filter output `conv_valid_out` = convolve(ringbuffer signal in, filter tap array, mode='valid') Dennis van Gils, 23-05-2021

`rb` stands for ring buffer

Test scenario 1:

Perform convolve every incoming block, ring buffer is 2x BLOCK_SIZE

Not optimal

BLOCK_SIZE	10	samples	
N_blocks	2	int	
rb_capacity	20	samples	
N_taps	7	samples	== must be odd number
N_sig_into_conv	16	samples	
offset_sig_into_conv	4	samples	
N_conv_valid_out	10	samples	== BLOCK_SIZE
offset_conv_valid_out	3	samples	
idx_rb_valid_start	7	samples	

block:	#0: start-up										#1										#2										#3										
conv_valid_out:								1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	3	3	3	3	3	3	3	3	3	3				
offset_conv_valid_out:					1*	1*	1*								2*	2*	2*	1*	1*	1*					3*	3*	3*	2*	2*	2*								3*	3*	3*	
wallclock idx:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
					sig_into_conv			conv_valid_out							conv_valid_out			sig_into_conv																							
ringbuffer idx:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19																					
ringbuffer idx:											0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19											
ringbuffer idx:																					0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	

Test scenario 2:

Perform convolve every incoming block, ring buffer is 3x BLOCK_SIZE

Maximized N_taps given N_blocks

BLOCK_SIZE	10	samples	
N_blocks	3	int	
rb_capacity	30	samples	
N_taps	21	samples	== must be odd number
N_sig_into_conv	30	samples	
offset_sig_into_conv	0	samples	
N_conv_valid_out	10	samples	== BLOCK_SIZE
offset_conv_valid_out	10	samples	
idx_rb_valid_start	10	samples	

block:	#0: start-up										#1: start-up										#2										#3										#4										
conv_valid_out:											2	2	2	2	2	2	2	2	2	2	3	3	3	3	3	3	3	3	3	3	4	4	4	4	4	4	4	4	4	4											
offset_conv_valid_out:	2*	2*	2*	2*	2*	2*	2*	2*	2*	2*	3*	3*	3*	3*	3*	3*	3*	3*	3*	3*	2*	2*	2*	2*	2*	2*	2*	2*	2*	2*	3*	3*	3*	3*	3*	3*	3*	3*	3*	3*	4*	4*	4*	4*	4*	4*	4*	4*	4*	4*	
																					4*	4*	4*	4*	4*	4*	4*	4*	4*	4*																					
wallclock idx:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
ringbuffer idx:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29																					
ringbuffer idx:											0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29											
ringbuffer idx:																					0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	

Optimal scenario:

Perform convolve every incoming block, maximize N_taps given N_blocks

Fs	20000	Hz	
BLOCK_SIZE	2000	samples	
N_blocks	21	int	
rb_capacity	42000	samples	= BLOCK_SIZE * N_blocks
N_taps	40001	samples	= BLOCK_SIZE * (N_blocks - 1) + 1 == max that fits using rb_capacity

Fred Harris' approximation — Multirate Signal Processing for Communication Systems, Fredric J. Harris, 2004, page 216, equation (8.16)

f_pass	49	Hz	
f_stop	50	Hz	
filter attenuation	44.00	dB	= N_taps * 22 * ((f_stop - f_pass) / Fs)

N_sig_into_conv	42000	samples	= BLOCK_SIZE + N_taps - 1 == rb_capacity by optimal design	
offset_sig_into_conv	0	samples	= rb_capacity - N_sig_into_conv == 0 by optimal design	
N_conv_valid_out	2000	samples	= rb_capacity - N_taps + 1 == BLOCK_SIZE by optimal design	
offset_conv_valid_out	20000	samples	= INT((N_taps - 1) / 2)	
idx_rb_valid_start	20000	samples	= offset_conv_valid_out + offset_sig_into_conv == offset_conv_valid_out by optimal design	
T_valid_start	1.00	s	= idx_rb_valid_start / Fs	i.e. Starting index within the input-signal ring buffer corresponding to the time stamps of the computed valid filter output
T_settle_filter	2.00	s	= (N_blocks - 1) * BLOCK_SIZE / Fs	i.e. When the filter starts outputting (not to be confused with the filter theoretical response time to an impulse)