	[-2.46813998 -7.3003779 -2.91425156 2.82909317].	-170.1764091752233
3	[8.5668058 2.92682213 8.40866209 -9.14225066].	-636.9625206197511
4	[8.76589534 -3.7692322 0.38597262 -4.05301551].	-171.4098073829174
5	[0.40212877 5.87654958 -6.74210512 -8.93336634].	-524.8174583999808
6	[-9.57187238 4.8761458 -5.96218891 3.90472017].	-306.8047847038760
7	[6.45130653 -7.70340012 7.43044673 -4.70207094].	-414.3766032051364
8	[6.63688003 -6.08240957 -3.75262294 -5.63404841].	-287.2561312275786

```
9       [ 1.38380213 -6.05647422 -9.74783179 -9.20728444].       -699.4336893349999
10      [ 0.01626441 -5.441819    4.18137887 -1.59234324].       -121.8210682508479
11      [ 7.11903192 -7.10840682 -3.70390674 -0.4859192 ].       -193.84075563495688
12      [ 3.41314276  7.39306433 -0.25317011 -0.71476575].       -123.20018963353341
13      [ 4.20641113 -2.1133109   2.2765559   1.21529123].       -48.08191183894172
14      [-7.1993438  -9.20362182 -0.3916158  -1.86759383].       -235.6555759082105
15      [ 9.69592013 -5.84480625  4.17324565 -8.23515375].       -485.8533541656186
16      [-4.94462415 -1.46993221  9.55860225 -3.21514837].       -344.2200563624783
17      [ 7.47977131 -2.49700451  5.68213219 -1.90089034].       -179.73045701991214
18      [3.89164787 3.97619773 0.52806077 3.16374216].           -87.63882221674211
19      [-6.55740597 -8.48195204  7.41906164  7.02169754].       -549.2309662447117
20      [ 0.7171927  -8.25397114 -6.37154321 -7.07199278].       -458.6124608668655
21      [-9.44533482 -3.69691898 -3.31658743  9.25891862].       -492.45832198468145
22      [-4.77974029  4.4732243  -0.45397362 -2.86226686].       -96.25395078503722
23      [-9.21795226  2.71476159 -5.99198225 -6.6637393 ].       -385.04374439197966
24      [-0.12265947  3.66898976  4.96724137  9.01897889].       -426.3263982387445
25      [-6.31817008 -7.90763218  7.25451881 -7.85973695].       -569.9665553407875
26      [ 6.10093659  5.64790105 -7.90934599 -1.88509981].       -302.90666713851715
27      [-2.32720378 -0.44141586 -9.30505364 -1.91542506].       -280.23305566499056
28      [6.29872458 7.22784721 7.929719   2.39774872].           -355.79560816666776
29      [ 0.26473028  7.96300463  7.77799867 -5.30152514].       -420.8054329295035
30      [ 5.81582384  2.62152108 -4.57770466  9.5115626 ].       -472.31398517595926
31      [ 4.65942108 -5.58147692 -6.96884962  6.8803859 ].       -419.06940946777536
32      [-4.548936    8.57580646  0.00935452  7.85312187].       -414.46808689868635
33      [9.36659365 2.03213989 2.70519731 9.33090731].           -466.209864278647
34      [ 9.42753538 -4.45432105  7.90433013  7.33298397].       -531.0862949399032
35      [-8.58674822  8.97786362  9.96749075 -3.17070739].       -573.2024723135487
36      [ 0.2617658   9.04037674 -8.35631646  5.455243  ].       -492.04812403801554
37      [-7.78667689  8.21015527 -0.89730407 -9.20268219].       -536.6185380444982
38      [-8.90818169  8.87497474  5.77863875  3.89057944].       -397.610485165921
39      [ 0.0899502   0.2500267   7.2864013  -9.81793586].       -544.9755076005299
40      [-5.23980383 -1.97589742  3.58761538  3.17968032].       -114.31830550232989
```

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

```
Evaluation       Proposed point           Current eval.          Best eval.
init     [-6.91674315  4.80099393 -4.7336997   0.67478787].       -162.98551438332169
init     [-9.70850075  8.37494016  8.01429708 -9.33157145].       -775.5340081417003
init     [ 9.13898673 -7.25581357 -4.32343294  2.12166369].       -262.896783965427
init     [ 8.88450272  7.05471082 -9.95481533  0.42452054].       -476.48819368144933
init     [ 1.04075267 -0.29245173  5.36268308 -6.78566494].       -271.7103262017248
1        [ 9.00627049  5.34951301  6.50018506 -1.86719396].       -279.05035795587855
2        [8.11179655 4.25164412 2.83994041 0.51622188].           -127.21592334362715
3        [7.07957877 4.20923579 8.31123577 5.90287485].           -432.1614135764528
4        [ 3.65158786 -4.292633    3.33049006  6.75429274].       -265.94586422179236
5        [ 3.07120909 -3.55916591 -5.97833992 -2.63162   ].       -169.69098906486255
6        [-8.34831975  3.35126666 -1.83244524 -0.35201807].       -102.72565265308921
7        [-6.77288607  5.95533828  4.43070962  2.50634443].       -200.82470642809477
8        [ 9.11568286 -9.45162454  9.96288854  5.4127943 ].       -676.7328994714871
9        [-0.54343888  6.9130662  -3.08529747  0.79842711].       -126.98341914051196
10       [ 9.87704008 -9.57194896 -7.30925724 -2.85091279].       -473.5868734997437
```

```
11      [-8.01556949  4.33863986 -9.09107963 -5.10325364].      -454.0129230656223
12      [-7.70853825  9.04000734 -1.47997073  6.83741383].      -416.4368787746895
13      [ 8.2042246  -8.88295657 -0.07055749  6.74550618].      -407.1454856720997
14      [-5.69729876  6.22751761  8.3350766   9.36945888].      -669.5907092612638
15      [-5.77909101 -5.04328215 -7.7933351  -1.05081511].      -270.8923482234484
16      [ 4.71575825  4.71416309 -2.11686206  5.24056754].      -189.9825505895613
17      [-7.8093924   0.25400768 -8.20800109 -6.68611527].      -442.0460447682908
18      [-9.43342891  5.47058984 -1.80600297  0.74344179].      -160.8400504984310
19      [ 3.45355371  3.20265039 -5.41025361  2.04079201].      -136.9128325139473
20      [-8.69701704 -3.37114475 -9.15259224 -0.82149058].      -352.3765606174113
21      [ 9.94633892  4.14929651  9.20784367 -1.98509533].      -403.4785500618005
22      [-9.44413253  7.85647531 -5.46813037 -2.76954875].      -333.0229982624914
23      [-6.34363966 -3.26687378  8.41196286 -0.3285002 ].      -274.301699618427
24      [ 4.20796658  8.79614205 -5.71197443  7.9768651 ].      -524.8526751197423
25      [ 2.30229124 -0.93404272  8.38361118 -2.73516381].      -247.8247100223423
26      [-4.51712007  1.04990949  6.57777777  4.88865804].      -248.0063841418865
27      [ 3.59342012 -4.54813843  8.78430934 -4.99047946].      -385.3956074922122
28      [ 2.8327017  -6.12881065  4.97078983  3.7126317 ].      -212.4096302678987
29      [ 2.40471286 -1.77558974  0.60610937 -2.25237082].      -33.48288474225401
30      [ 0.96407724  3.41164969 -3.18561901 -6.74939201].      -236.8698276729727
31      [-2.69182932e+00 -3.74057312e-03 -4.12328311e+00  6.63963874e+00].       -23
32      [ 2.1826784   7.09850311  9.34939513 -7.58720838].      -598.0380695859511
33      [-9.37407816 -4.87473518  3.48020019  5.57927244].      -296.2479315566874
34      [ 8.21600758  6.17182177 -4.06401172 -9.55146003].      -558.1556768677428
35      [-4.82867996 -5.81918983  6.49892693 -8.99566378].      -541.4381114503258
36      [-4.61329278 -8.66242128 -0.58239269 -3.4873965 ].      -221.022836159711
37      [ 8.94092981 -7.86645078  2.77085448 -2.37052644].      -249.2128076152337
38      [ 8.18669268  2.17729847 -9.70838708  8.06387937].      -619.3661355586727
39      [-8.38927662  5.10475433 -8.66079188  8.47480466].      -634.8141996833203
40      [-9.28072647  1.80933599  0.81593588 -8.55403877].      -387.3628486026613
```

```
### 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].       -559.2187253383422
init     [ 9.45202228 -0.93101505  2.18084926  5.51053029].       -226.8063900209954
init     [ 2.8322669   4.44036459 -9.29926952 -4.03101058].       -371.880837115183
init     [-8.82975016  7.14121885 -2.54291944  3.59695903].       -251.1102762788094
init     [-4.87440101 -3.0483757  -9.8117446  -2.83332435].       -363.2668776074504
1        [ 9.1114829  -9.99975933 -5.06042598  4.24465356].       -431.9015624918471
2        [-9.72135464 -7.69244795 -6.22844507  7.60723681].       -560.7130380018729
3        [-8.34192968  0.21366179 -9.02437947 -5.9293723 ].       -454.6271916844164
4        [ 5.90709134  3.33477276 -6.9423112  -1.38736973].       -209.4213805092718
5        [3.60696134 2.18226175 0.08284273 4.5407231 ].           -105.0279565392094
6        [-8.12476166 -3.94190857 -9.00862896  0.63619536].       -342.1742038873105
7        [-8.87693875 -6.00070012 -7.77180414  6.2427699 ].       -487.9083684337456
8        [ 0.65604441 -6.68523471  7.23179469 -8.11619975].       -510.202477173446
9        [ 7.28906482 -5.39350739  9.45774561  2.8390882 ].       -411.8988533422961
10       [-8.6173938   7.4519909  -9.59943625 -2.01230926].       -477.9688957311798
11       [-5.02166508 -3.62694175 -6.53971562 -2.35341132].       -201.9843533943209
12       [ 9.32398288 -7.33660406  5.53755865  8.82281307].       -597.9499642728872
```

```
13      [-0.75891568  3.78629    -1.56788422 -2.47440102].       -61.11336149575307
14      [ 8.77564289e+00 -8.65206420e+00  2.48741980e+00  5.80705633e-03].       -24
15      [-0.11208207 -9.37475273 -4.74485642 -5.54088705].       -366.1312448050868
16      [ 1.88062263 -3.7928668   1.11976485 -0.41307752].       -36.752570804531274
17      [ 1.35184205  2.8137708  -5.95001573  9.33790363].       -472.6559278604251
18      [-5.95616198 -6.51829453  3.11491075 -5.269219  ].       -260.6188749589722
19      [-8.04866188  5.49103451  1.85986981 -8.93405996].       -454.7309344445191
20      [-6.49205315  1.64236452  2.72273901  0.71872096].       -71.84763898887022
21      [ 7.492922    9.63861344 -0.96382108 -8.02431679].       -502.2951114079942
22      [ 6.80069021 -6.2200273  -6.945806   -8.55049092].       -560.8031093869174
23      [ 0.36021503  1.57341243  6.50797923 -1.95073231].       -147.36381512419118
24      [ 9.08087578  1.57911542  4.78823804 -8.22972753].       -427.144847432456
25      [ 5.73806422  8.92019293  7.19589597 -6.50032196].       -516.4245636402535
26      [6.59092781 5.95539015 9.06337472 2.02599095].       -377.2265145484946
27      [-8.69981076  9.82177991  6.66216341  5.7821664 ].       -535.5084858066066
28      [-6.35598742 -7.27908156  5.61501831  6.75881706].       -423.6803567760462
29      [ 8.25106717  9.07404702 -7.66322012  8.13279655].       -673.5011151878934
30      [ 2.77519336 -8.75741623 -1.16622916  9.13241349].       -498.7705523563129
31      [-4.82546615  2.79950892  8.00901169  8.80113831].       -541.2325709683021
32      [-1.60742232 -4.72458753  9.93864492  0.87456997].       -346.6167402705331
33      [-5.46982301  0.19457571  9.54171892 -7.89731155].       -552.5980019035239
34      [-9.22219849e-03 -9.41507799e+00 -7.98808083e+00  4.13870015e+00].       -43
35      [8.07821828 9.24282107 2.68655038 7.35346785].       -474.0637098247653
36      [-2.54787989 -2.73086776 -4.16336494  5.38455445].       -189.381498998145
37      [-1.39950129  9.80866879  1.05065622  9.78159231].       -580.4083991129226
38      [-4.2899857  -9.65626394 -0.99979073  2.44906824].       -231.88132930574838
39      [ 5.36768215  9.70592648  2.14490831 -0.56135479].       -232.28440129399903
40      [-2.75553328  9.6891664   7.18746512 -3.24277359].       -392.3941415040586
```

```
1 ### EXACT GP EI GRADIENTS
2
3 np.random.seed(run_num_14)
4 surrogate_exact_14 = dGaussianProcess(cov_func, optimize=opt)
5
6 exact_14 = dGPGO(surrogate_exact_14, Acquisition_new(util_grad_exact), objfunc, param)
7 exact_14.run(init_evals=n_init, max_iter=iters)
8
```

```
Evaluation       Proposed point           Current eval.          Best eval.
init     [ 0.27886688  5.46330104  7.40855371 -9.83906103].       -611.6615754592598
init     [-3.80528149  9.15207479  0.26233425 -3.6343115 ].       -235.04045120202608
init     [ 0.78399875 -5.57490115  6.12962716 -3.15490749].       -215.30445206449784
init     [ 0.77777698 -9.88252429  3.46304956 -5.79951476].       -366.4491321572651
init     [ 8.65115186 -2.51510501  5.04837844  5.26278004].       -274.73972442923775
1        [ 4.8502271  -1.84168373  9.82765998 -0.95725402].       -323.7223438501245
2        [-4.77473587 -9.74252485  0.96090492 -9.55982645].       -580.9628252026488
3        [-9.92742519 -4.56358619 -0.03785913 -6.46617548].       -307.4564097025707
4        [0.19437843 2.05845053 1.65792191 5.18484457].       -124.28878831402214
5        [-1.31829204 -2.32031879 -2.5251031  -6.82350009].       -217.87470331061357
6        [ 0.24276126 -4.12052041  2.21586778  7.93881754].       -300.8458156729556
7        [ 4.79563633  4.13838411 -7.01002178  7.65374311].       -438.99092445744697
8        [-8.61484726  8.56315525  8.59846951 -0.87156311].       -445.71037165155775
9        [-1.22137626  2.83846894  3.6206131   5.92588181].       -197.3963904411382
10       [ 2.54244816 -2.26775385  6.69847887 -4.22542869].       -222.77530569702088
11       [ 5.03921566 -0.10483683 -6.92496957  4.51444489].       -250.80213727545723
12       [ 6.82279714 -0.09161239 -2.028231    8.81142257].       -369.47318014324986
13       [-8.96893842 -9.80111106 -6.18093935 -0.65507665].       -388.89394807455415
14       [-6.79730927  6.67022322 -4.17180278 -1.34367333].       -194.62081643323853
```

```
15      [-6.15510499  2.90554086  9.18832867 -8.99547985].      -631.7204350497341
16      [-3.45035456 -3.84880861  6.14592988 -4.05004414].      -220.4603945883092
17      [ 6.31982958 -8.10015369  6.28733636  3.72133219].      -345.1502741051454
18      [-4.1962459   8.74327257  3.07792362 -3.43828096].      -246.2060553607907
19      [ 7.56622239  1.19315477  7.79884993 -2.17352434].      -261.4579708896355
20      [-7.77223206  2.88456515 -5.76179988  4.45343241].      -255.9762773464268
21      [-0.12388008  8.42440271  0.8938253  -5.69235191].      -273.9647205762494
22      [-6.17602376  5.74038283  2.81136066 -3.07356876].      -165.5458055258876
23      [ 5.22498155  9.63662115  0.71281827 -6.55975151].      -386.6750556756115
24      [ 3.16163099 -7.41787886  9.3383186  -7.47424887].      -605.1159315450858
25      [-5.41504132 -6.2939085   8.79453795 -5.38270416].      -456.4749505823591
26      [-2.19162723 -1.00437391 -6.54627731 -8.64645046].      -434.426426104876
27      [-8.01189683 -0.39981096  2.62732101  7.36608046].      -302.2552006610292
28      [-1.57727485  2.09123977 -0.53319287  7.30830626].      -225.7326087991825
29      [1.349273   6.70863794 1.93835752 6.58155122].          -276.3711392771723
30      [ 6.15573528 -8.96314526 -0.77934299  2.38573627].      -223.1580996941950
31      [ 2.0109895   0.71797124 -0.34174846 -6.35836892].      -167.1408414098029
32      [ 3.66886588 -6.97835601  4.98345433 -8.59838295].      -481.088690679187
33      [ 3.30130157 -6.57503668 -9.3979205   1.10333169].      -367.1928992268907
34      [ 8.58796337 -3.84470179 -9.43423321 -1.58187498].      -380.340160783244
35      [-8.21361433  2.64733509 -3.12534377  0.08413083].      -110.8118595582445
36      [ 0.67700616 -9.84109187 -5.33539649 -4.94960676].      -377.5463111769999
37      [ 8.89834037 -5.46605614 -6.42257075 -5.19338335].      -370.5691683505234
38      [ 9.49126672  7.54900651 -5.0903024   1.16455439].      -287.217425686323
39      [ 0.62477561 -8.91363205 -2.55768355  8.92968243].      -497.8781657834703
40      [ 4.24459215 -8.48876313 -3.33299209  0.63828868].      -197.0909199801751
```

```
1 ### EXACT GP EI GRADIENTS
2
3 np.random.seed(run_num_15)
4 surrogate_exact_15 = dGaussianProcess(cov_func, optimize=opt)
5
6 exact_15 = dGPGO(surrogate_exact_15, Acquisition_new(util_grad_exact), objfunc, param)
7 exact_15.run(init_evals=n_init, max_iter=iters)
8
```

```
Evaluation       Proposed point           Current eval.          Best eval.
init     [ 6.97635395 -6.4220815  -8.91273571 -2.76923108].      -400.1409127715913
init     [-4.49198143  0.6000045  -3.88162169 -3.91051282].      -127.2673106314054
init     [-7.76517448 -5.00201972  8.35259796 -4.71706293].      -408.6387460656439
init     [ 4.35547375  7.31430068  6.14158964 -5.78898835].      -373.1750549864325
init     [-6.65513937 -9.06587217 -9.21155376 -5.9953838 ].      -607.007632459987
1        [ 2.8544977  -1.6003636  -1.93515132 -2.19159348].      -43.717244223241735
2        [4.92526955 5.96067446 4.10838652 1.54460964].          -155.4973553761591
3        [ 7.60284646 -9.83670929  6.07418637  6.51677002].      -531.8853593071874
4        [-3.39221728 -3.69203211  3.60318038  4.66933918].      -164.9289803785776
5        [ 9.63144016  3.76660236 -9.40395266  2.32534909].      -408.0711964371901
6        [-0.46099561  9.58086516 -0.66748888 -7.87674906].      -433.3077986758719
7        [3.72122725 0.49817372 5.20397899 9.4368106 ].          -451.8016557305748
8        [-1.96659974 -2.21436105  9.64281216 -9.29814465].      -638.4477590949542
9        [-0.8289478  -8.11611777 -5.43376832  1.07172118].      -225.6017494454485
10       [-0.90690653  4.09601032  6.89554636 -6.11770942].      -326.7282335927336
11       [-8.83646821 -9.52000508 -3.03305299  9.46202552].      -645.0621025739723
12       [-5.20285545  0.11177617 -7.97281939  3.50515359].      -266.9366462934659
13       [ 6.16909346  9.45497947 -1.19118224 -8.24835539].      -493.2491995569618
14       [ 0.93691127 -4.95007973 -0.40679652  9.18253617].      -387.6567136784933
15       [-7.20628386  8.66608008  6.24591119  2.13266135].      -337.3596125159138
16       [-6.14779635 -0.43746793  3.56596824 -2.19771089].      -95.64627741914462
```

17	[1.30223912 -8.21616194 4.9325588 9.93275327].	-604.335219875349
18	[4.42929495 8.64709913 2.69581539 8.66421089].	-491.2397638236875
19	[1.14120333 4.18919902 -5.55151656 -7.17257482].	-334.6424488926178
20	[-0.96678485 -9.96823422 5.44886466 -3.6837932].	-343.0177673690593
21	[2.37201846 9.77107763 -3.75463112 -4.47165852].	-318.8490715089949
22	[-8.39053544 9.20499859 -6.65884284 8.09859265].	-635.234458453372
23	[-9.21357548 -6.74167545 1.0076465 -2.03849531].	-195.4582557300187
24	[-5.45975356 7.48081612 -2.98212522 -2.50654362].	-193.5443847163550
25	[9.54989234 -4.02397049 1.45709121 0.52136911].	-131.0417680584376
26	[8.57773458 -7.80604479 3.32206499 -5.22684115].	-337.8340217953778
27	[5.60066467 -3.87589042 -5.27284941 -7.26059882].	-355.6865014047286
28	[9.08733404 -5.64771707 -7.37824419 5.82540836].	-445.4300482738199
29	[1.13244521 9.49865276 -8.11469415 5.63624992].	-506.3452768017615
30	[8.36219672 9.80306436 9.96457014 1.990924].	-575.8595635674714
31	[6.2784228 -7.31871577 0.31170191 8.68142997].	-448.3061736856189
32	[-9.71396708 -2.10459634 6.69994244 7.91517354].	-488.4873827048736
33	[-8.14471549 0.62143936 -9.73831406 -8.84037068].	-664.2216611475224
34	[-1.5364281 6.887651 0.84061874 2.88926397].	-132.7513886921427
35	[7.17039774 1.88398051 4.17739472 -9.42407261].	-466.1178267830978
36	[-7.34093766 9.86020908 -7.42130562 -6.60187143].	-587.9029687025339
37	[-1.93342917 -4.84402422 -8.9163958 9.21804539].	-629.0630753401065
38	[-9.98712406 7.81056755 -6.70440634 6.67862108].	-535.0156893708369
39	[-7.5403735 6.64154191 3.56054747 9.93414717].	-577.8590049798895
40	[-7.52112501 8.79216844 9.18982572 -7.76365058].	-705.6275448275153

```
### 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].		-364.5052686545986
init	[-2.78542329 -5.53838117 3.77452324 -6.7253715].		-292.7694790431682
init	[-8.59350266 8.8202172 1.2736276 -8.44015321].		-519.2518779260945
init	[4.45281022 -6.83095653 -4.99437387 -4.13025488].		-256.2187854356974
init	[3.93221428 -0.71471824 -5.69875713 -1.06347476].		-118.4353660802023
1	[-0.64007705 3.2211867 3.79933264 -4.73341031].		-154.0872642720883
2	[5.98502725 -7.94806028 -2.90882686 9.11344119].		-519.7669380533303
3	[0.47384394 -7.09622115 -6.44235327 3.14296454].		-264.9618889185273
4	[5.52616367 4.65190076 3.38866017 -9.07435877].		-437.6438475693719
5	[4.28544997 -2.78264932 9.08402216 2.69441349].		-310.4491882302683
6	[1.87133514 5.63089801 3.59434817 4.26424392].		-178.4090409147218
7	[7.45907335 5.14332641 9.11613095 8.2671896].		-631.2426140402113
8	[1.24349389 9.55943585 -9.57929245 8.27200487].		-733.3046944193816
9	[-3.08885414 6.03421433 8.15398859 0.37041672].		-282.3759287866137
10	[-1.22767225 3.39992625 -0.74204049 -9.66914731].		-400.2476871959373
11	[8.89995329 -8.45639035 4.20210818 2.73243867].		-305.0682678921419
12	[-3.05457595 8.6513061 -8.42526724 -0.14730159].		-372.0628037710214
13	[6.62003549 -4.05866358 -4.45226831 -1.87309798].		-150.2724334285981
14	[2.62583757 0.46449641 0.44128596 2.38948669].		-30.7493233028567
15	[-4.41578116 -0.58548658 -4.47862442 5.06802468].		-183.0984391906081
16	[-6.05422889 7.09537147 0.12574541 1.03270007].		-141.6555937219469
17	[9.57679297 4.69037918 0.54002543 -2.39908886].		-159.6116691462973
18	[0.32007534 8.91304125 8.80232332 -9.7890119].		-774.7287599185722

```
19       [-9.28865296 -4.59916929 -9.1837648   7.6448698 ].       -615.3845345672883
20       [-8.09573386 -0.93369103  8.83705044  0.93227014].       -305.0413562565172
21       [-2.27514105  7.082857   -1.13091427  9.91441809].       -502.5296391753138
22       [-7.81128991  9.32828099 -7.39172325 -7.31409589].       -612.9466149146473
23       [ 3.90903595 -9.29697973  8.66198784 -2.99769656].       -449.183064922232
24       [-9.80337607 -9.9481815  -3.97856961  3.61462538].       -393.7879278061264
25       [-9.97009275 -8.42436269  6.62253575 -4.39221729].       -450.0827532080511
26       [ 8.0030186   5.27227521 -8.88730344 -2.93812009].       -391.1247645611562
27       [-0.06292031  3.28777901 -9.14027762 -6.50004619].       -441.2593673651187
28       [-4.76599638 -4.23637349  3.30560799  5.97208919].       -234.0529718164282
29       [ 0.538688   -9.45668017  5.20639036  9.83213664].       -647.1509297973578
30       [-9.19424087  0.9011799  -6.84776533 -5.52750894].       -349.0474058849186
31       [-3.2856625  -5.10697535 -8.72753993 -6.52804627].       -461.9293846989898
32       [ 9.89841122 -5.45567637  4.81688809 -6.81160717].       -412.7065553136335
33       [ 2.88365439 -4.50916064  3.88716858  4.45474821].       -173.6898699362777
34       [ 4.17202516 -0.86942292 -9.35266964  6.84379188].       -468.6848237407342
35       [-8.82583007 -8.21127663  9.05100865  9.62584087].       -829.1349267741234
36       [-0.65241218  9.60736151  9.67068509  8.52727514].       -756.4525675956647
37       [ 8.71670976  7.06490479 -6.87651402  8.95659051].       -638.5481778852303
38       [ 8.32095004 -4.12199346 -4.28042452  6.33822232].       -318.8782207258316
39       [-9.92192902 -2.31656907  0.50267992 -0.18318107].       -110.0699424548504
40       [-9.22064279  6.51989365  5.90473959  6.12475725].       -424.6867342069503
```

```
1 ### EXACT GP EI GRADIENTS
2
3 np.random.seed(run_num_17)
4 surrogate_exact_17 = dGaussianProcess(cov_func, optimize=opt)
5
6 exact_17 = dGPGO(surrogate_exact_17, Acquisition_new(util_grad_exact), objfunc, param)
7 exact_17.run(init_evals=n_init, max_iter=iters)
8
```

```
Evaluation       Proposed point           Current eval.          Best eval.
init     [-4.10669995  0.61173511 -6.16958426 -8.64199284].       -430.5408947278945
init     [5.7397092  3.12667044 2.75041792 1.51205788].          -84.33607001372373
init     [-9.21874168 -2.84372791  8.91366374 -8.79910639].       -649.2160718740936
init     [ 7.28084207  7.54581052 -8.97612669  3.04837231].       -445.7720200383797
init     [ 1.03502737  1.95026506 -0.32942751 -4.34023678].       -84.3545380937588
1        [-6.82082511  3.53524771 -7.6305547  -1.10007868].       -251.0363955498036
2        [ 6.36132887 -5.79483735 -1.80465394 -5.23795116].       -227.1416416194343
3        [-9.93358214e+00  6.76426027e-04 -8.86041968e-01 -1.86772363e+00].       -1
4        [-3.57261465 -5.71224609 -1.88427713  4.42421021].       -166.9691310040166
5        [-6.06192646 -5.22204103 -5.99548203 -7.48165956].       -423.024711028853
6        [-9.15358294  6.24405427 -7.34906493 -3.27748646].       -366.7584440566941
7        [-8.32522503  1.10793827 -8.79126558  8.08570378].       -565.1378998269158
8        [ 9.08506073 -7.59845958  3.69354279 -6.90608989].       -429.7145899308244
9        [-1.16236813 -7.73189354  3.93657692  3.54340093].       -217.6281294637466
10       [-9.19308579 -8.75896264 -4.61339917  6.673686  ].       -479.9543743857215
11       [-3.55419883 -1.3121061   6.26693875  4.48480965].       -214.3532082540939
12       [ 2.54749487 -9.29639909  6.99658479 -6.71299771].       -506.4497511543947
13       [ 3.37722984  0.46032172 -5.15280505  5.104616  ].       -195.7120911255773
14       [-9.67252227 -8.84826579  4.65918096  0.87645251].       -318.3378799110803
15       [ 4.34389407  6.23064858  6.70660755 -5.49281777].       -352.1313218446256
16       [-3.31878925 -7.0607148   8.23210147  8.97015571].       -635.8790069647349
17       [-4.02064857 -8.99836262  1.60115438  8.52912304].       -476.781519823831
18       [ 8.85510862  4.75071066  1.0351052  -1.66500369].       -137.8547296827695
19       [6.20008629 7.73399648 6.15686017 0.04789374].          -271.8004300661943
20       [-4.9003479  -9.03322285  6.16587778  1.19907682].       -307.016926806265
```

```
21      [-3.9804138   5.24617004  2.12715187  4.24226727].      -156.4499457908753
22      [-6.17626749 -8.46700681  8.99586277 -2.9797324 ].      -459.8185502973863
23      [ 2.2532422  -6.07277145 -1.50076647 -5.72195561].      -216.5542107800661
24      [ 4.47269207 -2.43605748  1.21501839 -0.91049035].      -39.61850629978485
25      [ 1.47089665 -7.51420962 -8.49603791  4.79799444].      -423.7212128871204
26      [ 4.77645923 -1.42601501  8.98250226 -8.3499408 ].      -547.8236863033205
27      [-8.16220936  6.4390444   7.89501557 -2.69417632].      -365.5724041785769
28      [ 7.40441237 -7.47492651  8.19629674  0.80056796].      -370.6758518470961
29      [-7.6835079   2.55147681  0.33005867  6.73855288].      -254.0155572030121
30      [ 8.88462278 -1.8863526   4.48954681 -8.39811614].      -428.6346849269804
31      [ 6.72932012  7.92538302 -2.29796252  9.53403681].      -550.3404681864292
32      [-0.87457974  1.00084996  1.21196131 -0.29406502].      -7.520738610594414
33      [ 9.58190765  0.91295312 -9.02272206 -2.85332303].      -370.2742701694138
34      [5.61298991 7.24122791 8.73210687 9.06939404].          -694.1411233742432
35      [ 9.89717695 -3.96793954 -0.65229232  8.43551932].      -415.3516007716323
36      [ 2.56432668  6.10803492 -0.24395397  0.04946162].      -81.38027887864618
37      [ 9.99737949  9.43741522 -5.87023887 -9.40704913].      -735.42661571412
38      [ 1.36579201  9.85146971 -9.27097417 -9.34370222].      -803.0402697147695
39      [ 6.56063674 -4.51038055 -9.79828685 -9.44980647].      -728.9436650697719
40      [-0.82153825  7.81312618 -7.02770348  9.65791157].      -644.0316793338778
```

```
1 ### EXACT GP EI GRADIENTS
2
3 np.random.seed(run_num_18)
4 surrogate_exact_18 = dGaussianProcess(cov_func, optimize=opt)
5
6 exact_18 = dGPGO(surrogate_exact_18, Acquisition_new(util_grad_exact), objfunc, param)
7 exact_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].      -343.0366725069492
init     [7.04466137 5.00272572 3.32203335 9.75790897].         -513.6566494158249
init     [-4.86063155 -9.43388149  2.71438231  6.94624775].     -416.7270243155722
init     [ 4.7234925  -9.58385776 -7.76793739 -4.04552516].     -452.4996897110763
init     [ 3.73940383  7.23252112 -6.02731282  3.14378061].     -267.1207899915831
1        [-6.1664519   4.28513481  2.79740714  0.66407663].     -99.9903410594818
2        [ 5.61099687  3.24280582 -9.3946128  -2.53775904].     -343.0519975384772
3        [ 3.30844576  7.22382341 -9.24562554  8.22016932].     -642.0425720582784
4        [ 8.03656335 -7.89614249  7.70734605  1.23688198].     -373.6135402668539
5        [ 9.562822   -7.89156436  1.83282657  5.62935728].     -352.8375538330246
6        [-5.31480829 -4.60774758 -2.97712464 -7.93522102].     -349.1706066518353
7        [ 4.54666061  8.69770438  0.01712558 -6.54510361].     -343.3266506149231
8        [ 2.31421647 -2.51006802  8.67058127  9.73721889].     -622.7471464704078
9        [ 8.39884901 -4.60470488 -7.7957923   9.97387584].     -693.1832086099516
10       [ 8.47467859 -1.0716028   7.04016464 -4.60023784].     -307.4573495762565
11       [-4.41945067  2.73837136  0.20384538  5.36041706].     -149.5898426538008
12       [-0.50618251  2.84808342  2.56143498 -7.45357454].     -258.3853201868007
13       [-8.79601456 -6.39611748  6.71849908 -7.75307177].     -535.0456864449842
14       [-5.37332728  7.94915331  0.37919817 -7.51567844].     -381.6237858896168
15       [-6.68510816 -6.15755534 -5.39539851 -2.06645774].     -224.9336123979594
16       [ 6.58567988 -3.25808927 -0.548504   -0.50180546].     -66.51127573185336
17       [-9.953162   -4.99542568 -2.13845079 -6.90241283].     -353.2661160857886
18       [-5.57956085 -7.22330468 -4.69720516  9.17151356].     -538.1416129636684
19       [ 2.00360707 -5.45715288 -1.03909538  9.47938148].     -426.2493268346466
20       [7.6035649  0.92120319 0.39864644 3.53645041].         -110.014112904619
21       [ 8.3677107   3.02920206 -3.05384275 -9.72173724].     -494.3972790505191
22       [-8.89719428  5.70166694  7.53764301 -8.80536608].     -624.7641513042951
```

```
23      [ 9.72445796 -4.87535282 -9.99546196 -8.2498416 ].      -714.0705377371825
24      [-6.69867140e+00 -9.34293864e+00  9.22919429e+00 -1.76365303e-04].      -47
25      [-3.02570566  7.5626429  -9.84103965  8.64216327].      -712.8281585142872
26      [ 0.603239    8.56375124 -6.99659236 -6.12136988].      -443.7811588473856
27      [3.47426713 5.81482046 6.59216091 1.73752403].      -222.14052153653665
28      [-2.81688776 -6.53120589  3.82918318  7.02029228].      -334.3741037451189
29      [-5.07201697  1.66464445 -7.40256954  0.05193562].      -195.67233503867888
30      [-8.62520888  8.53278751 -7.55767807  2.96646422].      -426.56628694381965
31      [1.29450659 7.41517571 4.45168654 4.77050416].      -262.12878756930735
32      [-0.43584091 -6.50223005  2.28261046 -1.12190431].      -105.41355727602479
33      [-6.17713725  9.54397218  0.80398352  9.11279071].      -554.4428213116671
34      [ 3.84859483  8.91190915  8.54433635 -6.0913698 ].      -541.0921267026
35      [ 1.01610546 -7.01798484 -7.88546948  3.26559047].      -328.73490426184594
36      [ 0.0450659  -0.50352339 -6.72446429  7.54026565].      -363.586786820952
37      [-2.39473123  1.3953035  -8.15416796 -8.57677166].      -503.34389514037906
38      [-5.56893334  1.71990049  8.85958323  6.2163384 ].      -426.9772312874401
39      [-9.35010075  5.13448104 -6.17933967 -8.50123274].      -543.786723790176
40      [ 7.32528391 -9.70927954  0.45190876 -5.71232345].      -373.33522401606376
```

```
1 ### EXACT GP EI GRADIENTS
2
3 np.random.seed(run_num_19)
4 surrogate_exact_19 = dGaussianProcess(cov_func, optimize=opt)
5
6 exact_19 = dGPGO(surrogate_exact_19, Acquisition_new(util_grad_exact), objfunc, param)
7 exact_19.run(init_evals=n_init, max_iter=iters)
8
```

```
Evaluation      Proposed point          Current eval.          Best eval.
init    [-8.04932797  5.22499433 -5.06124054 -7.23736625].      -405.7591605426394
init    [-3.37106873 -8.3400087   3.43954163  6.13187596].      -336.36654573762826
init    [ 9.65483829  2.7132147  -5.68153488  0.98054864].      -208.6243886962717
init    [ 9.11199160e-01 -5.31847854e+00 -7.72548314e+00 -6.81465902e-03].      -23
init    [-6.95783155  0.65372161 -2.25986463  3.76654767].      -121.3346135965923
1       [-6.5477408   0.83865275 -8.96461914  6.51115227].      -454.95319082784266
2       [-2.30389925 -5.27595204 -9.70176571 -1.31398656].      -350.2583079766517
3       [ 4.66969615 -0.45575046 -6.64393505 -5.40706888].      -271.59267326043584
4       [-9.13977787 -5.66464753  4.94279629  8.45310839].      -506.8258744362434
5       [-4.03600104  9.52409788  4.77638865 -9.1959919 ].      -604.4129191150145
6       [-4.83369566 -2.4386249   4.54159233  6.11885823].      -246.8982833108634
7       [-1.74139105 -2.25943387 -4.66754333 -7.46934292].      -301.764742107778
8       [ 3.38990203 -3.49236324 -8.09259704  9.98332142].      -631.0218446479364
9       [ 0.48549986 -6.62909349  6.4319956  -6.53556085].      -383.0913959392533
10      [-0.99362165  7.76510241 -3.3125504  -8.02417682].      -412.0495401571854
11      [ 1.66645895 -8.36624855 -7.84433287  7.99584163].      -583.0999231748925
12      [-5.10416772  3.72102212 -2.58654699  5.58265898].      -198.479540412442
13      [ 2.86565519 -0.33761145  0.96097259 -4.59742072].      -95.75545659214191
14      [-9.56069024  1.08179853 -2.00683023  4.38882877].      -182.87674863965128
15      [ 2.34954077 -3.13748972  3.15811446 -8.40909356].      -337.9805041983478
16      [-0.58070913  1.68475454  1.86922719  4.30763163].      -90.71881066985442
17      [2.41764449 1.87228964 5.94389184 3.41628568].      -165.5295239842262
18      [ 9.11708637 -4.55121628  1.89198796  7.70152498].      -372.54120637843994
19      [-2.30605317 -5.20595438 -0.16009014  4.3386911 ].      -134.89565147844593
20      [ 9.51568311 -8.82578504 -2.38869497 -2.78357922].      -294.4480324191947
21      [ 1.40856514 -0.68378535 -2.05805581  4.56833676].      -99.10476462698388
22      [ 9.47335062 -2.82596526 -9.98254377 -9.14656194].      -739.3084528981631
23      [ 2.51530768 -4.62186548  9.97645257  4.84711249].      -441.6168693747741
24      [-6.97129621  0.24238742  5.42427763  6.34874102].      -298.21088787177143
```

```
25      [-4.0793351   1.80214266 -3.34734211 -6.24787259].      -212.8941565641893
26      [-9.42856911  8.98664152  3.80558549 -5.85683224].      -431.0747456640331
27      [ 4.54762806 -0.20468487 -1.40689689 -3.80297829].      -84.55336473877665
28      [ 8.01410845  2.35469457 -6.26141828  6.51240303].      -362.5767565111792
29      [-1.35226567 -4.29413716  6.11966642 -7.64814281].      -385.0351552851985
30      [-3.1156522   0.15294593 -2.91082318 -3.69003264].      -89.63811185417782
31      [-0.25259458  3.44934982  5.3577388  -1.83043753].      -123.37793380084048
32      [ 3.48506606  9.86842239  6.83677971 -2.25957315].      -367.5645601164437
33      [5.71950689 3.41725053 1.95649279 2.58774267].          -94.33720214270983
34      [ 7.2052829   0.12871334 -1.41519373 -3.82625005].      -116.51831371205623
35      [-9.87217663 -3.42357815  6.86830815 -0.41111743].      -263.0986868654941
36      [ 7.7670299   5.6968523  -1.97128227 -9.1879163 ].      -474.56409105866277
37      [ 1.74657984 -8.41100693  0.75411934 -2.6998782 ].      -175.40407355515558
38      [ 1.81559099 -7.1068488   3.58237457 -7.22161856].      -351.41829127830357
39      [-6.0350497  -0.46255327  3.66257593  1.55122853].      -86.71836318038275
40      [ 5.01697203 -9.11312334 -3.14545699 -9.55509343].      -586.1489836426866
```

```
### 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].          -473.20865659642135
init     [-9.28220829  3.83515164 -2.42638116  0.37021891].      -133.78639158661557
init     [ 3.15902931 -6.12299564 -4.55367196  4.37211867].      -223.63108911163158
init     [ 5.66007219  7.0065528   5.50489788 -9.26671387].      -564.6196272537535
init     [-7.6661253   5.02561399 -5.21563568 -4.90387972].      -287.0837808154094
1        [-0.38031735 -3.41587185  0.21282112 -4.72742343].      -113.01100973396937
2        [ 2.2515589  -8.62950692  7.2874785   9.15451767].      -648.549100750603
3        [-2.00279263 -5.09620423  0.51102349  6.68881567].      -235.6982290011646
4        [-3.33679846  5.09475498 -0.14129379 -8.38664837].      -344.4506560372816
5        [-3.75442852  7.02008144  4.58355261 -6.10855163].      -324.943296034215
6        [ 4.26761042  1.24919415 -4.83066711 -6.15878884].      -243.06222494906058
7        [ 2.6185745   0.28739637 -8.78495862 -6.04256041].      -384.5987651129375
8        [ 7.25959653  3.0895583  -0.80389347  9.92972863].      -468.1292594707154
9        [-0.16988855 -8.40538175  0.41458404 -7.27792385].      -353.71808906261083
10       [-3.49997926  0.57776952  2.16068958 -2.13900545].      -45.22460573983692
11       [-9.19853355 -3.76188052  3.90442908  6.53321015].      -329.38154832642016
12       [-2.50232563 -8.25576548 -9.16975052 -0.52445896].      -395.93016354238904
13       [6.63294021 8.78254327 1.17529344 3.77776296].          -259.49194462274477
14       [ 0.69964944  8.14493356 -7.61528925 -3.43144779].      -354.2466217033996
15       [-8.02835277  3.38776435  9.30420836  4.26835017].      -419.98847492600896
16       [-4.5446278  -7.19815277  8.80947259  2.66802409].      -385.5742803771812
17       [-4.38378523 -2.62352062 -8.38409406 -9.96564204].      -641.1184785965473
18       [-2.25218145  6.59461508 -1.33537048  0.21013843].      -97.57649309920389
19       [ 9.82323278 -7.44187573  5.4849252   4.16735614].      -366.979573192001
20       [ 8.69610211 -4.83147655  2.30105213 -8.06879304].      -398.61473023237465
21       [ 9.94326022  5.89059493 -7.07853366  1.04477436].      -322.94977116904477
22       [ 5.13550599  2.02586059  5.52714892 -2.28980739].      -147.20264120882348
23       [-4.61072293  3.89013286 -7.89070456  5.84820065].      -375.12049235870035
24       [-8.43615413 -5.1255552   4.13795332 -9.8570129 ].      -563.7221146654842
25       [ 5.71811787 -8.64430875 -8.49295218 -8.02648515].      -656.2335852094399
26       [-0.81838675  8.7971258  -3.60485225  9.86807956].      -583.9494577012838
```

27	[-2.99132783 -1.49146415 -9.70724606 4.49308209].	-376.8399975243280
28	[-7.25701887 -8.36095808 -7.85151116 8.5115448].	-667.1998246489352
29	[9.80888347 -8.95843028 -9.8862586 3.36759389].	-595.2982227356134
30	[4.91253395 -9.54707138 6.5280506 -2.84486272].	-366.6454431981636
31	[-0.23811104 1.99927023 6.38129647 0.93804175].	-133.7333830579090
32	[-9.75809951 -7.08201376 -0.06854532 -0.55203876].	-196.7634263553898
33	[9.03920666 -1.08353422 -1.33043237 0.0695733].	-89.3848625426502
34	[9.98702901 9.15213347 -3.82445865 -6.66959404].	-489.0772331638205
35	[2.20862081 1.02179984 9.94977153 -9.58444806].	-671.4065945027759
36	[4.28456461 1.90692336 -6.99356526 4.40461416].	-249.9625760411938
37	[8.36049103 5.50223377 9.831103 -5.17953668].	-527.7091228774024
38	[6.16863697 -1.07459565 8.13307736 4.21486655].	-309.8628356370176
39	[-8.53707476 -9.69123096 8.4335272 9.72960446].	-852.7555157037436
40	[-5.9671341 9.71806023 3.37840405 8.48186325].	-546.4969373654435

```
end_exact = time.time()
end_exact

time_exact = end_exact - start_exact
time_exact
```

```
231.51595520973206
```

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

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

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

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

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

min_simple_regret_approx_1, min_simple_regret_exact_1
```

```
(4.5852133435359415, 4.238895465474788)
```

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

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

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

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

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

```
14
15 min_simple_regret_approx_2, min_simple_regret_exact_2
```

```
(4.129668896849149, 4.129668896849149)
```

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

```
(3.812215334872664, 3.812215334872664)
```

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

```
(3.67738086164022, 3.67738086164022)
```

```
 1 ### Simple regret minimization: run number = 5
 2
 3 approx_output_5 = np.append(np.min(approx_5.GP.y[0:n_init]),approx_5.GP.y[n_init:(n_ini
 4 exact_output_5 = np.append(np.min(exact_5.GP.y[0:n_init]),exact_5.GP.y[n_init:(n_init+i
 5
 6 regret_approx_5 = np.log(-approx_output_5 + y_global_orig)
 7 regret_exact_5 = np.log(-exact_output_5 + y_global_orig)
 8
 9 simple_regret_approx_5 = min_max_array(regret_approx_5)
10 simple_regret_exact_5 = min_max_array(regret_exact_5)
11
12 min_simple_regret_approx_5 = min(simple_regret_approx_5)
```

```
min_simple_regret_exact_5 = min(simple_regret_exact_5)

min_simple_regret_approx_5, min_simple_regret_exact_5
```

```
(4.50051346775536, 4.486902976200048)
```

```
### 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
```

```
(4.691322064785611, 3.5829521711756844)
```

```
### 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
```

```
(4.764413764658034, 4.511003020737738)
```

```
### 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
```

```
(4.629238403819344, 4.3840723898213705)
```

```
### 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
```

```
(4.519513422497501, 4.827751963190094)
```

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

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

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

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

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

min_simple_regret_approx_10, min_simple_regret_exact_10
```

```
(3.707314539376269, 3.6393444267981203)
```

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

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

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

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

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

min_simple_regret_approx_11, min_simple_regret_exact_11
```

```
(2.993160936600768, 3.8729060531415076)
```

```
### 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
```

```
(3.5110344050592053, 3.5110344050592053)
```

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

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

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

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

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

min_simple_regret_approx_13, min_simple_regret_exact_13
```

```
(4.3791614962060335, 3.604208177076985)
```

```
### 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
```

```
(4.871940135457522, 4.707833804309497)
```

```
### 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
```

```
(4.330687503198666, 3.7777426289164455)
```

```
### 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
```

```
(3.425867987599218, 3.425867987599218)
```

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

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

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

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

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

```
(4.399132969437249, 2.0176643526324676)
```

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

```
(4.197371493165651, 4.197371493165651)
```

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

```
(4.59617751916085, 4.437382870398724)
```

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

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

```
(3.8116413134951426, 3.8116413134951426)
```

```
# 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_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 :
```

```
 2
 3 slice33 = 32
 4
 5 approx33 = [simple_regret_approx_1[slice33],
 6        simple_regret_approx_2[slice33],
 7        simple_regret_approx_3[slice33],
 8        simple_regret_approx_4[slice33],
 9        simple_regret_approx_5[slice33],
10        simple_regret_approx_6[slice33],
11        simple_regret_approx_7[slice33],
12        simple_regret_approx_8[slice33],
13        simple_regret_approx_9[slice33],
14        simple_regret_approx_10[slice33],
15        simple_regret_approx_11[slice33],
16        simple_regret_approx_12[slice33],
17        simple_regret_approx_13[slice33],
18        simple_regret_approx_14[slice33],
19        simple_regret_approx_15[slice33],
20        simple_regret_approx_16[slice33],
21        simple_regret_approx_17[slice33],
22        simple_regret_approx_18[slice33],
23        simple_regret_approx_19[slice33],
24        simple_regret_approx_20[slice33]]
25
26 exact33 = [simple_regret_exact_1[slice33],
27        simple_regret_exact_2[slice33],
28        simple_regret_exact_3[slice33],
29        simple_regret_exact_4[slice33],
30        simple_regret_exact_5[slice33],
31        simple_regret_exact_6[slice33],
32        simple_regret_exact_7[slice33],
33        simple_regret_exact_8[slice33],
34        simple_regret_exact_9[slice33],
35        simple_regret_exact_10[slice33],
36        simple_regret_exact_11[slice33],
37        simple_regret_exact_12[slice33],
38        simple_regret_exact_13[slice33],
39        simple_regret_exact_14[slice33],
40        simple_regret_exact_15[slice33],
41        simple_regret_exact_16[slice33],
42        simple_regret_exact_17[slice33],
43        simple_regret_exact_18[slice33],
44        simple_regret_exact_19[slice33],
45        simple_regret_exact_20[slice33]]
46
47 approx33_results = pd.DataFrame(approx33).sort_values(by=[0], ascending=False)
48 exact33_results = pd.DataFrame(exact33).sort_values(by=[0], ascending=False)
49
50 ### Best simple regret minimization IQR - approx:
51 lower_approx33 = np.asarray(approx33_results[4:5][0])[0]
52 median_approx33 = np.asarray(approx33_results[9:10][0])[0]
53 upper_approx33 = np.asarray(approx33_results[14:15][0])[0]
54
55 lower_exact33 = np.asarray(exact33_results[4:5][0])[0]
56 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)
```

```
49
50 ### Best simple regret minimization IQR - approx:
51 lower_approx14 = np.asarray(approx14_results[4:5][0])[0]
52 median_approx14 = np.asarray(approx14_results[9:10][0])[0]
53 upper_approx14 = np.asarray(approx14_results[14:15][0])[0]
54
55 lower_exact14 = np.asarray(exact14_results[4:5][0])[0]
56 median_exact14 = np.asarray(exact14_results[9:10][0])[0]
57 upper_exact14 = np.asarray(exact14_results[14:15][0])[0]
```

```
 1 # Iteration24 :
 2
 3 slice24 = 23
 4
 5 approx24 = [simple_regret_approx_1[slice24],
 6        simple_regret_approx_2[slice24],
 7        simple_regret_approx_3[slice24],
 8        simple_regret_approx_4[slice24],
 9        simple_regret_approx_5[slice24],
10        simple_regret_approx_6[slice24],
11        simple_regret_approx_7[slice24],
12        simple_regret_approx_8[slice24],
13        simple_regret_approx_9[slice24],
14        simple_regret_approx_10[slice24],
15        simple_regret_approx_11[slice24],
16        simple_regret_approx_12[slice24],
17        simple_regret_approx_13[slice24],
18        simple_regret_approx_14[slice24],
19        simple_regret_approx_15[slice24],
20        simple_regret_approx_16[slice24],
21        simple_regret_approx_17[slice24],
22        simple_regret_approx_18[slice24],
23        simple_regret_approx_19[slice24],
24        simple_regret_approx_20[slice24]]
25
26 exact24 = [simple_regret_exact_1[slice24],
27        simple_regret_exact_2[slice24],
28        simple_regret_exact_3[slice24],
29        simple_regret_exact_4[slice24],
30        simple_regret_exact_5[slice24],
31        simple_regret_exact_6[slice24],
32        simple_regret_exact_7[slice24],
33        simple_regret_exact_8[slice24],
34        simple_regret_exact_9[slice24],
35        simple_regret_exact_10[slice24],
36        simple_regret_exact_11[slice24],
37        simple_regret_exact_12[slice24],
38        simple_regret_exact_13[slice24],
39        simple_regret_exact_14[slice24],
40        simple_regret_exact_15[slice24],
41        simple_regret_exact_16[slice24],
42        simple_regret_exact_17[slice24],
43        simple_regret_exact_18[slice24],
44        simple_regret_exact_19[slice24],
45        simple_regret_exact_20[slice24]]
```

```

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

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

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

```
# Iteration34 :

slice34 = 33

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

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

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

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

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

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

```
# Iteration5 :

slice5 = 4

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

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

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

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

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

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

```
# Iteration15 :

slice15 = 14

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

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

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

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

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

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

```
# Iteration25 :

slice25 = 24

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

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

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

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

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

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

```
# Iteration35 :

slice35 = 34

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

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

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

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

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

lower_exact35 = np.asarray(exact35_results[4:5][0])[0]
median_exact35 = np.asarray(exact35_results[9:10][0])[0]
upper_exact35 = np.asarray(exact35_results[14:15][0])[0]
```

```
# Iteration6 :

slice6 = 5

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

```
        simple_regret_approx_20[slice6]]

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

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

### Best simple regret minimization IQR - approx:
lower_approx6 = np.asarray(approx6_results[4:5][0])[0]
median_approx6 = np.asarray(approx6_results[9:10][0])[0]
upper_approx6 = np.asarray(approx6_results[14:15][0])[0]

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

```
# Iteration16 :

slice16 = 15

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

```
        simple_regret_approx_16[slice16],
        simple_regret_approx_17[slice16],
        simple_regret_approx_18[slice16],
        simple_regret_approx_19[slice16],
        simple_regret_approx_20[slice16]]

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

approx16_results = pd.DataFrame(approx16).sort_values(by=[0], ascending=False)
exact16_results = pd.DataFrame(exact16).sort_values(by=[0], ascending=False)

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

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

```
# Iteration26 :

slice26 = 25

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

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

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

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

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

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

```
# Iteration36 :

slice36 = 35

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

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

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

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

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

lower_exact36 = np.asarray(exact36_results[4:5][0])[0]
median_exact36 = np.asarray(exact36_results[9:10][0])[0]
upper_exact36 = np.asarray(exact36_results[14:15][0])[0]
```

```
# Iteration7 :

slice7 = 6

approx7 = [simple_regret_approx_1[slice7],
        simple_regret_approx_2[slice7],
        simple_regret_approx_3[slice7],
        simple_regret_approx_4[slice7],
        simple_regret_approx_5[slice7]
```

```
        simple_regret_approx_5[slice7],
        simple_regret_approx_6[slice7],
        simple_regret_approx_7[slice7],
        simple_regret_approx_8[slice7],
        simple_regret_approx_9[slice7],
        simple_regret_approx_10[slice7],
        simple_regret_approx_11[slice7],
        simple_regret_approx_12[slice7],
        simple_regret_approx_13[slice7],
        simple_regret_approx_14[slice7],
        simple_regret_approx_15[slice7],
        simple_regret_approx_16[slice7],
        simple_regret_approx_17[slice7],
        simple_regret_approx_18[slice7],
        simple_regret_approx_19[slice7],
        simple_regret_approx_20[slice7]]

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

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

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

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

```
# Iteration17 :

slice17 = 16

approx17 = [simple_regret_approx_1[slice17],
```

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

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

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

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

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

```
# Iteration27 :
```

```
slice27 = 26

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

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

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

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

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

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

```
# Iteration37 :

slice37 = 36

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

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

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

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

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

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

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

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

```
# Iteration18 :

slice18 = 17

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

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

```

```

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]
```

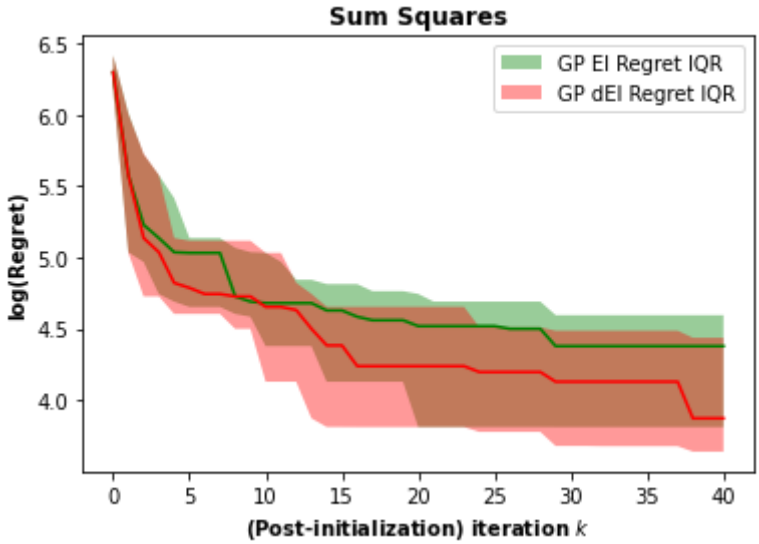
```
### Summarize arrays: 'exact'

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 = 'Sum Squares'

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
```

```
(1217.5402565002441, 231.51595520973206)
```

```
1
```