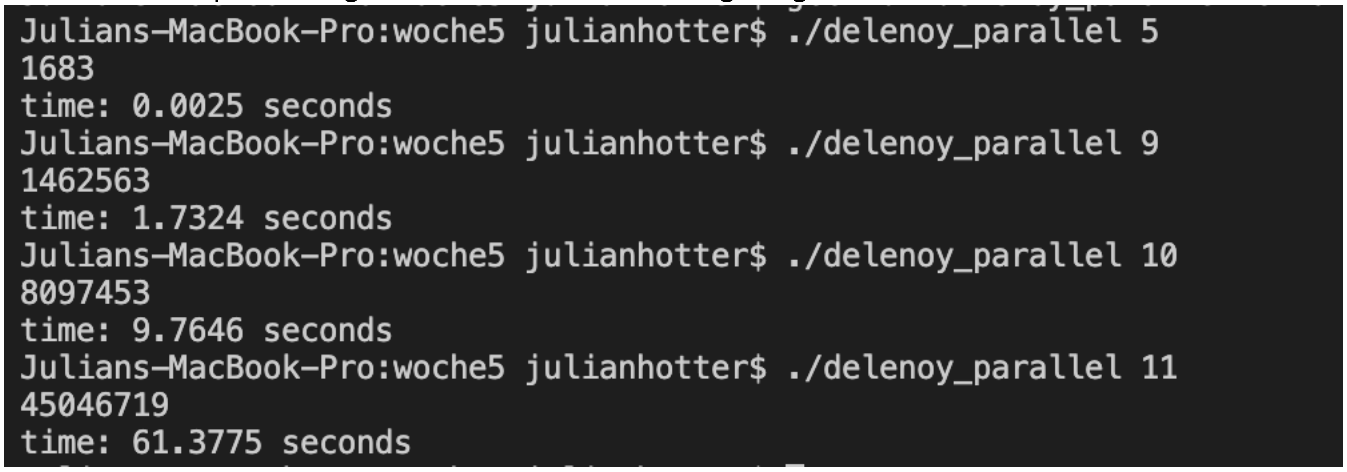
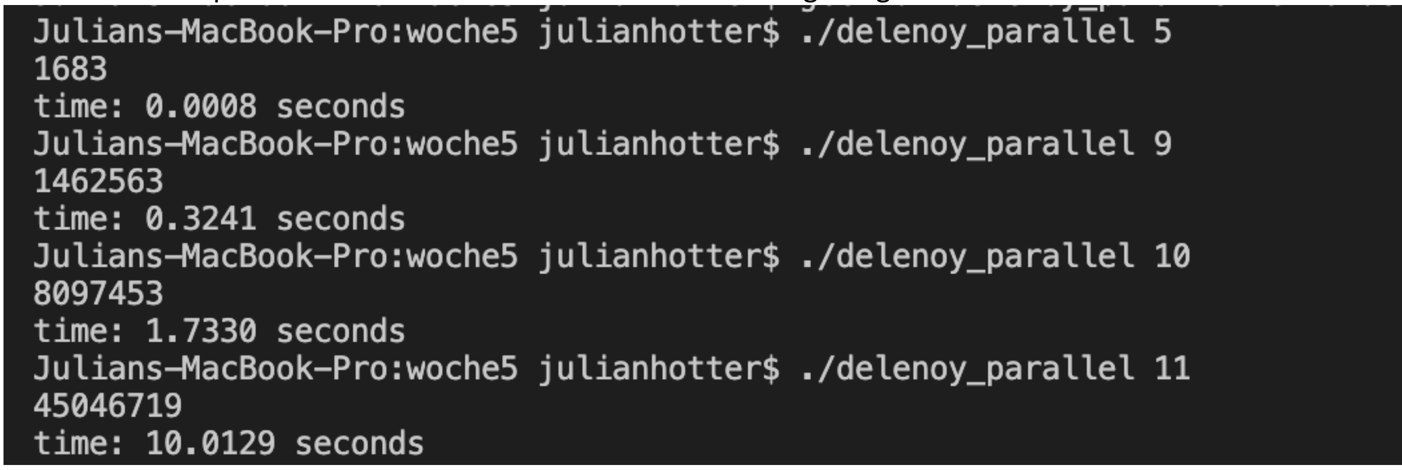
Parallel:

Values with the parallel region in the main and with single region



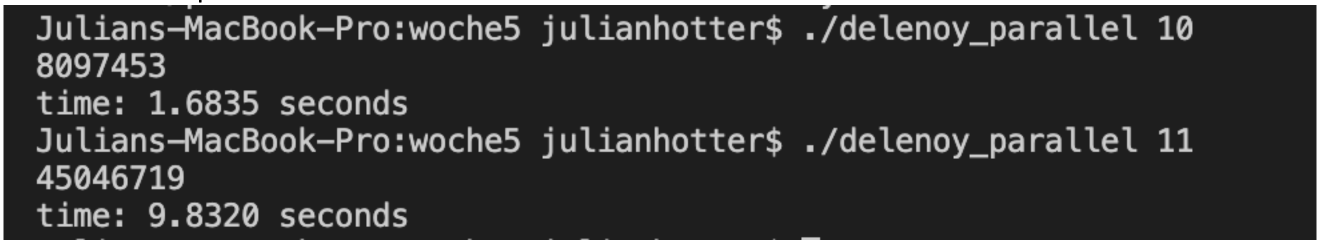
```
Julians-MacBook-Pro:woche5 julianhotter$ ./delenoy_parallel 5
1683
time: 0.0025 seconds
Julians-MacBook-Pro:woche5 julianhotter$ ./delenoy_parallel 9
1462563
time: 1.7324 seconds
Julians-MacBook-Pro:woche5 julianhotter$ ./delenoy_parallel 10
8097453
time: 9.7646 seconds
Julians-MacBook-Pro:woche5 julianhotter$ ./delenoy_parallel 11
45046719
time: 61.3775 seconds
```

Values with the parallel function in the function itself with single region



```
Julians-MacBook-Pro:woche5 julianhotter$ ./delenoy_parallel 5
1683
time: 0.0008 seconds
Julians-MacBook-Pro:woche5 julianhotter$ ./delenoy_parallel 9
1462563
time: 0.3241 seconds
Julians-MacBook-Pro:woche5 julianhotter$ ./delenoy_parallel 10
8097453
time: 1.7330 seconds
Julians-MacBook-Pro:woche5 julianhotter$ ./delenoy_parallel 11
45046719
time: 10.0129 seconds
```

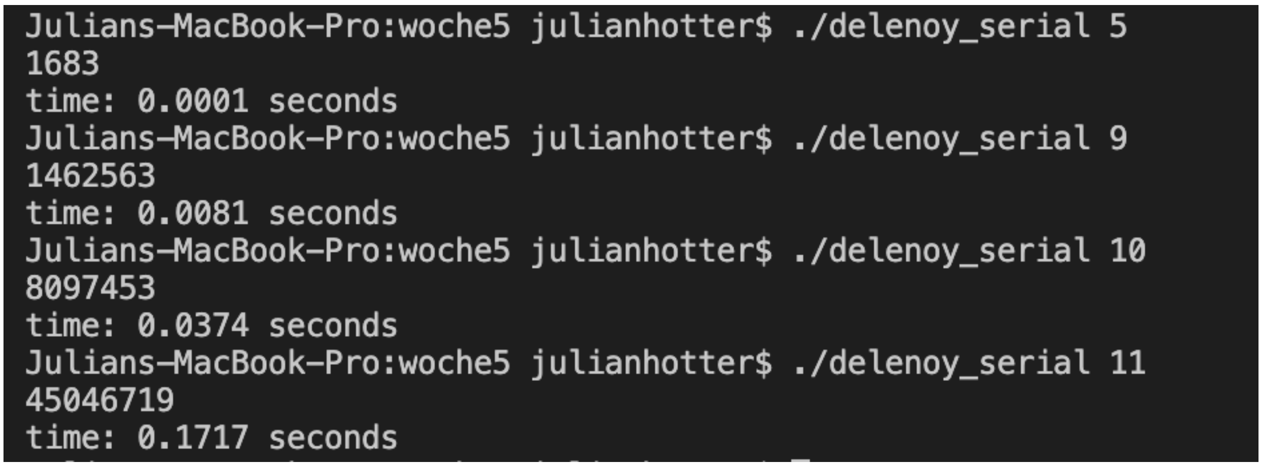
Then I tried to sum the variables up before returning but these also got us just a few miliseconds improvement.



```
Julians-MacBook-Pro:woche5 julianhotter$ ./delenoy_parallel 10
8097453
time: 1.6835 seconds
Julians-MacBook-Pro:woche5 julianhotter$ ./delenoy_parallel 11
45046719
time: 9.8320 seconds
```

It is better but not even as good as the serial version.

Serial:

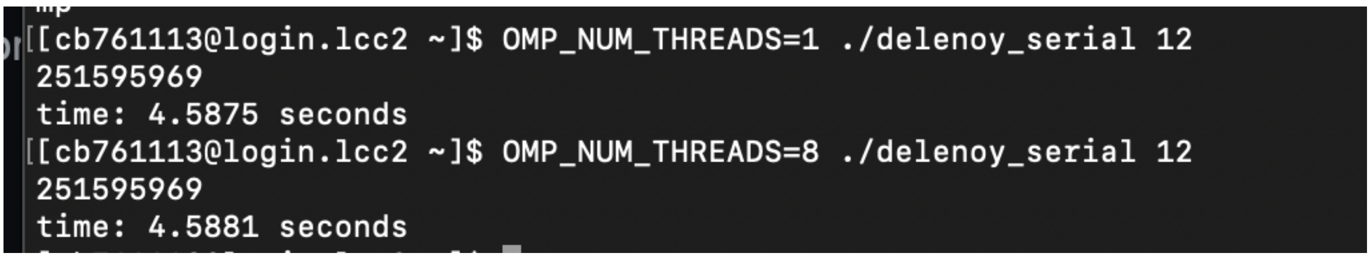


```
Julians-MacBook-Pro:woche5 julianhotter$ ./delenoy_serial 5
1683
time: 0.0001 seconds
Julians-MacBook-Pro:woche5 julianhotter$ ./delenoy_serial 9
1462563
time: 0.0081 seconds
Julians-MacBook-Pro:woche5 julianhotter$ ./delenoy_serial 10
8097453
time: 0.0374 seconds
Julians-MacBook-Pro:woche5 julianhotter$ ./delenoy_serial 11
45046719
time: 0.1717 seconds
```

On my local computer the serial version is way faster than the tasked version, I have trief many different approaches , but it never gets faster than the sequential version.

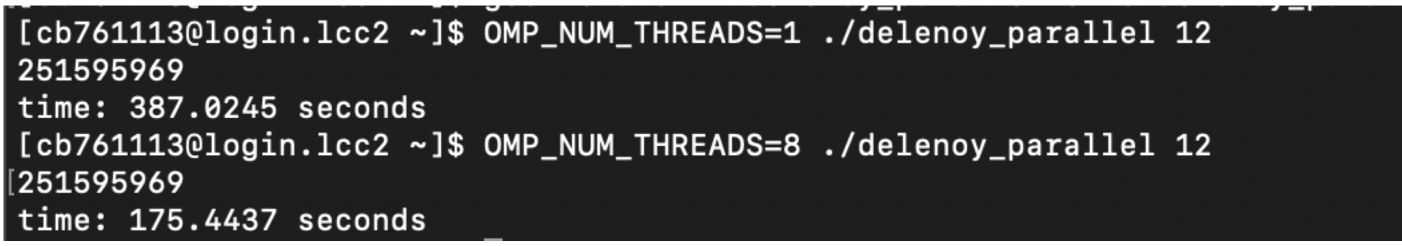
Now on the LCC2.

SERIAL :
N = 12
With 1 and 8 threads



```
[[cb761113@login.lcc2 ~]$ OMP_NUM_THREADS=1 ./delenoy_serial 12
251595969
time: 4.5875 seconds
[[cb761113@login.lcc2 ~]$ OMP_NUM_THREADS=8 ./delenoy_serial 12
251595969
time: 4.5881 seconds
```

Parallel
N = 12
With 1 and 8 threads



```
[cb761113@login.lcc2 ~]$ OMP_NUM_THREADS=1 ./delenoy_parallel 12
251595969
time: 387.0245 seconds
[cb761113@login.lcc2 ~]$ OMP_NUM_THREADS=8 ./delenoy_parallel 12
[251595969
time: 175.4437 seconds
```

So again, de serial version is way better. But with 8 Threads there is a big improvement on the parallel version.

Main Bottleneck:
The main bottleneck could be that the threads have to share the variables and have to wait (taskwait) , when one task is slow the others also get slow. Therefore, you don`t have that good times as you expect. We tried very much different OMP variation (see above) but nothing did the big thing for better times.

Improve?:
We tried much to improve but I think that would not be possible unless it changes the underlying algorithm.
Maybe don`t make the taskwait , but then you have synchronization problems und false results get here sometimes.

Maybe tasks are not the best idea for that kind of task ?