# Reservoir calc

Lets design a python model in anaconda notebook which does the following. Python variable names are in brackets, units are after the # symbol)

## Constant inputs

* Gravity (g) = 9.81 # m/s^2
* Seconds in an hour (secs\_hr) # seconds
* Efficiency turbine mode (eff\_t) = 90%
* Efficiency pump turbine (eff\_p) = 91%
* Initial rated headloss (hl\_rated\_ini\_percent) = 5% # percent of gross head
* Rating point for turbine (rating\_point).
  + This can be either:
    - hg\_avg (default), or
    - hg\_min

## Model parameters

There are several user inputs and parameters. The user inputs some of these, and the algorithm solves the remaining parameters, using optimization routines such as SciPy:

* **Upper reservoir** 
  + full supply level (u\_fsl) # metres above sea level (masl)
  + minimum operating level (u\_mol) # masl
  + total volume at u\_fsl (u\_fsl\_vol) # m^3
  + minimum volume at u\_mol (u\_mol\_vol) # m^3
  + upper live volume (u\_live\_vol) = u\_fsl\_vol – u\_mol\_vol # m^3
* **Lower reservoir**
  + full supply level (l\_fsl) # masl
  + minimum operating level (l\_mol) # masl
  + total volume at l\_fsl (l\_fsl\_vol) # m^3
  + minimum volume at l\_mol (l\_mol\_vol) # m^3
  + upper live volume (l\_live\_vol) = l\_fsl\_vol – l\_mol\_vol # m^3
* **Gross head, head range, head loss and net head**
  + Maximum gross head (hg\_max) = u\_fsl - l\_mol # metres (m)
  + Minimum gross head (hg\_min) = u\_mol - l\_fsl # m
  + Average gross head (hg\_avg) = (hg\_max + hg\_min) / 2 # m
  + Head range - gross (hr\_gross) = hg\_max / hg\_min # -
  + Rated gross head (hl\_rated\_ini) = based on inputs (1 or 2) above # m
  + Rated initial head loss (hl\_rated\_ini) = hg\_rated \* hl\_rated\_ini\_percent # m
  + Rated net head generation mode (hn\_rated\_gen\_ini) = hg\_rated - hl\_rated\_ini # m
* **Power parameters**
  + Cycle time in hours (cycle\_hours\_gen) # hours
  + Live vol for generation (live\_vol\_gen) = minimum(l\_live\_vol, u\_live\_vol) # m^3
  + Rated flow rate in turbine mode (q\_t\_rated) # m^3
  + q\_t\_rated = live\_vol\_gen / (cycle\_hours\_gen \* secs\_hr)
  + power in generation mode (power\_gen\_rated) :
    - power\_gen\_rated = q\_t\_rated \* hn\_rated\_gen\_ini \* g \* eff\_t / 1000 # megawatts (MW)

# Solutions:

The algorithm needs to solve for the following scenarios:

## A. Case 1

The simple case is the user supplying two water levels for each reservoir, which sets the volumes, determines the head, head loss, range, and power etc. In this case the reservoirs do not need to balance the live volume. The live volume for gneration is as above, live\_vol\_gen = minimum(l\_live\_vol, u\_live\_vol).

## B. Case 2

The second case involves the user supplying 2 water levels for a single reservoir (governing reservoir), then solving for the other reservoir (secondary reservoir). In this case the user needs to give one water level for the secondary reservoir, to fix a point so that the other water level can be determined, by either:

1. balancing the live volume (l\_live\_vol = u\_live\_vol)
2. solving for desired power output. This is achieved by finding the live volume to target in the secondary reservoir, which results in the correct power, according to power equation above. Because the gross head and head loss changes as water levels change, this process needs to be iterative (for example use SciPy optimisation).
3. the desired head range. This is achieved by adjusting the water level in secondary reservoir to meet the head range given. As above, this required iteration too.

## C. Case 3

The third case is when the reservoir are not balanced. I.e. one can be larger than the other (referred to as independent). In this case:

1. The user defines the independent reservoir with two water levels.
2. The water levels in the secondary reservoir are given by either:
   1. solving for power
   2. solving for head range