Our SMT model can be described as follows:
Note that, in the explanation we have taken 10 equations, whereas the problem occurs when the number of equations is quite large.
Let us consider three cases:

- **Case 0:** Values of a set of equations are given in correct form, e.g., $v_1, v_2, \ldots, v_{10}$ are given. Then the system of equations can be formed as follows:

$$\text{for i in } \{1, 2, \ldots, 10\} : \quad equation_i == v_i$$

- **Case 1:** Instead of $v_1, v_2, \ldots, v_{10}$ we know $b_1, b_2, \ldots, b_{10}$ where,

$$b_1 = v_1 + 2; \quad b_2 = v_2 - 2; \quad b_3 = v_3 + 1; \quad b_4 = v_4 - 1; \quad b_5 = v_5$$
$$b_6 = v_6 - 2; \quad b_7 = v_7 + 2; \quad b_8 = v_8 - 1; \quad b_9 = v_9 + 1; \quad b_{10} = v_{10}$$

  Here, we know only $b_1, b_2, \ldots, b_{10}$ and the highest error which is ± 2. Since we do not know the positions of error, we can form the system of constraints as follows:

$$\text{for i in } \{1, 2, \ldots, 10\} : \quad b_i - 2 \leq equation_i \leq b_i + 2$$

  Note that the errors here are distributed uniformly, i.e., equal number of +2,-2,+1,-1 and 0.

- **Case 2:** Instead of $v_1, v_2, \ldots, v_{10}$ we know $b_1, b_2, \ldots, b_{10}$ where,

$$b_1 = v_1 + 2; \qquad b_2 = v_2 - 1; \quad b_3 = v_3 + 1$$

$$b_i = v_i \ \text{ for i in } \{4, \ldots, 10\}$$

  Errors are injected at 30% places. Since we do not know the positions of error, we have to model it as follows:

$$\text{for i in } \{1, 2, \ldots, 10\} : \quad b_i - 2 \leq equation_i \leq b_i + 2$$

Problem: Now in our case, the system of equations is large and highly non-linear. We observed that **Case 0 is solvable in 10-12 sec., Case 1 is solvable in 10-20 sec., but for Case 3 we are not getting any solution even after 24 hours even when the number of errors in Case 2 is much lesser than Case 1.** We want to solve the system of constraints for Case 2 with solution time near to the solution time of Case 1. **We observed that as soon as the b_i approaches towards v_i, the solver takes more time to solve,i.e., Normally distributed error has higher solution time as compared to**

uniformly distributed error. In our scenario, we encounter errors with normal distribution.

Note:– While solving the system of constraints in Case 2, we have the information that given sequence b_i is near to v_i for most of the cases, e.g., we know that $b_i = v_i$ for approximately 70% times; $b_i = v_i \pm 1$ for approximately 20% times; and so on.

Toy Example:– Since the actual code is quite large, we illustrated the same scenario in a toy example (script and readme file are available at `https://github.com/Anonychn/SMT_Toy_Example.git`). In the script, we first collected the values from a random array and its updated version. Next, we inject errors to the values as per Case 1 and Case 2 separately. Thereafter, we form the SMT constraints to recover the original array with the help of given erroneous values. Refer to the README file and comments in the scripts for explanation.

Target:– we want our model to work well with errors in normal distribution (Case 2), as in this case, errors are very less.