

# sP mini-project for exam

Sebastian Truong - study number: 20206054

## CMakeLists.txt

```
cmake_minimum_required(VERSION 3.0)
project(ExamProject)

set(CMAKE_CXX_STANDARD 20)

add_executable(ExamProject main.cpp)
add_executable(ExamProjectTest unitTests.cpp)
add_executable(ExamProjectBenchmark main_bm.cpp)
```

## Pretty-printing

for figure 1b

human format:

```
---Pretty printing for Figure 1b---
Reactions:
  Reaction: A + C -->(0.001000) B + C
--- Middle of pretty printing for Figure 1b---
```

graph format:

```
--- Middle of pretty printing for Figure 1b---
0.001000 -> B C
A -> 0.001000
B ->
C -> 0.001000
---End of pretty printing for Figure 1b---
```

for Circadian Rhythm

human format:

```
---Pretty printing for Circadian Rhythm---
Reactions:
  Reaction: A + DA -->(1.000000) D_A
  Reaction: D_A -->(50.000000) DA + A
  Reaction: A + DR -->(1.000000) D_R
  Reaction: D_R -->(100.000000) DR + A
  Reaction: D_A -->(500.000000) MA + D_A
  Reaction: DA -->(50.000000) MA + DA
  Reaction: D_R -->(50.000000) MR + D_R
  Reaction: DR -->(0.010000) MR + DR
  Reaction: MA -->(50.000000) MA + A
  Reaction: MR -->(5.000000) MR + R
  Reaction: A + R -->(2.000000) C
  Reaction: C -->(1.000000) R
  Reaction: A -->(1.000000)
  Reaction: R -->(0.200000)
  Reaction: MA -->(10.000000)
  Reaction: MR -->(0.500000)
--- Middle of pretty printing for Circadian Rhythm---
```

graph format:

```
--- Middle of pretty printing for Circadian Rhythm---
0.010000 -> MR DR
0.200000 ->
0.500000 ->
1.000000 -> D_A
1.000000_ -> D_R
1.000000__ -> R
1.000000___ ->
10.000000 ->
100.000000 -> DR A
2.000000 -> C
5.000000 -> MR R
50.000000 -> DA A
50.000000_ -> MA DA
50.000000__ -> MR D_R
50.000000___ -> MA A
500.000000 -> MA D_A
A -> 1.000000 1.000000_ 2.000000 1.000000___
C -> 1.000000__
DA -> 1.000000 50.000000_
DR -> 1.000000_ 0.010000
D_A -> 50.000000 500.000000
D_R -> 100.000000 50.000000__
MA -> 50.000000___ 10.000000
MR -> 5.000000 0.500000
R -> 2.000000 0.200000
---End of pretty printing for Circadian Rhythm---
```

## for seihr

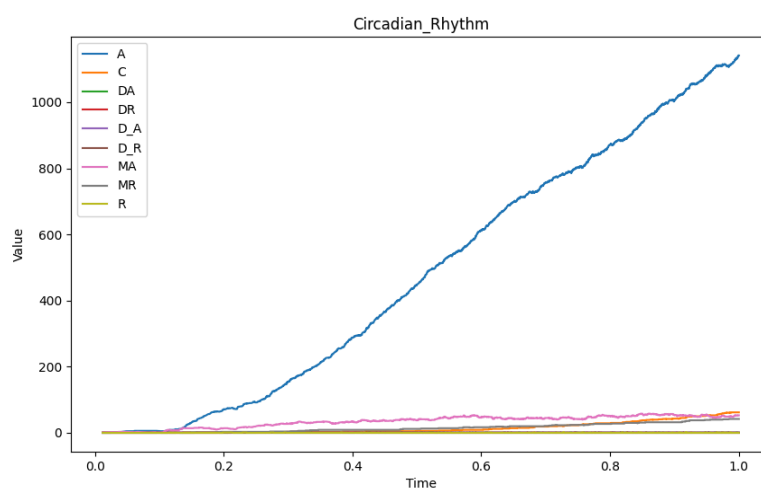
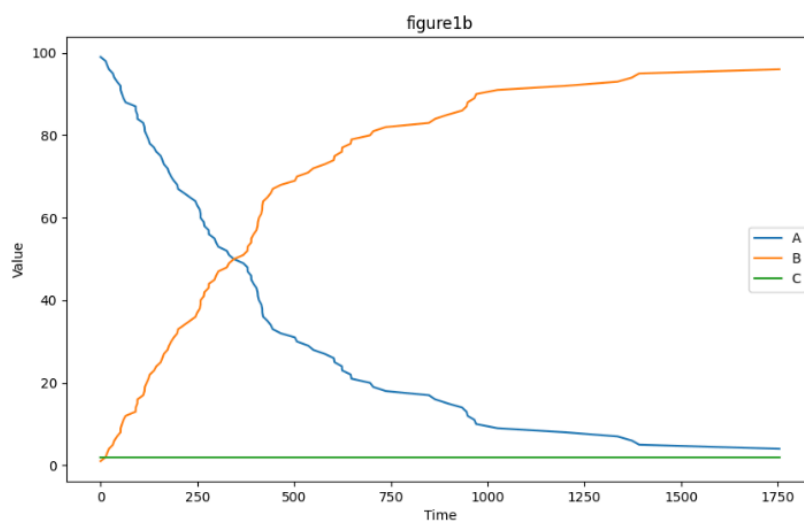
human format:

```
---Pretty printing for COVID19 SEIHR: 10000---
Reactions:
  Reaction: S + I -->(0.000077) E + I
  Reaction: E -->(0.196078) I
  Reaction: I -->(0.322581) R
  Reaction: I -->(0.000290) H
  Reaction: H -->(0.098814) R
--- Middle of pretty printing for COVID19 SEIHR: 10000---
```

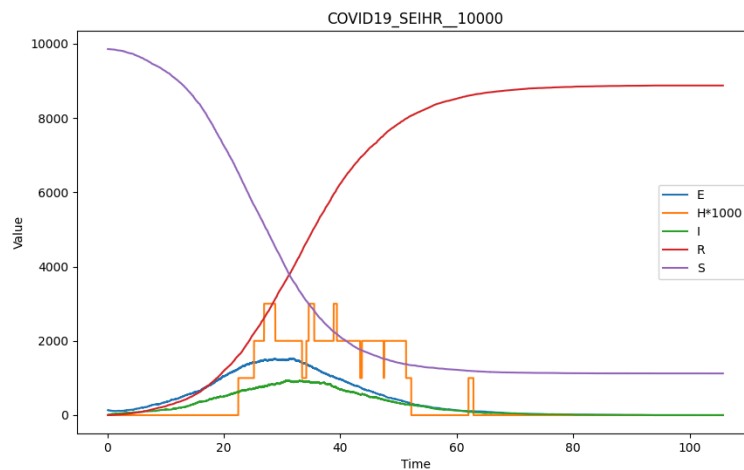
graph format:

```
--- Middle of pretty printing for COVID19 SEIHR: 10000---
0.000077 -> E I
0.000290 -> H
0.098814 -> R
0.196078 -> I
0.322581 -> R
E -> 0.196078
H -> 0.098814
I -> 0.000077 0.322581 0.000290
R ->
S -> 0.000077
---End of pretty printing for COVID19 SEIHR: 10000---
```

## Trajectories (externals: python + matplotlib + pandas)



(cutoff at time=1.0 due to performance time)



Observer peak hospitalizations with multi core  
(time=5, simulations=100)

for seihr  $N_{NJ}=589'755$

```
---End of simulation for COVID19 SEIHR: 589755 iteration 95---
Peak hospitalization: 3
---End of simulation for COVID19 SEIHR: 589755 iteration 96---
Peak hospitalization: 6
---End of simulation for COVID19 SEIHR: 589755 iteration 97---
Peak hospitalization: 2
---End of simulation for COVID19 SEIHR: 589755 iteration 98---
Peak hospitalization: 0
---End of simulation for COVID19 SEIHR: 589755 iteration 99---
Peak hospitalization: 2
Average peak hospitalization over 100 simulations: 3.66
```

for seihr  $N_{DK}=5'822'763$

```
---End of simulation for COVID19 SEIHR: 5822763 iteration 95---
Peak hospitalization: 27
---End of simulation for COVID19 SEIHR: 5822763 iteration 94---
Peak hospitalization: 44
---End of simulation for COVID19 SEIHR: 5822763 iteration 97---
Peak hospitalization: 27
---End of simulation for COVID19 SEIHR: 5822763 iteration 96---
Peak hospitalization: 33
---End of simulation for COVID19 SEIHR: 5822763 iteration 99---
Peak hospitalization: 36
---End of simulation for COVID19 SEIHR: 5822763 iteration 98---
Peak hospitalization: 33
Average peak hospitalization over 100 simulations: 31.12
```

Benchmarking for single and multi core  
 with figure1b, circadian rhythm, seihr n=10'000  
 (time=5, simulations=100)

```
Starting chrono for Single Core
Single Core took 6193 milliseconds to run.
Starting chrono for Multi Core
Multi Core took 97275 milliseconds to run.
```









(in main\_bm.cpp)

## Conclusions

Overall, I believe that my implementation is decent. It though has several issues / possible improvements:

- issue: concurrency does not work optimally, and I suspect it is due to my laptops limited computational power. Since my computer already uses a lot of memory just having CLion open, see Appendix A.1, I believe the spreading of resources into different threads just makes it slower in total.
  - It could also be problems with the code that I have missed.
- issue: import with .cpp multiple instances, should be .h but could not figure out why it did not work for me
- possible improvement: no visual graph representation but printed instead
- possible improvement: performance optimizations possible with symbolTable

## Appendix A.1 (CLion opened without any task running)

Name	Status	9% CPU	86% Memory	2% Disk	0% Network
Apps (6)					
>  CLion		0%	1.886,1 MB	0 MB/s	0 Mbps
>  Discord (6)		0,6%	47,6 MB	0 MB/s	0 Mbps
>  Google Chrome (15)		0,6%	495,8 MB	0,2 MB/s	0,3 Mbps
>  Spotify (8)		0%	163,8 MB	0,1 MB/s	0,1 Mbps
>  Task Manager		6,0%	51,2 MB	0,6 MB/s	0 Mbps
>  Visual Studio Code (15)		0%	77,8 MB	0 MB/s	0 Mbps