Epidemiology modeling

Aleksandra Prorok
Marcin Dwużnik

May 2024

1 Task 1

As the aspect of epidemic outbursts to explore, we have chosen the Mortality, but also the influence of the vaccination effectiveness and fluctuation between gaining and losing immunity.

2 Task 2

Aims and general properties We aim to model the dynamics of an epidemic with a focus on the impacts of vaccination and mortality. We would like to understand how vaccination influences the spread of the disease and the overall mortality rate. By introducing additional compartments to the traditional SEIR model, we aim to capture the complexities introduced by vaccination and death cases.

To capture the targeted aspects of the epidemic, we will:

1. Introduce a death compartment to account for fatalities due to infection.
2. Include a compartment for vaccinated individuals who are still susceptible, recognizing that vaccines are not always fully effective.
3. Incorporate a vaccination compartment for the individuals who have been vaccinated.
4. Use differential equations to model the transitions between compartments based on specific parameters.
5. Moreover, we also provided a parameter for those, whose vaccination failed - we allow them to return to *Susceptible* compartment and get another vaccination which this time might be successful and give them immunity.

Our main goal is to investigate the *Dead*, *Vaccinated*, and *Failed vaccination* and also therefore analyze the influence of the change of the corresponding parameters. The compartments and parameters are presented and described in Task 3.

To estimate the parameters of the model, the potential sources of data could be:

- the vaccine efficacy from clinical trial data,
- statistics about mortality or severity of the disease for a specific period of time
- information about possible mobility of infectious individuals, lockdowns, and restrictions in society.

3 Task 3

The key elements of the model include the following compartments:

- **Susceptible (S):** Individuals who are not yet infected but can contract the disease.
- **Exposed (E):** Individuals who have been exposed to the disease but are not yet infectious.
- **Infected (I):** Individuals who are currently infectious.
- **Recovered (R):** Individuals who have recovered from the disease and gained immunity.
- **Vaccinated (V):** Individuals who have been vaccinated and gained immunity.
- **Failed vaccination (V_failed):** Individuals who have been vaccinated but remain susceptible due to ineffective vaccine.
- **Dead (D):** Individuals who have died because of the infection.

Due to the introduced compartments, we are obligated to include the following parameters:

- β: infection rate - it is influenced by the *number of contacts* and *probability of infecting*.
- γ: recovery rate,
- σ: infection rate in exposed individuals,
- α: rate at which individuals leave the *Recovered* compartment,
- α_**v**: rate of immunity loss after the vaccination,
- α_**v_failed**: rate at which individuals leave the *Failed vaccination* compartment - enables them to revaccinate,
- **v_rate**: vaccination rate,
- **v_success**: rate of vaccine efficacy,
- δ: mortality rate.

4 Task 4

We have decided to put our project into the Streamlit app, which lets us change the parameters easily. **Initial conditions:**

- Population size:
- Initial number of infected individuals:

5 Task 5

Scenario 1: Hospital Capacity

The parameters that can be controlled by social restrictions are:

- *prob_of_infecting*: 'Probability of Infecting', which can be reduced by social distancing, wearing masks, disinfecting social spaces, etc.;
- *avg_no_contacts_per_individual*: 'Average Number of Contacts per Individual', which casually can be minimized by the well-known approach *Stay Home, Save Lives*,

which have a direct impact on the β parameter.

Below, we present the way to find optimal values for the above parameters. Our attention should be attracted by the bold green line, which represents *Infected* individuals.

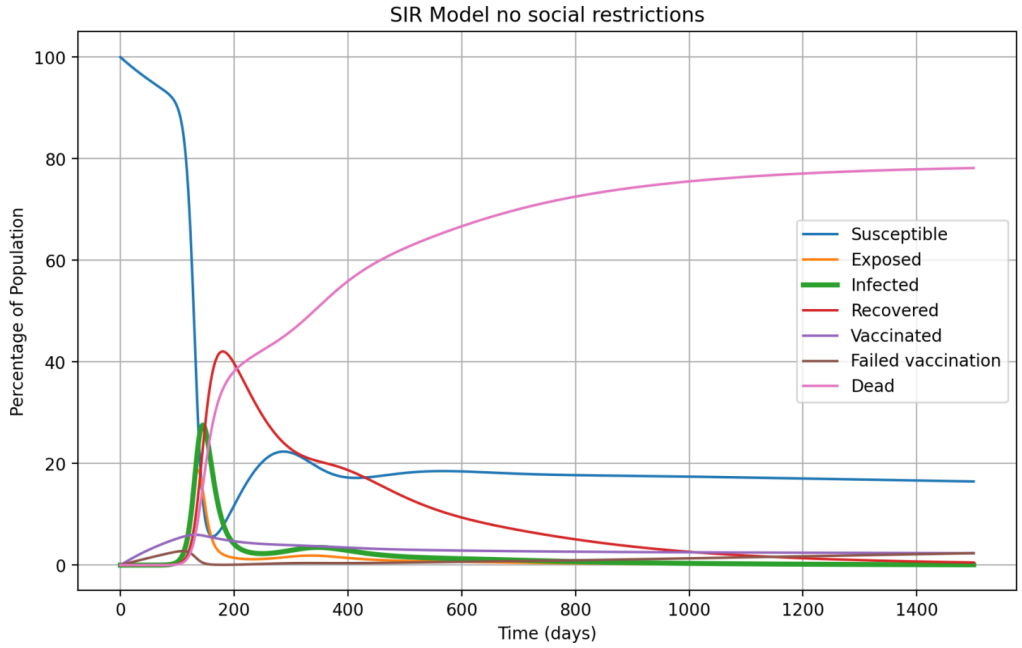

Figure 1: Parameters: *prob_of_infecting*: 0.02, *avg_no_contacts_per_individual*: 20

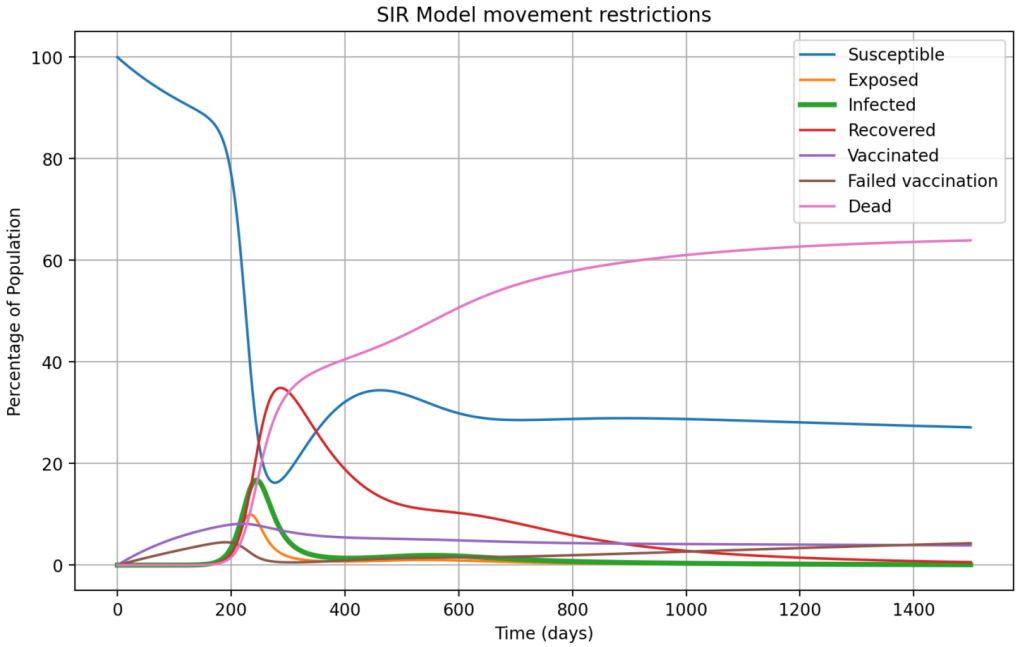

Figure 2: Parameters: *prob_of_infecting*: 0.02, *avg_no_contacts_per_individual*: 12

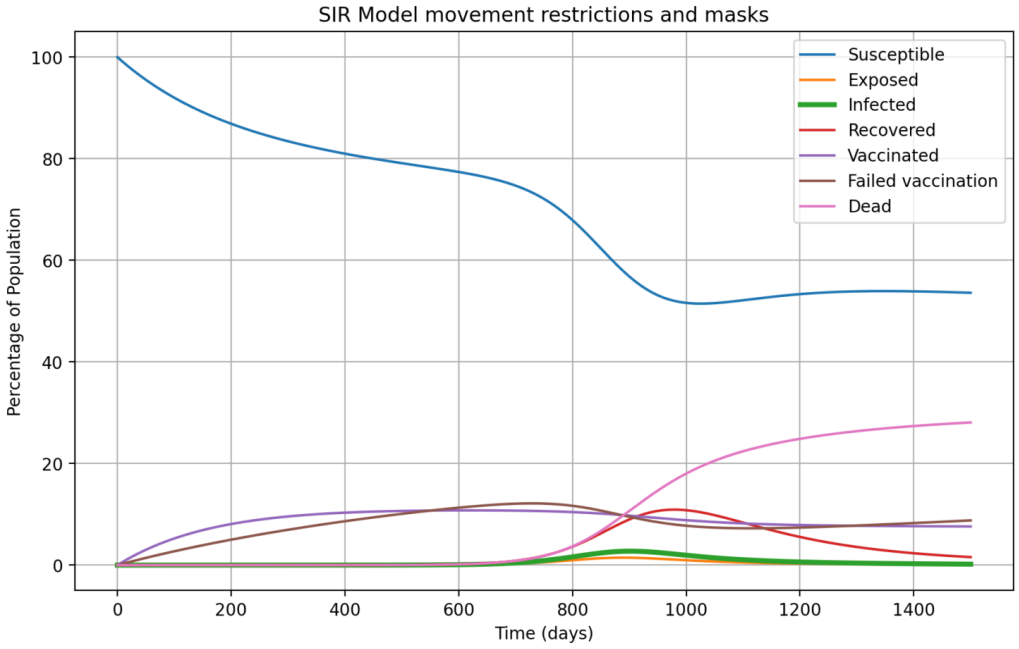

Figure 3: Optimal parameters proper for this scenario: *prob_of_infecting*: 0.01, *avg_no_contacts_per_individual*: 12. As we can see, the *Infected* line is below 10%.

Social restrictions led to a refined line, decreased the value of infections to only a few percent of the population, and delayed the "peak" of the infections, which led us to the conclusion that wearing masks, social distancing, and restrictions are effective ways to fight the epidemic.

Scenario 2: Seasonal Variation

To investigate how the epidemic would evolve with seasonal variation we created a special tab on our Streamlit app, which allows us to choose the parameter, number of changes per year, and the value of selected parameters in each part of the year.

I've decided to investigate the *gamma* - the recovery rate - the rate at which *Infected* individuals join the *Recovered* compartment. I've decided to check, how two seasonal variations per year would impact the development of the epidemic and the sizes of the compartments.

I tested five sets of parameters - some more and some less extreme. All parameters excluding *gamma* remained the same.

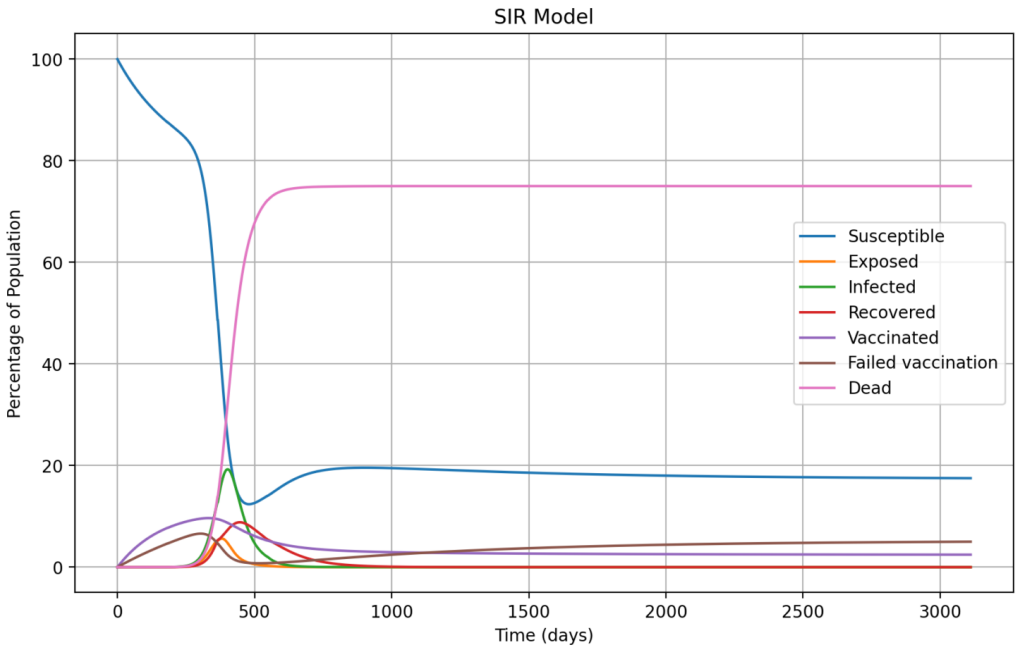

Figure 4: Parameter γ variation: 0.7%, 1.5%

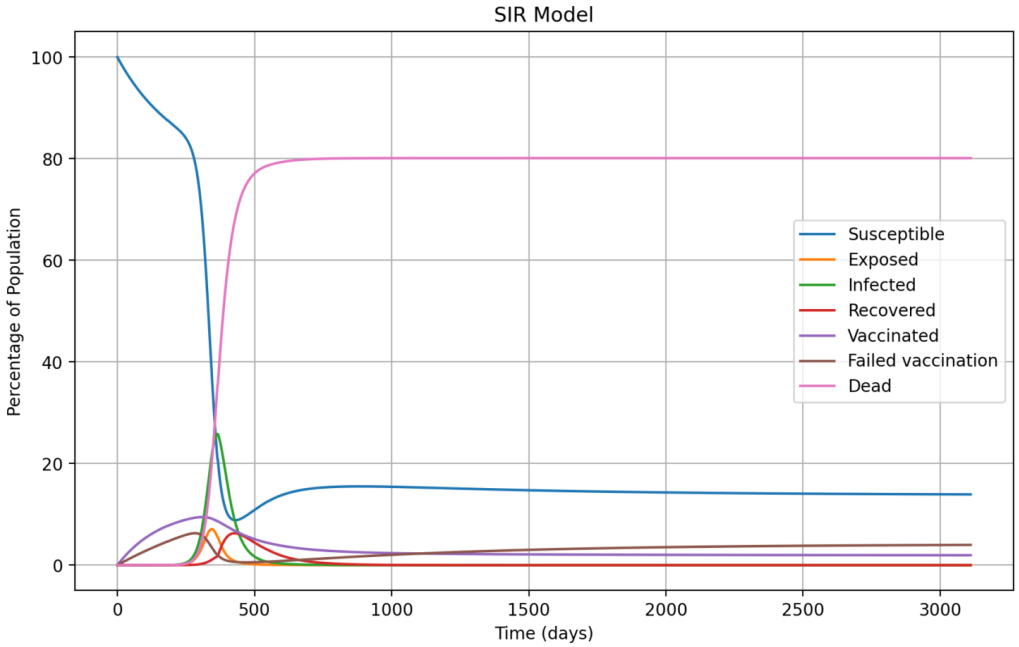

Figure 5: Parameter γ variation: 0.7%, 0.2%

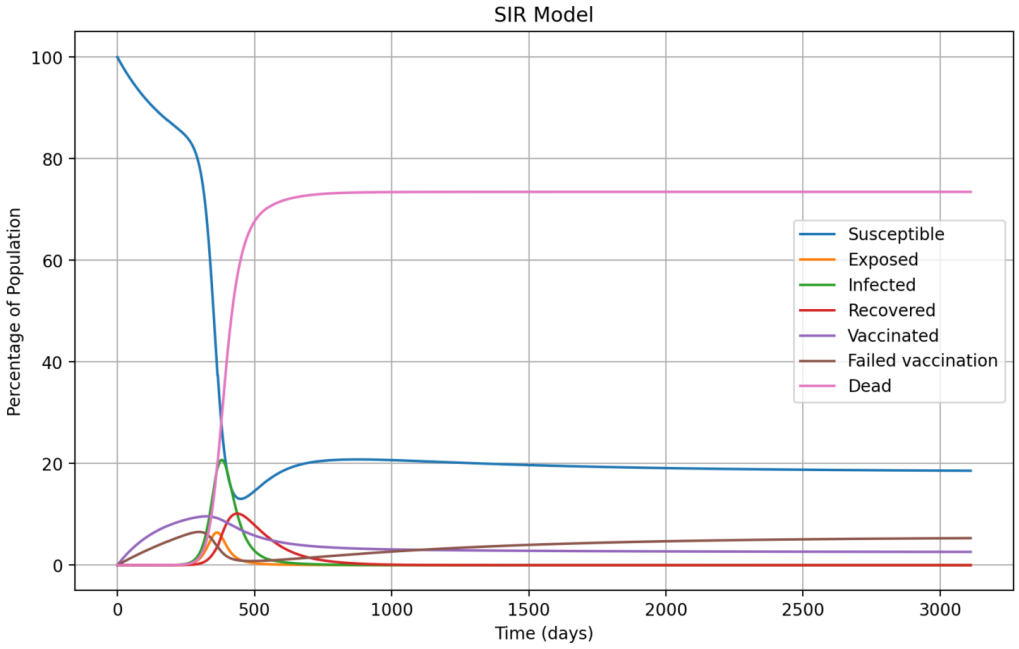

Figure 6: Parameter γ variation: 1%, 0.7%

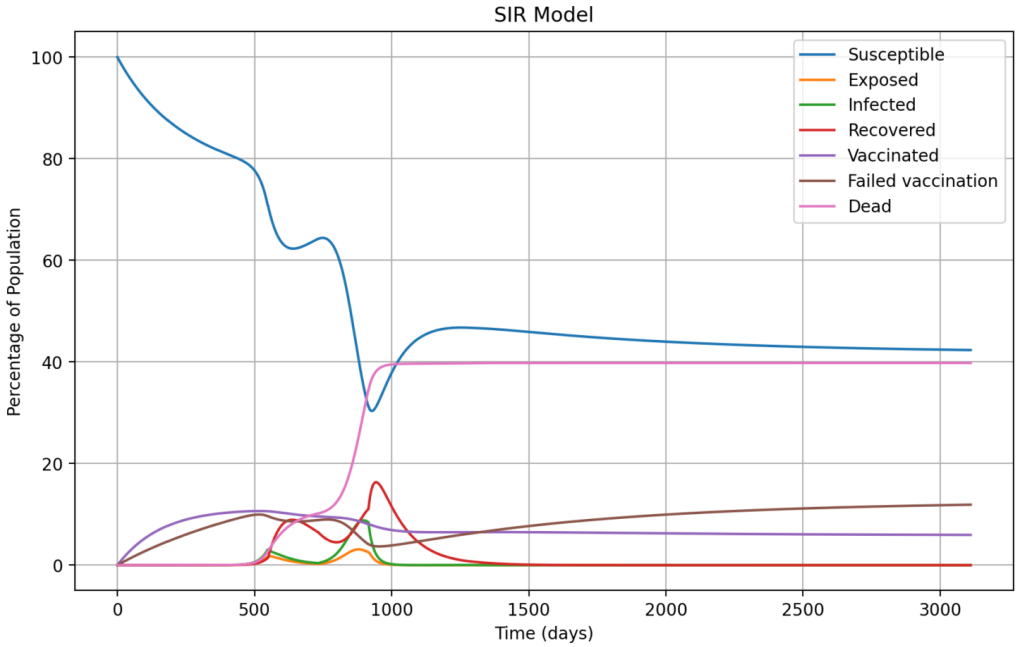

Figure 7: Parameter γ variation: 2%, 7.5%

This parameter set was the most extreme and high - we can notice the instant fall around the 1000th day of simulation and violent fluctuations between *Susceptible*, *Recovered*, and *Dead* compartments.

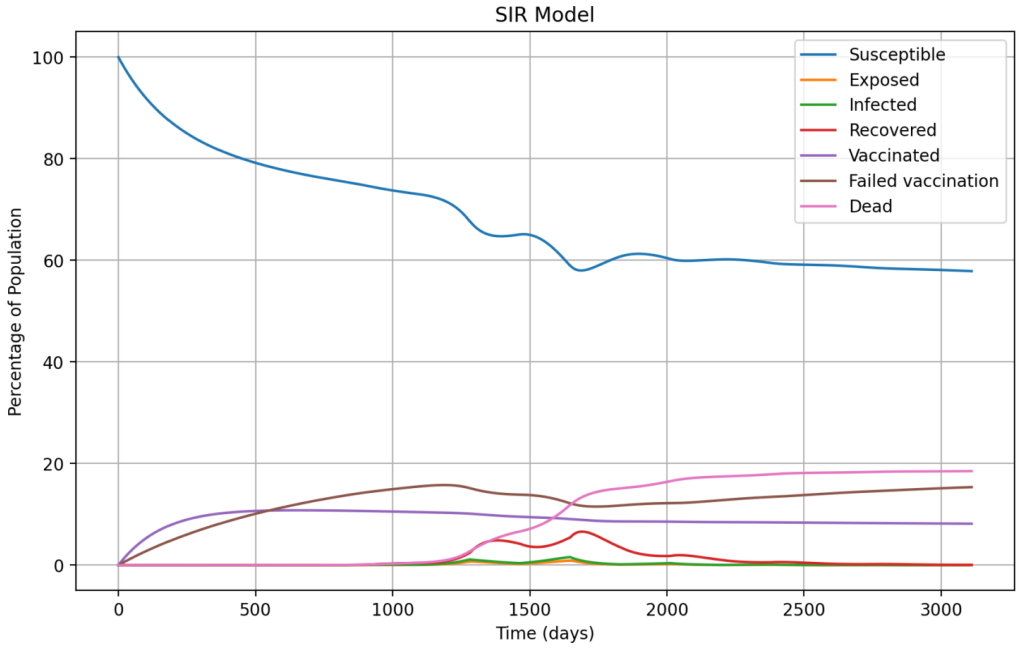

Figure 8: Parameter γ variation: 5%, 7.5%

In this case, both values of the parameter are high, and as a result, most individuals remain in *Susceptible* compartment.

In every situation, the seasonal variation is not noticeable at the beginning or end of the simulation. When *gamma* is low, the *Dead* compartment is common to achieve very high levels. By choosing bigger values, we can see the decrease of the *Dead* percentage.

The results might not seem obvious at first, because the variation happens every 183 days, but we don't observe any major changes in the trajectories that often, but it makes perfect sense. What this parameter dictates, is the rate at which individuals leave the *Infected* compartment, so we would not notice any significant changes unless there are many infections at a given time, that is why most of these simulations seem very similar before and after the peak of infections and are only visible when the gap between the two parameter values is large, and the values are high.

The conclusion is that the mild seasonal variations don't have a major impact on the epidemic development. It seems like what's changing the results of these simulations isn't exactly the seasonality of the parameter change, but rather what the average of the two seasonal values is.

Scenario 3: Vaccination rate vs vaccination success rate

As the third scenario, I've decided to check if there is a significant difference in the size of the *Dead* compartment, depending on the *v_rate* and *v_success*. All parameters excluding *v_rate* and *v_success* will be the same:

- 'beta': 1.3×10^{-5}

- 'sigma': 0.14,
- 'gamma': 0.05,
- 'alpha': 0.02,
- 'alpha_v': 0.005,
- 'alpha_v_failed': 0.001,
- 'delta': 0.03

1. in this scenario we explore a model without vaccinations.

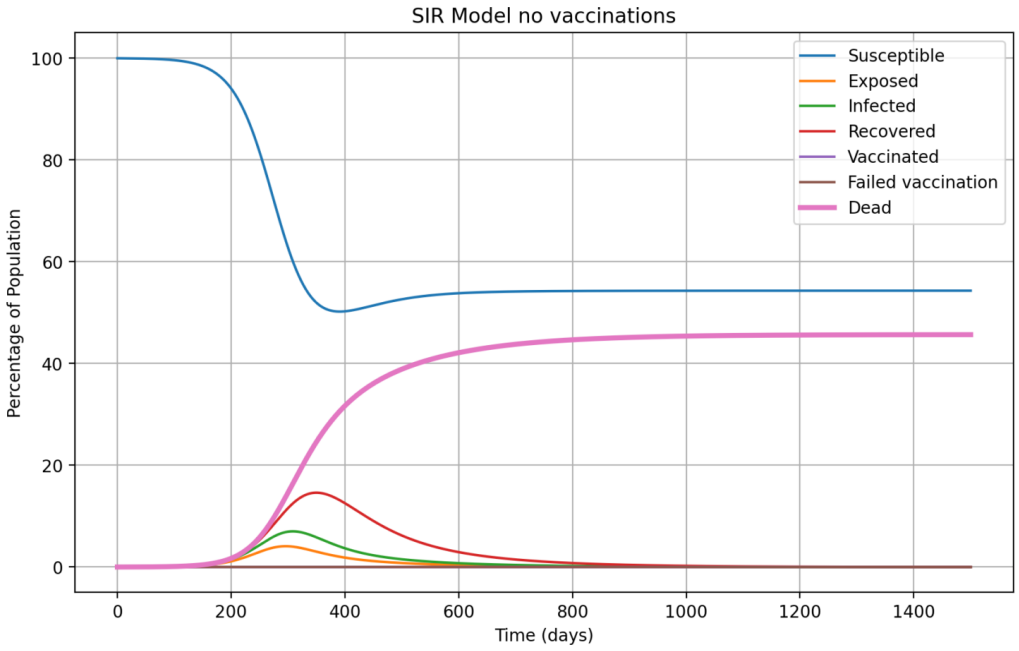

Figure 9: Parameters: *v_rate*: 0.0, *v_success*: 0.0

Almost 50% of people end up in the *Dead* compartment. The model stabilizes around day 800.

2. in this scenario we explore a high vaccination rate but a low vaccination success rate.

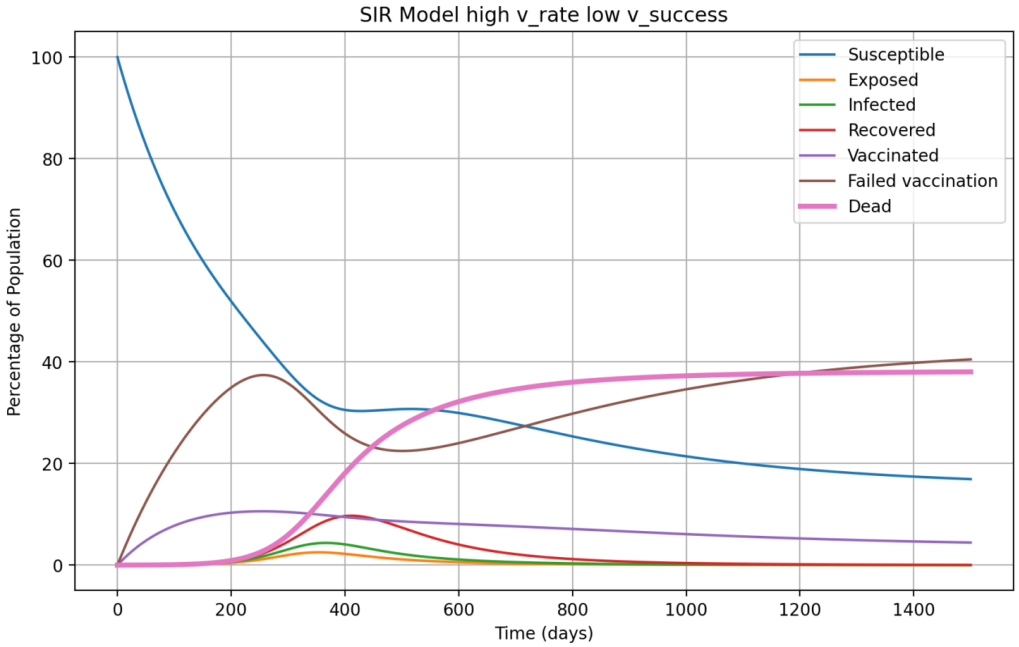

Figure 10: Parameters: *v_rate*: 0.004, *v_success*: 0.3

We can see that below 40% of people end up in the *Dead* compartment this time, which is less than in the no vaccination model. It's important to note that it's not **that much less**. We can see that at one point almost 40% of people tried to get vaccinated but they failed to gain immunity and were still able to get *Exposed* and *Infected*. We also see that the model took longer to stabilize, which flattened the curve, meaning fewer people were *Infected* at the same time than in the previous model.

3. in this scenario we explore a low vaccination rate but a high vaccination success rate.

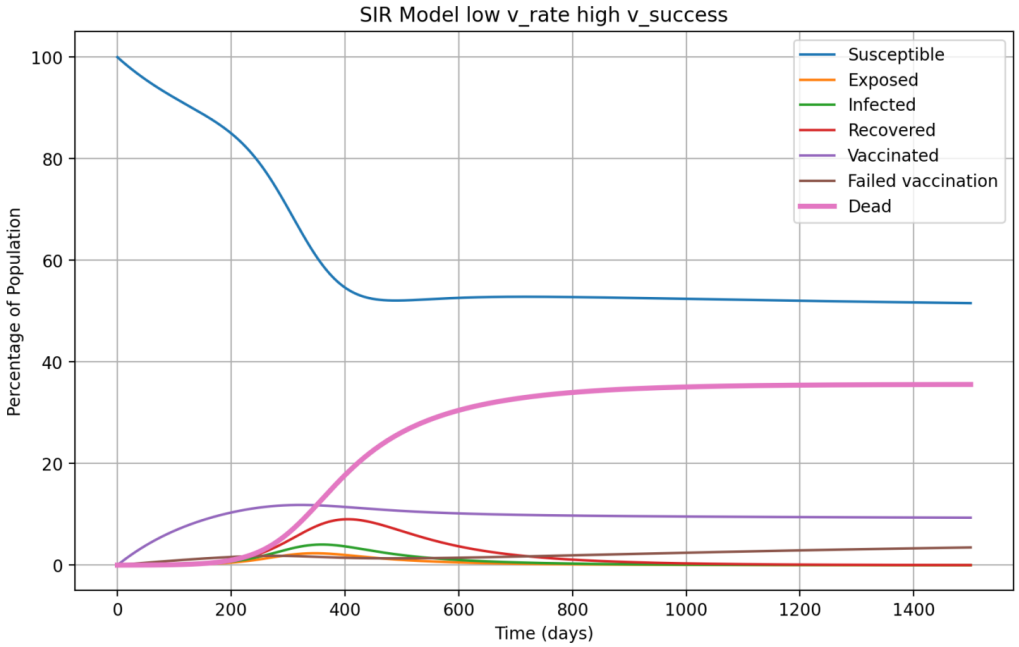

Figure 11: Parameters: *v_rate*: 0.001, *v_success*: 0.9

We can see very similar results to the previous model in terms of the *Dead* compartment, which is expected because the number of people in the *Vaccinated* (not counting *Failed vaccination* compartment is relatively the same.

4. in this scenario we explore a high vaccination rate and a high vaccination success rate.

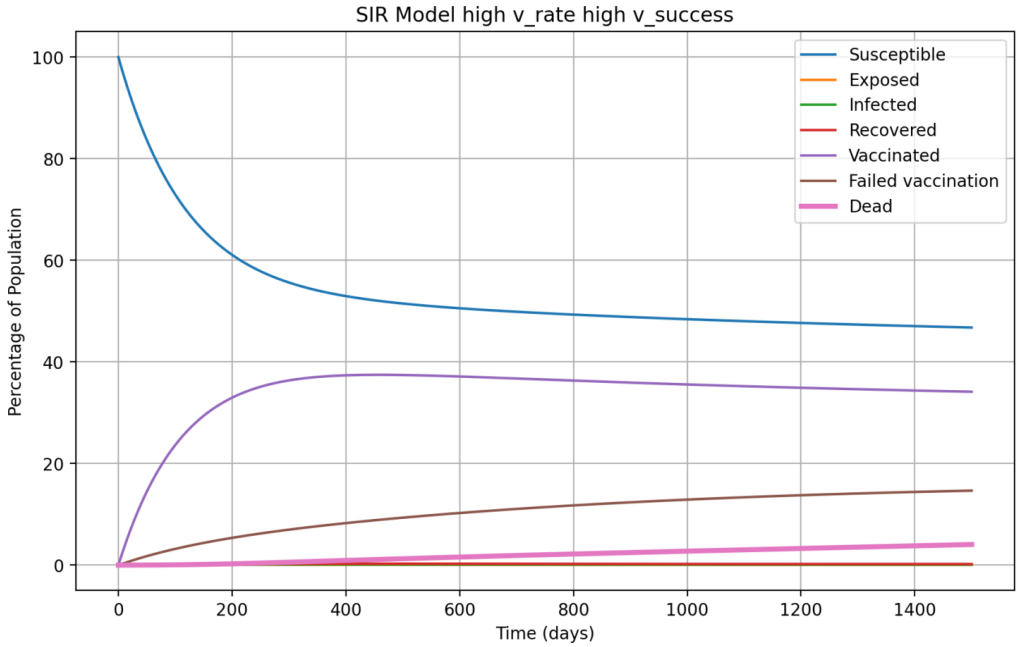

Figure 12: Parameters: *v_rate*: 0.004, *v_success*: 0.9

Finally, when combining an effective and popular vaccine we're able to keep the *Dead* compartment below 10% at day 1500. We can also see that the *Infected* compartment is very close to 0% at all times, proving that vaccinations can greatly reduce the stress on hospitals during a pandemic.

6 Task 6

Functions generating data disturbed by seasonal variations and random noise are included in the code.

7 Task 7

In this task, I decided to predict the parameters for data generated from scenario 1 - the hospitalization - and scenario 3 - the vaccinations with added noise and seasonal variations. I will be showcasing 3 prediction functions from scipy.optimize:

- minimize
- least_squares,
- differential_evolution,

Both **minimize** and **least_squares** require initial guesses to find optimal parameters, while **differential_evolution** does not. I will always use the same initial parameters for prediction purposes:

- 'beta': 0.000001,
- 'sigma': 0.15,
- 'gamma': 0.05,
- 'alpha': 0.01
- 'alpha_v': 0.005,
- 'alpha_v_failed': 0.001,
- 'delta': 0.03,
- 'v_rate': 0.001,
- 'v_success': 0.70,

These parameters are very similar to some of the ones used in earlier models, so it should help in getting better results using these prediction functions.

The **differential_evolution** function does not use initial guesses, instead, it tries to find optimal parameters in given bounds - we make reasonable assumptions on where each parameter we're trying to predict might be - the smaller the bounds, the more accurate (and less computationally intensive) the search. I also added bounds to the other two functions, for the **minimize** function these are the same as for the **differential_evolution** function.

Bounds:

- `'beta'`: (0, 0.1),
- `'sigma'`: (0, 0.4),
- `'gamma'`: (0, 0.4),
- `'alpha'`: (0, 0.4),
- `'v_rate'`: (0, 0.1),
- `'v_success'`: (0, 1),
- `'alpha_v'`: (0, 0.4),
- `'alpha_v_failed'`: (0, 0.4),
- `'delta'`: (0, 0.2),

For the **least_squares** function I used (0, 1) bounds for all parameters.

Before running the prediction functions, I added a de-noising layer using the **bilateralFilter** function with *kernel_size=23*, *sigma_color=777777*, *sigma_space=70* parameters, to try to mitigate the effect of random noise worsening the quality of the predictions.

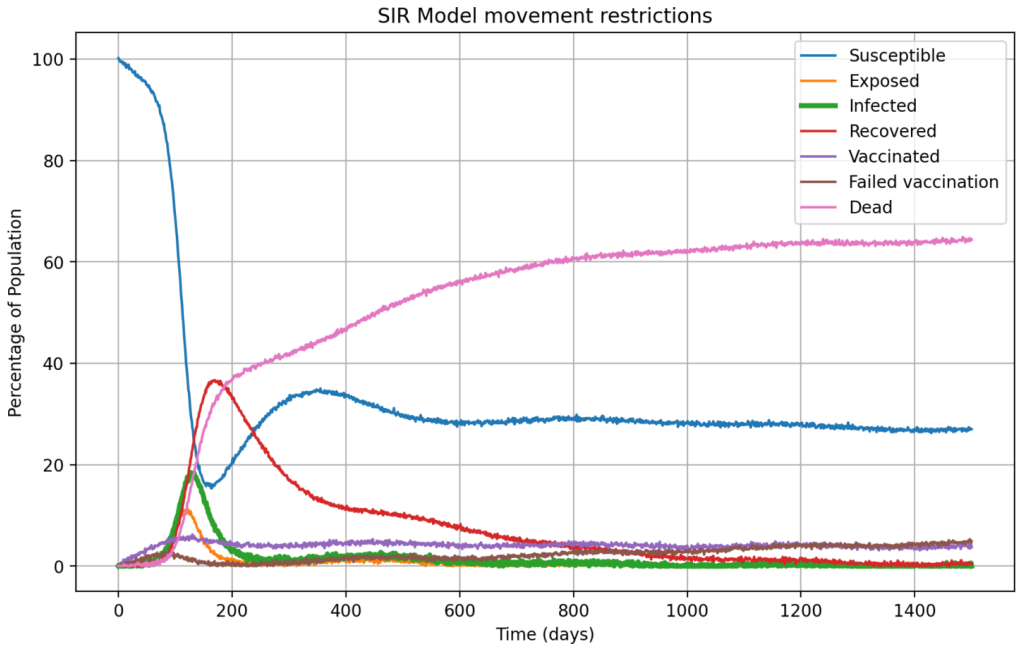

Figure 13: Movement restrictions data with noise

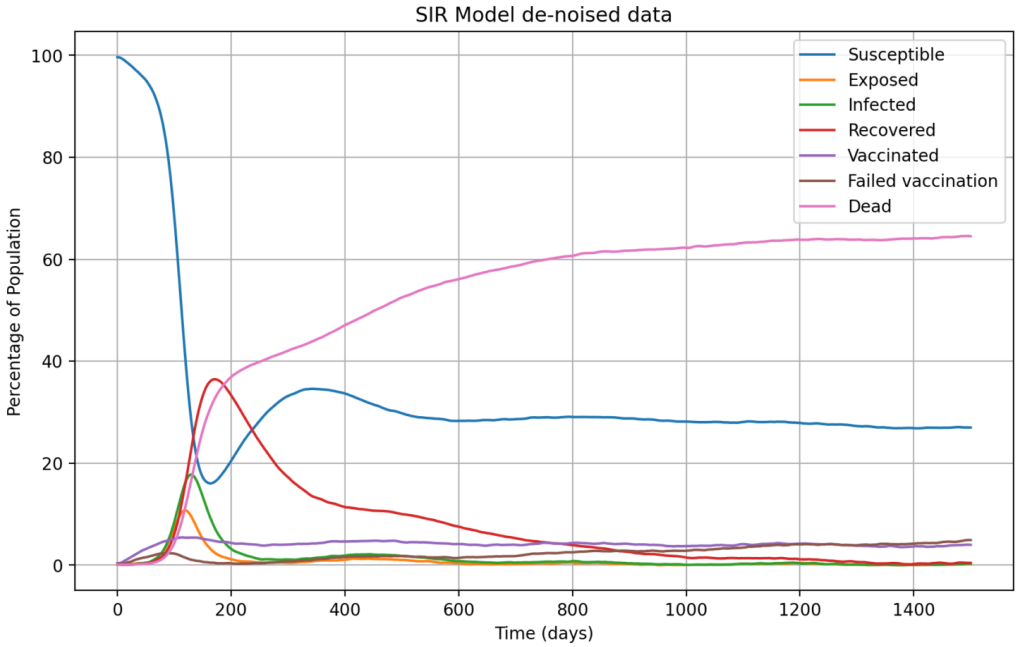

Figure 14: Movement restrictions data after de-noising

For each of the scenarios, I will use the prediction functions on only one of the generated plots - this will be enough to demonstrate how well each prediction function does in each scenario.

Hospitalisation

Parameter	**Original values**	**minimize**	**least_squares**	**differential_evolution**
beta	2.4e-05	8.177167e-05	0.121323350	2.408797e-05
sigma	0.15	0.058704769	0.098743299	0.149398775
gamma	0.05	0.179985298	0.558033007	0.050055468
alpha	0.01	0.009332747	0.001821988	0.010053484
v_rate	0.001	0.008348394	0.482369660	0.001004598
v_success	0.7	0.029630183	0.678233878	0.713358097
alpha_v	0.005	0.065554450	0.001633414	0.005194541
alpha_v_failed	0.001	0.087390781	0.116787221	0.000923955
delta	0.03	0.111804434	0.725754036	0.030029114

Table 1: Comparison of original and predicted values (rounded to 9 decimal places) for movement restriction from the Hospitalization scenario.

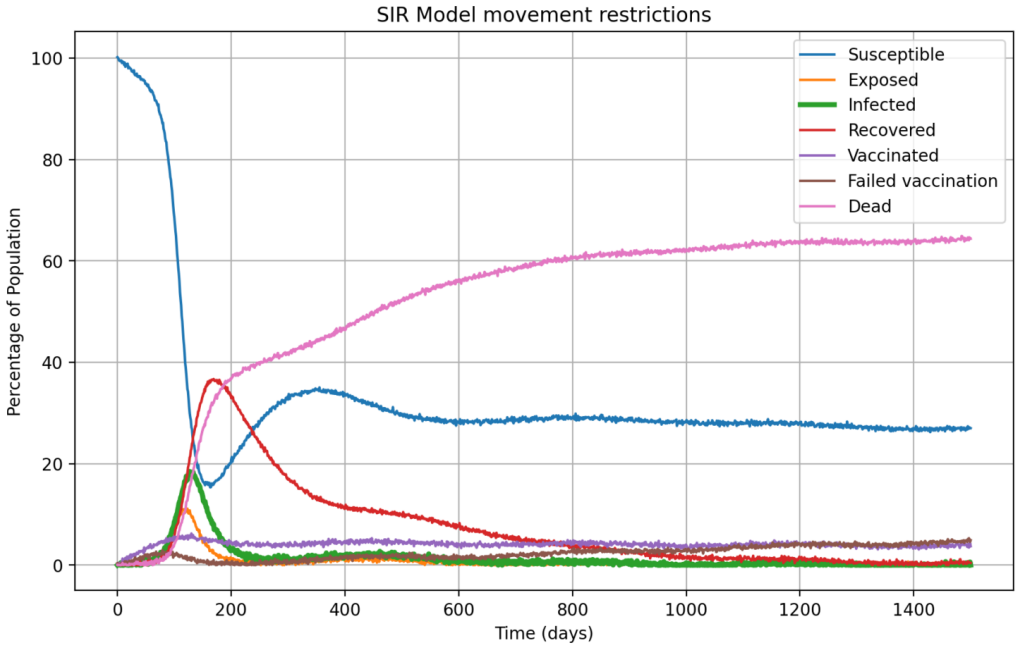

Figure 15: Model for movement restrictions with noise and seasonal variations.

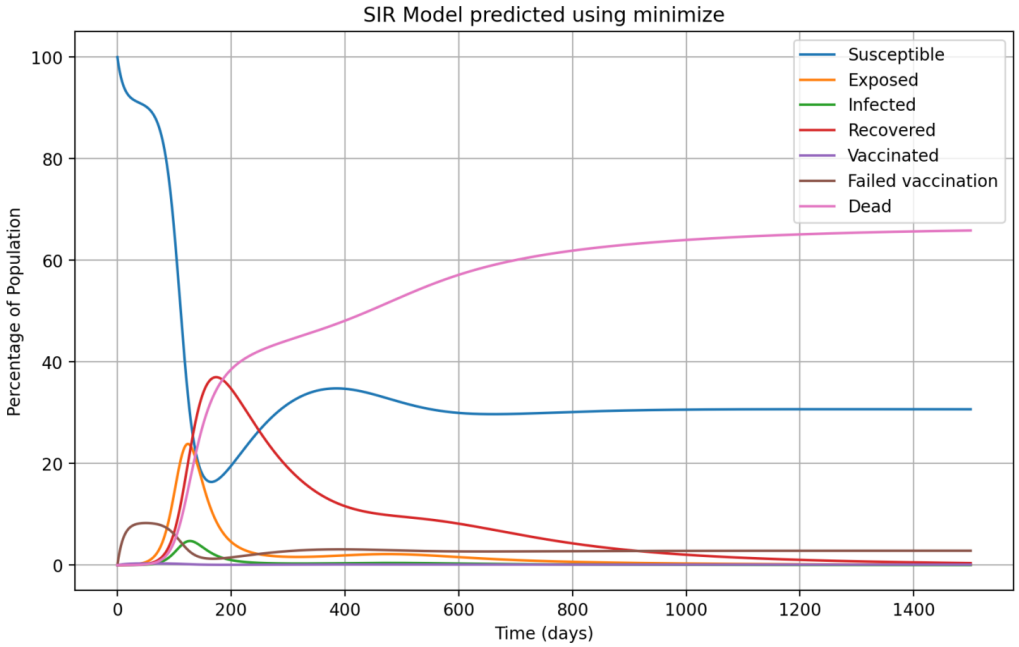

Figure 16: Prediction using minimize. *Summed Error: ∼1.128*

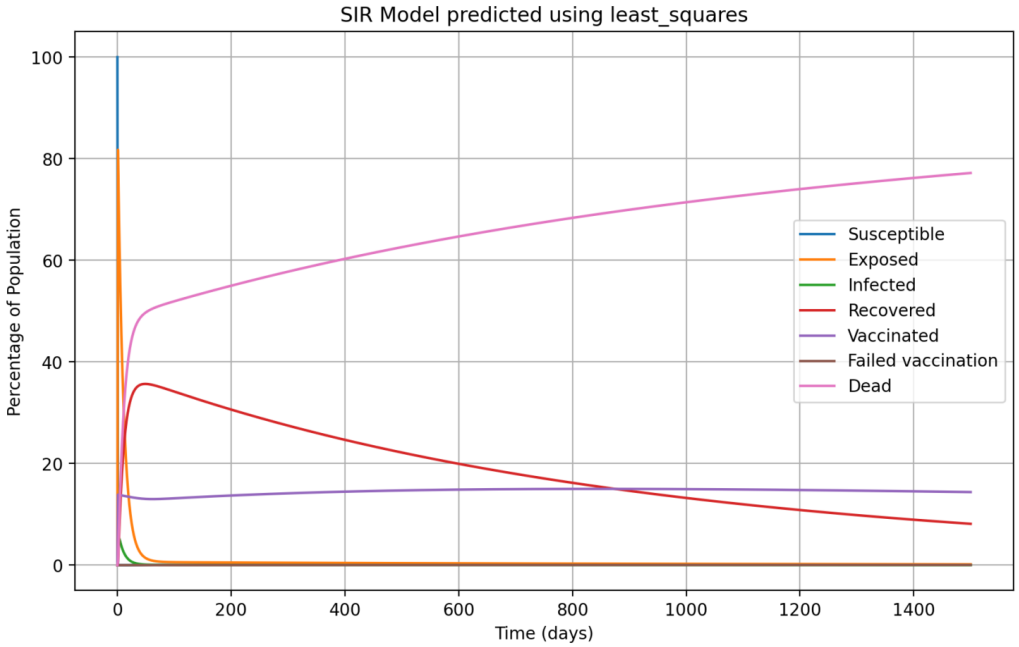

Figure 17: Prediction using least_squares. *Summed Error: ∼2.007*

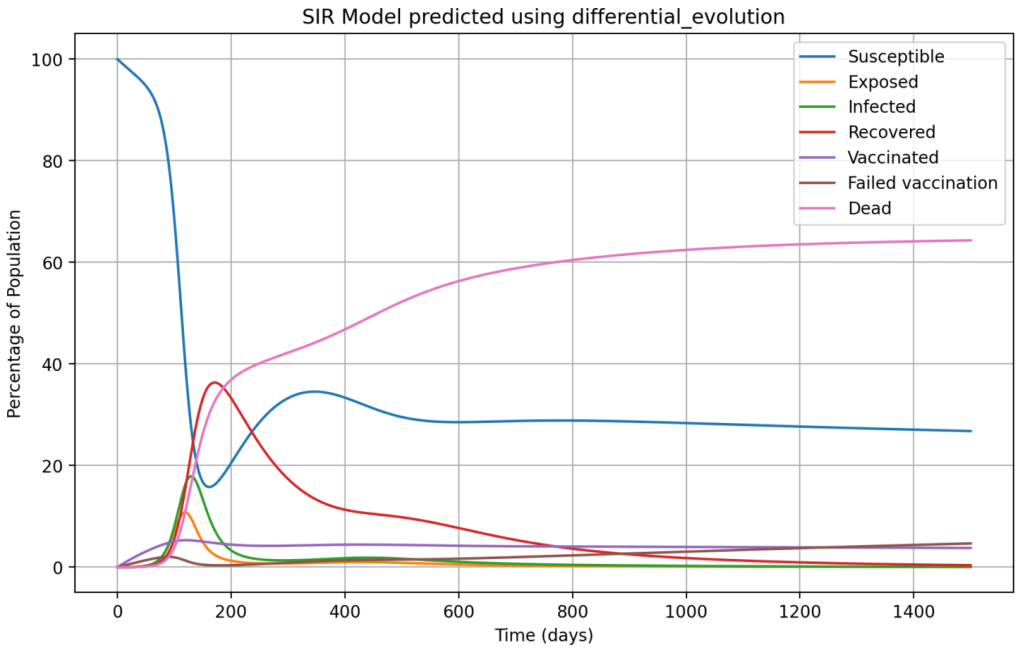

Figure 18: Prediction using differential_evolution. *Summed Error:∼0.014*

We can see that judging by the Summed Error and the shape of the trajectories, the **differential_evolution** function provided us with the best prediction of parameters, and the **minimize** function was the second best. It's worth noting, that despite its spectacularly accurate results, the **differential_evolution** function took significantly longer than the other prediction methods.

Vaccinations

Parameter	Original values	minimize	least_squares	differential_evolution
beta	1.300000000e-05	1.697203592e-05	1.704515801e-05	6.389009278e-02
sigma	0.14	0.151745152	0.149979293	0.299782535
gamma	0.05	0.064261578	0.054509	0.263046878
alpha	0.02	0.038736263	0.010435792	0.359946
v_rate	0.004	0.007690125	0.012610657	0.00097443
v_success	0.3	0.643744828	0.699875712	0.567541675
alpha_v	0.005	0.021278554	0.012611168	0.222249854
alpha_v_failed	0.001	0.001213086	0.000984912	0.356003338
delta	0.03	0.045301863	0.033172234	0.021664735

Table 2: Comparison of original and predicted values (rounded to 9 decimal places) for high *v_rate* and low *v_success* from the Vaccines scenario.

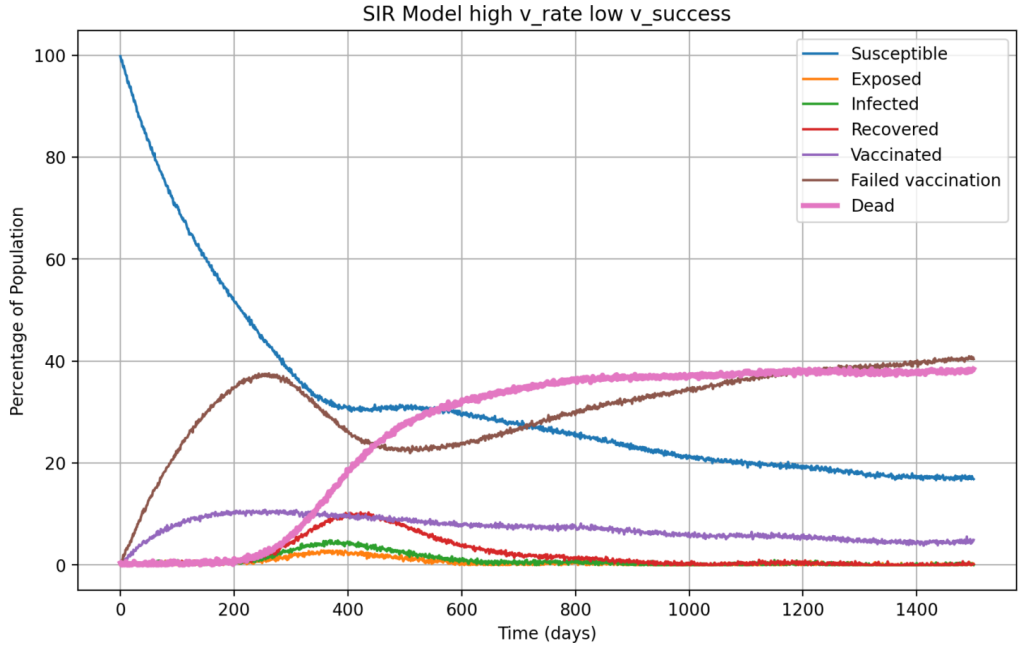

Figure 19: Model for high *v_rate* and low *v_success* with noise and seasonal variations.

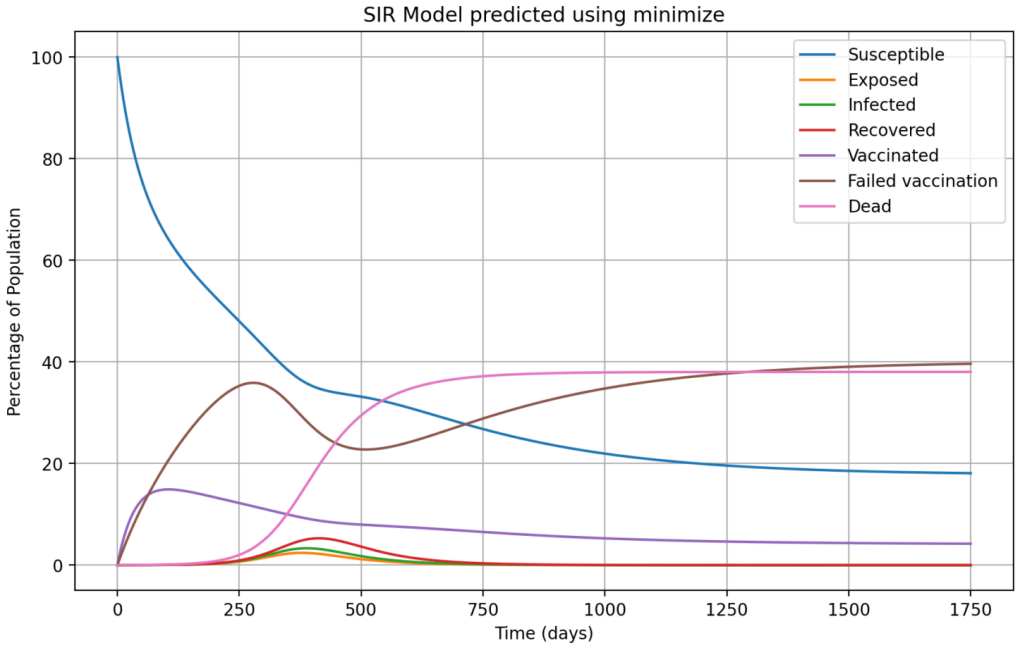

Figure 20: Prediction using minimize. *Summed Error: ∼0.424*

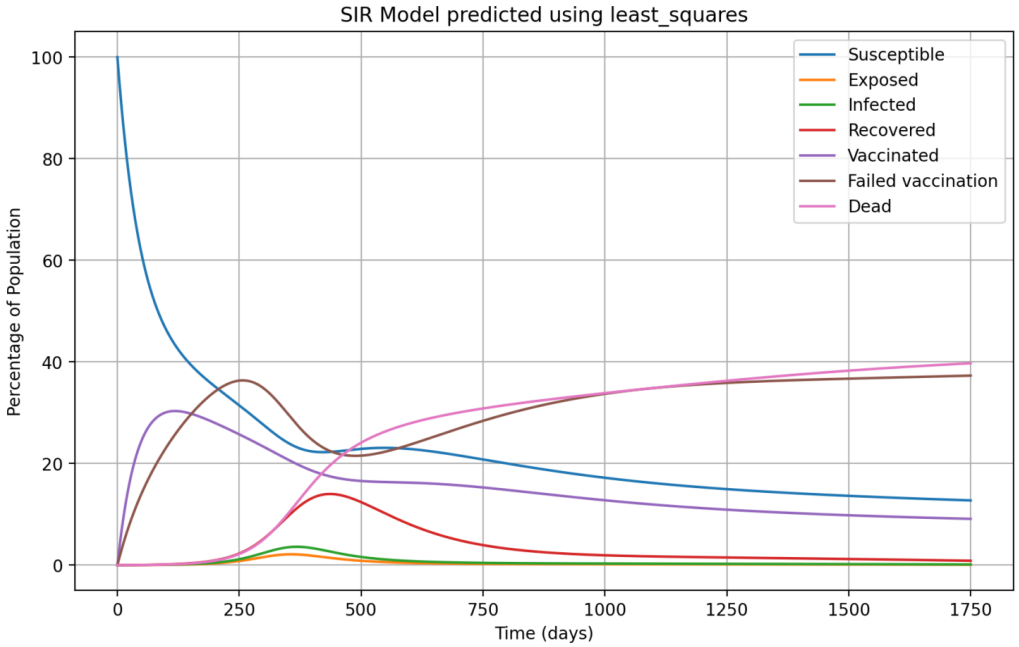

Figure 21: Prediction using least_squares. *Summed Error: ∼0.443*

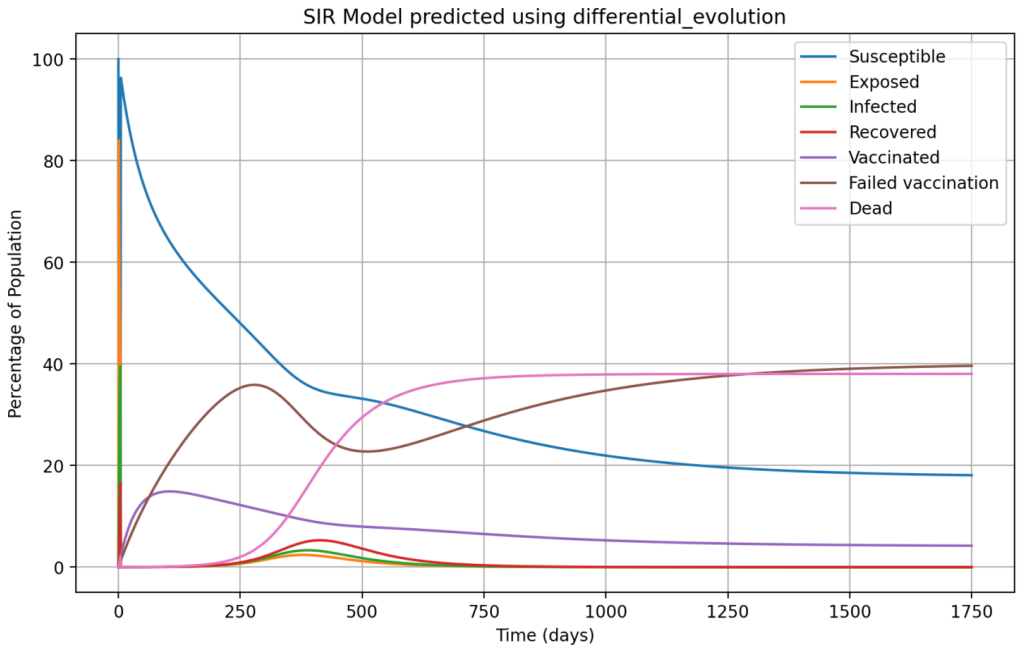

Figure 22: Prediction using differential_evolution. *Summed Error:∼1.628*

This time, it's the **minimize** function that has the smallest Summed Error of all of the functions, however looking at the trajectories, all of the prediction functions have produced reasonable results. What's worth taking a closer look at is the **differential_evolution** function result - on one hand, it has by far the highest error of the three prediction functions. On the other, I would say it's really impressive, that it achieved a trajectory this accurate with parameters this wrong.

Additionally, this result of the **differential_evolution** function is an outlier - when rerunning the prediction, it returned a result with an incredible 0.00077 Summed Error, meaning that pretty much every parameter was predicted perfectly. The other functions returned practically the same results as in the previous simulation. What we can learn from this, is that it's worth running the predictions at least a couple of times for a higher chance of an accurate result.

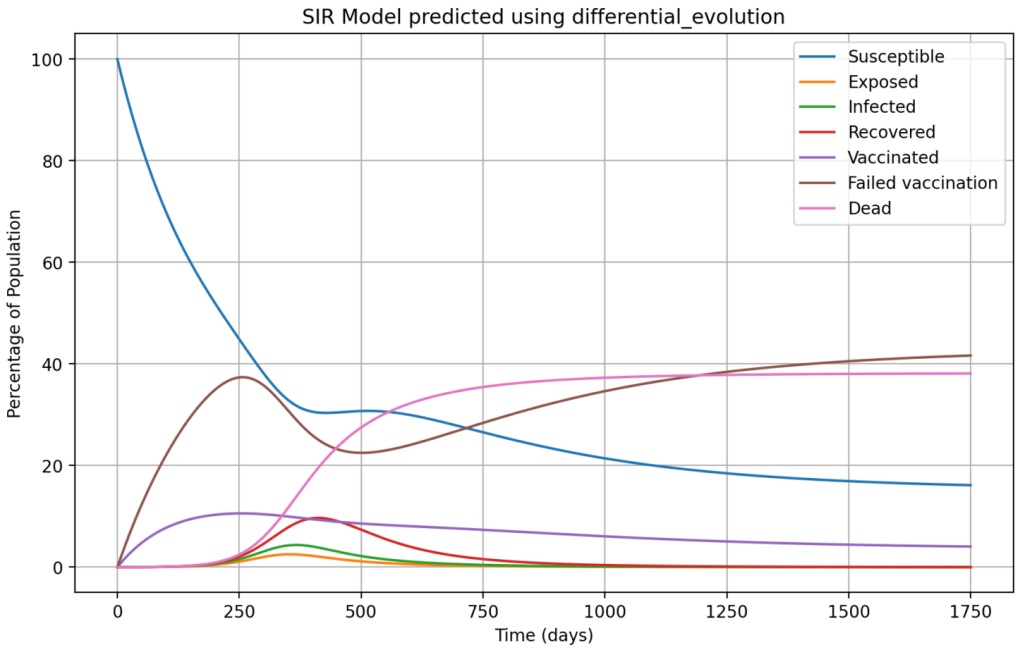

Figure 23: Another prediction using differential_evolution. *Summed Error:∼0.00077*