CS3205 Networks Assignment 2 Report

Author Name: **Anumala Venu Madhava Reddy**

Roll no: **CS18B051**

Department of Computer Science and Engineering

March 29, 2021

1 Aim/objective

- The Objective of this project is to emulate the modified AIMD TCP congestion control algorithm.

2 Introduction

- Congestion is a state occuring in network layer when the message traffic is so heavy that it slows down the network response time.
- And when the congestion occurs and is left unmanaged , it increases delay which increases retransmission , which in turn increases congestion.
- AIMD is a congestion control algorithm and it has 3 phases.
- Slow start phase : Here increment of congestion window is exponential , till we reach a threshold. because each time an ack is received we increase congestion window by 1, hence the exponential growth.
- Congestion aviodance phase : this is the phase when we arleady reached threshold , therefore the congestion window is increased by 1 for every RTT.
- Congestion Detection phase : If congestion occurs , sender goes back to one of the above two phases.Congestion is detected through retransmission.
  - Retransmission due to timeout - Congestion possibility is high.(fast retransmission)
    - we make, ssthresh = cwnd/2
    - set cwnd = initial value
    - start with slow phase again
  - Retransmission doe to 3 duplicate ACK's - Congestion possibility is less (fast recovery)
    - cwnd = ssthresh = cwnd/2
    - start with congestion aviodance phase
-

3 Experimental details

- We are going to use a modified AIMD algorithm
- Sender's MSS is 1kB and Each segment size is fixed length of one MSS
- Recievers Window size is 1MB and does not change during the entire duration of emulation
- Congestion window is always assumed to be multiple of MSS
- Sender always has data to be sent
- Congestion threshold is always set half of current congestion window
- Go back N sliding protocol is used but cumulative acknowledgement are not considered, for each segment an individual timer and ACK are used.

3.1 Experimental/Simulation setup

- The initial cw is given by
$$CW_{new} = K_i * MSS$$
- During exponential phase , when a segment's ACK is successfully received
$$CW_{new} = min\left(CW_{old} + K_m * MSS, RWS\right)$$
- During linear growth phase , when a segment's ACK is received
$$CW_{new} = min\left(CW_{old} + K_n * MSS * \frac{MSS}{CW_{old}}, RWS\right)$$
- When a timeout occurs ,
$$CW_{new} = max\left(1, K_f * CW_{old}\right)$$

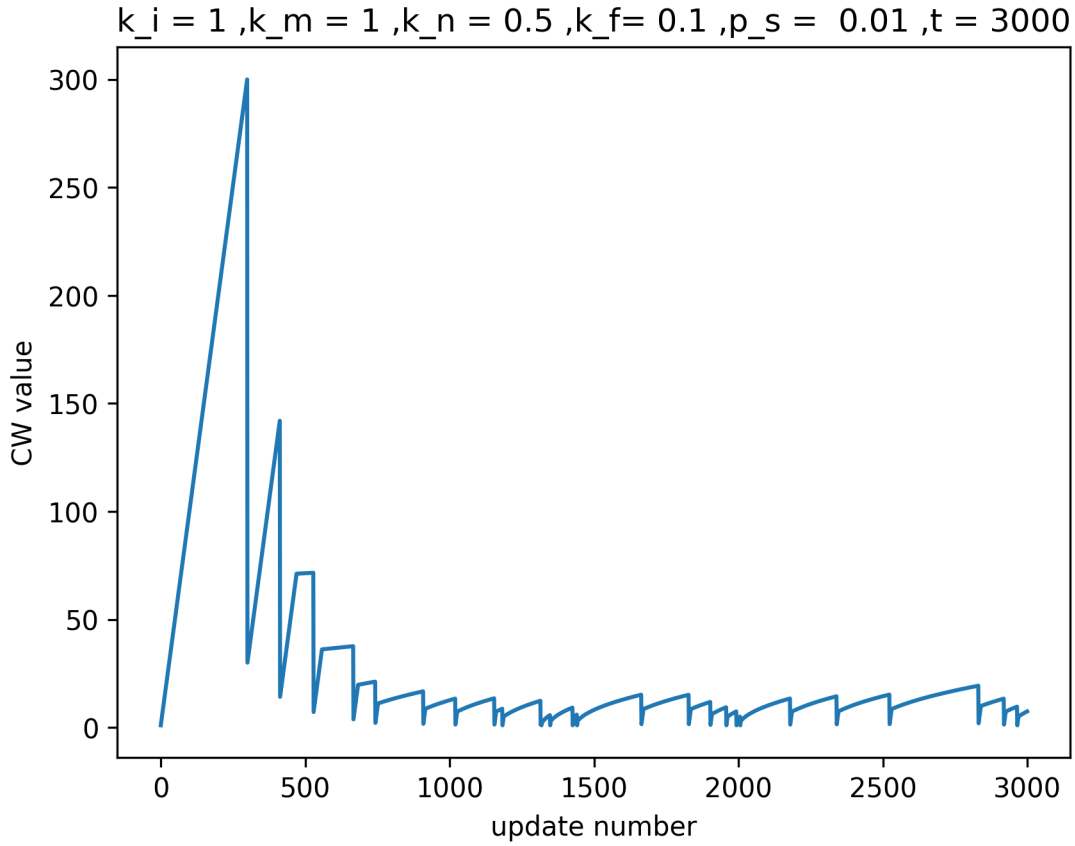
3.2 Entities Involved

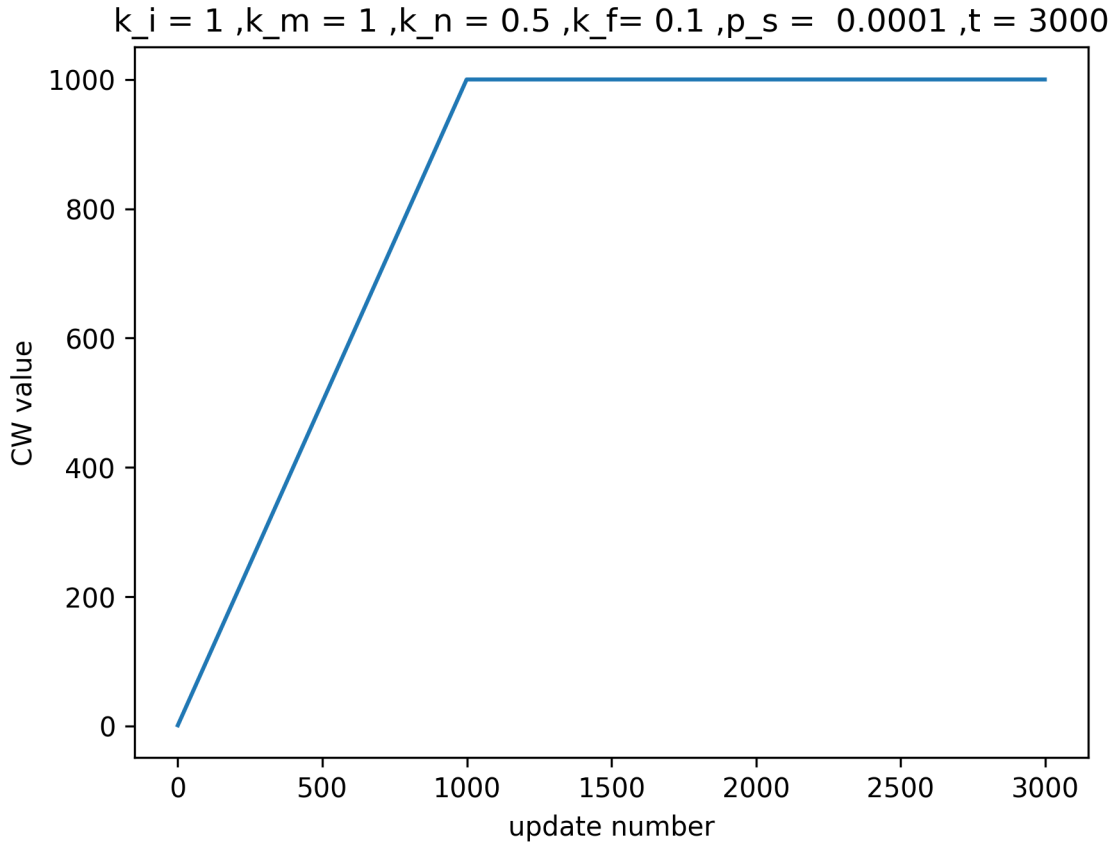
- K_i , $1 \leq K_i \leq 4$ This Denotes the initial congestion window CW , Default value is 1.
- K_m , $0.5 \leq K_m \leq 2$ denotes the multiplier of congestion window , during exponential growth phase
- K_n , $0.5 \leq K_n \leq 2$ denotes the multiplier of congestion window , during linear growth phase
- K_f , $0.1 \leq K_f \leq 0.5$ denotes the multiplier when a timeout occurs
- P_s , $0 < P_s < 1$ denotes the probability of occurence of a timeout
- T it denotes the number of segments to be processed before stopping

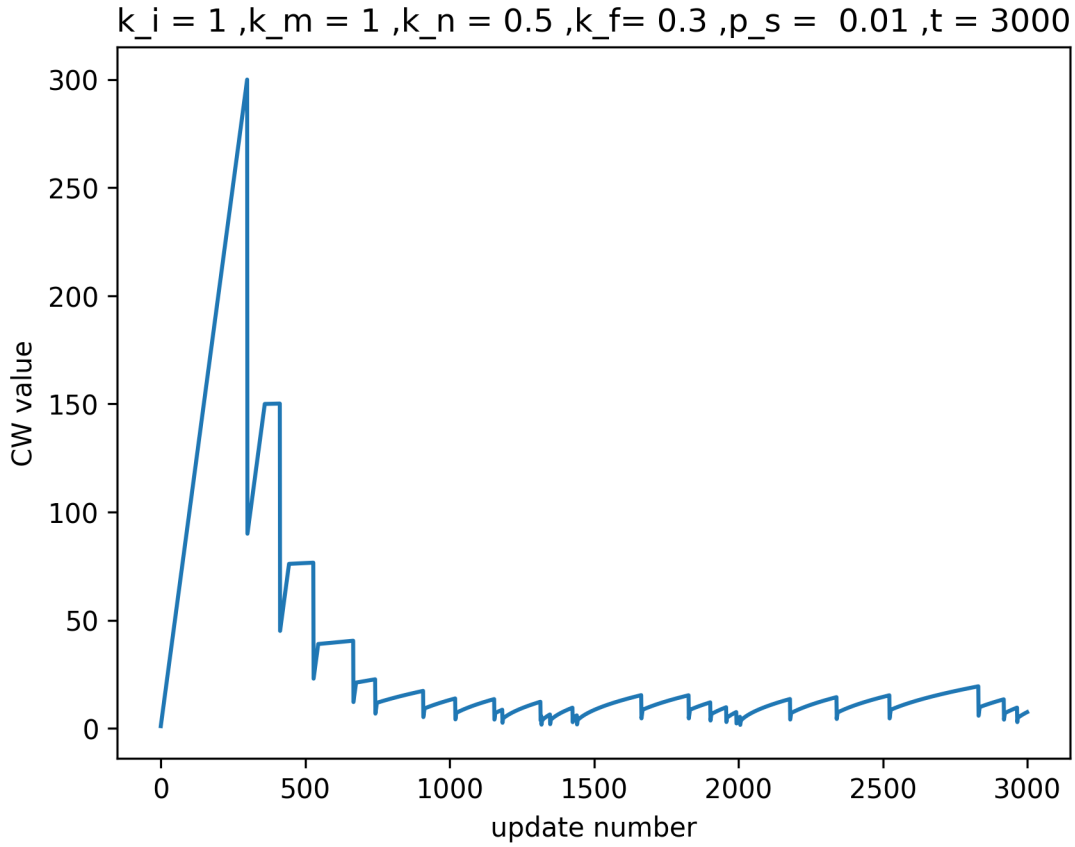
3.3 Additional Details

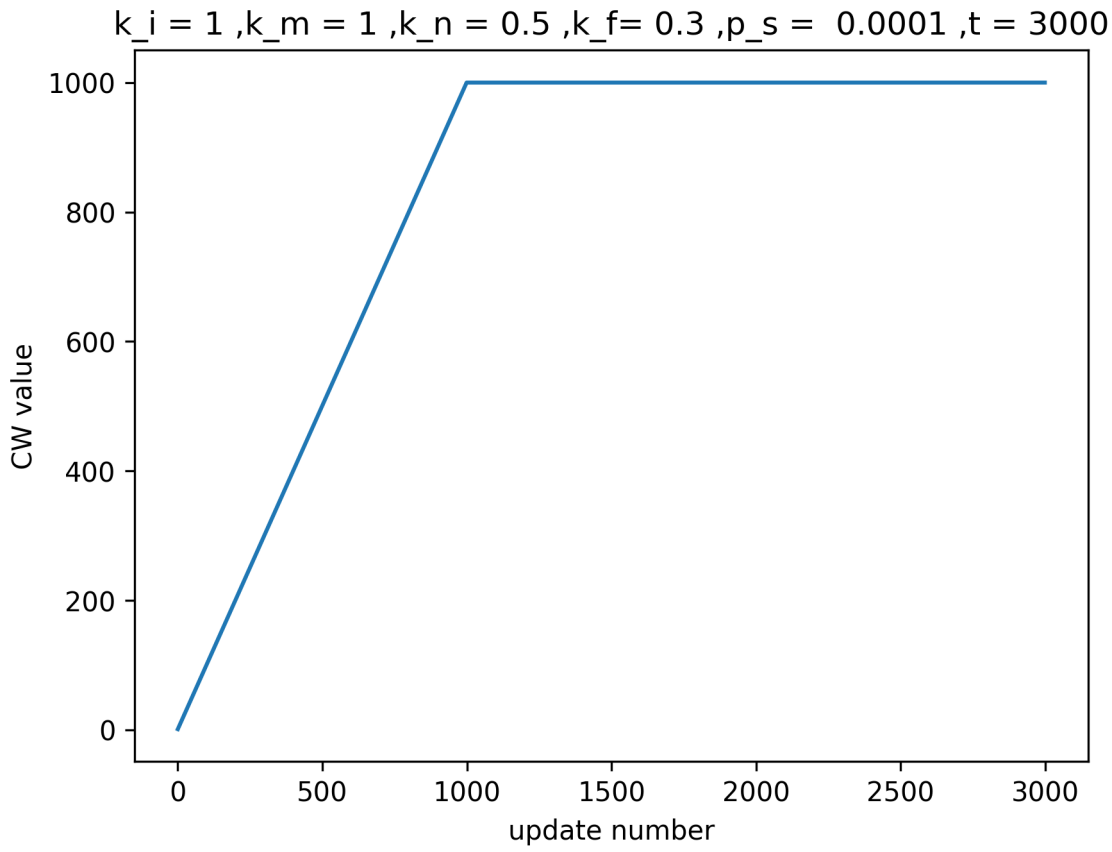
- As sender and reciever are not actually present , we use bernouli distribution generator to create the occurence of a timeout.

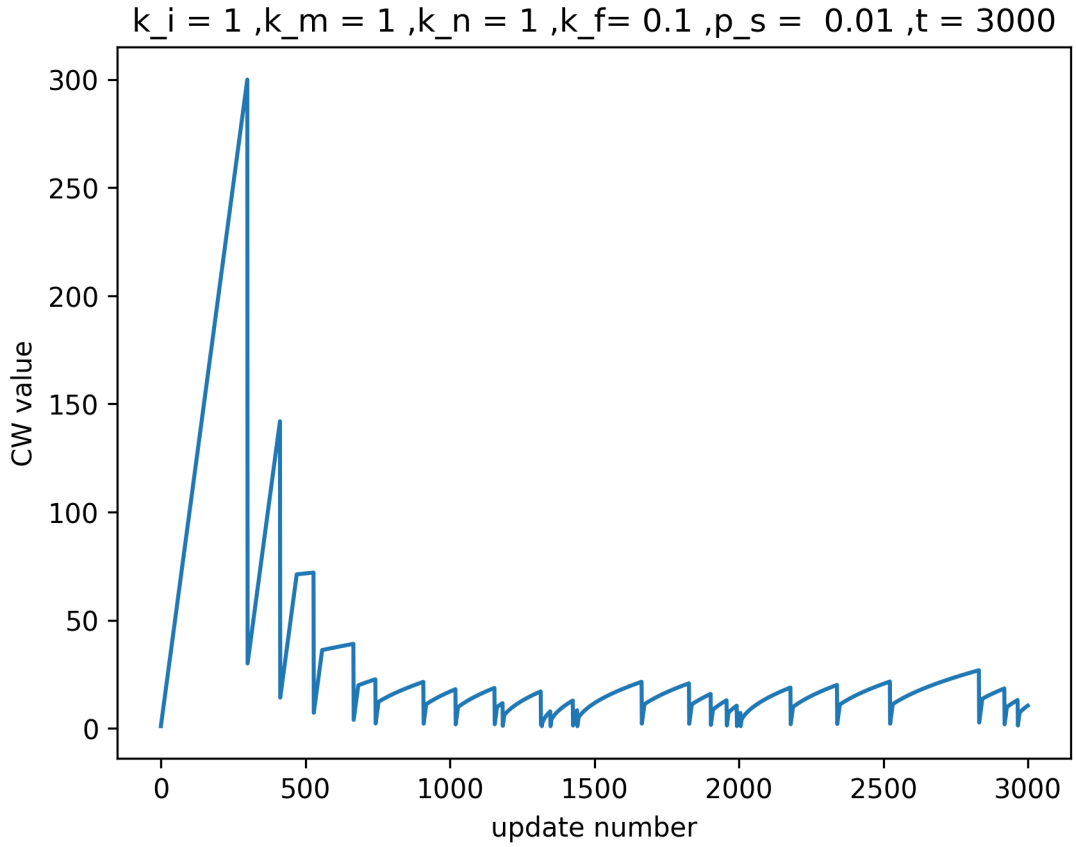
4 Results and Observations

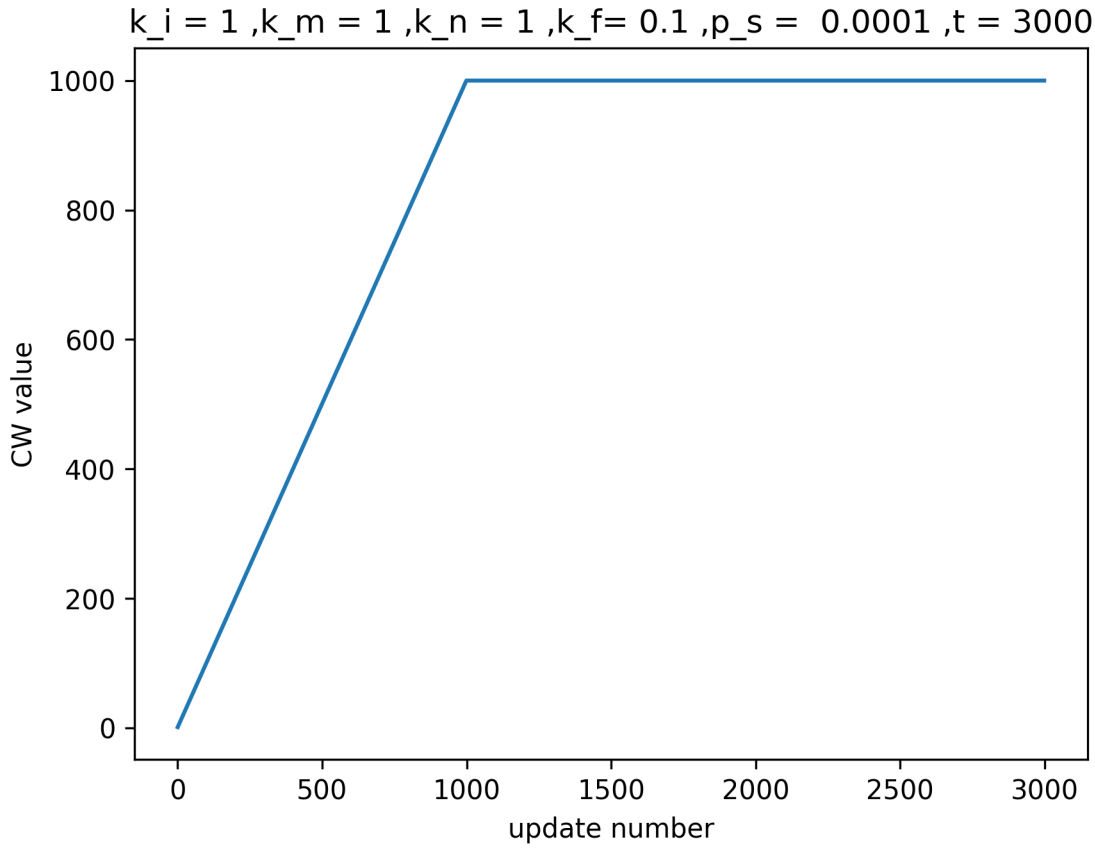


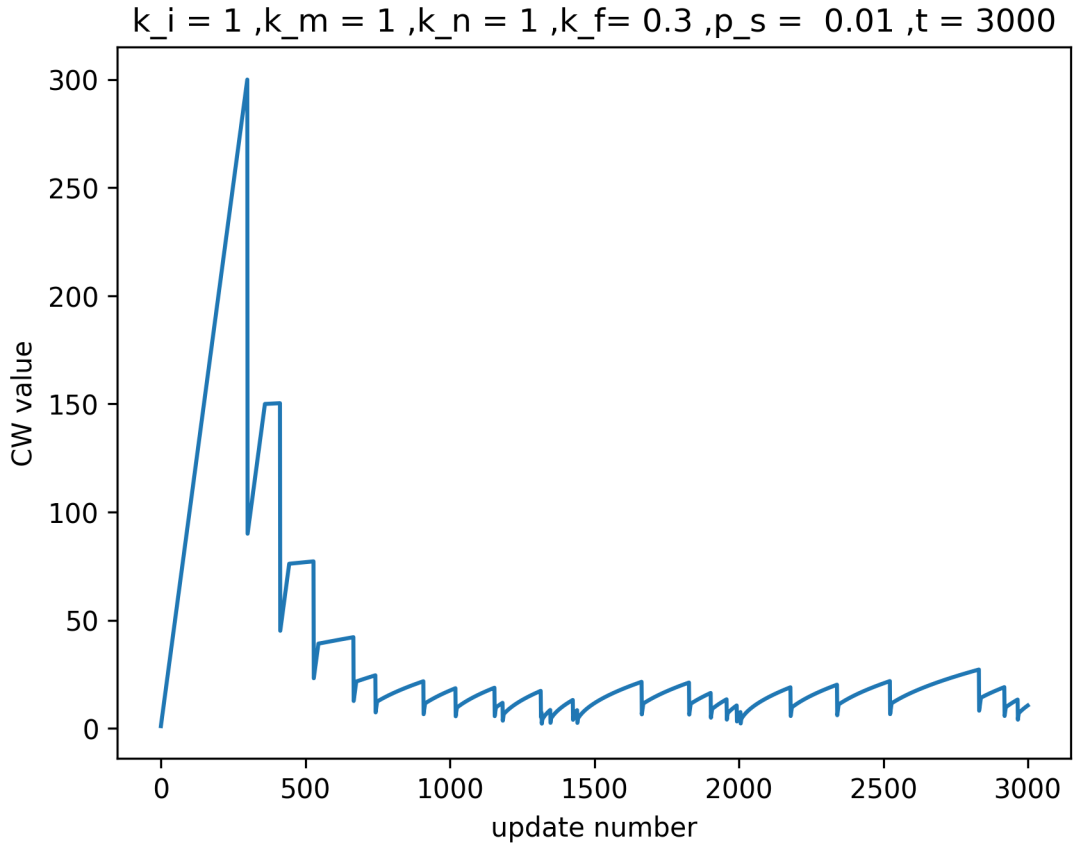


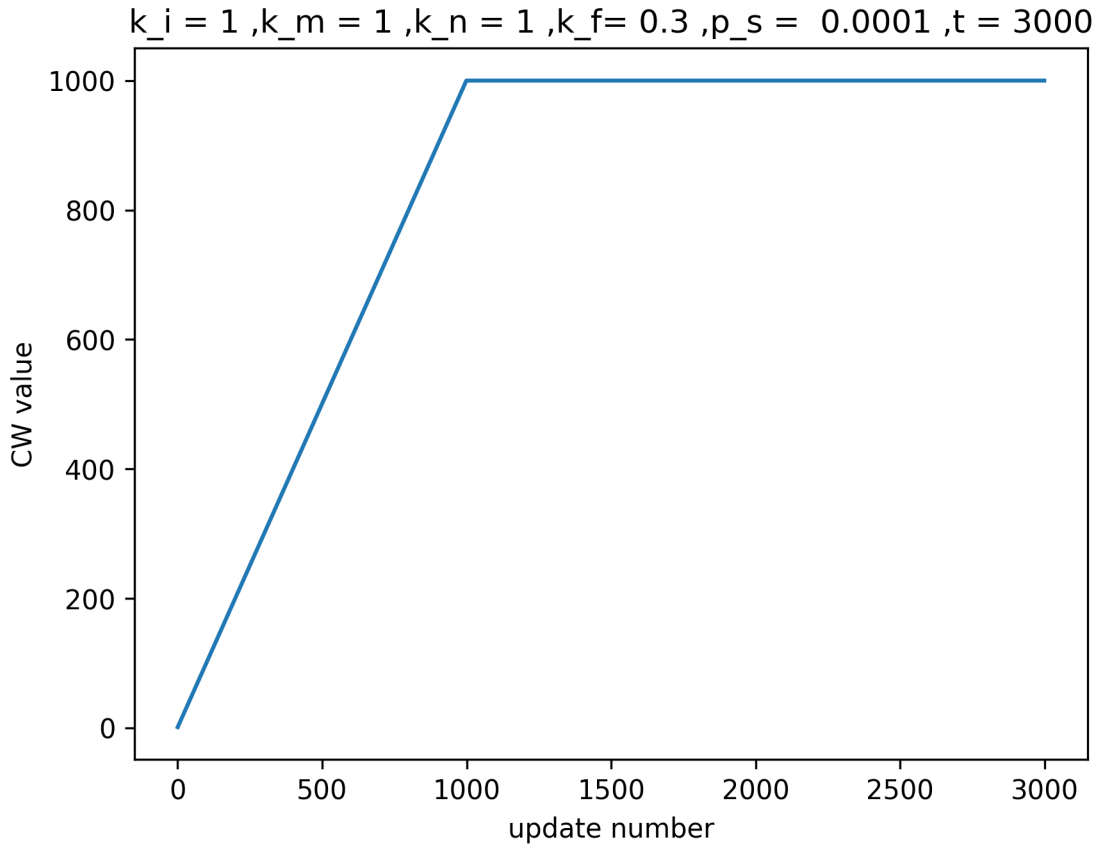

k_i = 1 ,k_m = 1 ,k_n = 0.5 ,k_f= 0.3 ,p_s = 0.01 ,t = 3000
CW value
update number
0
50
100
150
200
250
300
0
500
1000
1500
2000
2500
3000

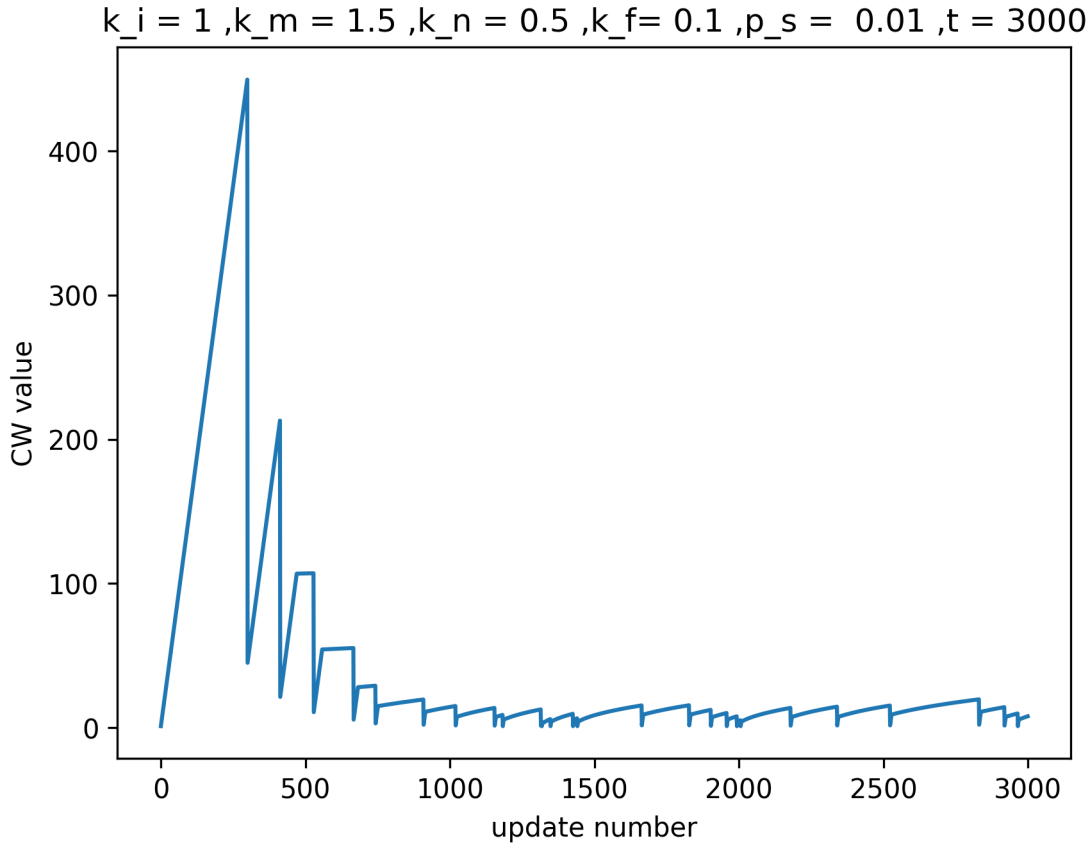

k_i = 1 ,k_m = 1 ,k_n = 0.5 ,k_f= 0.3 ,p_s = 0.0001 ,t = 3000
CW value
update number
0
200
400
600
800
1000
0
500
1000
1500
2000
2500
3000

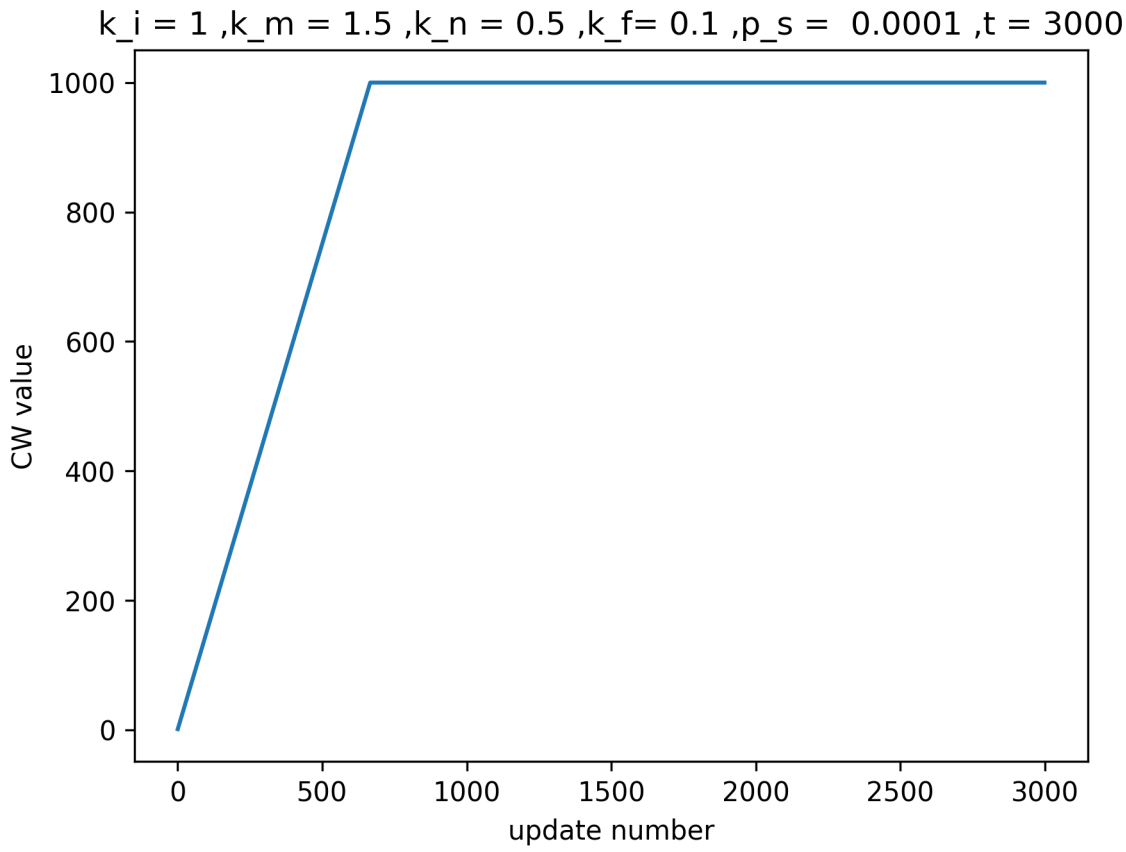

k_i = 1 ,k_m = 1 ,k_n = 1 ,k_f= 0.1 ,p_s = 0.01 ,t = 3000
CW value
update number
0
50
100
150
200
250
300
0
500
1000
1500
2000
2500
3000

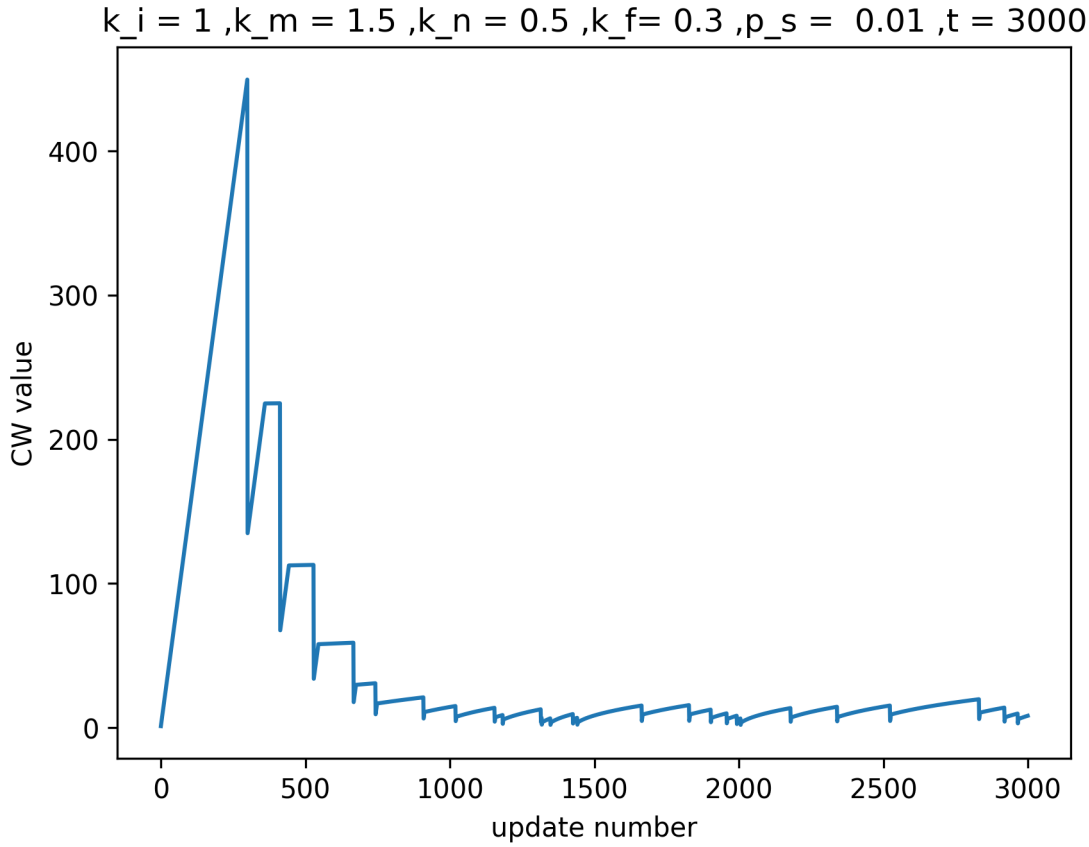

k_i = 1 ,k_m = 1 ,k_n = 1 ,k_f= 0.1 ,p_s = 0.0001 ,t = 3000
CW value
update number
0
200
400
600
800
1000
0
500
1000
1500
2000
2500
3000

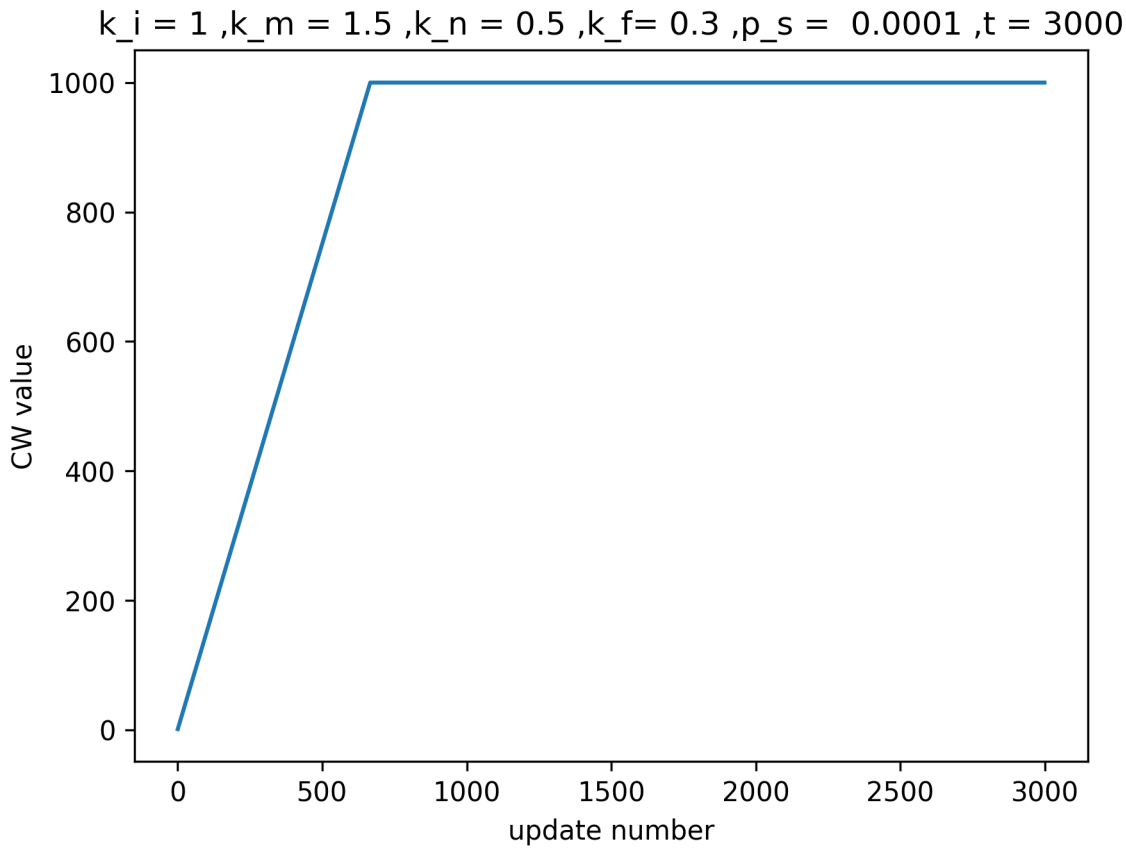

k_i = 1 ,k_m = 1 ,k_n = 1 ,k_f= 0.3 ,p_s = 0.01 ,t = 3000
CW value
update number
0
50
100
150
200
250
300
0
500
1000
1500
2000
2500
3000

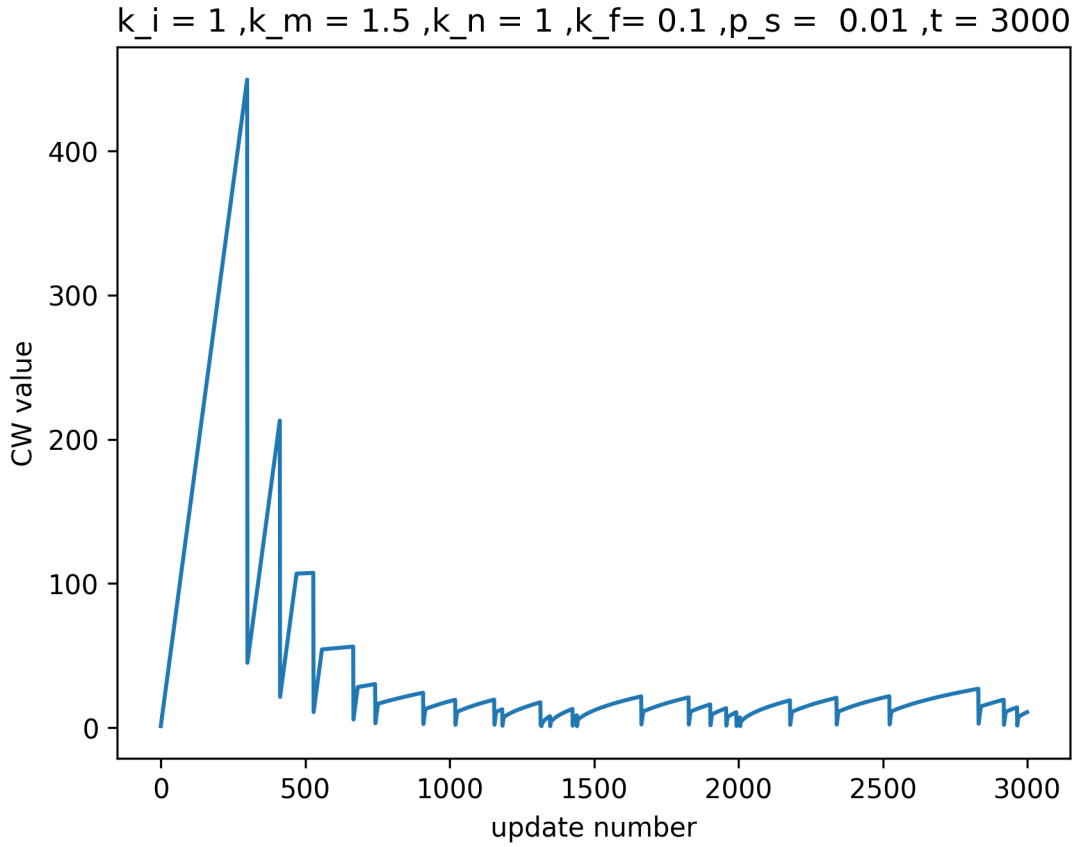

k_i = 1 ,k_m = 1 ,k_n = 1 ,k_f= 0.3 ,p_s = 0.0001 ,t = 3000
CW value
update number
0
200
400
600
800
1000
0
500
1000
1500
2000
2500
3000

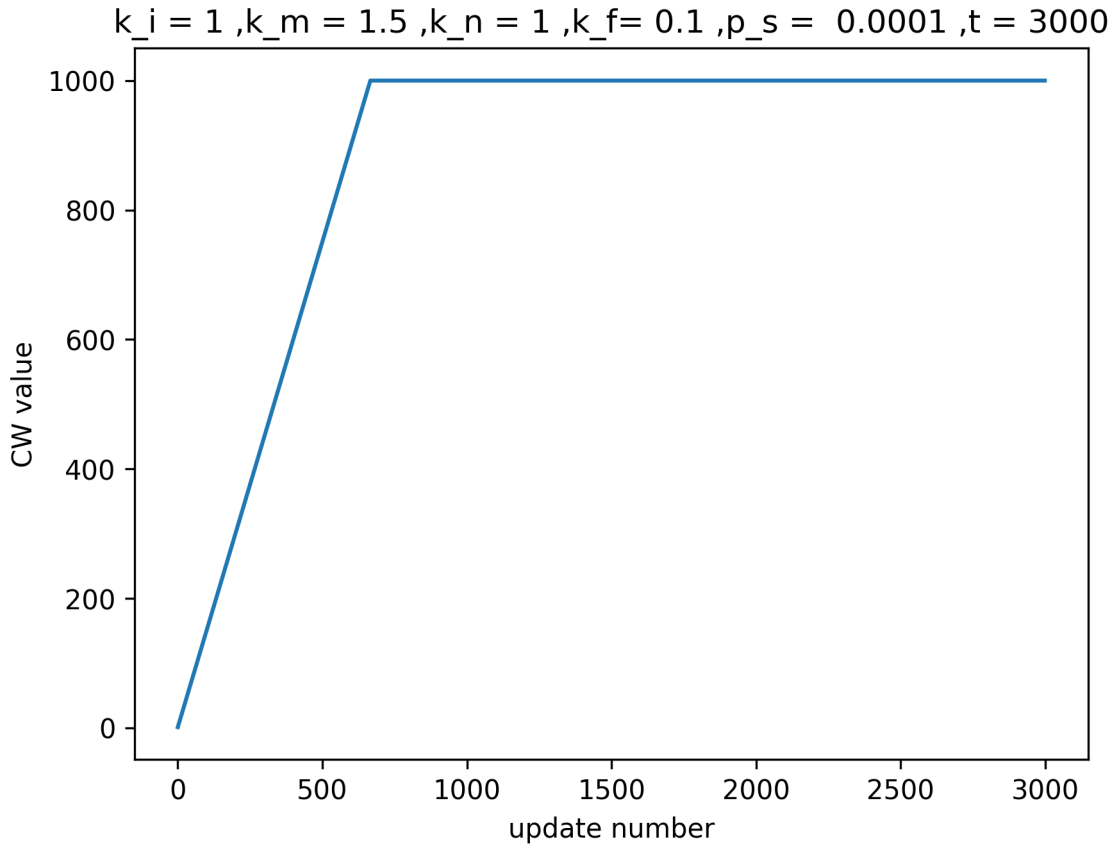

k_i = 1 ,k_m = 1.5 ,k_n = 0.5 ,k_f= 0.1 ,p_s = 0.01 ,t = 3000
CW value
update number
0
100
200
300
400
0
500
1000
1500
2000
2500
3000

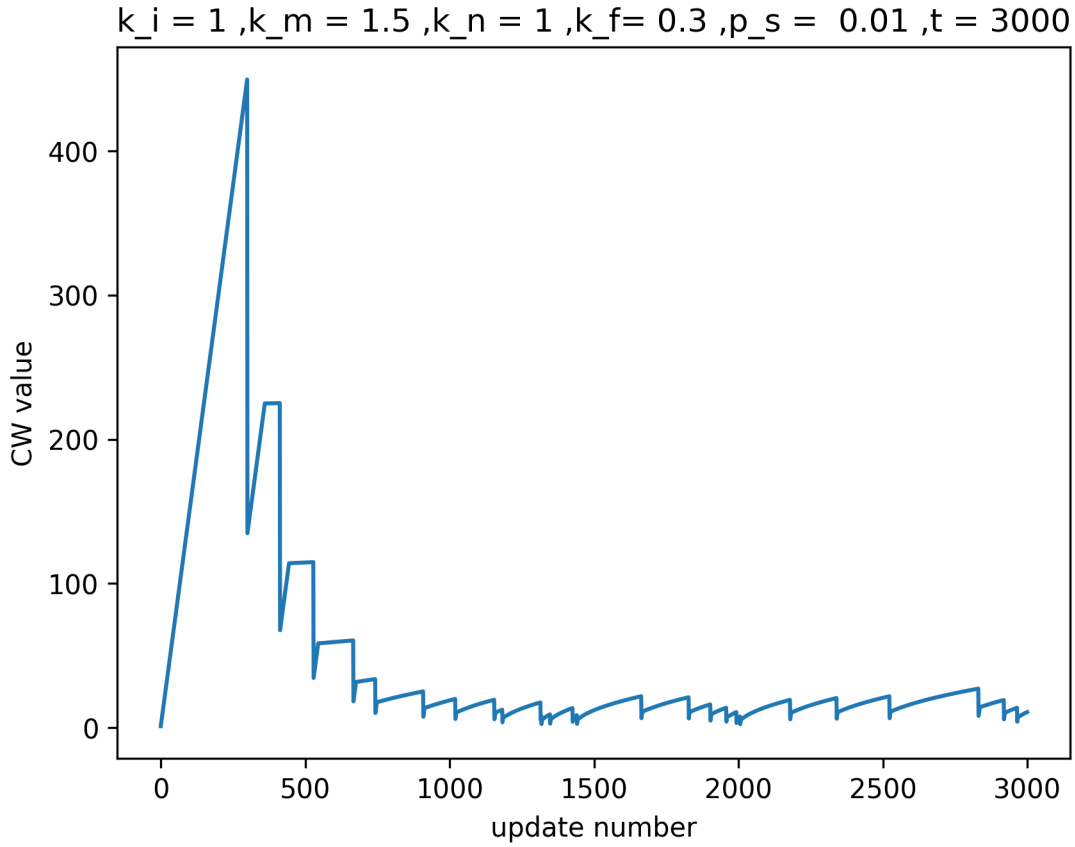

k_i = 1 ,k_m = 1.5 ,k_n = 0.5 ,k_f= 0.1 ,p_s = 0.0001 ,t = 3000
CW value
update number
0
200
400
600
800
1000
0
500
1000
1500
2000
2500
3000

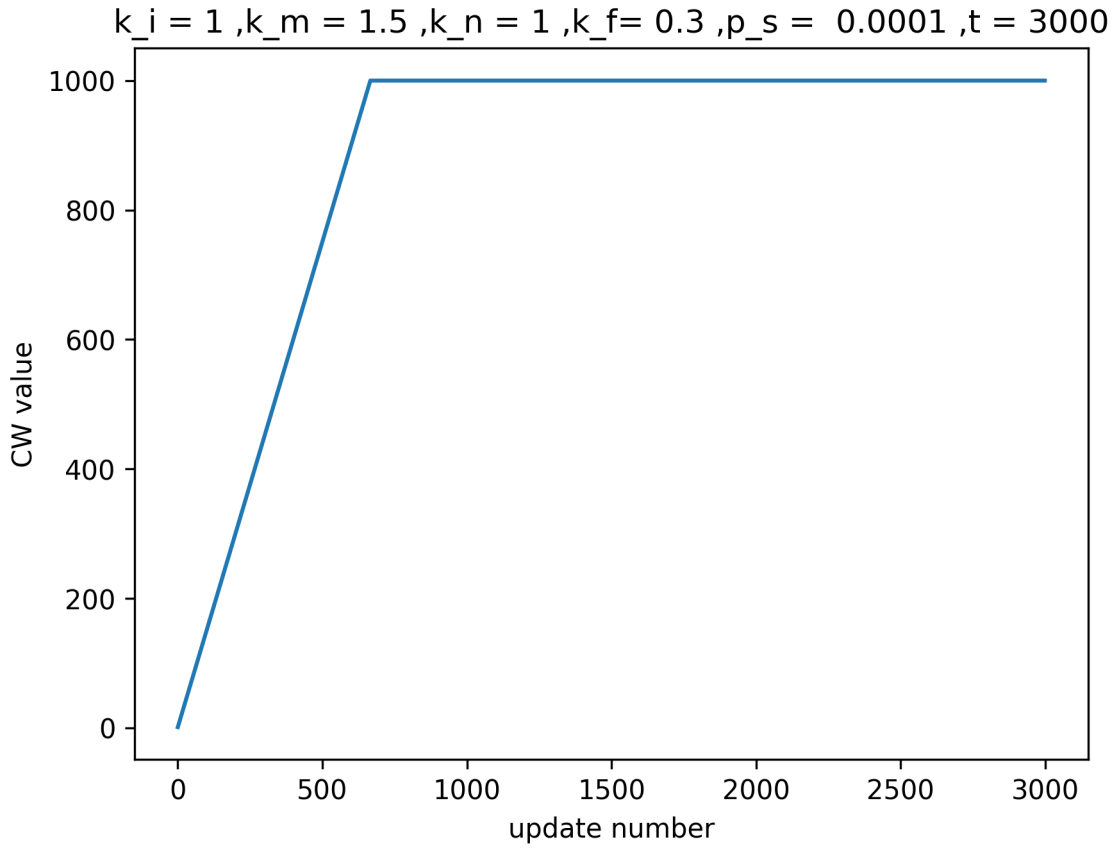

k_i = 1 ,k_m = 1.5 ,k_n = 0.5 ,k_f= 0.3 ,p_s = 0.01 ,t = 3000
CW value
update number
0
100
200
300
400
0
500
1000
1500
2000
2500
3000

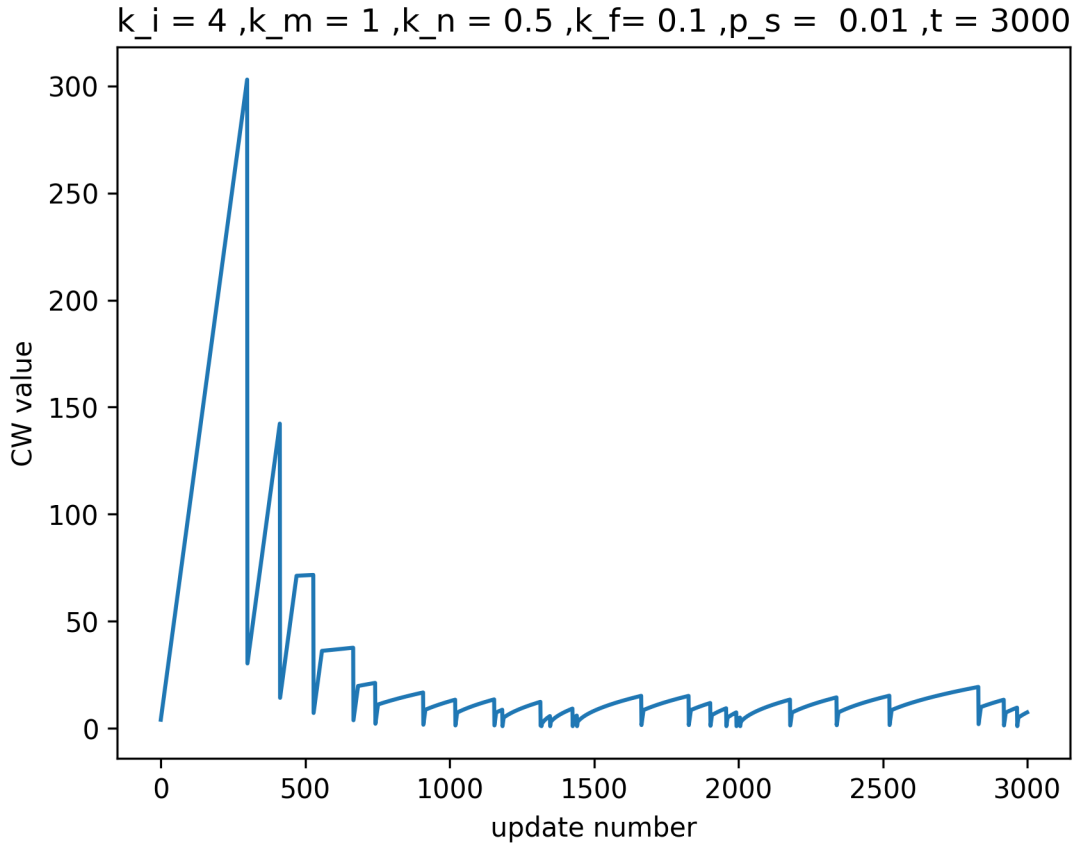

k_i = 1 ,k_m = 1.5 ,k_n = 0.5 ,k_f= 0.3 ,p_s = 0.0001 ,t = 3000
CW value
update number
0
200
400
600
800
1000
0
500
1000
1500
2000
2500
3000

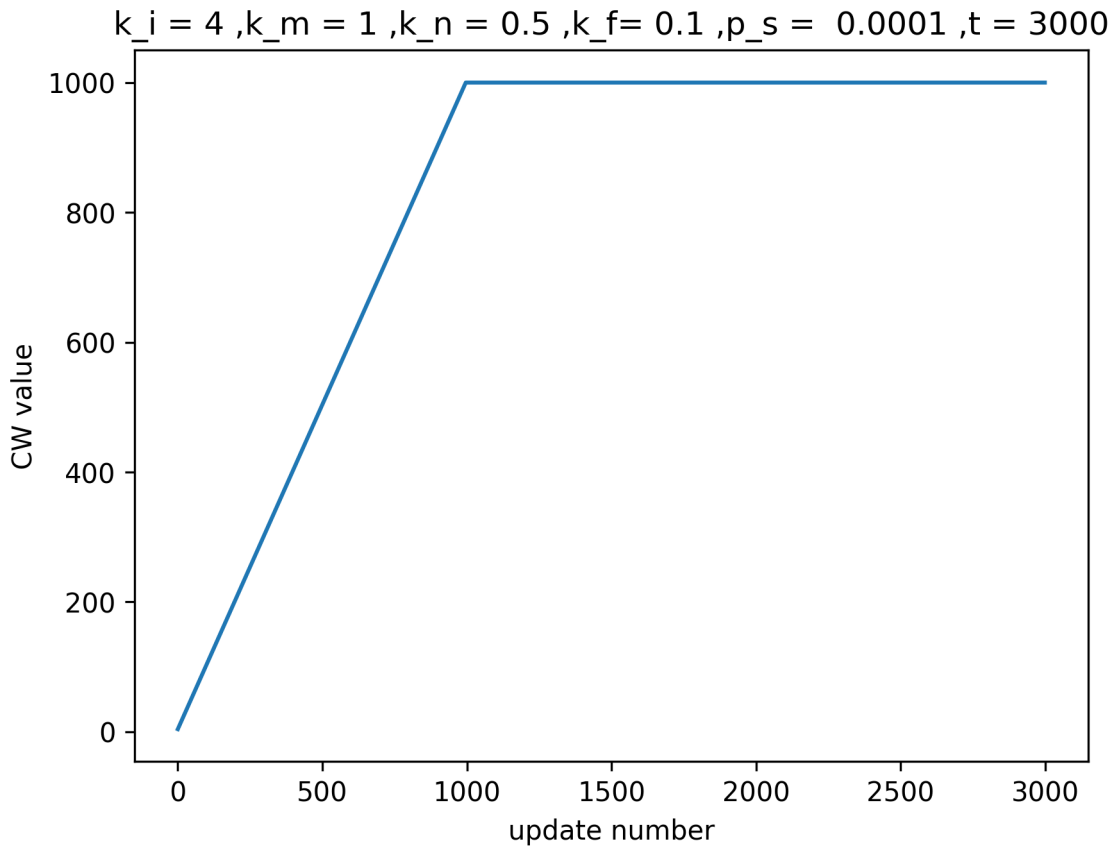

k_i = 1 ,k_m = 1.5 ,k_n = 1 ,k_f= 0.1 ,p_s = 0.01 ,t = 3000
CW value
update number
0
100
200
300
400
0
500
1000
1500
2000
2500
3000

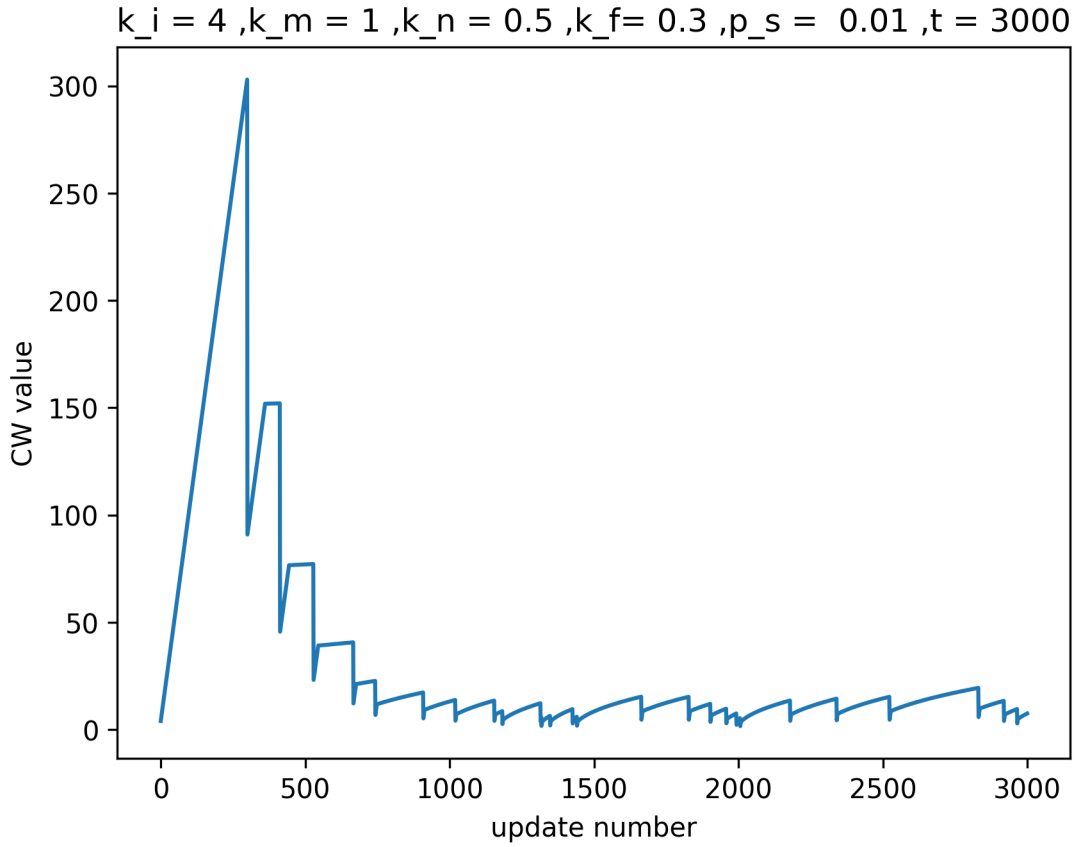

k_i = 1 ,k_m = 1.5 ,k_n = 1 ,k_f= 0.1 ,p_s = 0.0001 ,t = 3000
CW value
update number
0
200
400
600
800
1000
0
500
1000
1500
2000
2500
3000

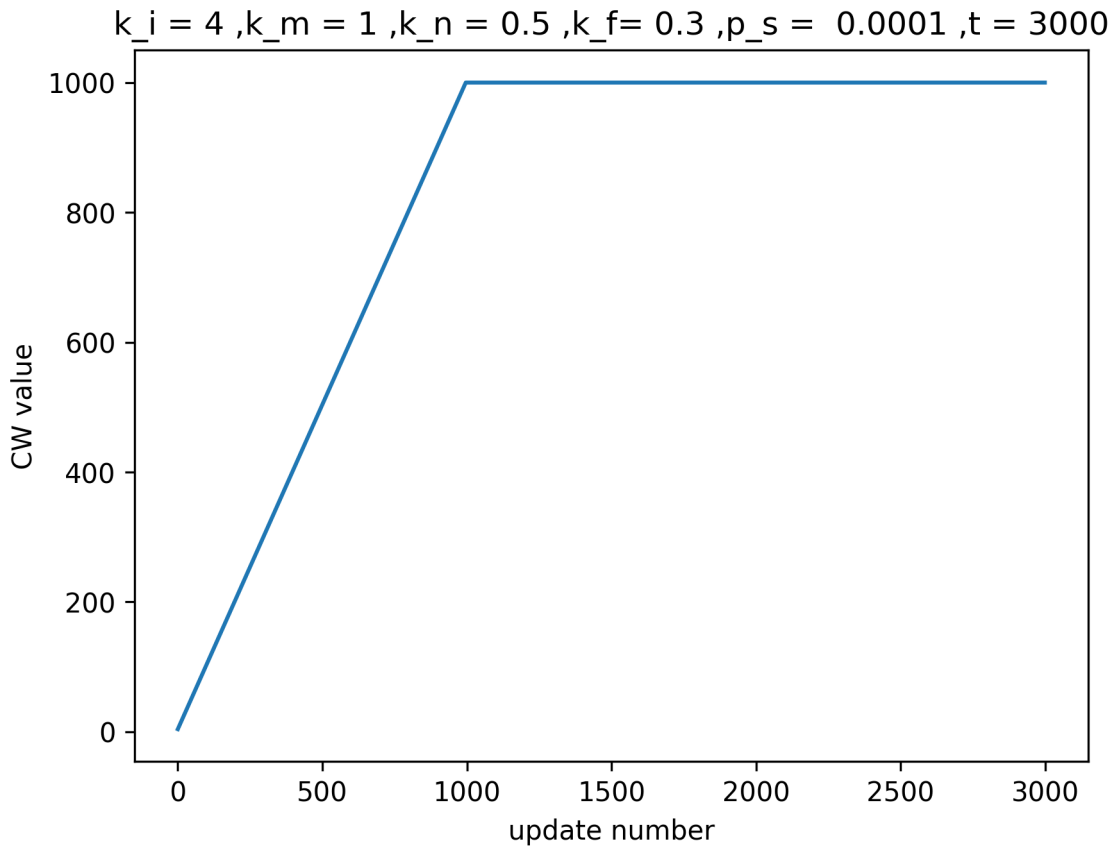

k_i = 1 ,k_m = 1.5 ,k_n = 1 ,k_f= 0.3 ,p_s = 0.01 ,t = 3000
CW value
update number
0
100
200
300
400
0
500
1000
1500
2000
2500
3000

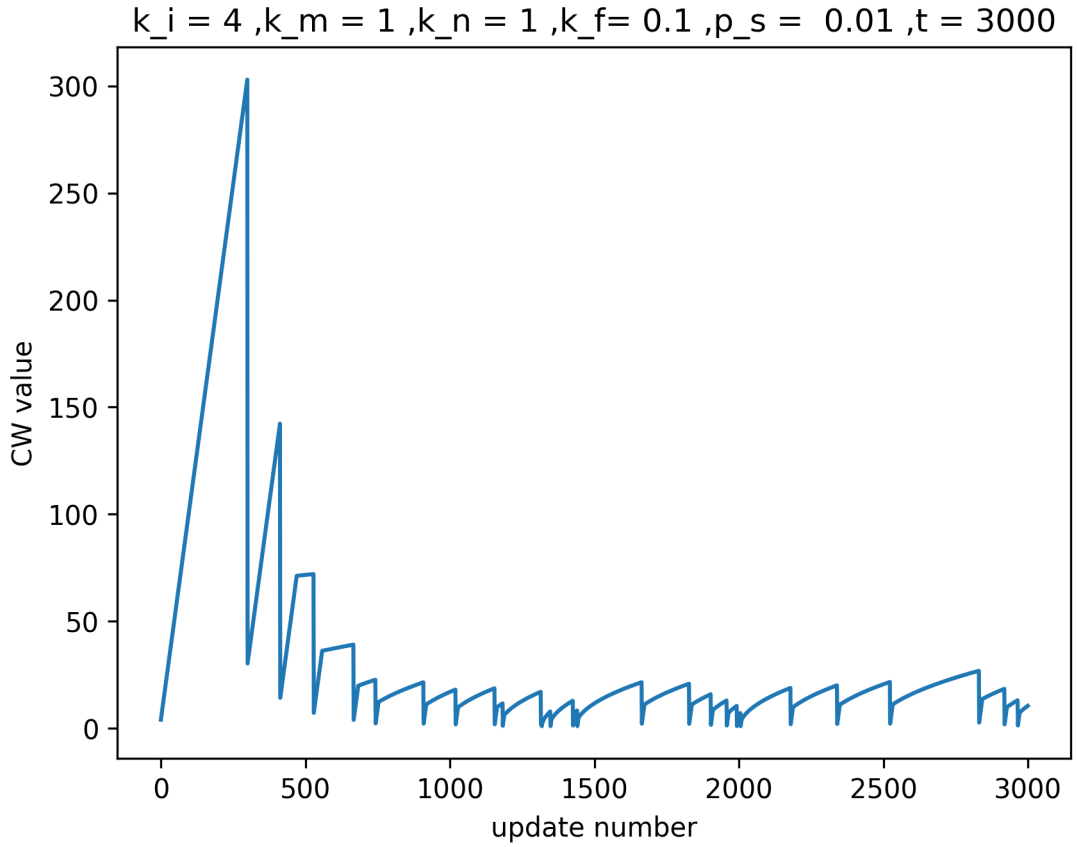

k_i = 1 ,k_m = 1.5 ,k_n = 1 ,k_f= 0.3 ,p_s = 0.0001 ,t = 3000
CW value
update number
0
200
400
600
800
1000
0
500
1000
1500
2000
2500
3000

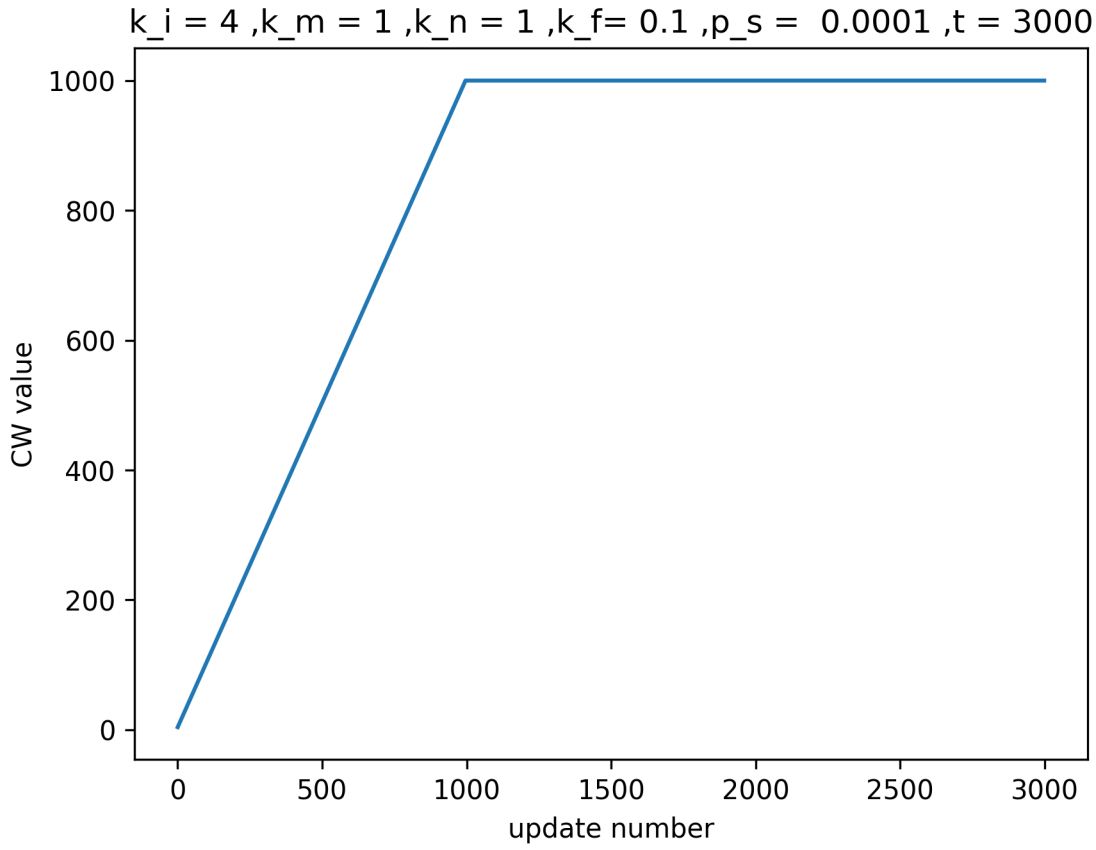

k_i = 4 ,k_m = 1 ,k_n = 0.5 ,k_f= 0.1 ,p_s = 0.01 ,t = 3000
CW value
update number
0
50
100
150
200
250
300
0
500
1000
1500
2000
2500
3000

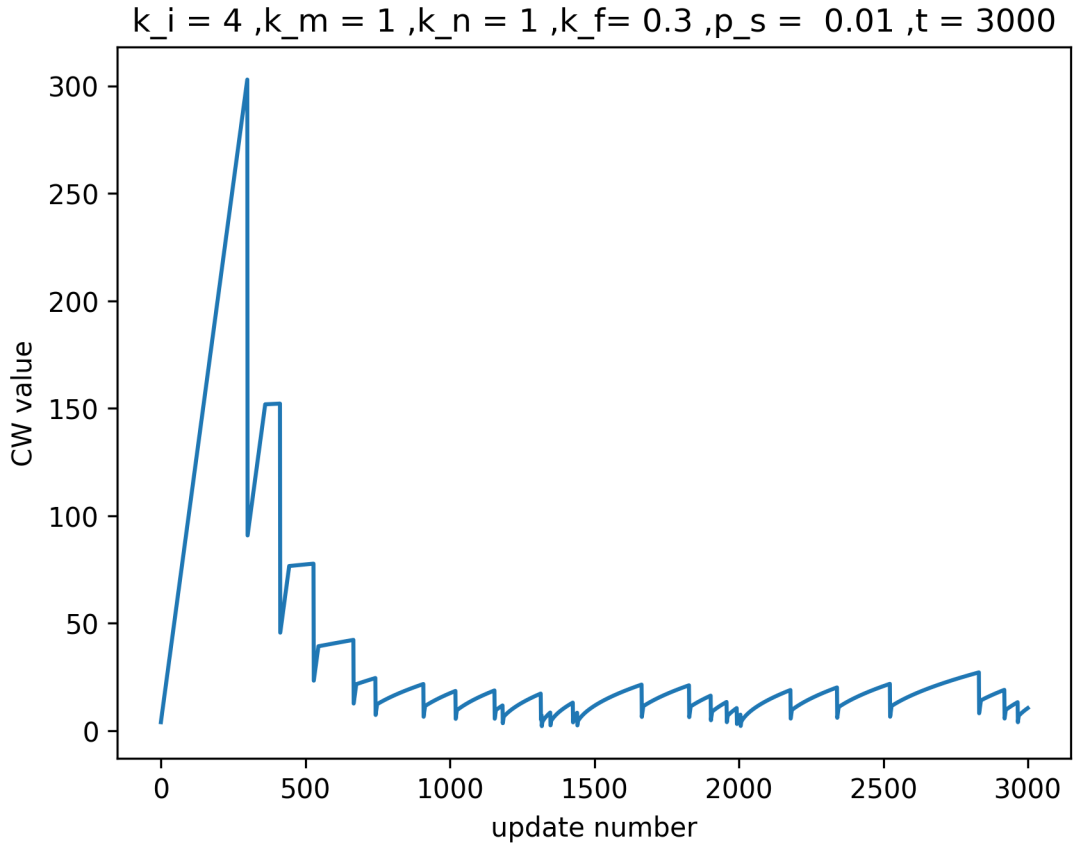

k_i = 4 ,k_m = 1 ,k_n = 0.5 ,k_f= 0.1 ,p_s = 0.0001 ,t = 3000
CW value
update number
0
200
400
600
800
1000
0
500
1000
1500
2000
2500
3000

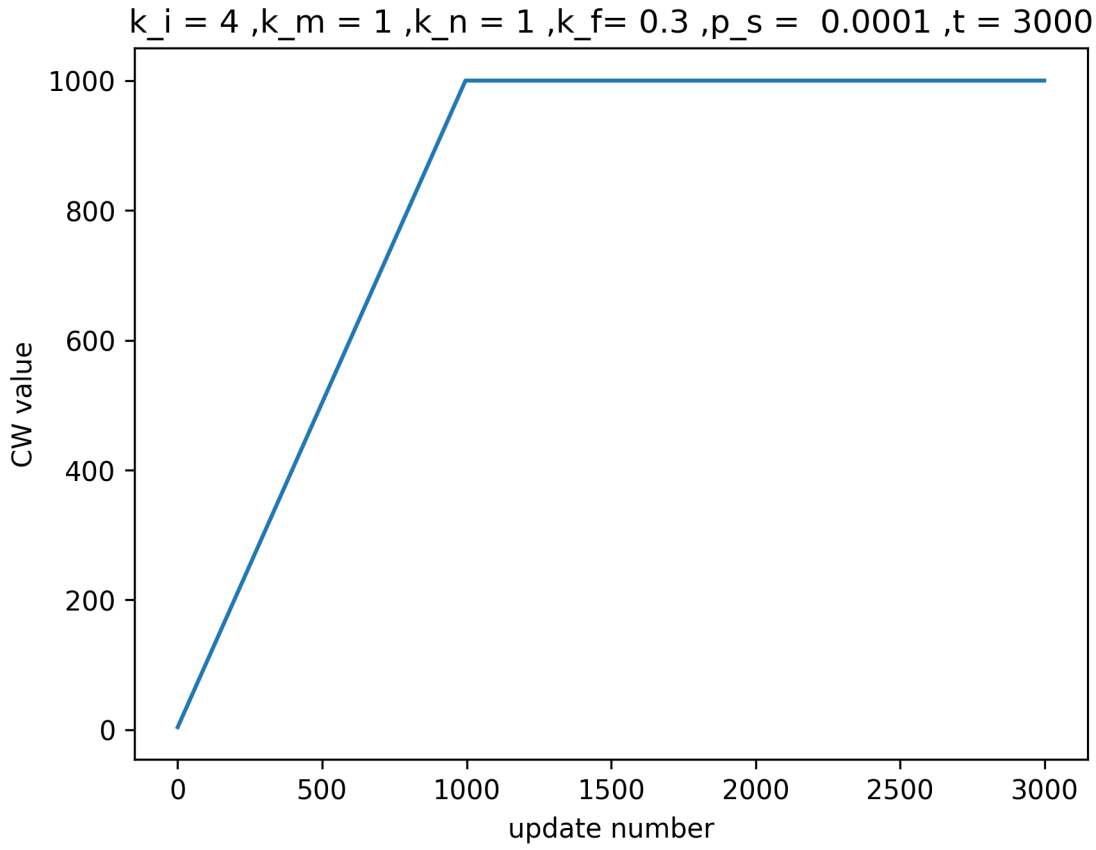

k_i = 4 ,k_m = 1 ,k_n = 0.5 ,k_f= 0.3 ,p_s = 0.01 ,t = 3000
CW value
update number
0
50
100
150
200
250
300
0
500
1000
1500
2000
2500
3000

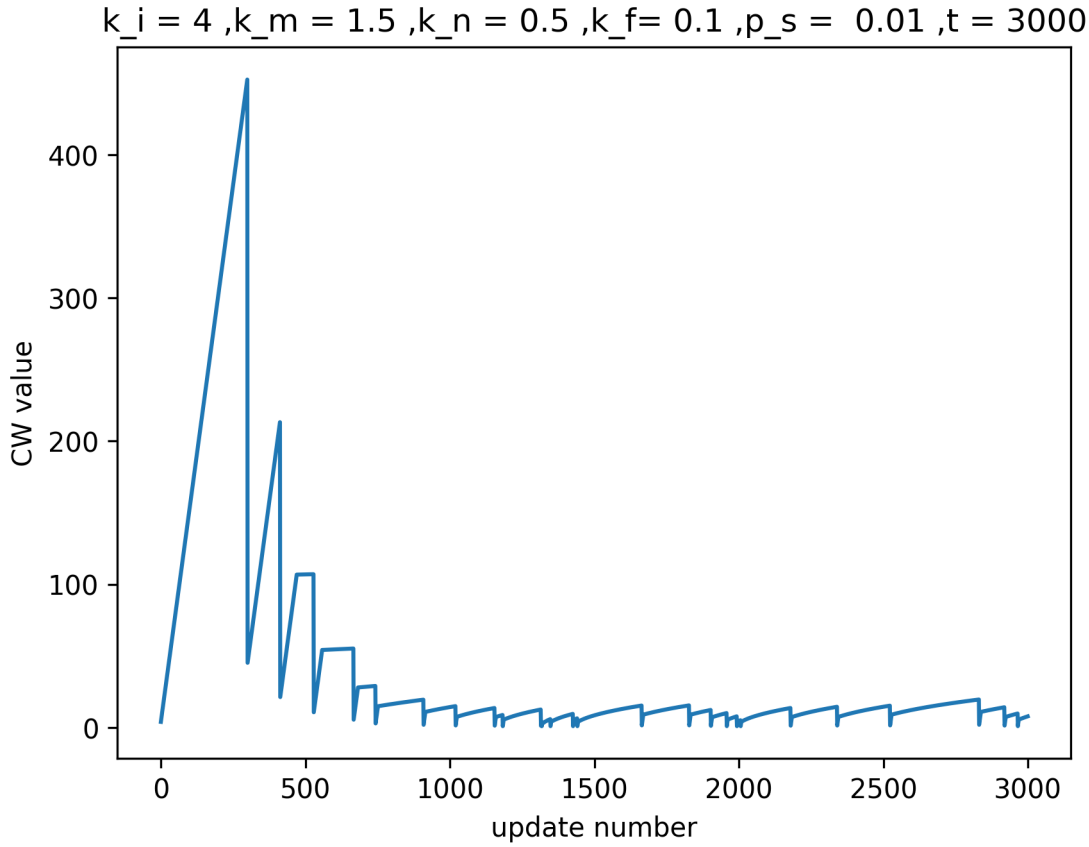

k_i = 4 ,k_m = 1 ,k_n = 0.5 ,k_f= 0.3 ,p_s = 0.0001 ,t = 3000
CW value
update number
0
200
400
600
800
1000
0
500
1000
1500
2000
2500
3000

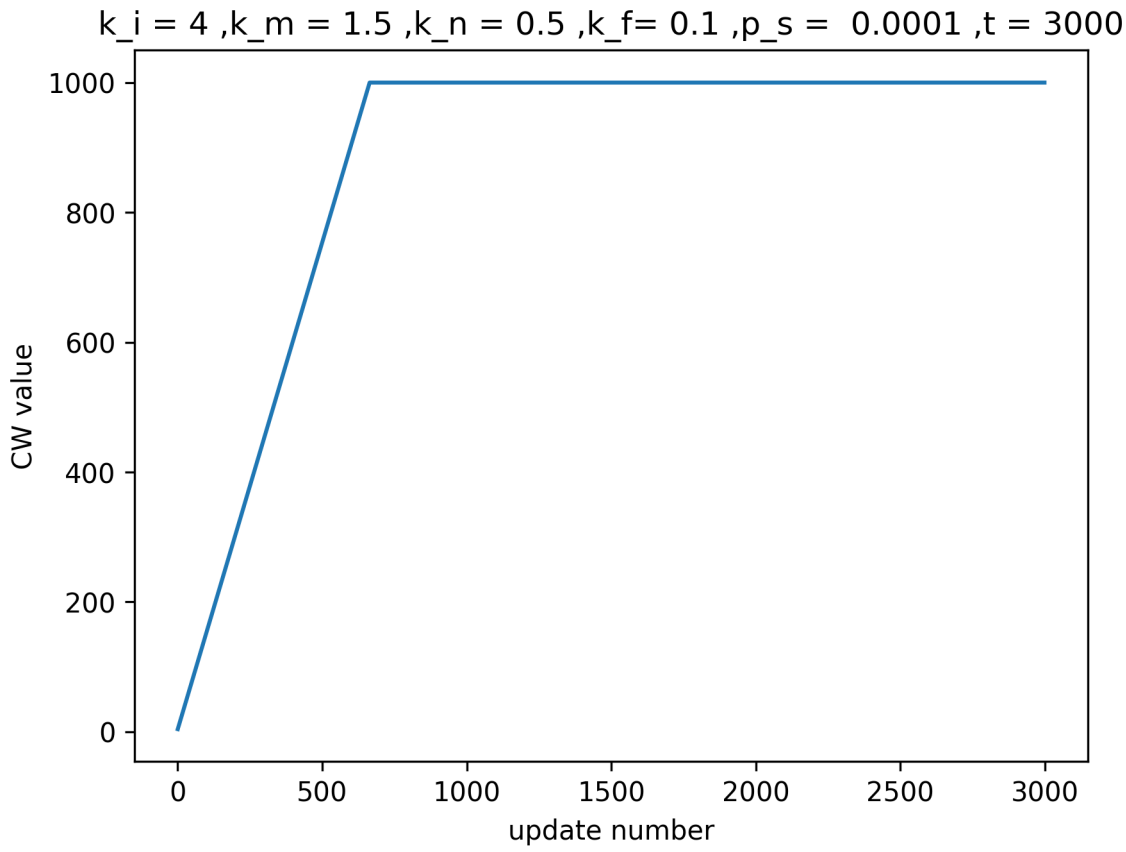

k_i = 4 ,k_m = 1 ,k_n = 1 ,k_f= 0.1 ,p_s = 0.01 ,t = 3000
CW value
update number
0
50
100
150
200
250
300
0
500
1000
1500
2000
2500
3000

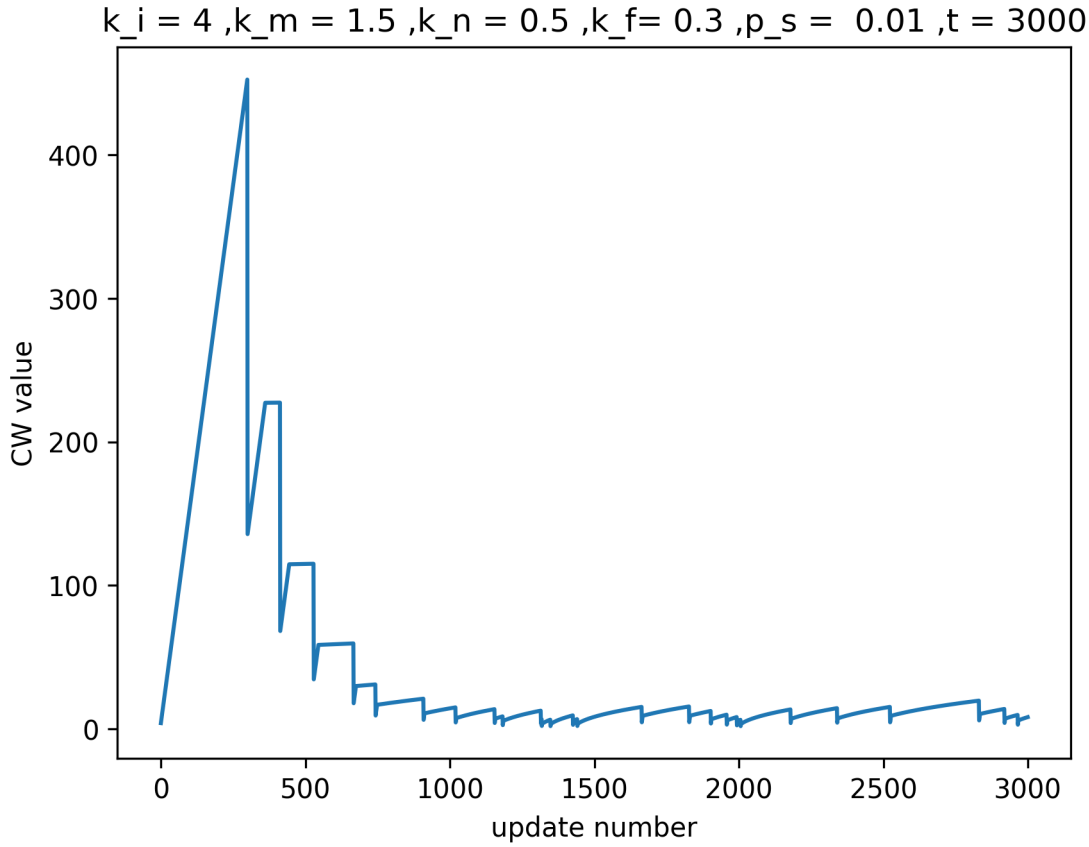

k_i = 4 ,k_m = 1 ,k_n = 1 ,k_f= 0.1 ,p_s = 0.0001 ,t = 3000
CW value
update number
0
200
400
600
800
1000
0
500
1000
1500
2000
2500
3000

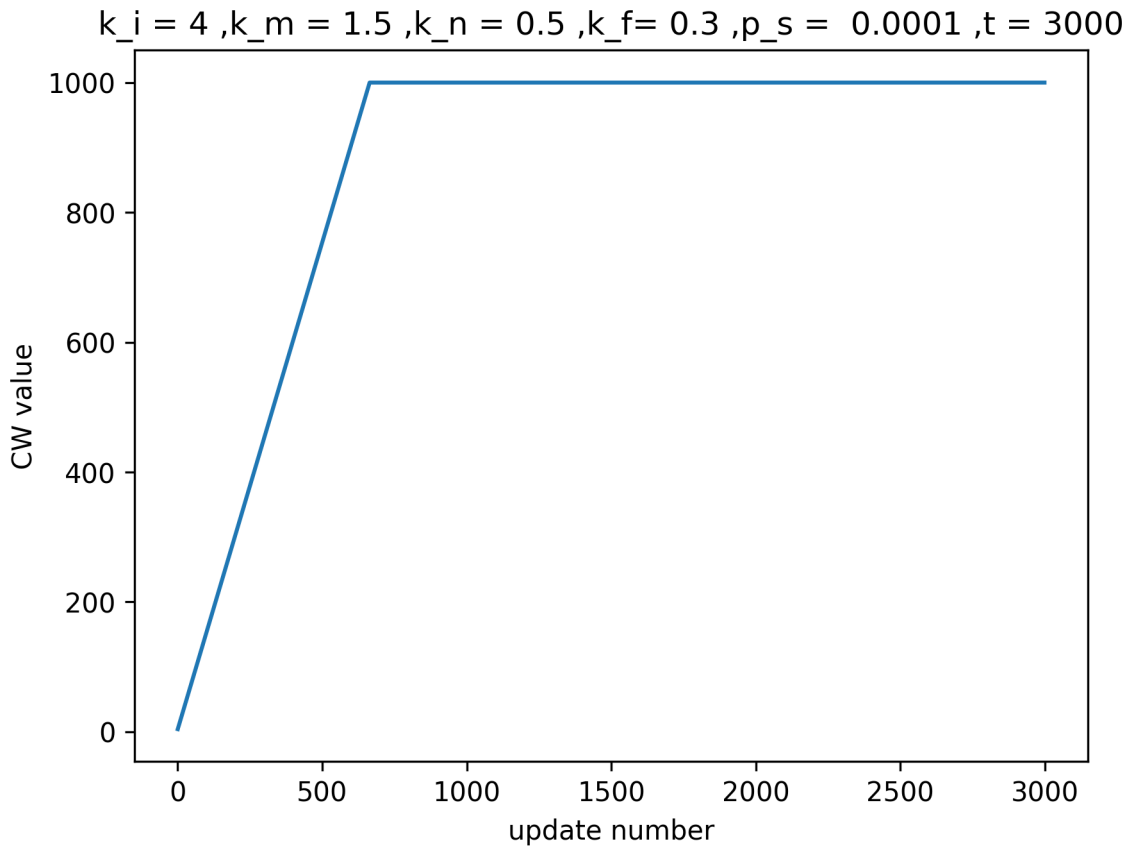

k_i = 4 ,k_m = 1 ,k_n = 1 ,k_f= 0.3 ,p_s = 0.01 ,t = 3000
CW value
update number
0
50
100
150
200
250
300
0
500
1000
1500
2000
2500
3000

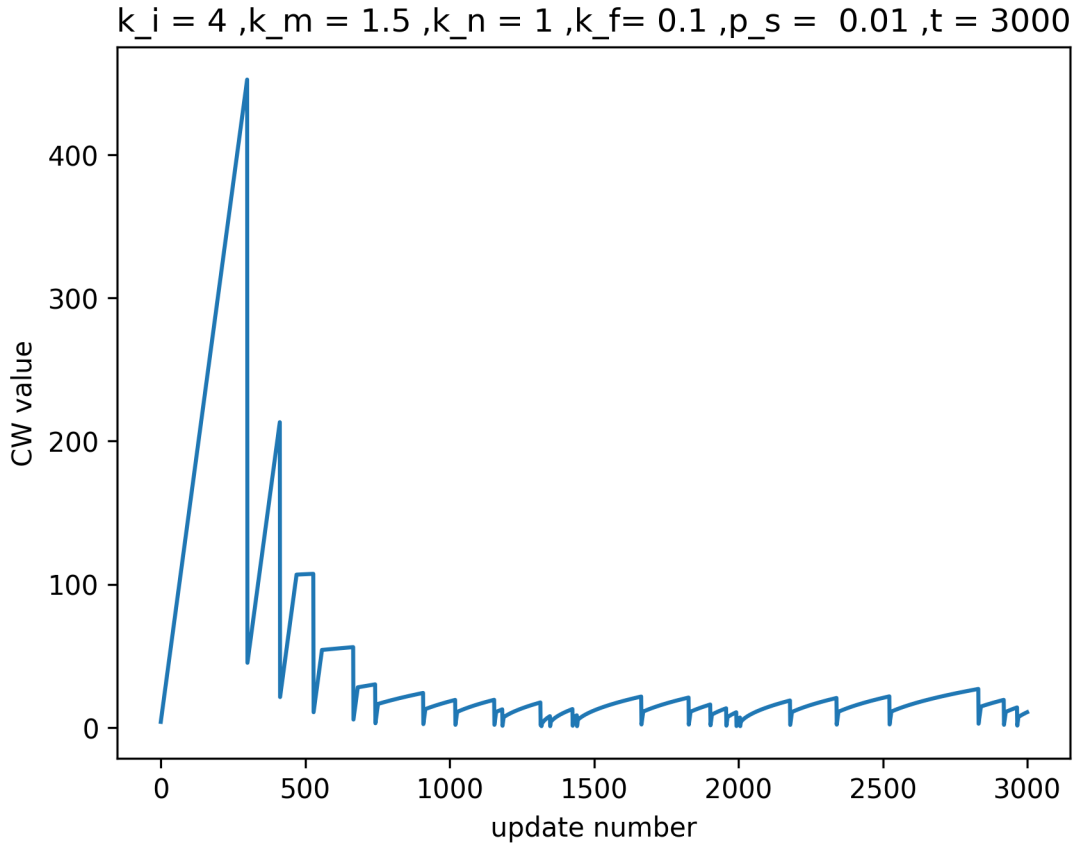

k_i = 4 ,k_m = 1 ,k_n = 1 ,k_f= 0.3 ,p_s = 0.0001 ,t = 3000
CW value
update number
0
200
400
600
800
1000
0
500
1000
1500
2000
2500
3000

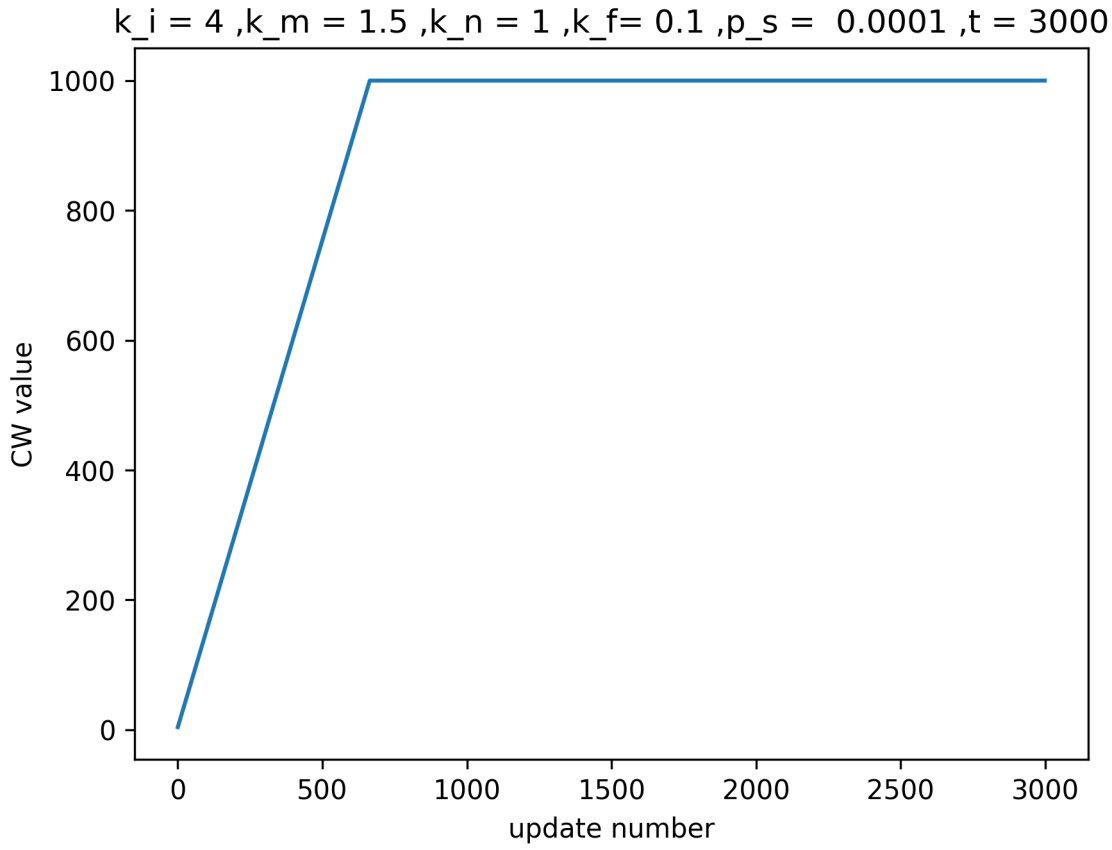

k_i = 4 ,k_m = 1.5 ,k_n = 0.5 ,k_f= 0.1 ,p_s = 0.01 ,t = 3000
CW value
update number
0
100
200
300
400
0
500
1000
1500
2000
2500
3000

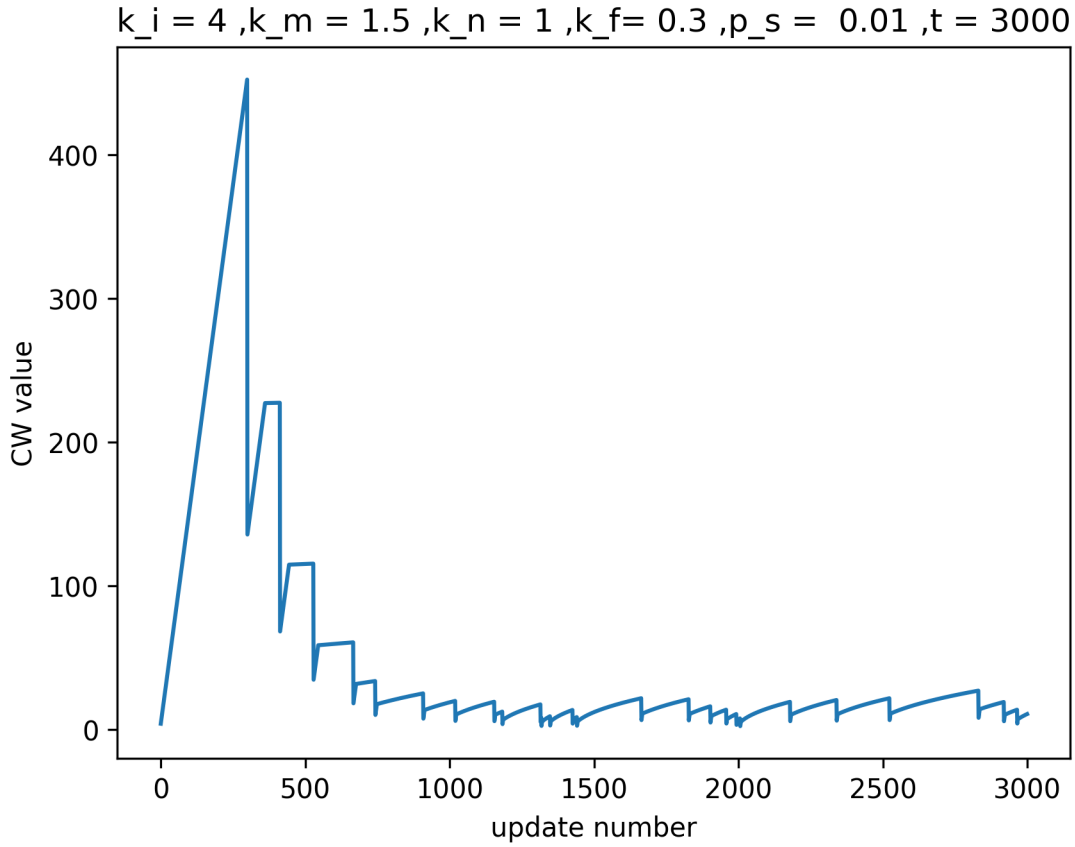

k_i = 4 ,k_m = 1.5 ,k_n = 0.5 ,k_f= 0.1 ,p_s = 0.0001 ,t = 3000
CW value
update number
0
200
400
600
800
1000
0
500
1000
1500
2000
2500
3000

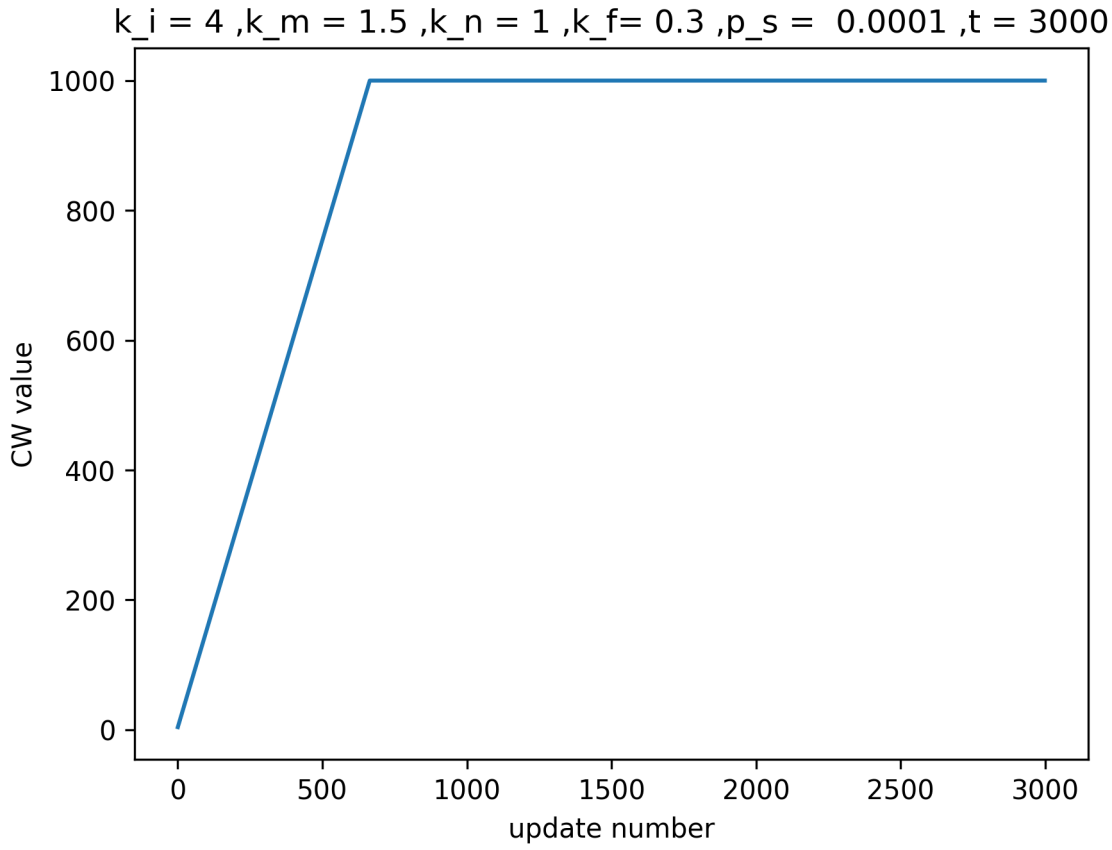

k_i = 4 ,k_m = 1.5 ,k_n = 0.5 ,k_f= 0.3 ,p_s = 0.01 ,t = 3000
CW value
update number
0
100
200
300
400
0
500
1000
1500
2000
2500
3000


k_i = 4 ,k_m = 1.5 ,k_n = 0.5 ,k_f= 0.3 ,p_s = 0.0001 ,t = 3000
CW value
update number
0
200
400
600
800
1000
0
500
1000
1500
2000
2500
3000


k_i = 4 ,k_m = 1.5 ,k_n = 1 ,k_f= 0.1 ,p_s = 0.01 ,t = 3000
CW value
update number
0
100
200
300
400
0
500
1000
1500
2000
2500
3000


k_i = 4 ,k_m = 1.5 ,k_n = 1 ,k_f= 0.1 ,p_s = 0.0001 ,t = 3000
CW value
update number
0
200
400
600
800
1000
0
500
1000
1500
2000
2500
3000


k_i = 4 ,k_m = 1.5 ,k_n = 1 ,k_f= 0.3 ,p_s = 0.01 ,t = 3000
CW value
update number
0
100
200
300
400
0
500
1000
1500
2000
2500
3000


k_i = 4 ,k_m = 1.5 ,k_n = 1 ,k_f= 0.3 ,p_s = 0.0001 ,t = 3000
CW value
update number
0
200
400
600
800
1000
0
500
1000
1500
2000
2500
3000

5 Learnings

- the first timeout is very important .
- All the 5 parameters we are using effect the graph.
- But The probability of timeout P_s affects the graph very much.

6 Additional Thoughts

- for a particular K_n , K_m , K_f , P_s if we keep changing K_i we can know what is the optimal senders window size is for that parameters.

7 conclusion

- TCP congestion control is very important , because as we have seen the graph increases exponenetially before timeout . and doing threshold = cw/2 makes that highest cw reached after sometime reduces.

8 References

https://www.geeksforgeeks.org/tcp-congestion-control/
https://www.geeksforgeeks.org/congestion-control-in-computer-networks/
https://www.geeksforgeeks.org/graph-plotting-in-python-set-1/