**RastriginDigitalTwin.py**

main:

kwargs = {‘niter’ : 2}

**cfg.**main(RastriginDigitalTwin(), args, \*\*kwargs)

class RastriginDigitalTwin(AlgoDigitalTwin):

\_\_init\_\_ (self):

dim = 20

bnds = np.full((dim,2), (-5.12, 5.12))

# a BASIC question I have is why do we want these class variables,

# as opposed to just making them part of the params\_dict ??

# and if we DO make them class variables (and we’re passing “self” into

# the function calls – why do we need to put them into the parameter list?

*# IF we define bounds, probably should pass them to initialization*

*# shouldn’t we also pass dimensions? probably maxiters?*

super().\_\_init\_\_(rastrigin, x0 = utils.get\_random\_x0(10,-5.12, 5.12))

class AlgoDigitalTwin():

# Change this to accept x0, bounds, dimensions, and maxiters ?

# we’re passing bounds – but no where do we assign it.

def \_\_init\_\_(self, of, flip=False, x0=None, bounds=None):

self.optimizer = Optimus()

self.algo\_objective\_func = of

self.flip = flip

self.x0 = x0

self.bounds = bounds

**configuration.py**

**main:**

res = algo\_wrapper.optimize(args, \*\*kwargs)

**RastriginDigitalTwin.py**

class RastriginDigitalTwin(AlgoDigitalTwin)

optimize(self, args, \*\*kwargs):

return super().optimize(args, \*\*kwargs)

**\_algo\_digital\_twin.py**

def optimize(self, args, \*\*kwargs):

self.optimizer.set\_objective\_function(self.algo\_objective\_func, self.flip)

*# Do we need to test to see if these(x0 and bounds) are even defined?*

*# probably OK…they’ll just be set to NONE ?*

self.optimizer.set\_starting\_point(self.x0)

self.optimizer.set\_bounds(self.bounds)

*# Do we need another function for “set\_dimensions” ?*

self.optimizer.set\_solver(args.solver)

self.optimizer.update\_solver\_params(args.solver, \*\*kwargs)

return self.optimizer.**solve()**

**\_optimus.py**

solve(self):

# Q: Why are we passing in the objective\_function and x0 ???

# these are defined within “self” ? shouldn’t we just pass in “self” ???

res = self.solver\_dict[self.solver\_name]**.solve**(self.objective\_function, x0=self.x0, \*\*self.solver\_params\_dict[self.solver\_name])

**\_pso\_solver.py**

# solve(below) is called from Optimus.solve (above)

# this call has \*args and \*\*kwargs where:

# args = self.objective\_function, x0=self.x0

# and kwargs = self.solver\_params\_dict[self.solver\_name]

solve(self, \*args, \*\*kwargs):

return **self.pso\_global\_optimize**(args[0], \*\*kwargs)

# Similarly – shouldn’t “maxiter” be set in the dictionary – not sent via the parameter list ?

**pso\_global\_optimize**(self, fun, dimension = None, x0=None, bounds=None, maxiter=1000, n\_particles=10, options={'c1':0.2,'c2': 0.6, 'w' : 0.95}, pso\_kwargs={}, fun\_kwargs={}):

*# dimension, x0, and bounds are all defined within the Optimus class – why are we defaulting them to None in the parameter list here?*

*optimizer = ps.single.GlobalBestPSO(n\_particles, dimensions, options, \*\*pso\_kwargs)*