Assignment 5

Vivek Kumar

(vivekkumar13@iisc.ac.in)

Code Architecture:

- *SEIRV* function takes *start_date*, *end_date* and initial parameters $P = \{\beta, S(0), E(0), I(0), R(0), CIR(0)\}$ as input and returns the numpy array of $\{S(t), E(t), I(t), R(t), \Delta i(t)\}$ for each day between start_date and end_date. This function also take one optional parameter *is_close_loop_control* as input to enable whether β is to be updated based on closed loop conditions.

- *get_new_beta* function is called from *SEIRV* function when *is_close_loop_control* is enabled to return the appropriate β based on average number of cases in previous week.

- *deltaV* function takes date as input and returns the number of vaccinations done on that date. If the date is greater than 27 April 2021 then it returns 2 Lakhs vaccinations.

- *compute_loss* function takes output of *SEIRV* function for a given set of parameters as input and returns the loss between daily cases from the dataset and daily cased produced by our *SEIRV* function. It first computes seven-day average for actual daily cases and estimated daily cases and computes the loss using the formula described in the assignment.

- *gradient_descent* function takes initial parameters P as inputs and returns parameters P^* as output such that $loss(P^*) \leq 0.01$. To compute the gradient, it perturbs β by $+0.01$, $CIR(0)$ by $+0.1$, and other parameters by $+1$, then it computes their respective losses. Gradient w.r.t. a parameter is the ratio of change in loss and change in parameter value. Now, the parameters are updated based on the formula described in assignment. We repeat the above process until P reaches P^*.

- *plot_cases* and *plot_susceptible* function is used to plot the number of cases and fraction of susceptible population respectively in all five scenarios for a given date range *date_start* to *date_end*.

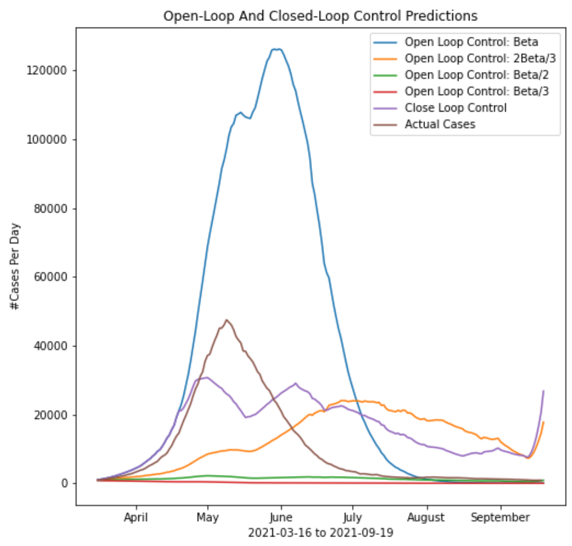
Experiment:

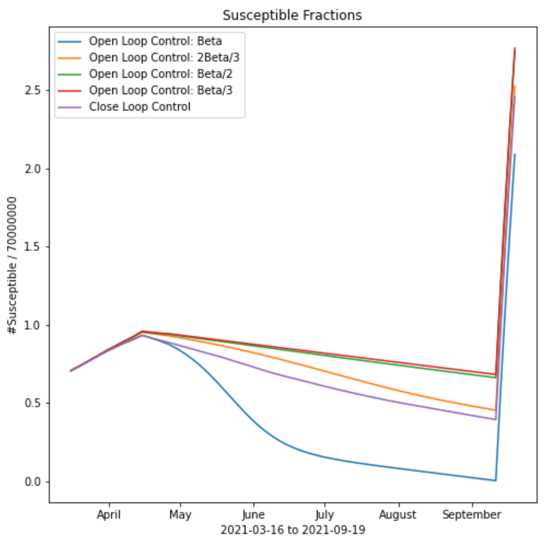
First, I choose my initial parameters by trying all possible combinations of parameters by iterating in long step sizes to pick the parameters which give the smallest loss so far. Then I ran gradient descent algorithm on these initial parameters but after certain number of iterations parameters were updating very slow as the updates are dependent on number of iterations. To speed up the process the process I reset the number of iterations to zero after certain number of iterations. But sometimes it causes overshooting from the minima.

Output:

Training Loss = `0.009351034143708413`

Parameters	**Initial**	**Optimal**
β	`4.5`	`0.4509617296888799`
$S(0)$	`48999999.9`	`48999999.89999955`
$E(0)$	`73499.918`	`73499.91798234466`
$I(0)$	`73499.9183`	`73499.91824215087`
$R(0)$	`20852999.9`	`20852999.899999607`
$CIR(0)$	`12.0`	`14.234270941158353`





* I didn't estimate the predictions after 19 September 2021 as we don't have *Tested* data afterwards.