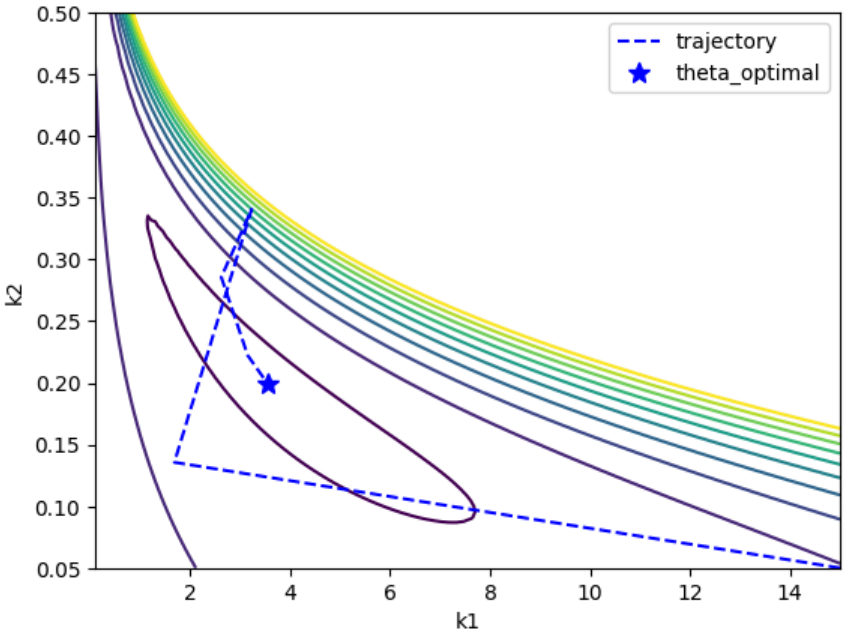
CaleyNovitskaia

HW1

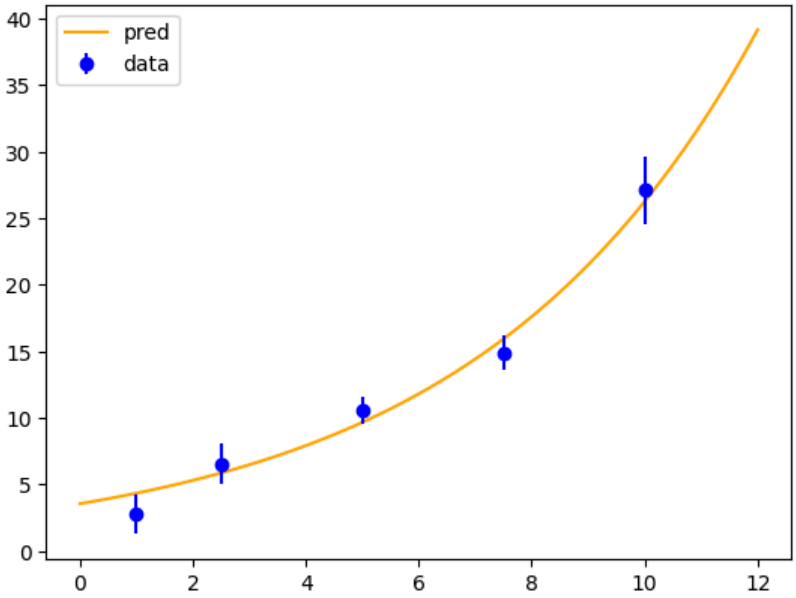
b)



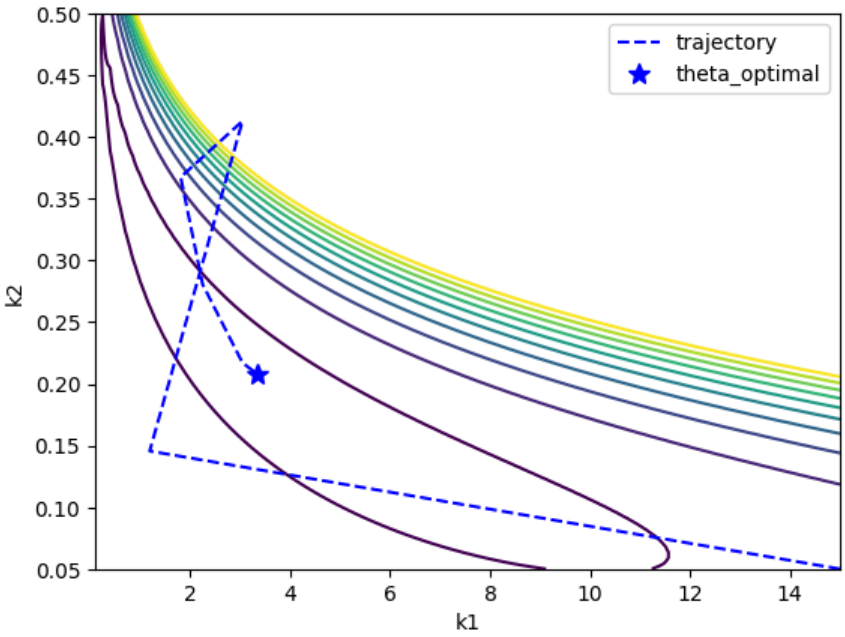
Algorithm converges to the optimal value. The trajectory for θ does not look smooth in the beginning, in contrast, near the solution it is smooth. Also the step lenght is not constant, it is larger in the beginning and smaller at the end of the run, near the optimal solution.

c)

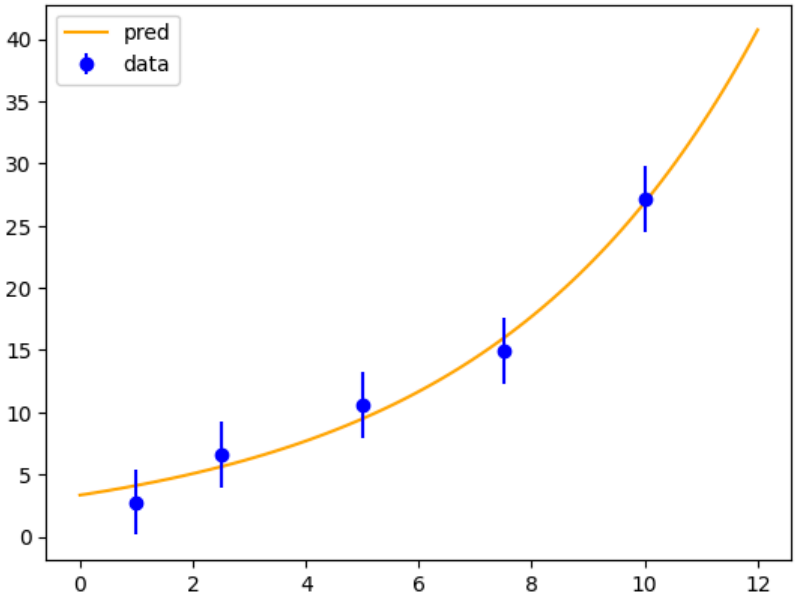
```
<matplotlib.legend.Legend at 0x108dc8e50>
```



d)




```
<matplotlib.legend.Legend at 0x108f14b50>
```



After changing σ^2, the optimal solution did not change dramatically, but plots became a bit different.

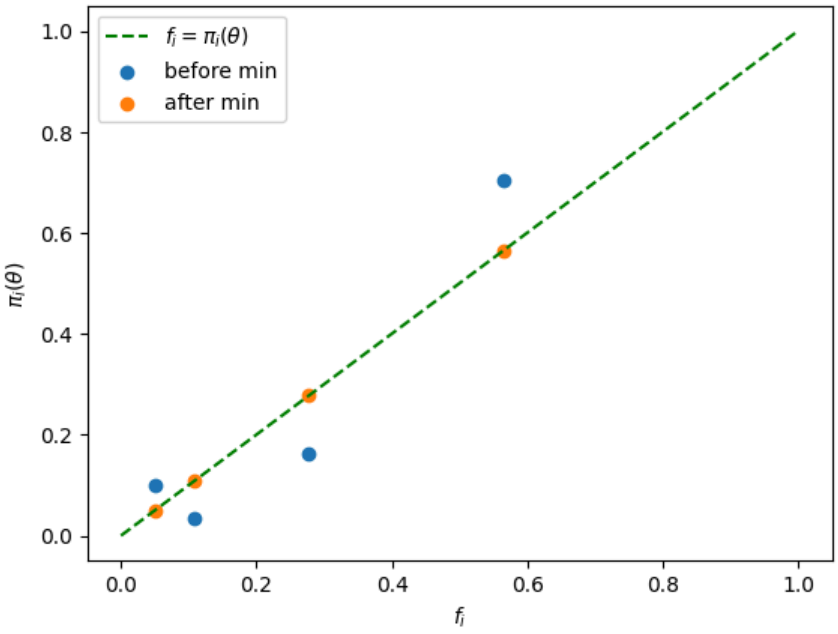
According to a contour plot, it seems that the levels are bit wider, additionally, some steps of the algorithm are bigger after changing σ^2, θ_s went out of the courtor lines on the second iteration.

According to the prediction-data plots, the error bars have different lenghts and have smaller values in comparison to the second plot (after changing σ^2). In addition, the predicted solution crosses all the error bars in the second plot (after changing σ^2) in contrast to the results on the first plot.

HW2

b)

```
<matplotlib.legend.Legend at 0x10f119790>
```



As we see, the orange points associated with the stationary distribution after minimization align the line $f_i = \pi_i(\theta)$, therefore the minimization works.

Discussion

Discuss if you think that the parameters of the model can be fitted well using your program

Parameter identifyability

The goal of the problem is to estimate following parameters of the Markov model: r_{10}, r_{20}, r_{31}, r_{32}.

The matrix G^T is following:

$$G^T = \begin{bmatrix} -r_{03} & r_{10} & r_{20} & 0 \\ 0 & -r_{10} & 0 & r_{31} \\ 0 & 0 & -r_{20} & r_{32} \\ r_{03} & 0 & 0 & -r_{31} - r_{32} \end{bmatrix}$$

After multiplying $G^T v = 0$,

where

$$v^T = \begin{bmatrix} v_0 & v_1 & v_2 & v_3 \end{bmatrix}$$

We would receive the system of linear equasions:

$$\begin{cases} -r_{03}v_0 + r_{10}v_1 + r_{20}v_2 = 0 \\ -r_{10}v_1 + r_{31}v_3 = 0 \\ -r_{20}v_2 + r_{32}v_3 = 0 \\ r_{03}v_0 - (r_{31} + r_{32})v_3 = 0 \end{cases}$$

After simplification:

$$\begin{cases} -r_{03}v_0 + r_{10}v_1 + r_{20}v_2 = 0 \\ v_1 = \frac{r_{31}}{r_{10}}v_3 \\ v_2 = \frac{r_{32}}{r_{20}}v_3 \\ v_0 = \frac{r_{31}+r_{32}}{r_{03}}v_3 \end{cases}$$

We can see that we have only 3 equations because after adding v_0, v_1, v_2 into the system, we will receive $0 = 0$. So, different sets of parameters r might determine the same eigenvector v.

Therefore, the task to find parameters for this Markov model has no unique solution. It means we have parameter non-identifyability in case we leave it as it is.

But according to the task description, we have an additional constriction on one of the parameters r_{32}, which limits the range of possible ways to determine the set of parameters.

Parameters could be fitted well when we stick to the usage of bounds.

This could be illustrated on two experiments.

We randomly initialize θ_0, and look at the parameter ranges of the obtained solutions with and without bounds. As we can see, solutions **with bounds** have **smaller range**, and θ^* are localized in relatively small area.

NB! The `RuntimeWarning` could be explained as the solver might return small or negative thetas during optimization steps which affects the eigenvector values. Ideally, it is good to bound other parameters above 0 at least (or even $> 10^{-5}$ for example) in case we want to run the optimization from different starting points.

Experiment 1

with bounds

```
/var/folders/6d/q9k5j9917lzfbk22pyg8733h0000gn/T/ipykernel_24146/1275534598.py:27: RuntimeWarning: invalid value encountered in
log
  KL += freq[k] * np.log(freq[k]/e_vector[k])
Parameter ranges, min-max:
r10: 9.310808448256866 - 9.463039233894376
r20: 0.6299284827311052 - 0.65826484780961
r31: 20.483033210811218 - 20.818934041834574
r32: 7.181264399455995 - 7.5
```

no bounds

```
/var/folders/6d/q9k5j9917lzfbk22pyg8733h0000gn/T/ipykernel_24146/1275534598.py:27: RuntimeWarning: invalid value encountered in
log
  KL += freq[k] * np.log(freq[k]/e_vector[k])
Parameter ranges, min-max:
r10: 3.3393552075349904 - 9.447194978926865
r20: 0.6329878099652996 - 1.8117334560695992
r31: 7.34701845634916 - 20.783490705907028
r32: 7.215913224266904 - 20.654127634104192
```

Discuss if convergence to an optimal parameter set is feasible from anywhere in parameter space

The previous example also shows that without constrains on the r_{32} we would not achive the convergence to the same optimum. With it the convergence is feasible from any starting points, in general. On the other hand, the answer is not entirely straightforward because solver might brings us to parameters which are meaningless for the optimization function, and raise the error. In this case we will not obtain the solution.

As we mentioned, some problems during the optimization might happen because the solver does not know which set of parameters causes the issues with the eigenvector and then with the optimization function. Therefore, we need to limit our parameters additionally to avoid negative/small thetas.

The following example illustrate this behaviour, when we initialize θ_0 with the values different from the values in the task, the solver finds intermediate θ with the negative values which leads to $\inf$ in the optimization function.

But limiting parameters halps.

Experiment 2

no additional bounds

```
/var/folders/6d/q9k5j9917lzfbk22pyg8733h0000gn/T/ipykernel_24146/1275534598.py:27: RuntimeWarning: invalid value encountered in
log
  KL += freq[k] * np.log(freq[k]/e_vector[k])
```

```
  message: ABNORMAL_TERMINATION_IN_LNSRCH
  success: False
   status: 2
      fun: nan
        x: [ 3.000e-03  1.000e-03  1.000e-03  7.000e+00]
      nit: 0
      jac: [ 3.629e+01 -4.353e+02 -1.089e+02  2.261e-02]
     nfev: 35
     njev: 7
 hess_inv: <4x4 LbfgsInvHessProduct with dtype=float64>
```

```
/var/folders/6d/q9k5j9917lzfbk22pyg8733h0000gn/T/ipykernel_24146/1275534598.py:27: RuntimeWarning: invalid value encountered in
log
  KL += freq[k] * np.log(freq[k]/e_vector[k])
```

```
theta: [3.e-03 1.e-03 1.e-03 7.e+00], KL: 2.814977612522357
theta: [-0.40007731  4.83569061  1.21066743  6.99974886], KL: nan
theta: [-6.84931434 82.19074041 20.56534635  6.9957306 ], KL: nan
theta: [-36.28470315 435.25203379 108.90331512   6.97739062], KL: nan
```

with additional bounds

```
  message: CONVERGENCE: NORM_OF_PROJECTED_GRADIENT_<=_PGTOL
  success: True
   status: 0
      fun: 1.0451156897528395e-10
        x: [ 9.318e+00  6.579e-01  2.050e+01  7.500e+00]
      nit: 33
      jac: [ 3.648e-07 -7.146e-06 -2.605e-07  5.204e-07]
     nfev: 220
     njev: 44
 hess_inv: <4x4 LbfgsInvHessProduct with dtype=float64>
```

Additionally: Conditioning

Data perturbations

Experiment 3

Technically, the stationary distribution based on the parameters from our model aligns with the statistics from frequencies obtained from the data. Hovewer, there is a question on if the number of entries is enough for states 1 and 3 in the data (which are equal to 11 and 5 respectively) to use them for the model, how would perturbations of the data affect the solution.

```
State 0 - 28 entries
State 1 - 11 entries
State 2 - 57 entries
State 3 - 5 entries
```

```
  0%|                                                                                                | 0/1000 [00:00<?, ?it/s]/var/folde
rs/6d/q9k5j9917lzfbk22pyg8733h0000gn/T/ipykernel_24146/1275534598.py:27: RuntimeWarning: divide by zero encountered in log
  KL += freq[k] * np.log(freq[k]/e_vector[k])
/var/folders/6d/q9k5j9917lzfbk22pyg8733h0000gn/T/ipykernel_24146/1275534598.py:27: RuntimeWarning: invalid value encountered in
scalar multiply
  KL += freq[k] * np.log(freq[k]/e_vector[k])
/var/folders/6d/q9k5j9917lzfbk22pyg8733h0000gn/T/ipykernel_24146/1275534598.py:27: RuntimeWarning: invalid value encountered in
log
  KL += freq[k] * np.log(freq[k]/e_vector[k])
100%|████████████████████████████████████████████████████████████████████████████████| 1000/1000 [00:06<00:00, 154.17it/s]
```

```
Successful fits: 995/1000
Parameter uncertainty:
r10: 10.306198031738386 +- 6.0144880492115504
r20: 0.6652731566304667 +- 0.304737214647837
r31: 27.871464430665487 +- 23.68003045474262
r32: 7.41853359687759 +- 0.22343750924509947

Noisy counts:
cnt_0: 27.546 +- 4.448807031103956
cnt_1: 10.936 +- 3.1394114098027996
cnt_2: 56.578 +- 5.076998719716206
cnt_3: 4.94 +- 2.0881570822138835

Noisy frequencies:
f0: 0.27546000000000004 +- 0.04448807031103957
f1: 0.10936000000000001 +- 0.031394114098027996
f2: 0.5657800000000001 +- 0.05076998719716206
f3: 0.049400000000000006 +- 0.020881570822138838
```

Looking at the results we can see that only the bounded parameter r_{32} does not vary significantly, meaning that the problem is ill-conditioned as a slight change in data counts of state entries might cause significant changes in the solution.

Solution: - include more state entries to expand the statistics, bound the rest of the parameters.