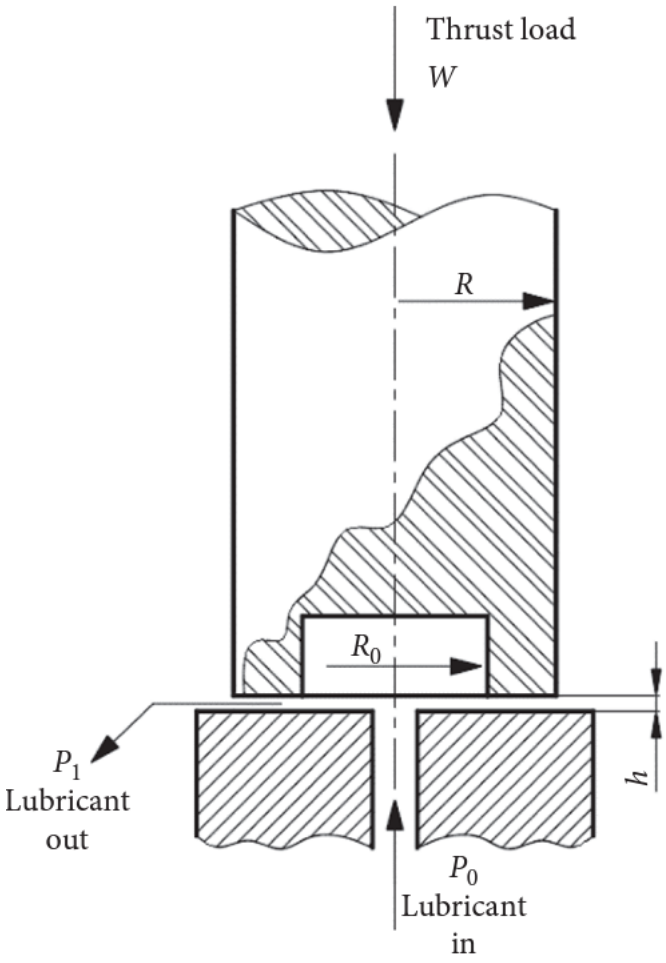

Figure 1: A sketch of a hydrostatic bearing (copied from [2]).

Problem description: Hydrostatic Thrust Bearing Design for Minimum Power Loss

In the design of hydrostatic thrust bearings [3], the primary performance indicator is the operational power loss. For this reason, the main objective is typically minimising the operational power loss when the thrust bearing is exposed to axial load in Figure 1.

The power loss depends on the decision vector $\mathbf{x} = (x_1, x_2, x_3, x_4)^\top \in \mathcal{X} \subset \mathbb{R}^4$, where, $x_1 = Q$ is the flow rate, $x_2 = R_0$ is the recess radius, $x_3 = R$ is the bearing step radius, and finally $x_4 = \mu$ is the viscosity of the fluid.

The optimisation problem is expressed as:

$$\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) = \frac{P_0(\mathbf{x})x_1}{0.7} + E_f(\mathbf{x}). \tag{1}$$

The feasible space is defined by the following constraints (as per Appendix C4 in [4]).

$$g_1(\mathbf{x}) = 101000 - W(\mathbf{x}) \leq 0, \tag{2}$$
$$g_2(\mathbf{x}) = P_0(\mathbf{x}) - 1000 \leq 0, \tag{3}$$
$$g_3(\mathbf{x}) = \Delta T(\mathbf{x}) - 50 \leq 0, \tag{4}$$
$$g_4(\mathbf{x}) = 0.001 - h(\mathbf{x}) \leq 0, \tag{5}$$
$$g_5(\mathbf{x}) = x_2 - x_3 \leq 0, \tag{6}$$
$$g_6(\mathbf{x}) = \frac{0.0307 x_1}{772.8 \pi P_0(\mathbf{x}) h(\mathbf{x}) x_3} - 0.001 \leq 0, \tag{7}$$
$$g_7(\mathbf{x}) = \frac{W(\mathbf{x})}{\pi(x_3^2 - x_2^2)} - 5000 \leq 0, \tag{8}$$
$$x_1 \in [1, 16], \tag{9}$$
$$x_2 \in [1, 16], \tag{10}$$
$$x_3 \in [1, 16], \tag{11}$$
$$x_4 \in [1, 16], \tag{12}$$

where $g_1(\mathbf{x})$ is for weight capacity, which must be greater than weight of generator, $g_2(\mathbf{x})$ is for inlet oil pressure, $g_3(\mathbf{x})$ is for oil temperature rise, $g_4(\mathbf{x})$ is for oil film thickness, $g_5(\mathbf{x})$ is for step radius and must be greater than recess radius, $g_6(\mathbf{x})$ is for limits on significance off exit loss and must be less than 0.001, and $g_7(\mathbf{x})$ is the limit for contact pressure and must be less than 5000.

Other parameters in this problem are defined through functions of $\mathbf{x}$:

$$W(\mathbf{x}) = \frac{\pi P_0(\mathbf{x})}{2} \left[\frac{(x_3^2 - x_2^2)}{\ln \frac{x_3}{x_2}} \right], \tag{13}$$
$$P_0(\mathbf{x}) = \frac{6 \times 10^{-6} x_4 x_1}{\pi h(\mathbf{x})^3} ln \frac{x_3}{x_2}, \tag{14}$$
$$E_f(\mathbf{x}) = 143.308 \Delta T(\mathbf{x}) x_1, \tag{15}$$
$$\Delta T(\mathbf{x}) = 2(10^{P(\mathbf{x})} - 560), \tag{16}$$
$$P(\mathbf{x}) = \frac{\log_{10}(\log_{10}(8.122 x_4 + 0.8)) - 10.04}{-3.55}, \tag{17}$$
$$h(\mathbf{x}) = \left(\frac{1500\pi}{60} \right)^2 \frac{2 \times 10^{-6} \pi x_4}{E_f(\mathbf{x})} \left(\frac{x_3^4}{4} - \frac{x_2^4}{4} \right), \tag{18}$$

where, $W(\mathbf{x})$ is the load carrying capacity, $P_0(\mathbf{x})$ is the inlet pressure, $E_f(\mathbf{x})$ is the friction loss, $\Delta T(\mathbf{x})$ is the temperature, $P(\mathbf{x})$ is the pressure, and $h(\mathbf{x})$ is the oil thickness.

Now, you have the following tasks.

1. Implement all the functions $f(\mathbf{x})$ and $g_i(\mathbf{x}); \forall i \in [1, 7]$, independently, where each function takes at least a Numpy array $\mathbf{x}$. Each function should have an independent counter that represents how many times a respective function has been called (or, in other words, evaluated).

2. Implement the Random Search (RS) method discussed in the lectures that can use the functions defined above and return an approximation of the optimum.

3. Implement the simulated annealing (SA) method that can use the functions defined above and return an approximation of the optimum solution $\mathbf{x}^*$.

4. For 21 repetitions of each of the algorithms implemented in 2 and 3, compare and comment on the performances of these optimisers. The number of evaluations for each individual function $f(\mathbf{x})$ or $g_i(\mathbf{x})$ that you are allowed at each instance of an optimisation run is 10000 at most.

In this assignment, you must use Python 3.x to develop your code in a Jupyter notebook. Please note that you are allowed to use basic and advanced Python modules, such as Numpy, Scipy, Matplotlib, etc. However, the core of the algorithmic implementations must be your own: for instance, you cannot just use a module that already implements the algorithms here. If you are in doubt, please feel free to contact the module coordinator for clarification.

Submission

You are required to submit the following on Canvas.

Code A single Jupyter notebook file (with the format `XXXXXXX.ipynb`[1]). You must ensure that all the results are pre-generated and repeatable[2], otherwise you may lose marks.

For validating your function implementations, you must use the following code in your notebook and ensure that your code is producing the same outputs (accurate up to 3 decimal places).

```
# validation code
x = np.array([4.19, 11.57,  6.69, 10.65])
print("Objective function output, f(x) = ", f(x))
print("Constraint function output, g1(x) = ", g1(x))
print("Constraint function output, g2(x) = ", g2(x))
print("Constraint function output, g3(x) = ", g3(x))
print("Constraint function output, g4(x) = ", g4(x))
print("Constraint function output, g5(x) = ", g5(x))
print("Constraint function output, g6(x) = ", g6(x))
print("Constraint function output, g7(x) = ", g7(x))
```

✓ 0.0s

```
Objective function output, f(x) =  -1784.0773507231925
Constraint function output, g1(x) =  101000.01528759542
Constraint function output, g2(x) =  -1000.0000598304209
Constraint function output, g3(x) =  -52.97118117373543
Constraint function output, g4(x) =  -0.9196377239881505
Constraint function output, g5(x) =  4.88
Constraint function output, g6(x) =  -0.14477999681964238
Constraint function output, g7(x) =  -4999.999945390436
```

[1]Please use your student ID to replace XXXXXXX in the name of the document.
[2]You may want to look at setting a seed using `numpy.random.seed` method.