An *In Situ* Approach for Approximating Complex Computer Simulations and Identifying Important Time Steps

Kary Myers, Statistical Sciences Group
Los Alamos National Laboratory

Joint work with Earl Lawrence, Mike Fugate, Claire Bowen, Larry Ticknor,
Joanne Wendelberger, Jon Woodring, and Jim Ahrens

Sponsored by the National Nuclear Security Administration under contract DE-AC52-06NA25396.
LA-UR-15-23050.

CoDA2016

March 2-4, 2016 | Santa Fe, New Mexico

Exploring Data-Focused Research across the Department of Energy

Join us for the third Conference on Data Analysis, bringing together scientists, statisticians, and data analysts from across the Department of Energy national laboratories along with their academic and industrial collaborators.

Banquet Speaker: Rayid Ghani, Program Director, Data Science for Social Good, University of Chicago.

Call for Posters: Deadline February 3, 2016

Present your data-focused work at the CoDA 2016 poster session!

The CoDA 2016 invited program features 6 themed sessions exploring these topics:

- Power Grid Data
- Subsurface Modeling
- Multisource Data
- Cyber Security
- Data Analysis at Exascale
- Really Expensive Data

Visit cnls.lanl.gov/coda for more information and to register.

Important announcement #2: Statistics and Beer Day

A new holiday to celebrate how the field of statistics has improved the world by focusing on how it has improved beer.

When? June 13, William Sealy Gosset's birthday.

Who? You might remember him by his pseudonym Student, as in Student's *t*-distribution.

Why? Gosset worked for Guinness Brewery where he developed and applied statistical methods to improve the beer.

How? Have a few pints with statisticians and other normal people. Start with a Guinness.

commons.wikimedia.org

An *In Situ* Approach for Approximating Complex Computer Simulations and Identifying Important Time Steps

Kary Myers, Statistical Sciences Group
Los Alamos National Laboratory

Joint work with Earl Lawrence, Mike Fugate, Claire Bowen, Larry Ticknor, Joanne Wendelberger, Jon Woodring, and Jim Ahrens

Sponsored by the National Nuclear Security Administration under contract DE-AC52-06NA25396.

LA-UR-15-23050

An *In Situ* Approach for Approximating Complex Computer Simulations and Identifying Important Time Steps!

Kary Myers, Statistical Sciences Group
Los Alamos National Laboratory

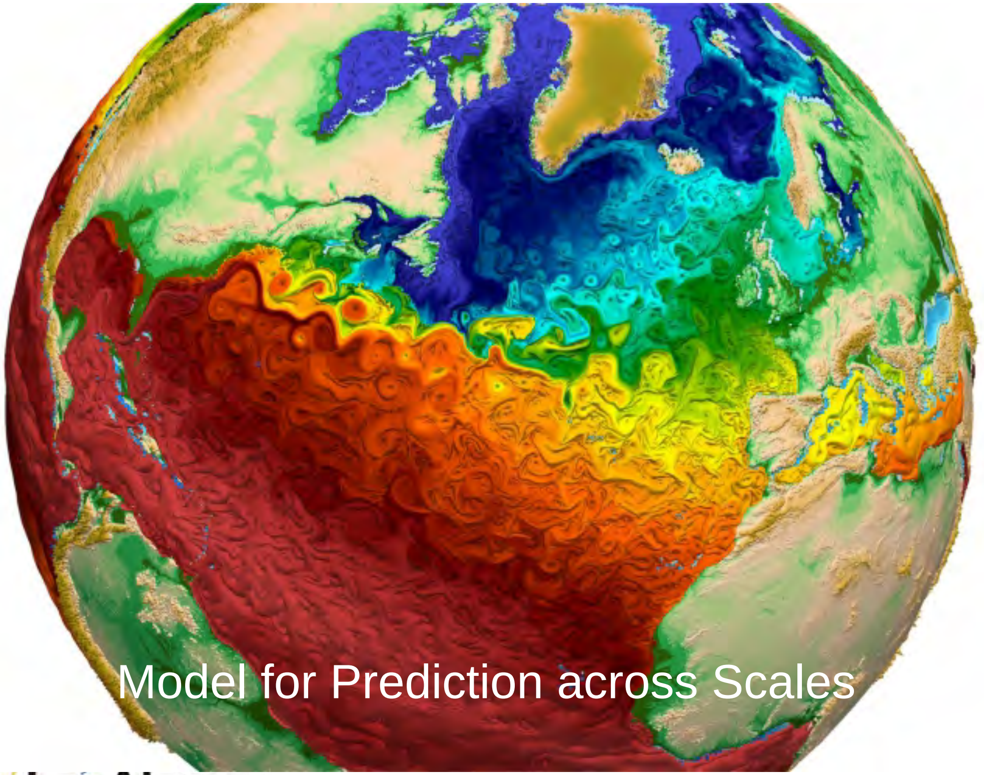
Joint work with Earl Lawrence, Mike Fugate, Claire Bowen, Larry Ticknor, Joanne Wendelberger, Jon Woodring, and Jim Ahrens

Sponsored by the National Nuclear Security Administration under contract DE-AC52-06NA25396.
LA-UR-15-23050.

What’s a complex computer simulation?

Running a numerical model of a system, typically on a supercomputer. Often used when physical experiments aren’t practical.

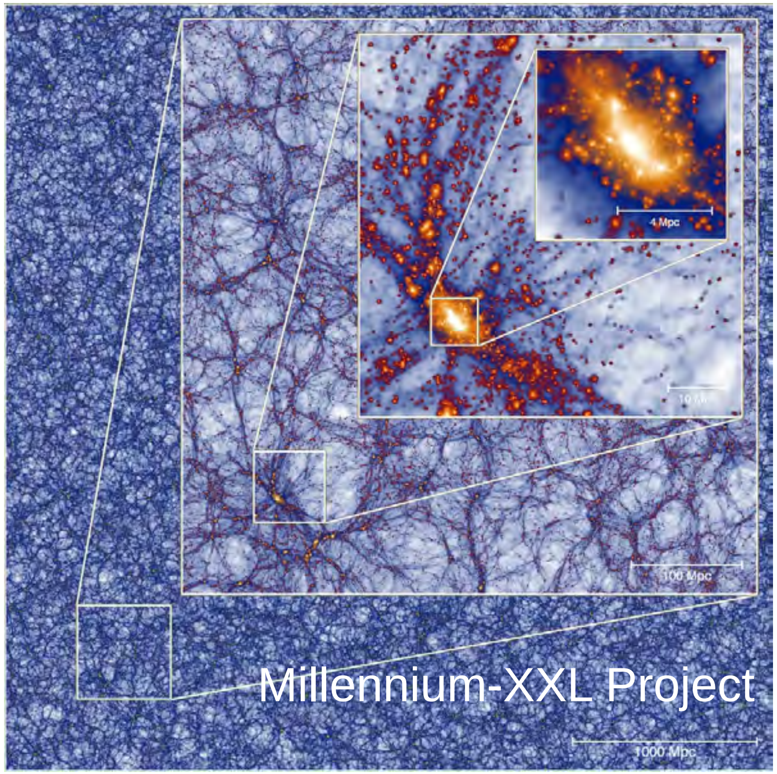
Modeling Climate



Model for Prediction across Scales

Los Alamos National Laboratory

Modeling the Universe



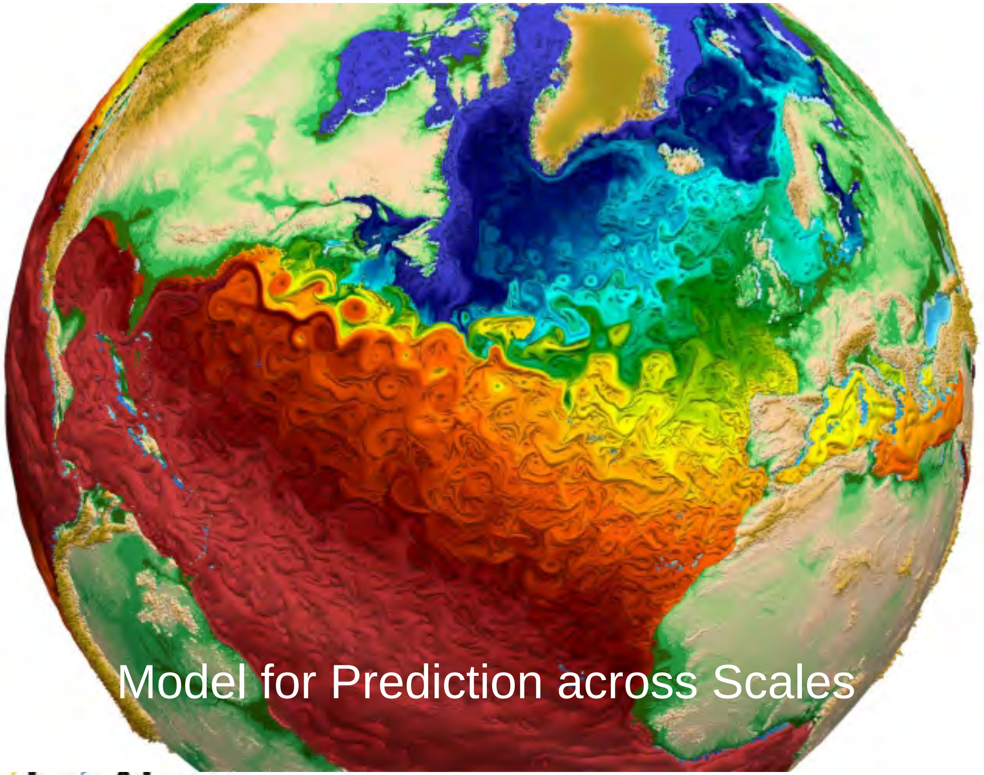
Millennium-XXL Project

Max-Planck-Institute for Astrophysics

Coming soon: Exascale computing

A billion billion (1,000,000,000,000,000,000) calculations every second. This means more science, but only if we can extract useful information.

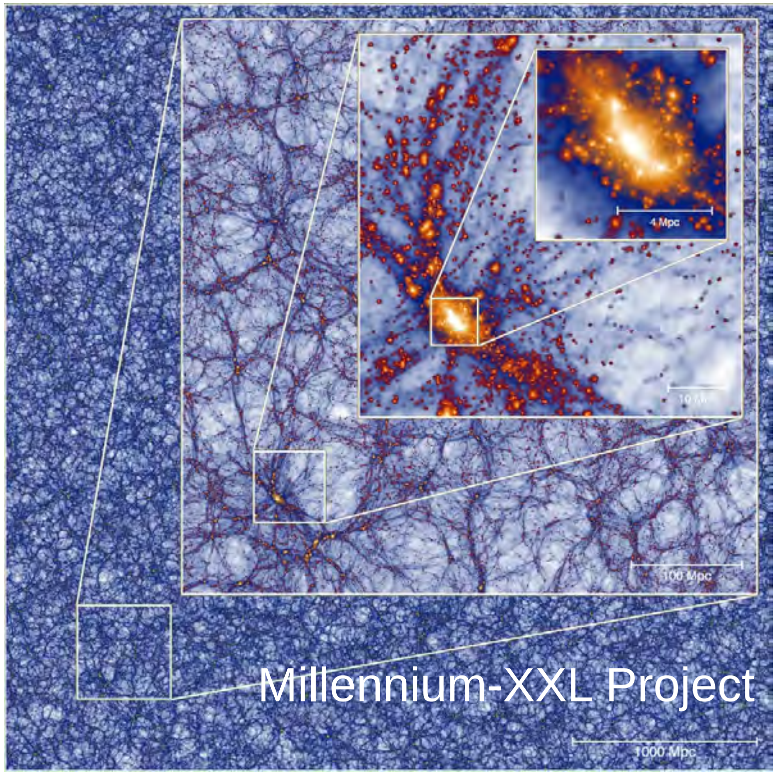
Modeling Climate



Model for Prediction across Scales

Los Alamos National Laboratory

Modeling the Universe

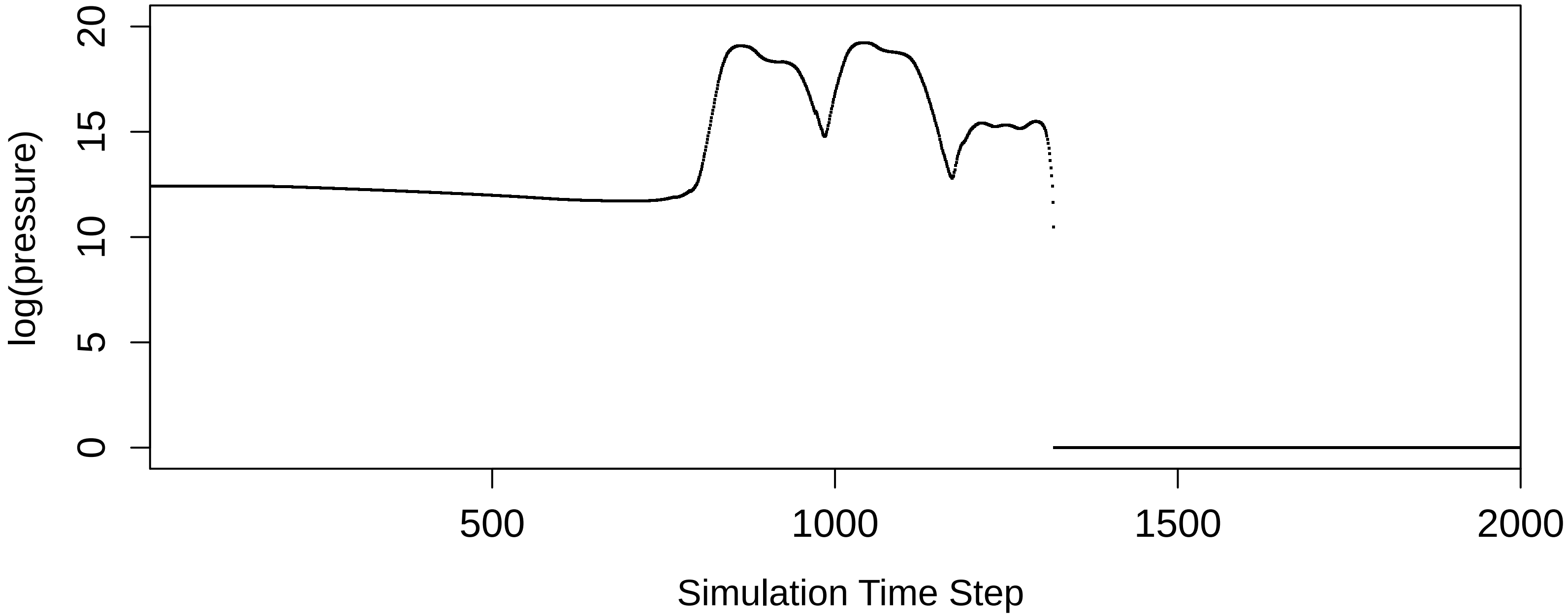


Millennium-XXL Project

Max-Planck-Institute for Astrophysics

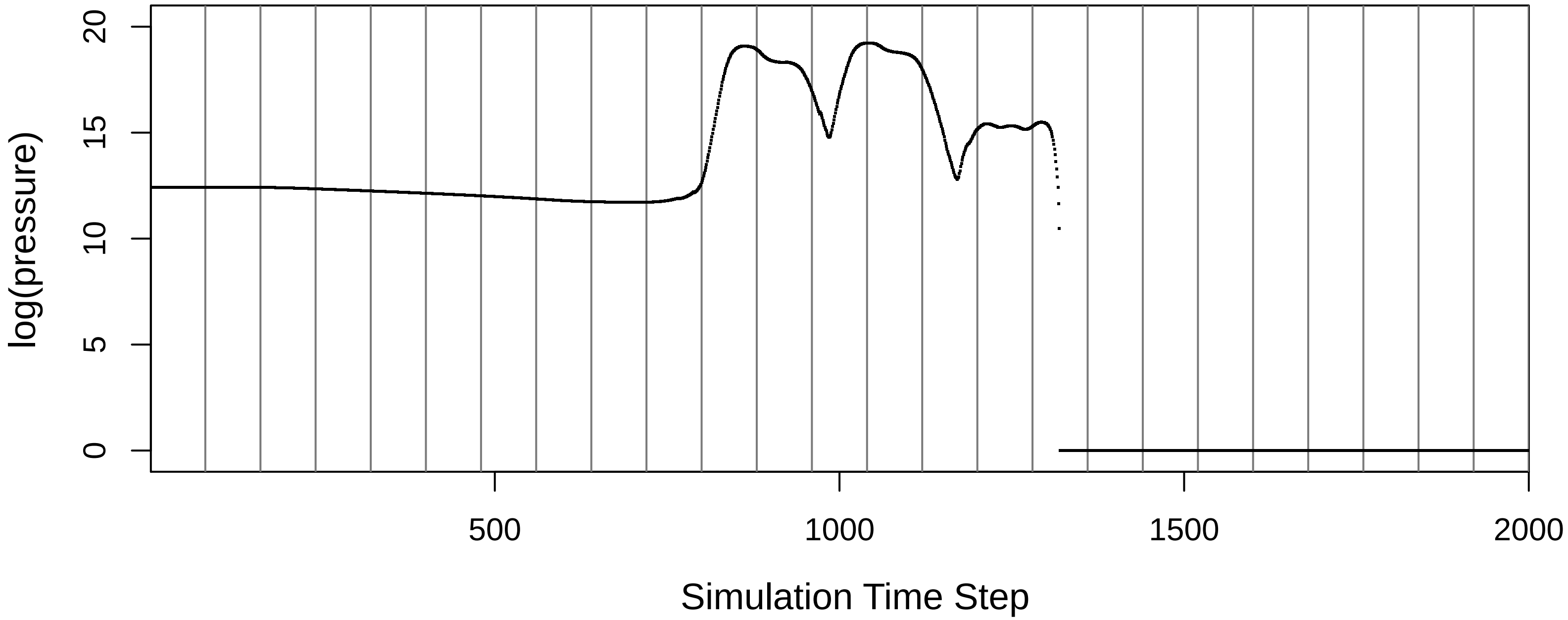
One idea: Only save a subset of simulation time steps

A 1-d example. (I'll talk about the simulation behind this example later.)



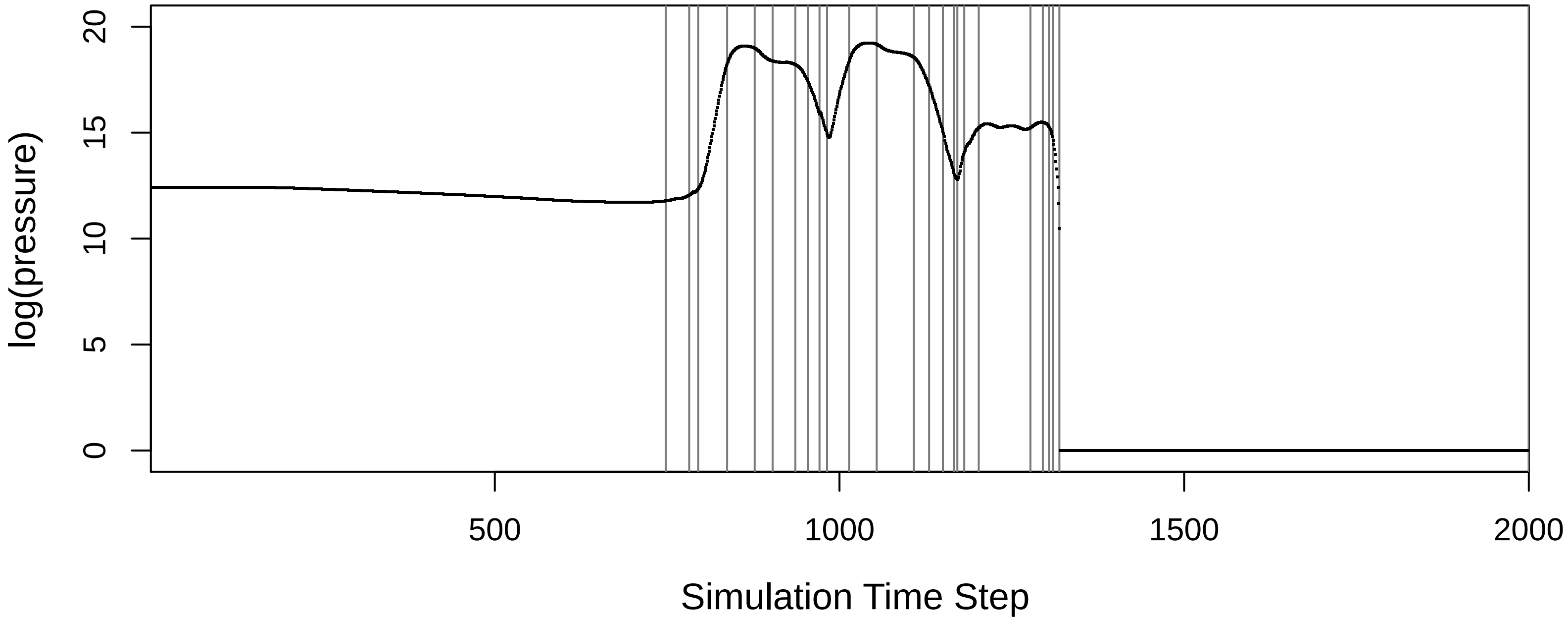
One idea: Only save a subset of simulation time steps

Standard practice: Save evenly spaced time steps.



One idea: Only save a subset of simulation time steps

Our approach: An *in situ* analysis to select “important” time steps in an online fashion. We do this by cheaply computing and comparing linear fits.

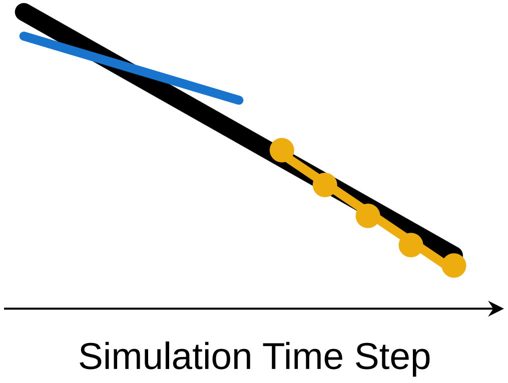


Our *in situ* approach: Compare linear fits

Cheap to compute and update within the simulation as it’s running.

Compare 3 lines in 2 temporal regions of interest:

- **curr:** Time steps currently characterized by a linear fit; only sufficient statistics are stored.
- **buff:** *B* time steps most recently computed by the simulation; stored in the buffer.



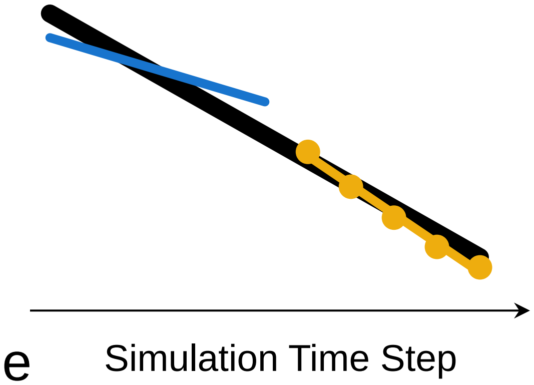
Our *in situ* approach: Compare linear fits

Cheap to compute and update within the simulation as it's running.

Consider 2 hypotheses:

H_0: **One line** fits best.

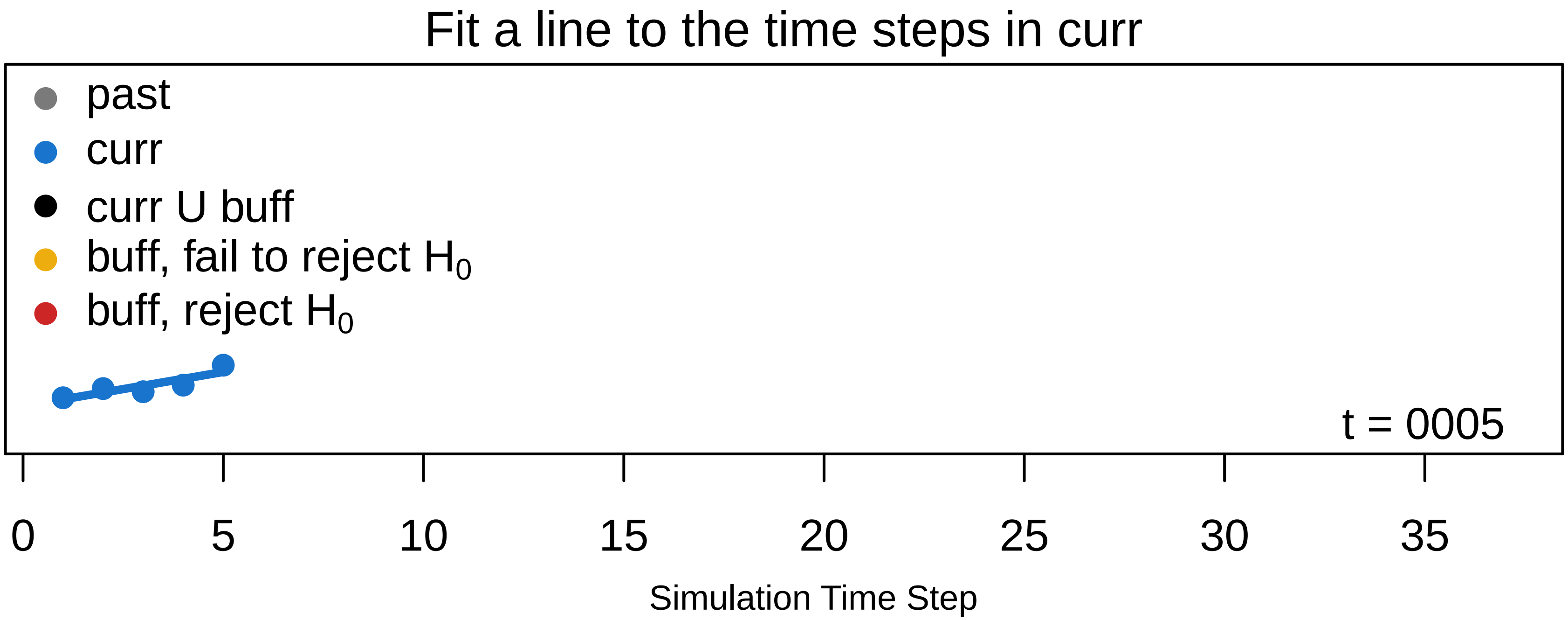
H_1: **Two lines** (**curr** + **buff**) fit best.



Use a **modified *F*-statistic** at some α level to decide when to reject H_0.

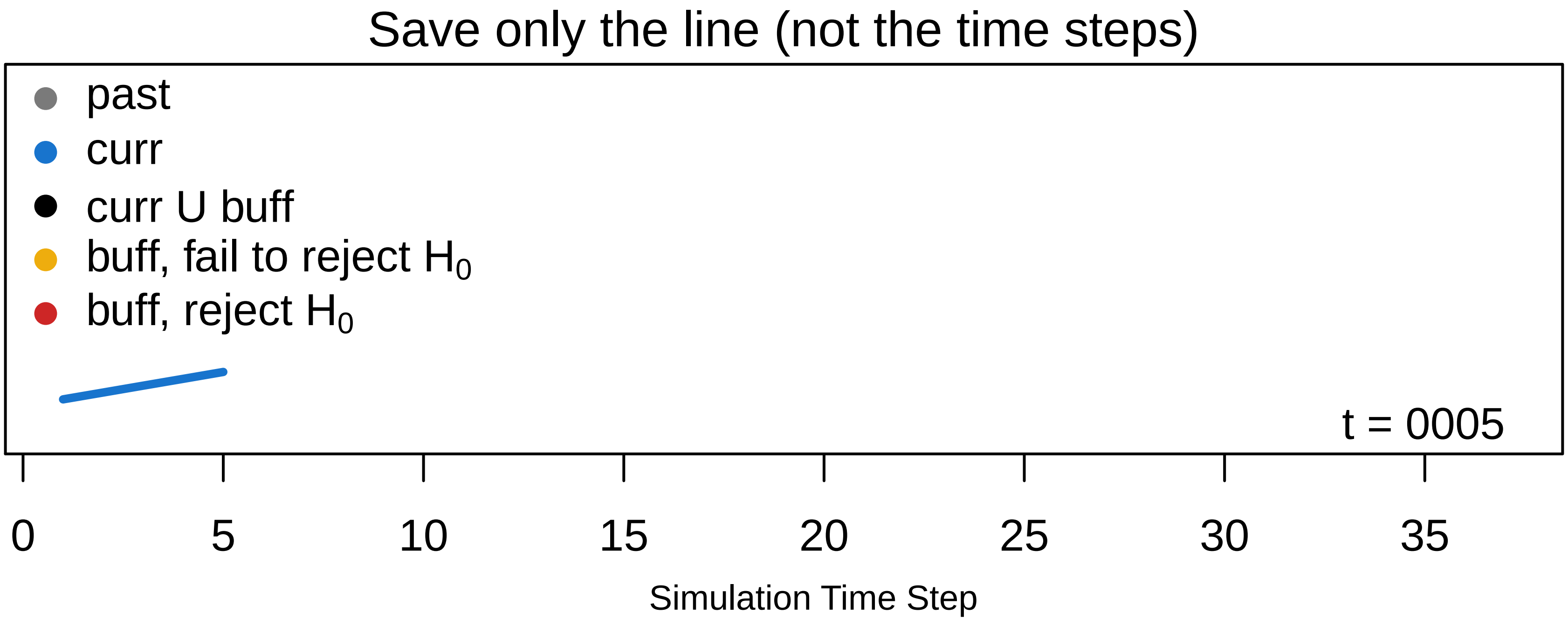
Our *in situ* approach: Compare linear fits

A toy example: Piecewise linear data with noise. Buffer size *B* = 5.



Our *in situ* approach: Compare linear fits

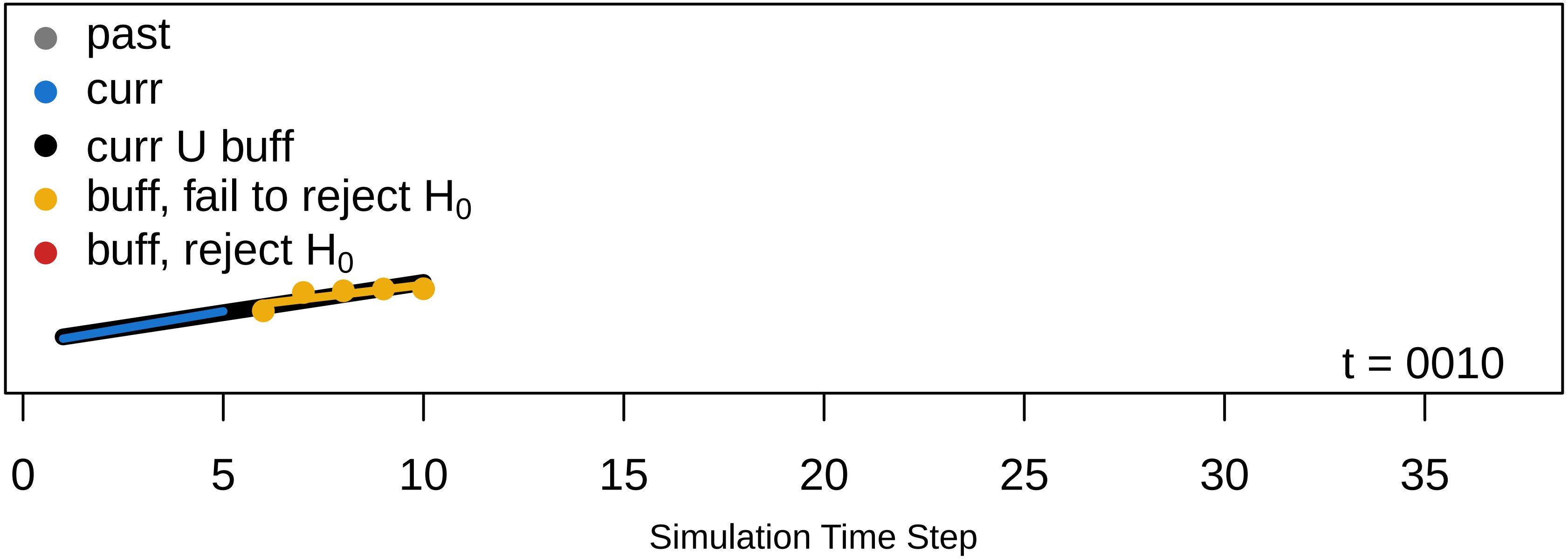
A toy example: Piecewise linear data with noise. Buffer size $B = 5$.



Our *in situ* approach: Compare linear fits

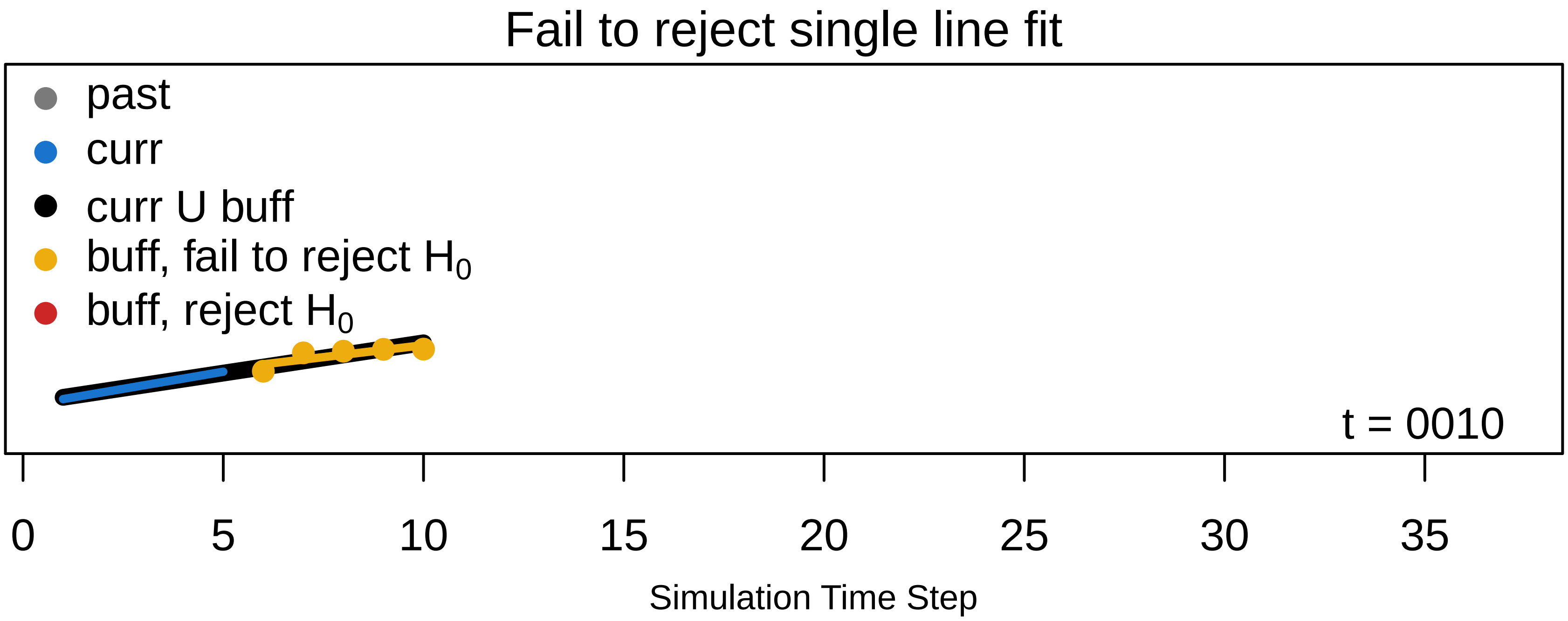
A toy example: Piecewise linear data with noise. Buffer size *B* = 5.

Acquire the next *B* time steps; fit and compare 3 lines



Our *in situ* approach: Compare linear fits

A toy example: Piecewise linear data with noise. Buffer size *B* = 5.

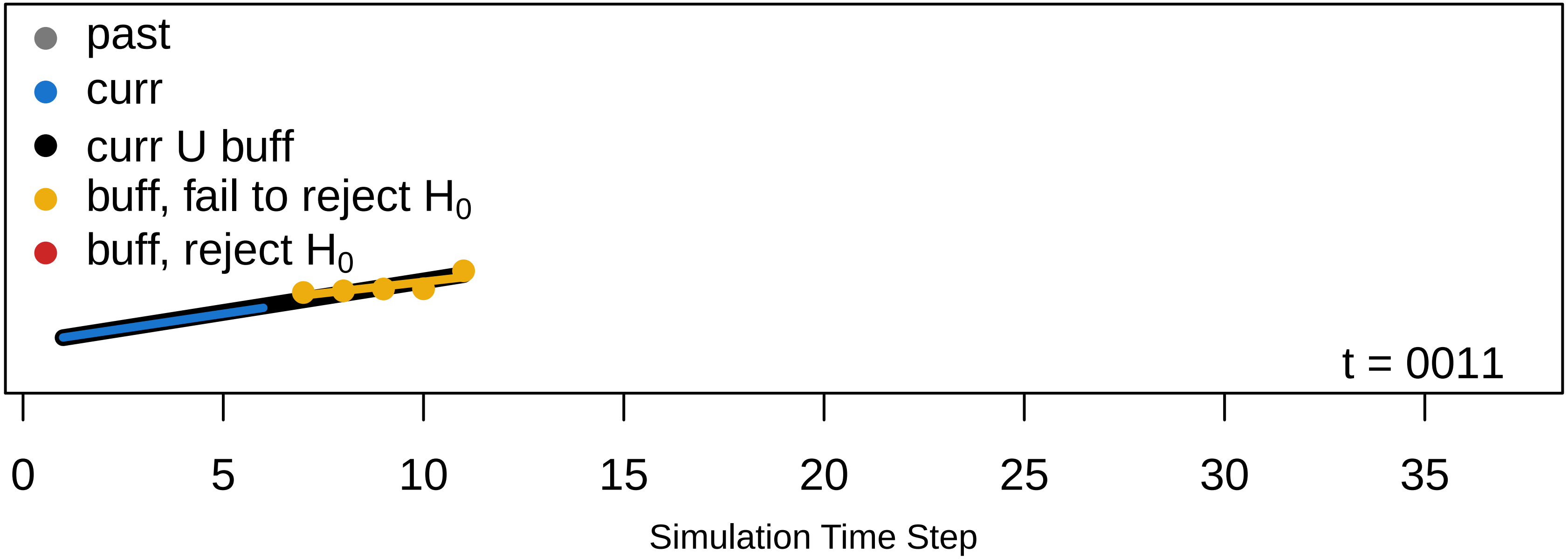


Our *in situ* approach: Compare linear fits

A toy example: Piecewise linear data with noise. Buffer size $B = 5$.

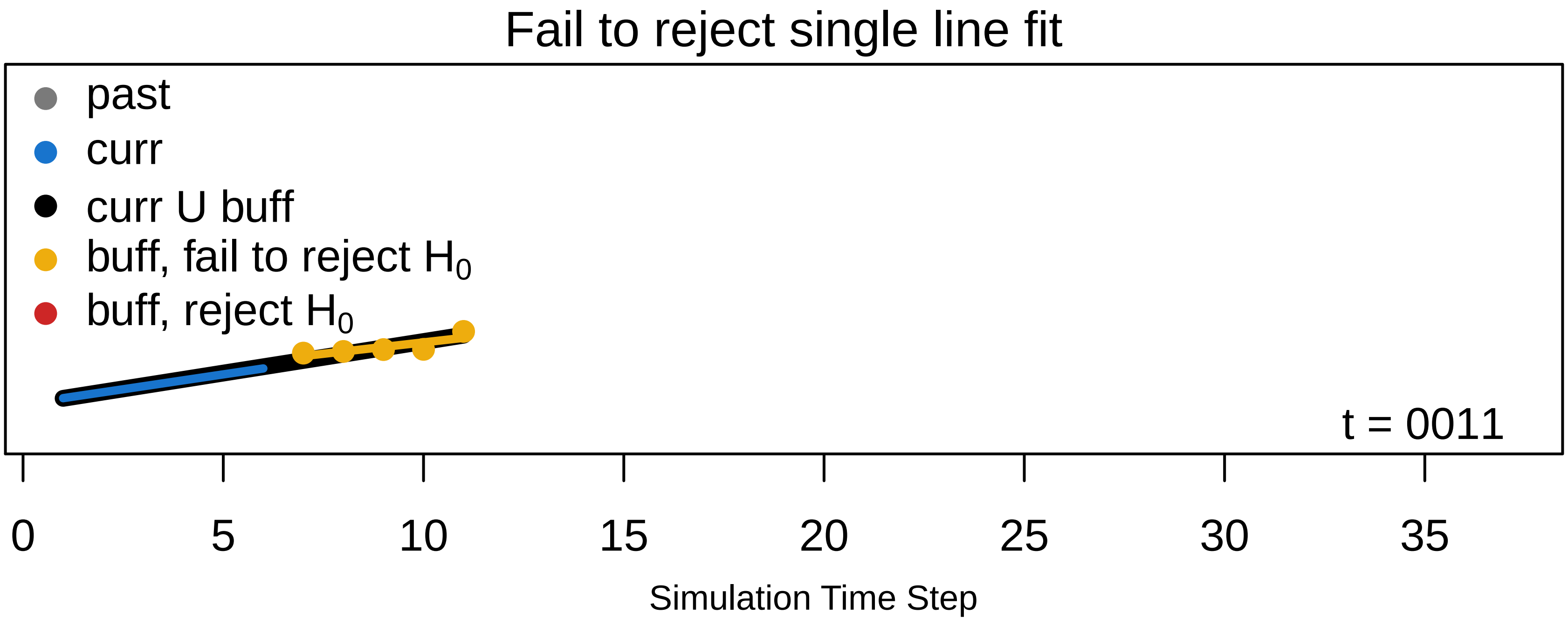
And throw it away!

Add new time step to buff, move oldest one to curr, update the 3 lines



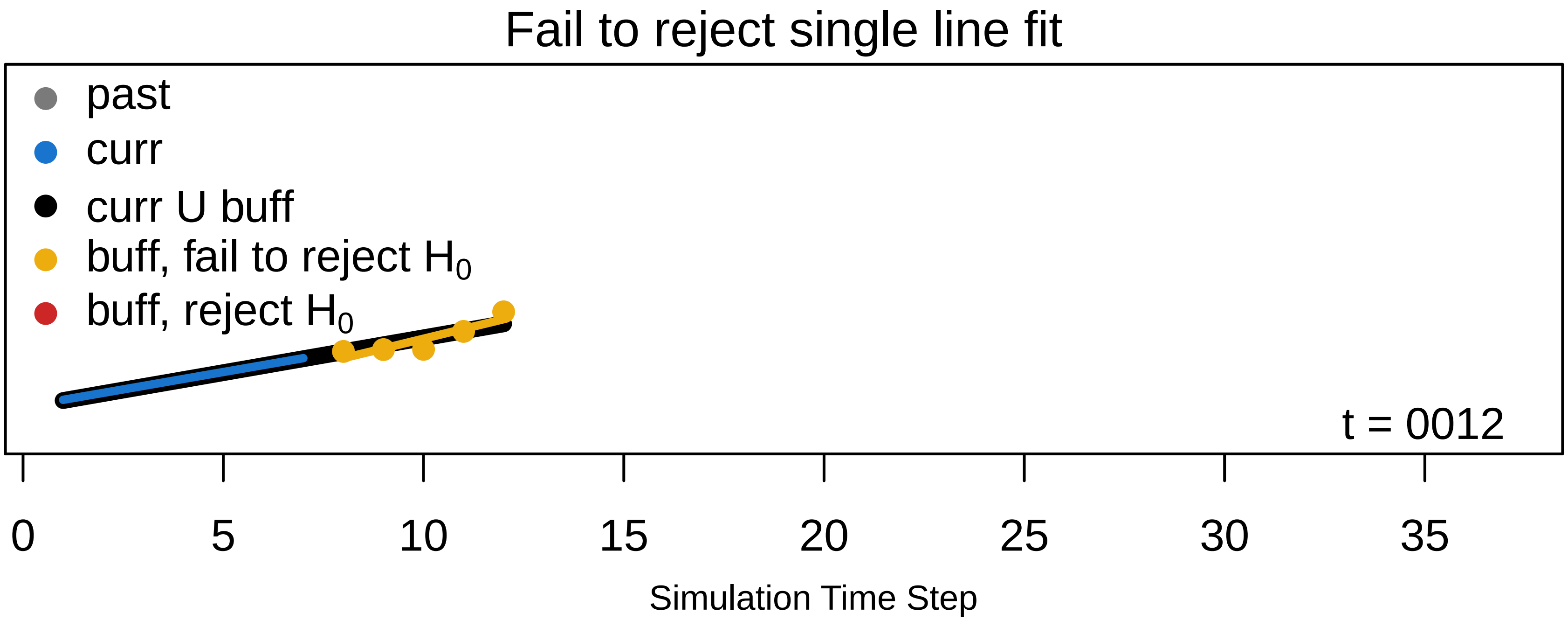
Our *in situ* approach: Compare linear fits

A toy example: Piecewise linear data with noise. Buffer size $B = 5$.



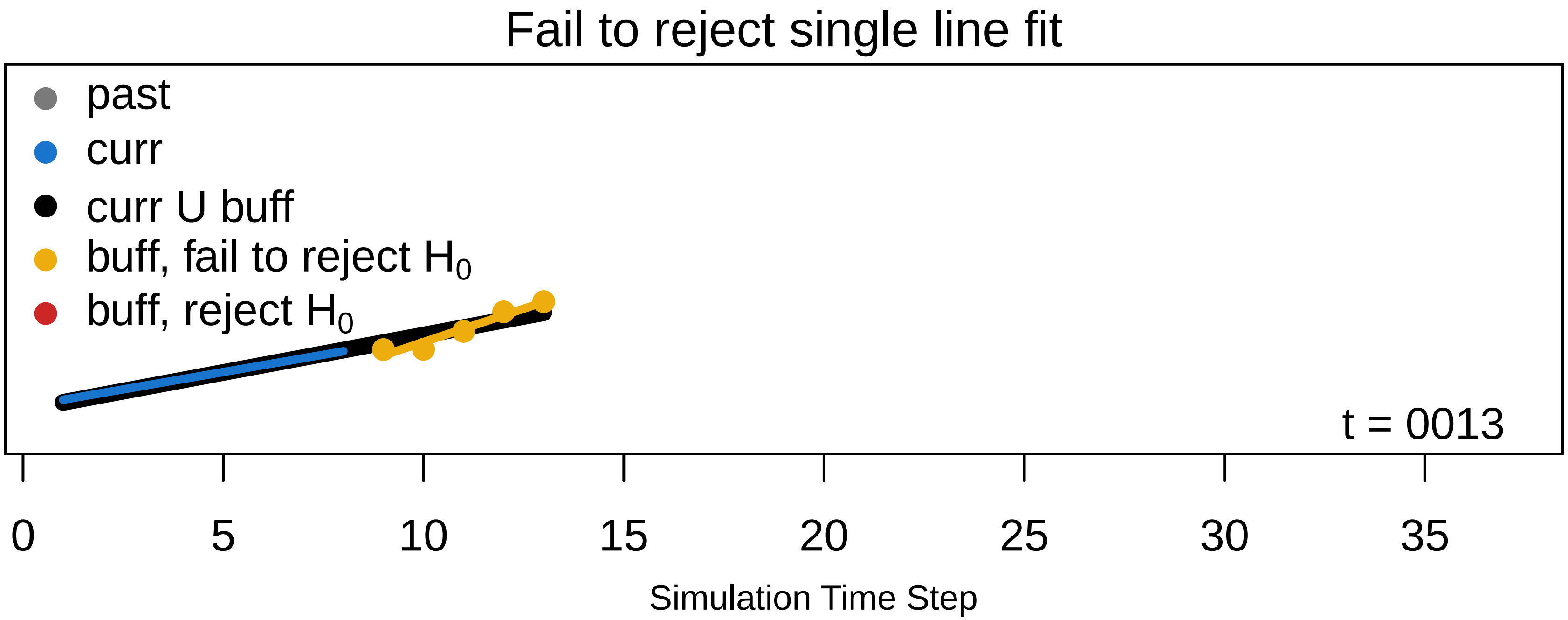
Our *in situ* approach: Compare linear fits

A toy example: Piecewise linear data with noise. Buffer size *B* = 5.



Our *in situ* approach: Compare linear fits

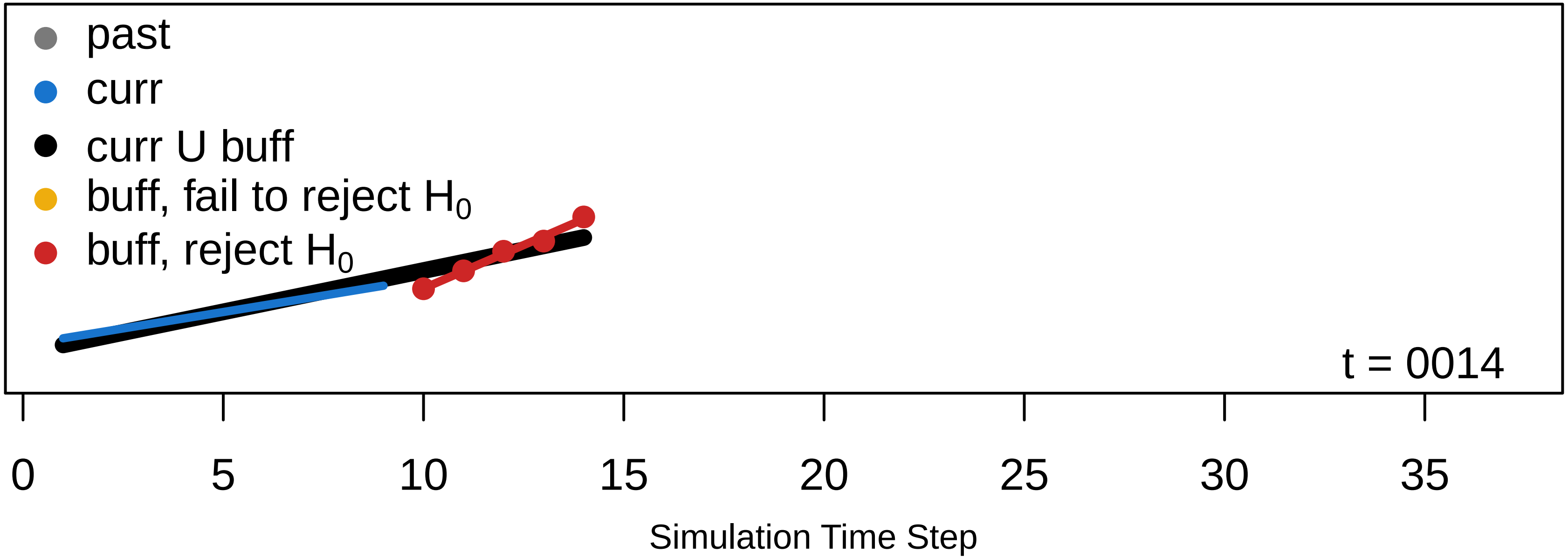
A toy example: Piecewise linear data with noise. Buffer size *B* = 5.



Our *in situ* approach: Compare linear fits

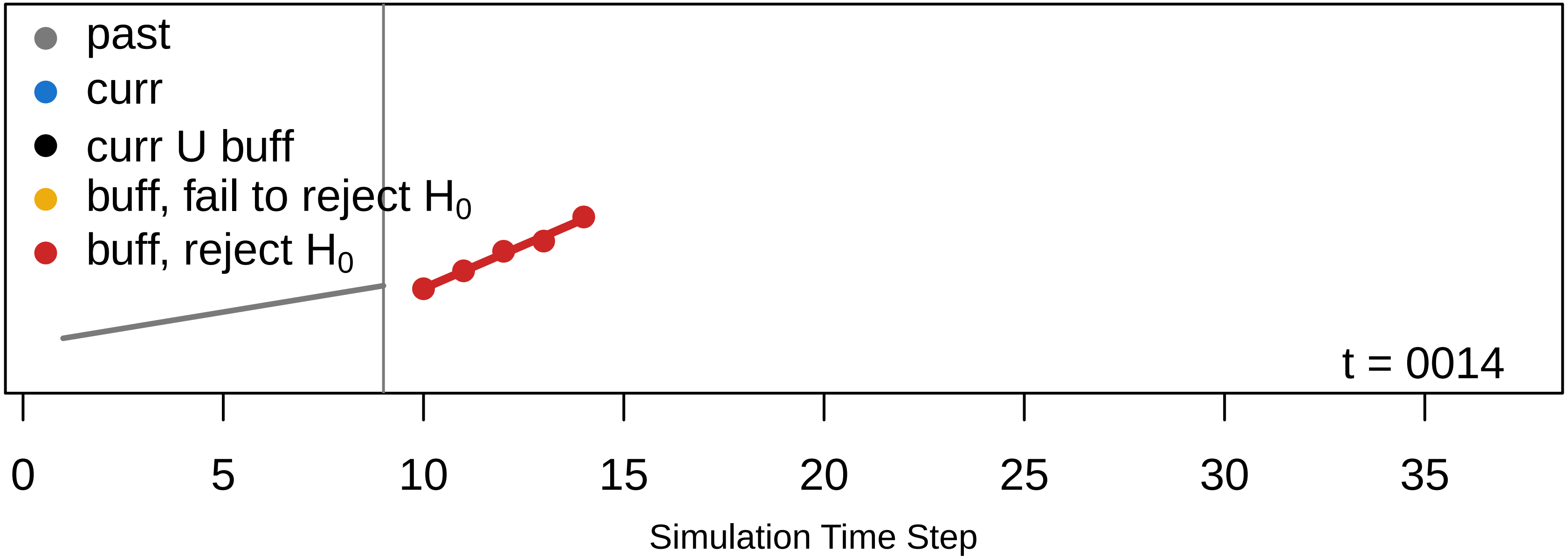
A toy example: Piecewise linear data with noise. Buffer size *B* = 5.

Reject single line fit



Our *in situ* approach: Compare linear fits

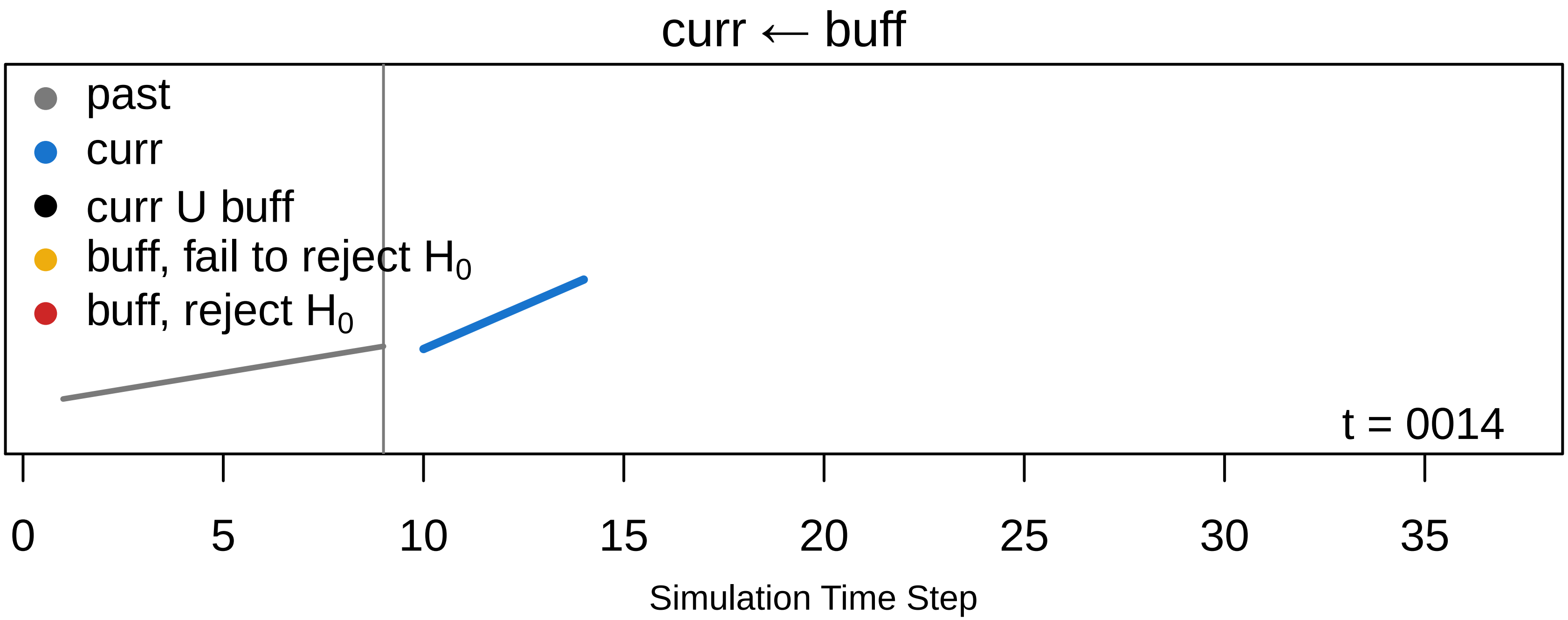
A toy example: Piecewise linear data with noise. Buffer size B = 5.



past ← past ∪ curr, write things to disk

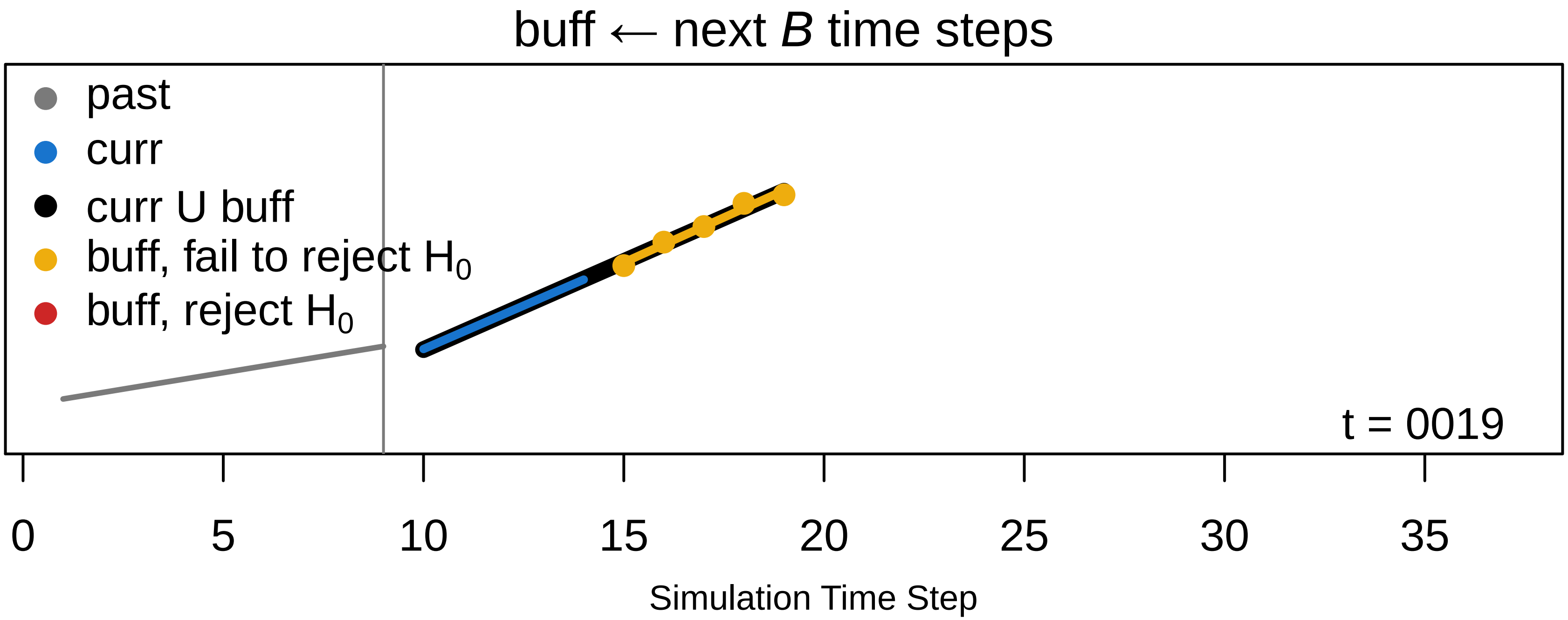
Our *in situ* approach: Compare linear fits

A toy example: Piecewise linear data with noise. Buffer size *B* = 5.



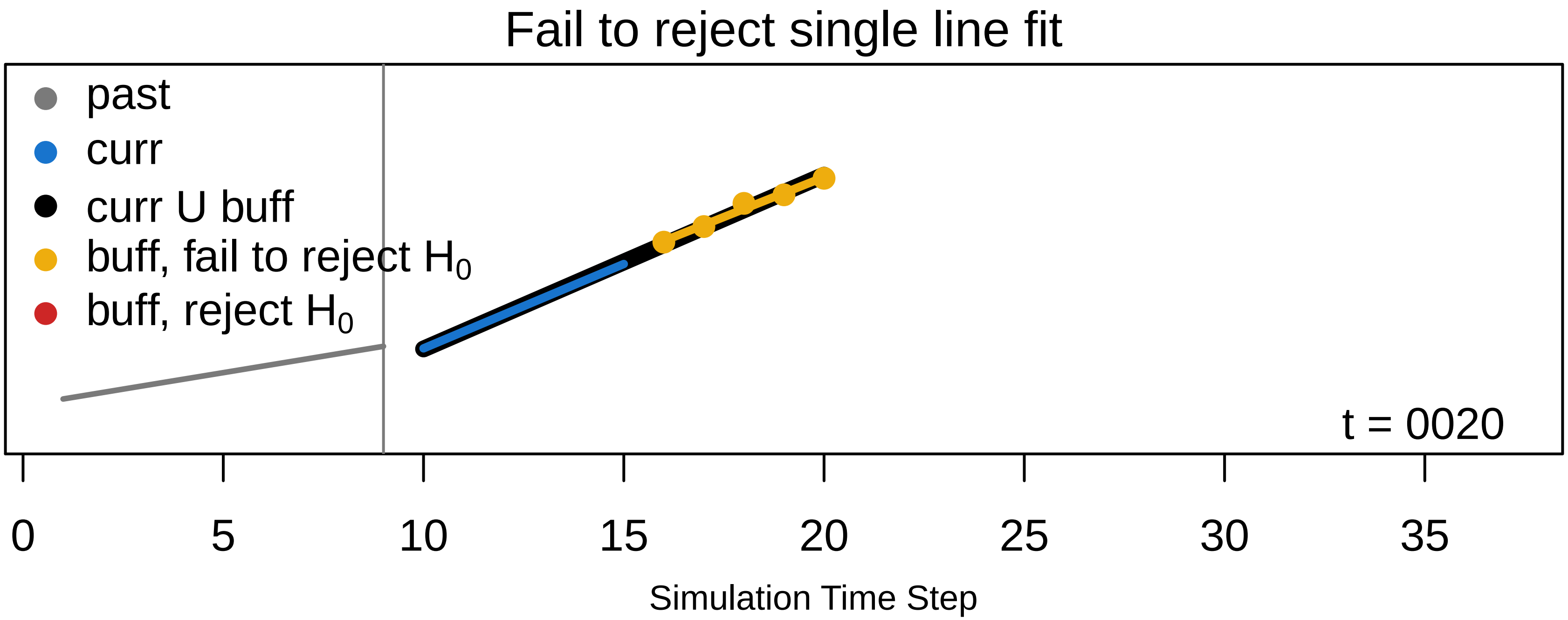
Our *in situ* approach: Compare linear fits

A toy example: Piecewise linear data with noise. Buffer size B = 5.



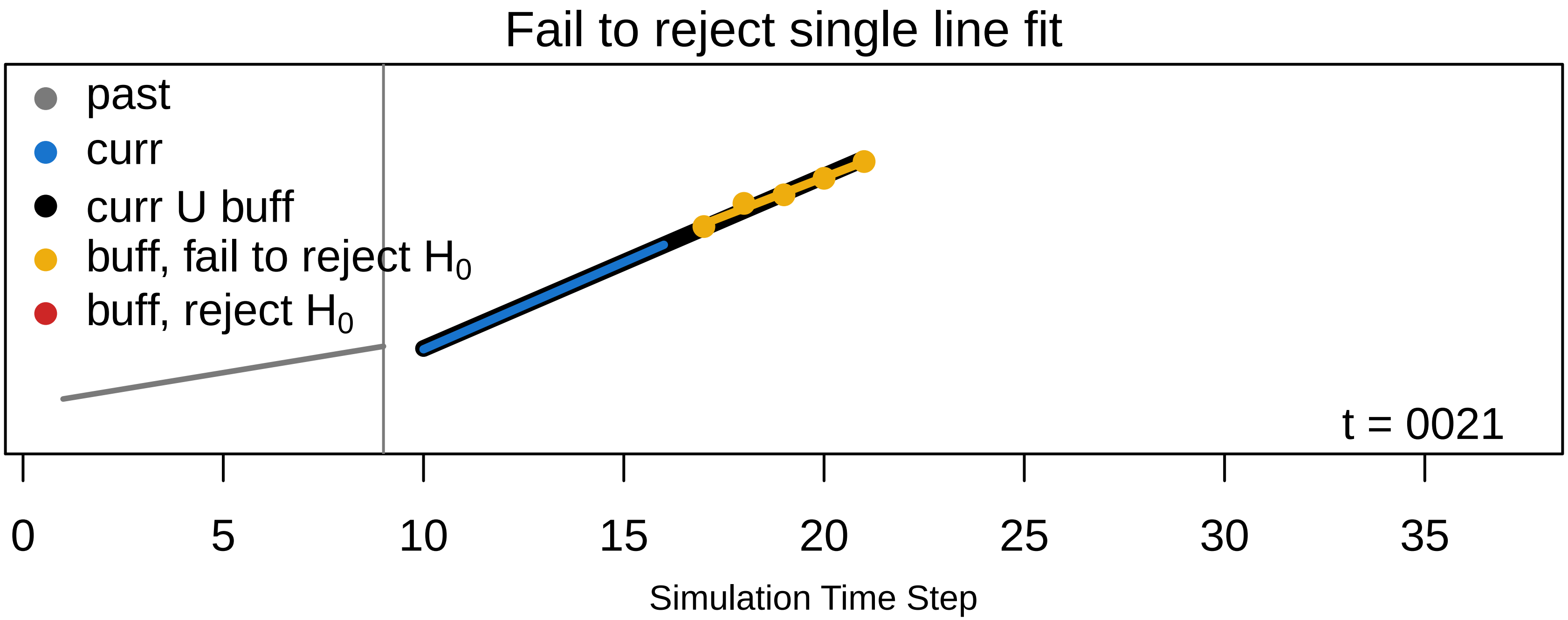
Our *in situ* approach: Compare linear fits

A toy example: Piecewise linear data with noise. Buffer size *B* = 5.



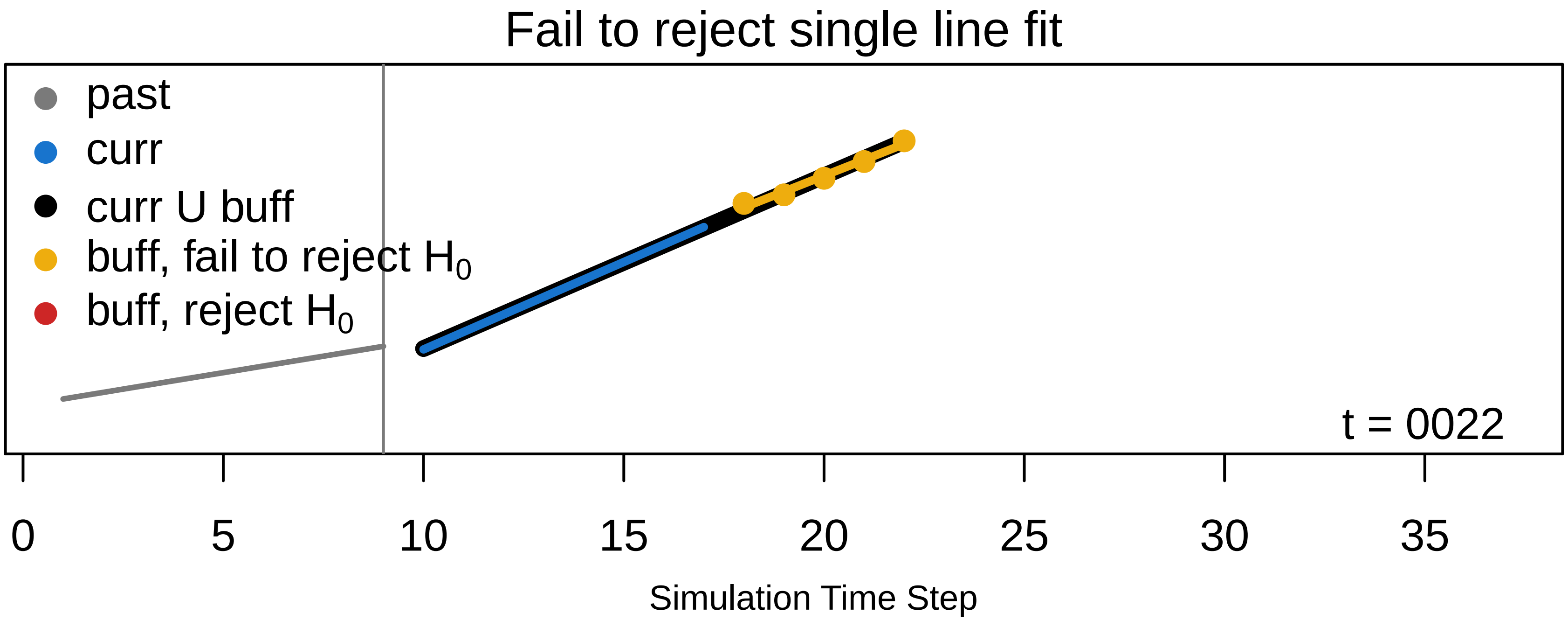
Our *in situ* approach: Compare linear fits

A toy example: Piecewise linear data with noise. Buffer size *B* = 5.



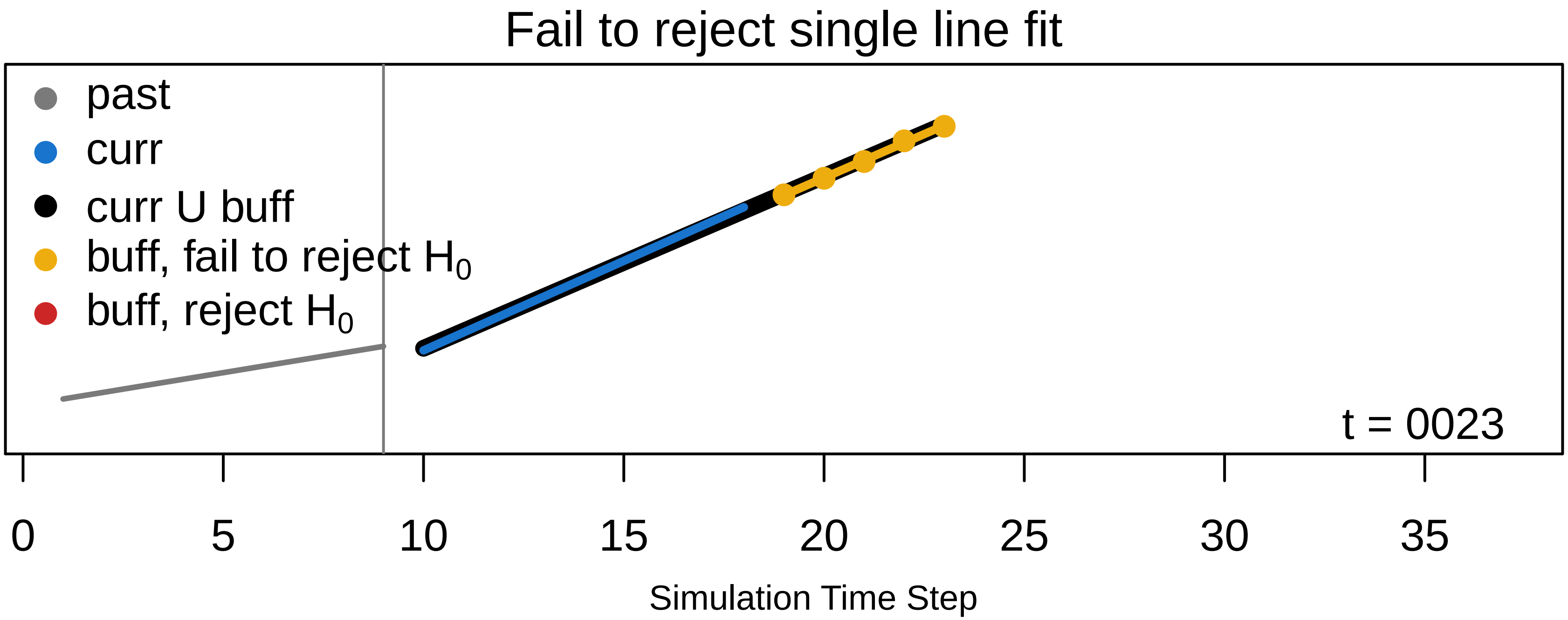
Our *in situ* approach: Compare linear fits

A toy example: Piecewise linear data with noise. Buffer size *B* = 5.



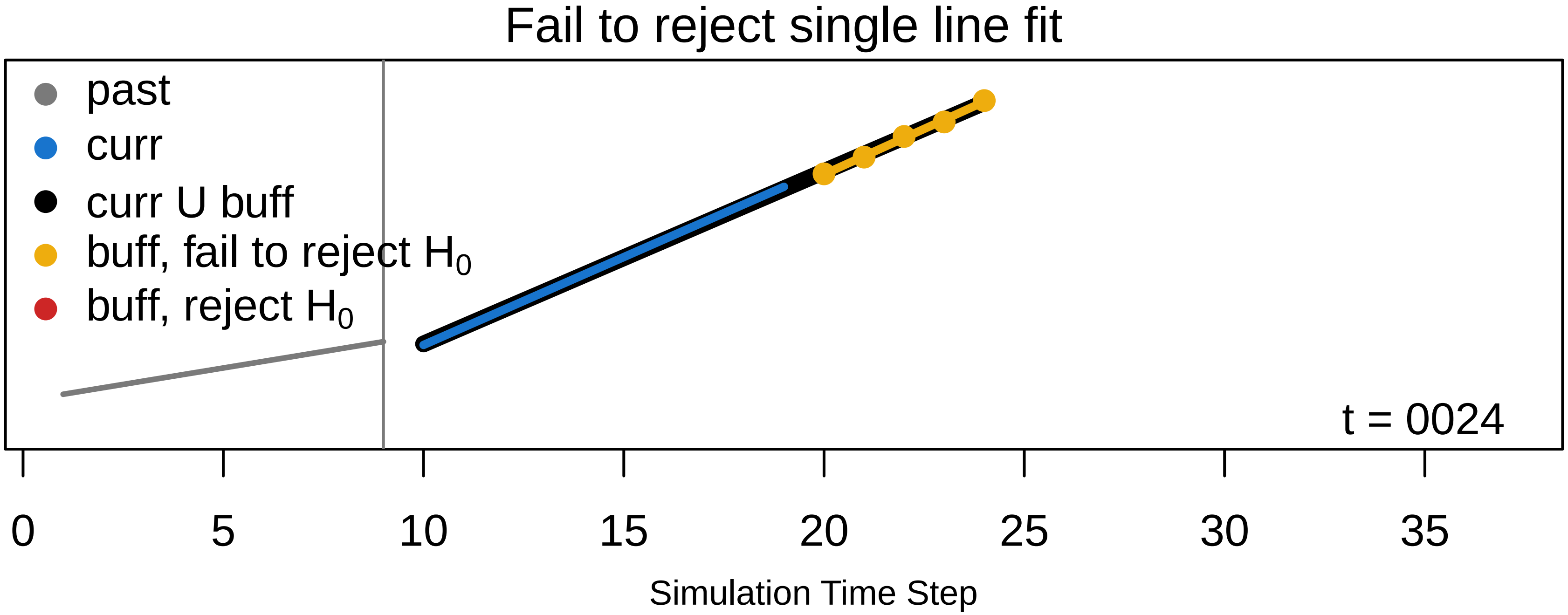
Our *in situ* approach: Compare linear fits

A toy example: Piecewise linear data with noise. Buffer size B = 5.



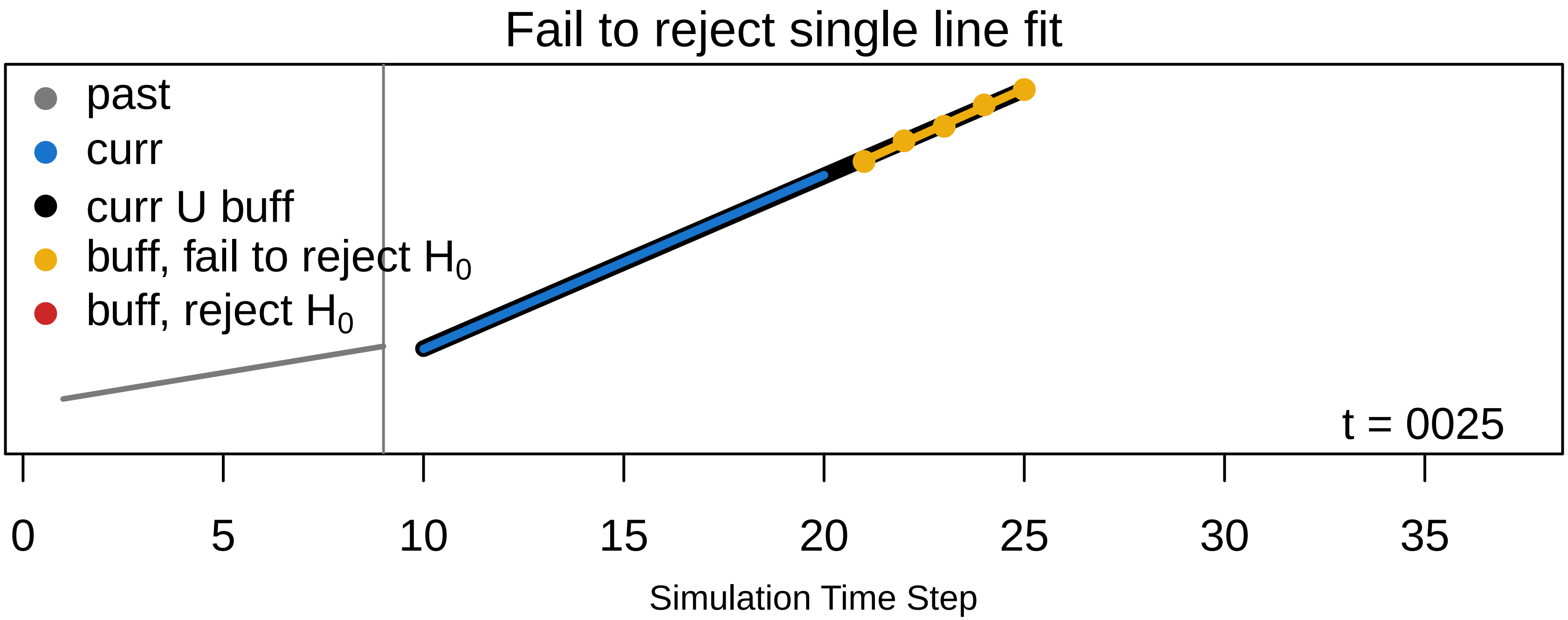
Our *in situ* approach: Compare linear fits

A toy example: Piecewise linear data with noise. Buffer size *B* = 5.



Our *in situ* approach: Compare linear fits

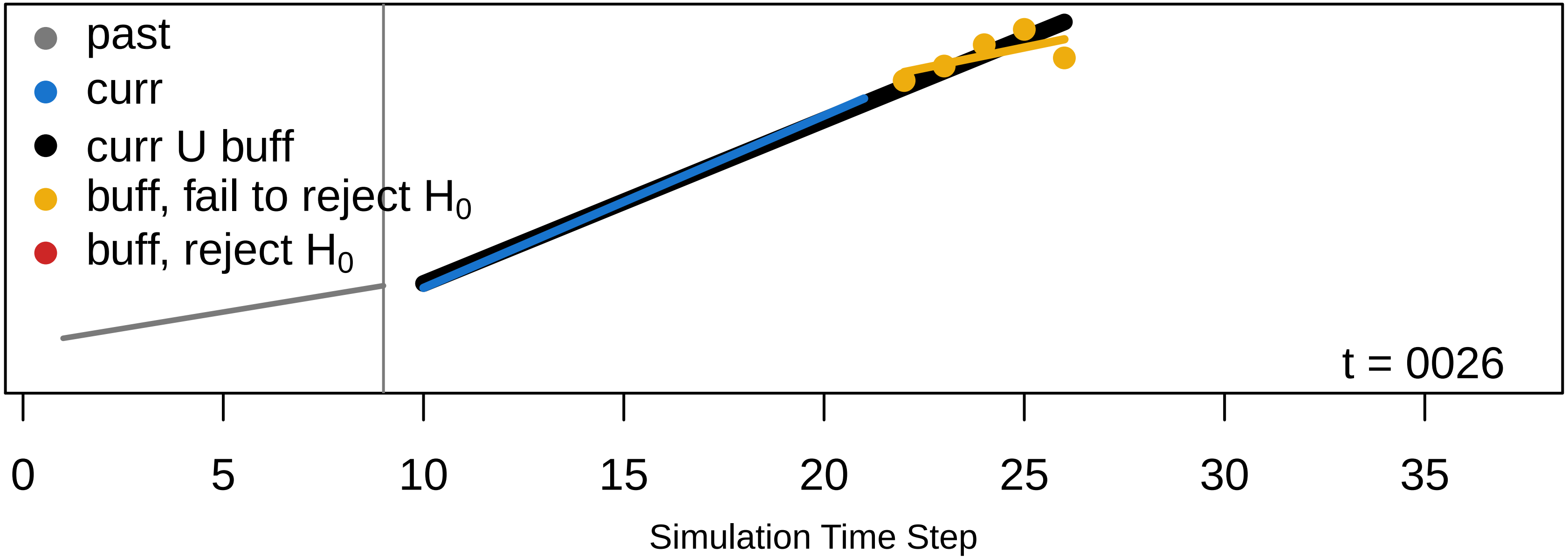
A toy example: Piecewise linear data with noise. Buffer size *B* = 5.



Our *in situ* approach: Compare linear fits

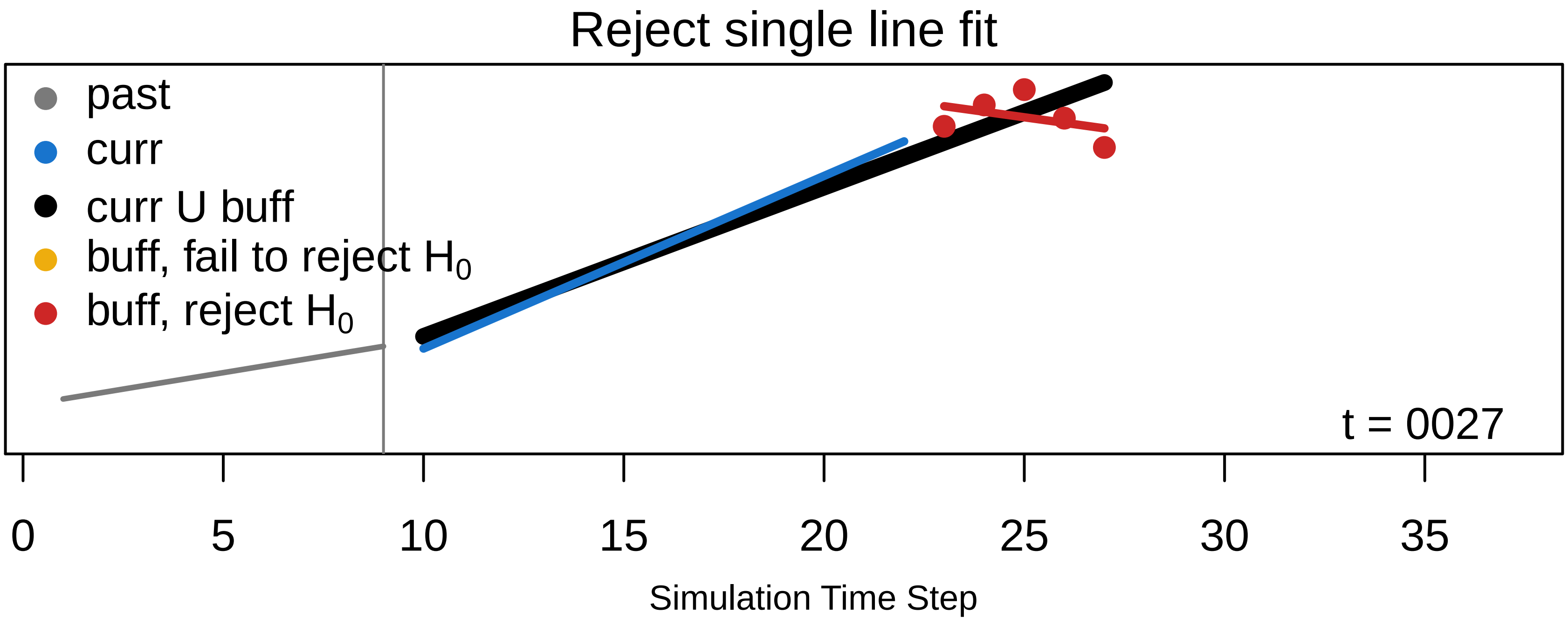
A toy example: Piecewise linear data with noise. Buffer size *B* = 5.

Fail to reject single line fit



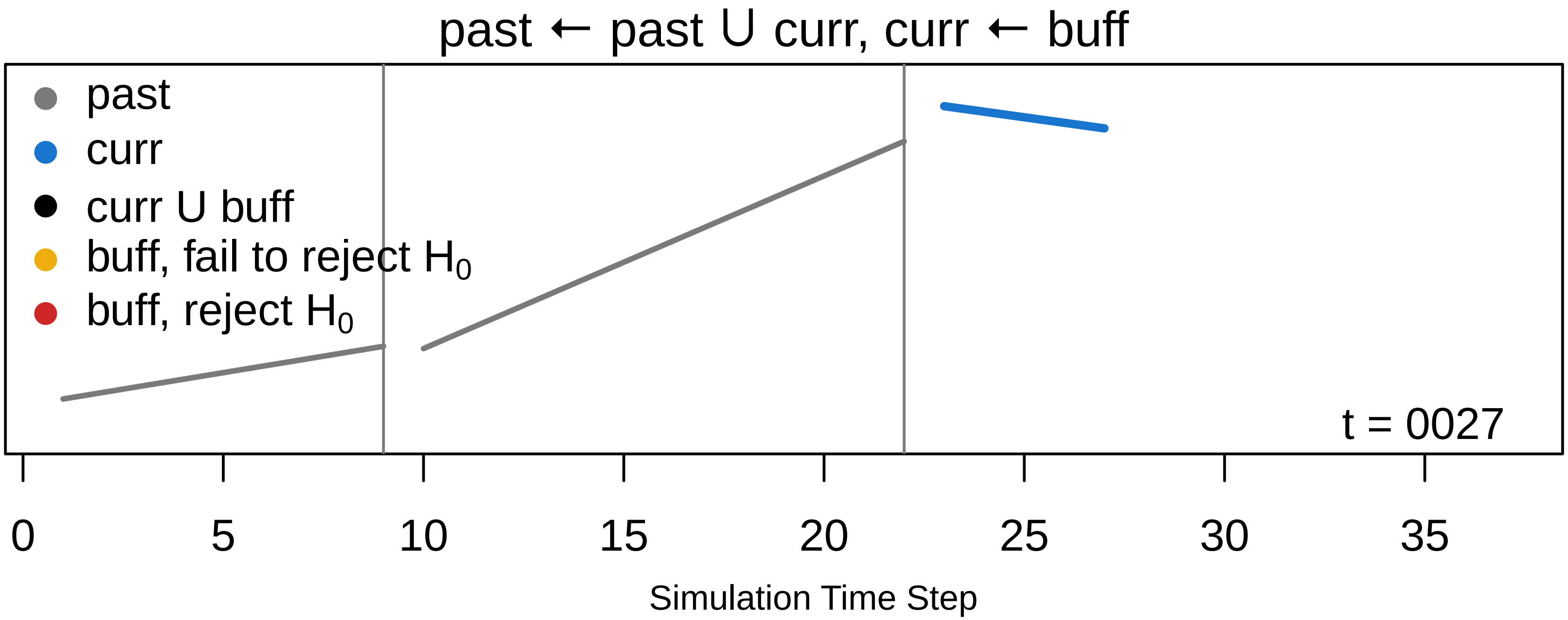
Our *in situ* approach: Compare linear fits

A toy example: Piecewise linear data with noise. Buffer size B = 5.



Our *in situ* approach: Compare linear fits

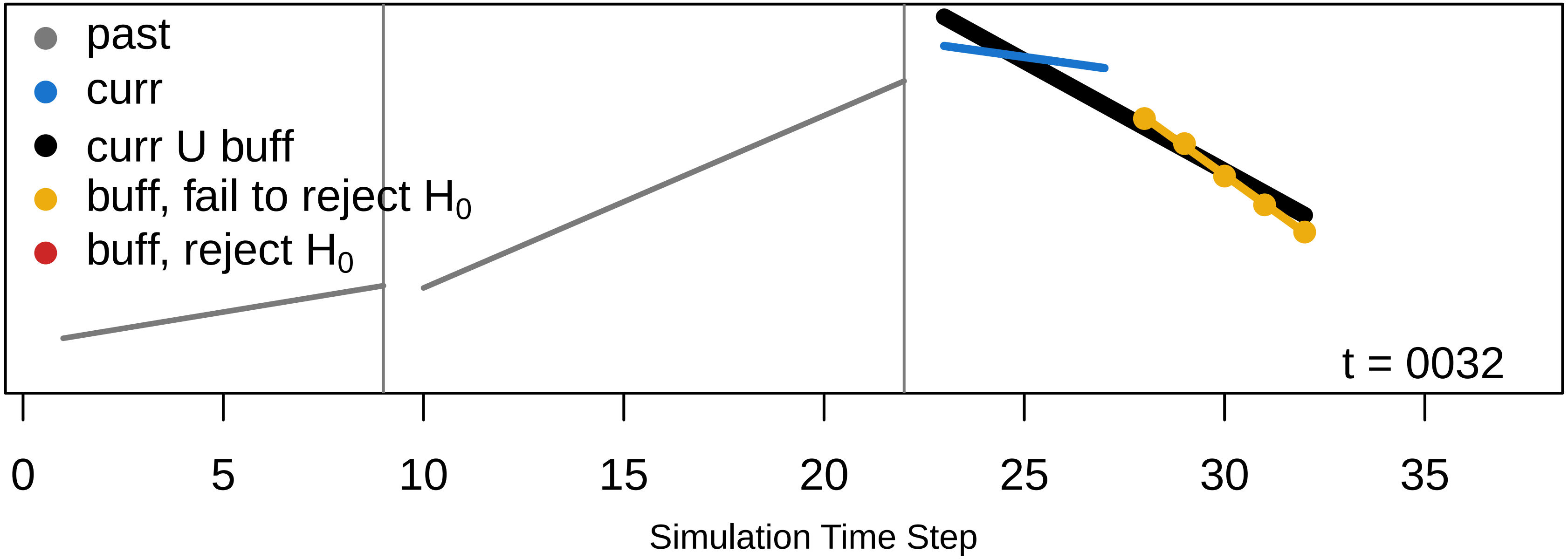
A toy example: Piecewise linear data with noise. Buffer size B = 5.



Our *in situ* approach: Compare linear fits

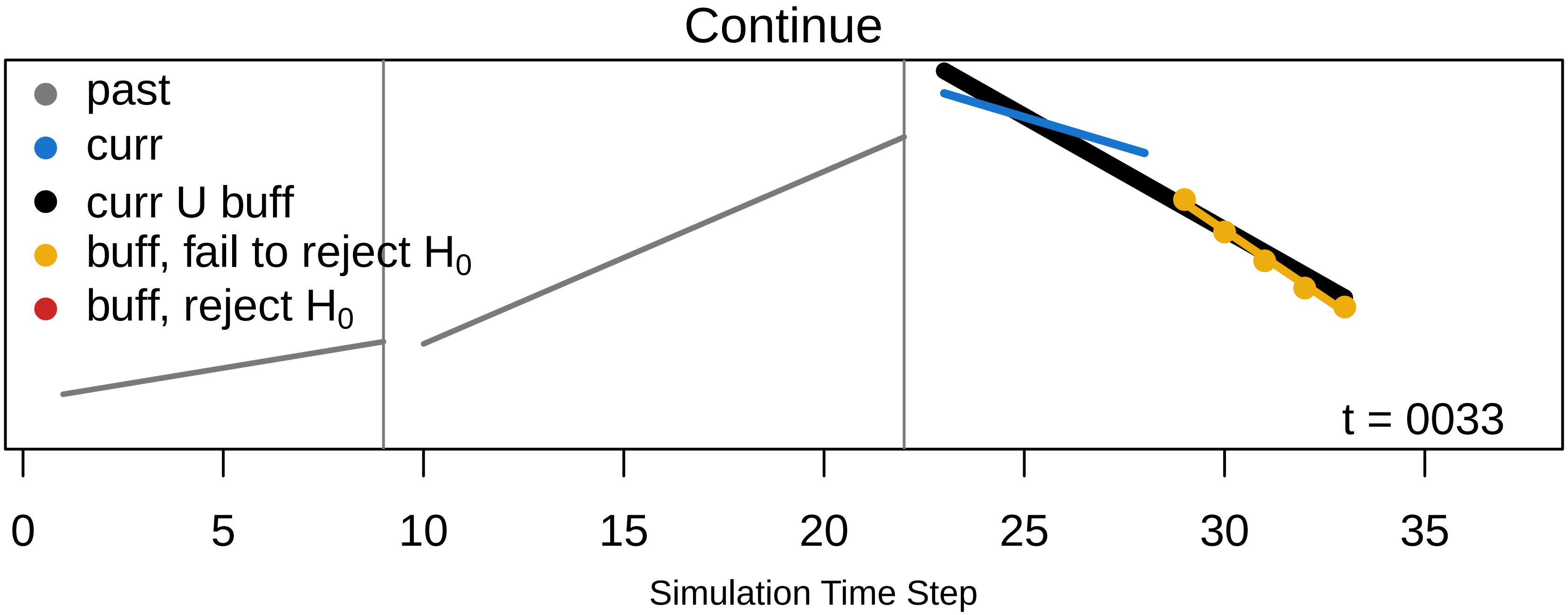
A toy example: Piecewise linear data with noise. Buffer size B = 5.

buff ← next B time steps



Our *in situ* approach: Compare linear fits

A toy example: Piecewise linear data with noise. Buffer size *B* = 5.



Our *in situ* approach: Compare linear fits

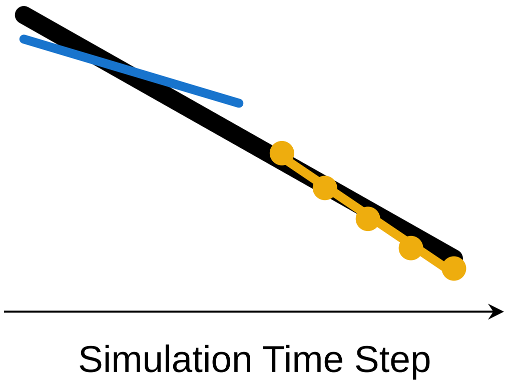
We capture each of the 3 lines with a set of **sufficient statistics**:

$$T_{\bullet},\ \sum t_i,\ \sum t_i^2,\ \sum y_i,\ \sum y_i^2,\ \sum t_i y_i$$

Number of time steps
in the line

Time

Response

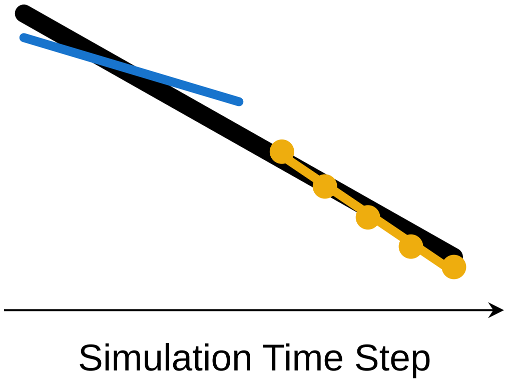


Our *in situ* approach: Compare linear fits

We capture each of the 3 lines with a set of **sufficient statistics**:

$$T_\bullet, \sum t_i, \sum t_i^2, \sum y_i, \sum y_i^2, \sum t_i y_i$$

- Update these in constant time, O(1), as the simulation progresses.
- Use to compute the **modified *F*-statistic** for our hypothesis test.
- Use to construct a linear approximation of the entire simulation with known error.



Our modified *F*-statistic

Here’s the standard formulation:

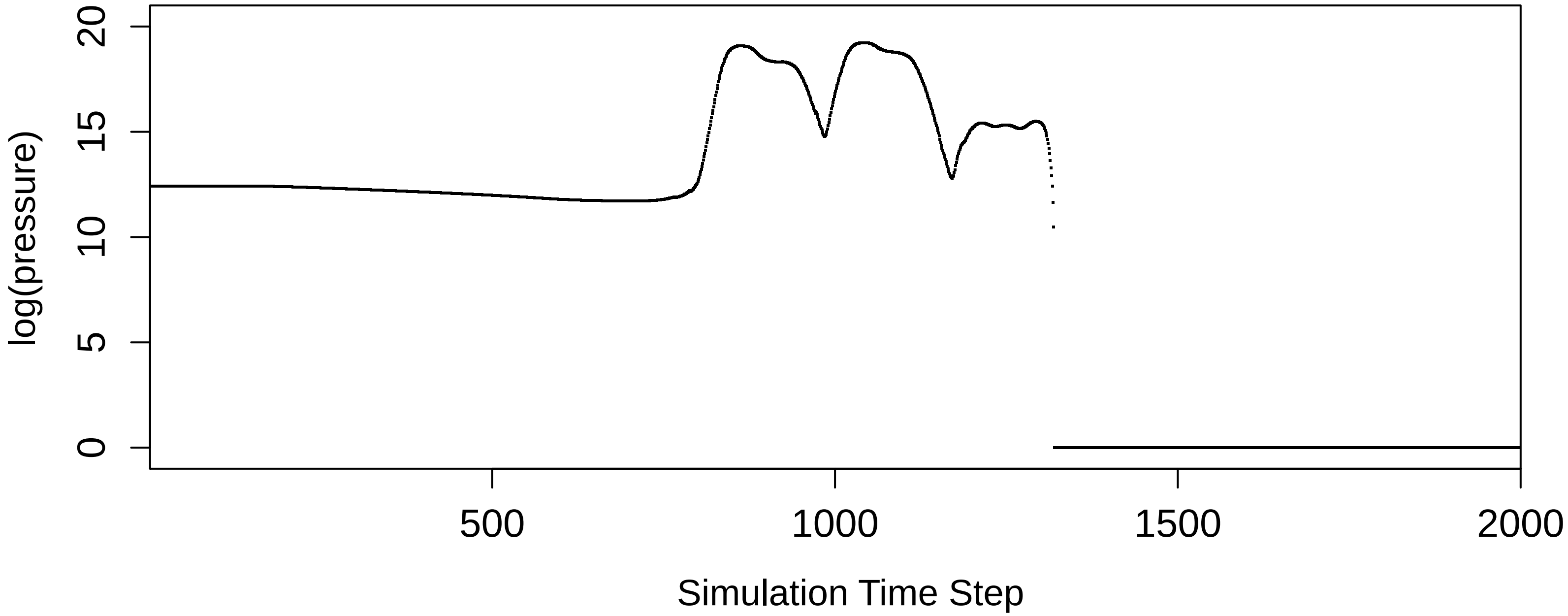
$$F = \frac{\left(\frac{\overset{\text{One line}}{RSS_1} - \overset{\text{Two lines}}{RSS_2}}{p_2 - p_1}\right)}{\left(\frac{RSS_2}{T_{\mathsf{curr}\cup\mathsf{buff}} - p_2}\right)}$$

Number of parameters required for each model

Total number of time steps currently under consideration

Our modified *F*-statistic

But this can reject H_0 when both **curr** and **buff** have extremely low RSS, which is common in these computer simulations.



Our modified *F*-statistic

So we add a “nugget” δ^2, scaled by $T_{\text{curr U buff}}$, to have the effect of adding white noise and encouraging less (or smarter) rejection.

$$F = \frac{\left(\dfrac{RSS_1 - RSS_2}{p_2 - p_1}\right)}{\left(\dfrac{RSS_2 \longleftarrow + T_{\text{curr}\cup\text{buff}} \times \delta^2}{T_{\text{curr}\cup\text{buff}} - p_2}\right)}$$

Now we have 3 “tuning parameters”: *α*, nugget δ^2, and buffer size *B*.
I’ll come back to this later. But first: A demo!

Demo: Is there water on the moon? NASA finds out!

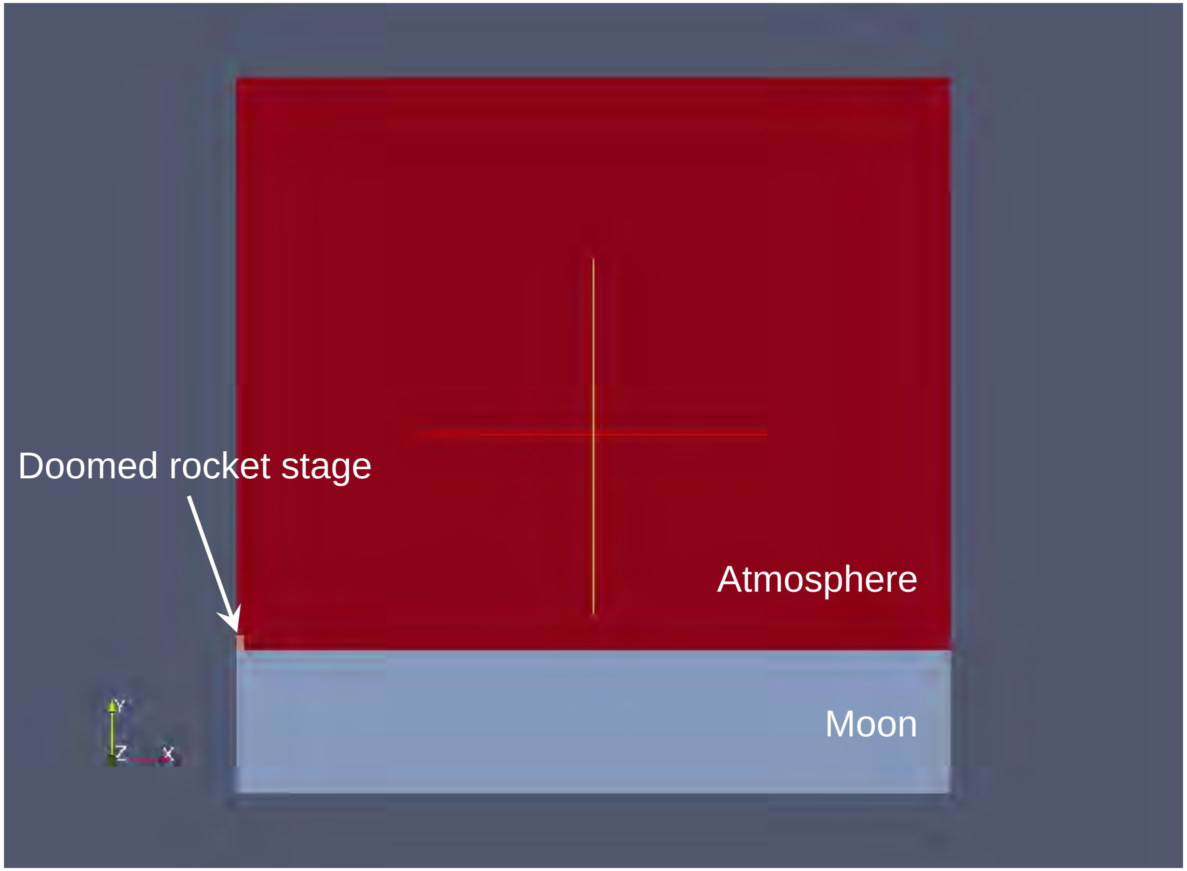
2009 LCROSS Mission: Lunar CRater Observation and Sensing Satellite

www.popularmechanics.com

But before NASA crashed the Moon…

Scientists used RAGE simulations to bound the expected results.

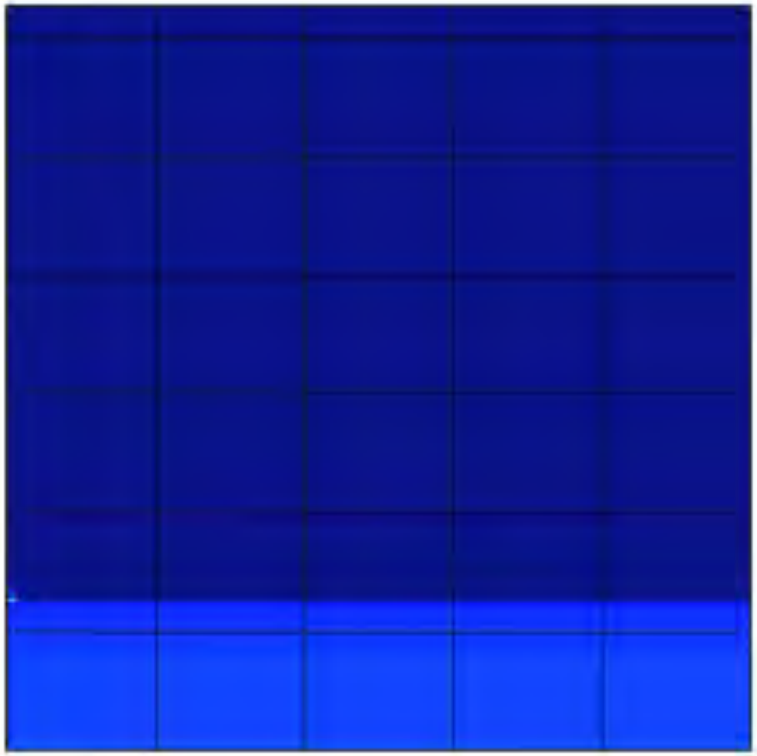
Korycansky et al. 2009



But before NASA crashed the Moon…

Scientists used RAGE simulations to bound the expected results.
Korycansky et al. 2009

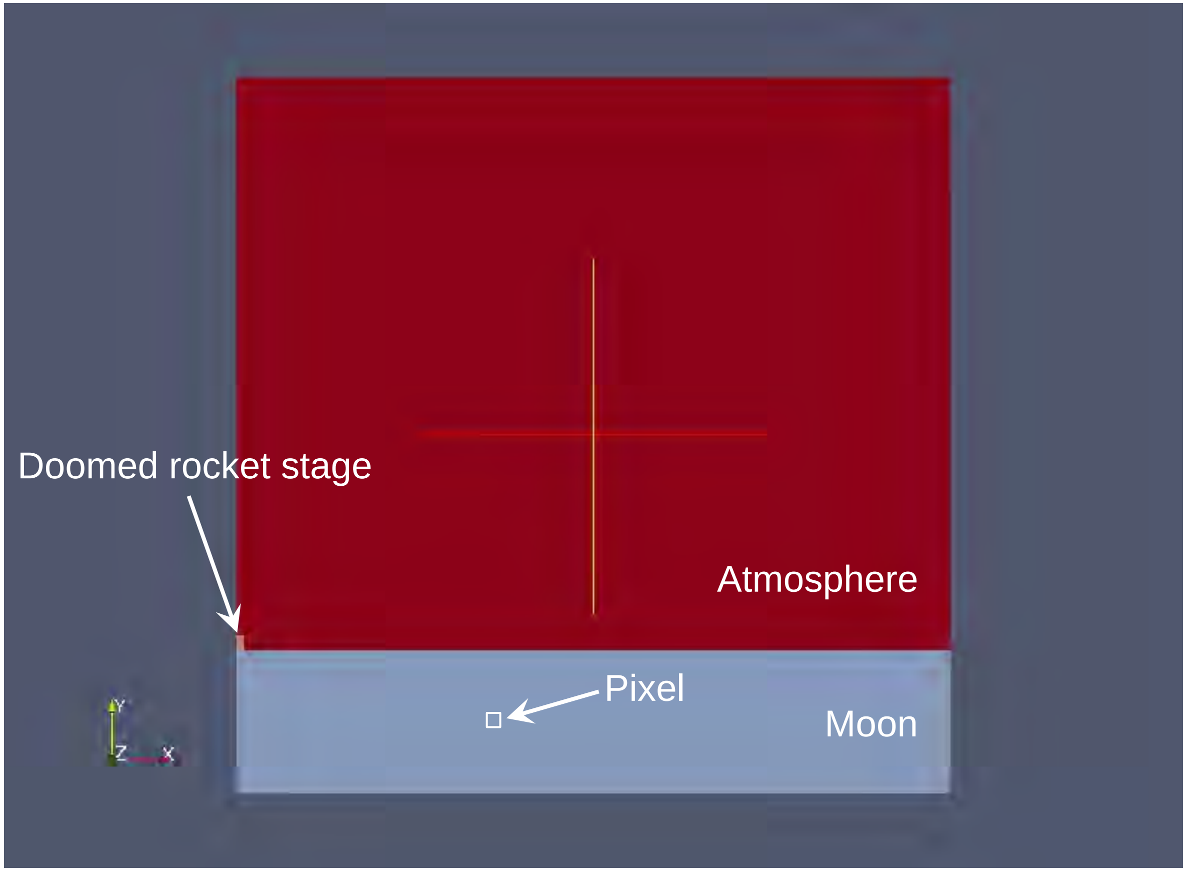
- **RAGE:** A massively parallel Eulerian code used to solve 1D, 2D, or 3D hydrodynamics problems.
  Gittings et al. 2008
- 2000 time steps, ~10 variables in 2D.
- Not a billion billion calculations per second, but a useful testbed.



Pressure

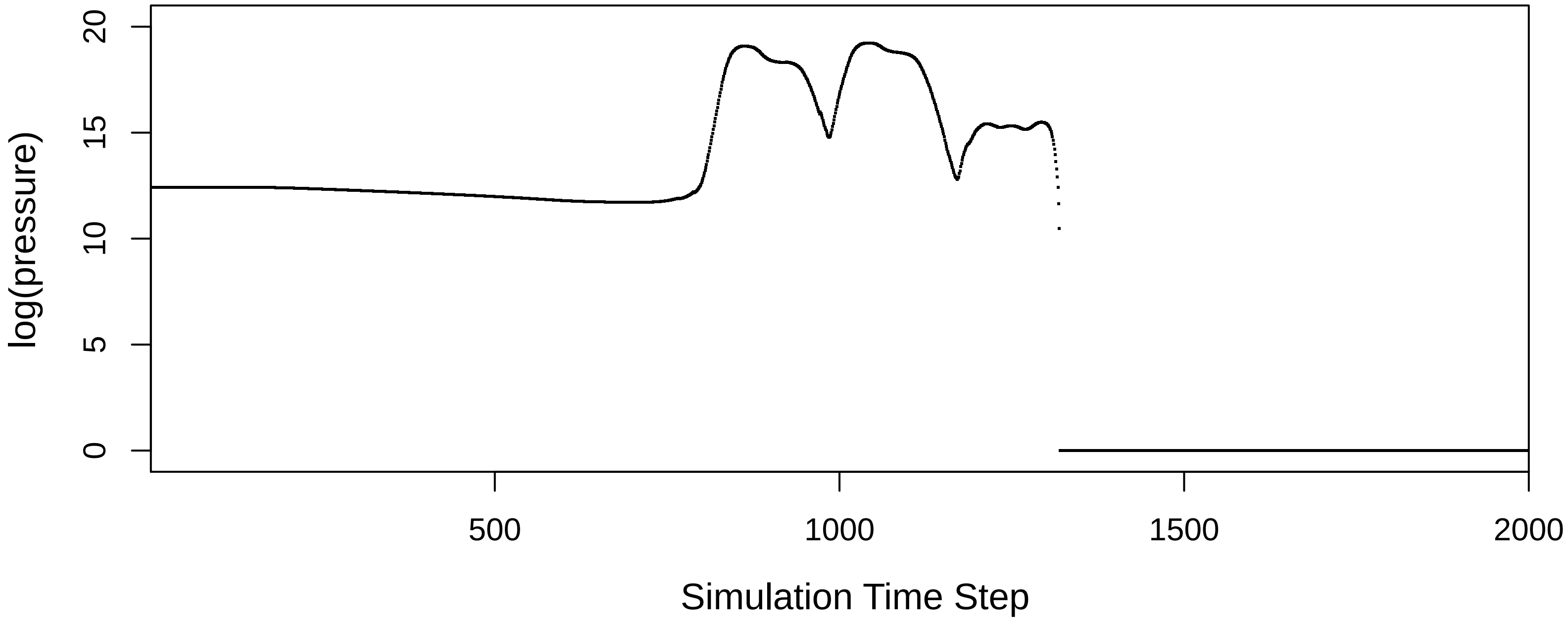
Demonstration with LCROSS simulation

First we'll track a pixel of the pressure variable.



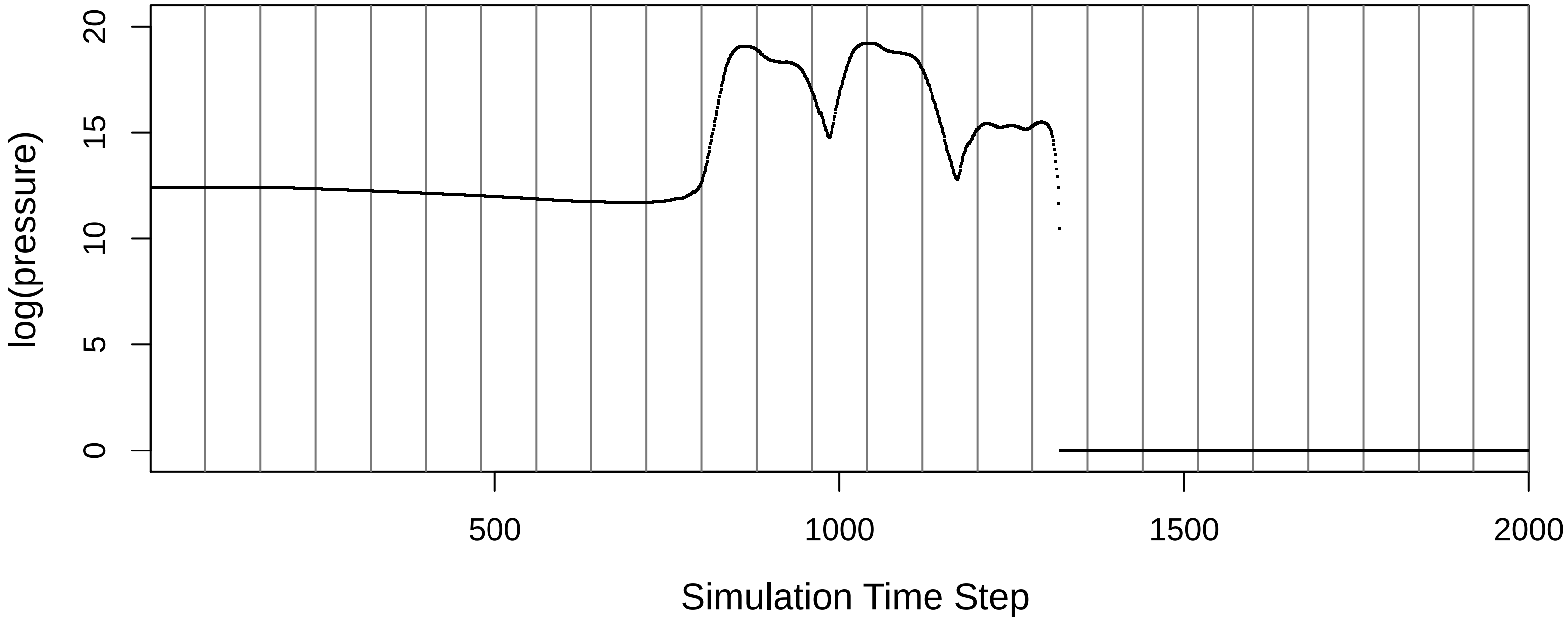
Demonstration with LCROSS simulation (single pixel)

First we'll track a pixel of the pressure variable.



Demonstration with LCROSS simulation (single pixel)

Standard practice: 25 evenly spaced partitions.

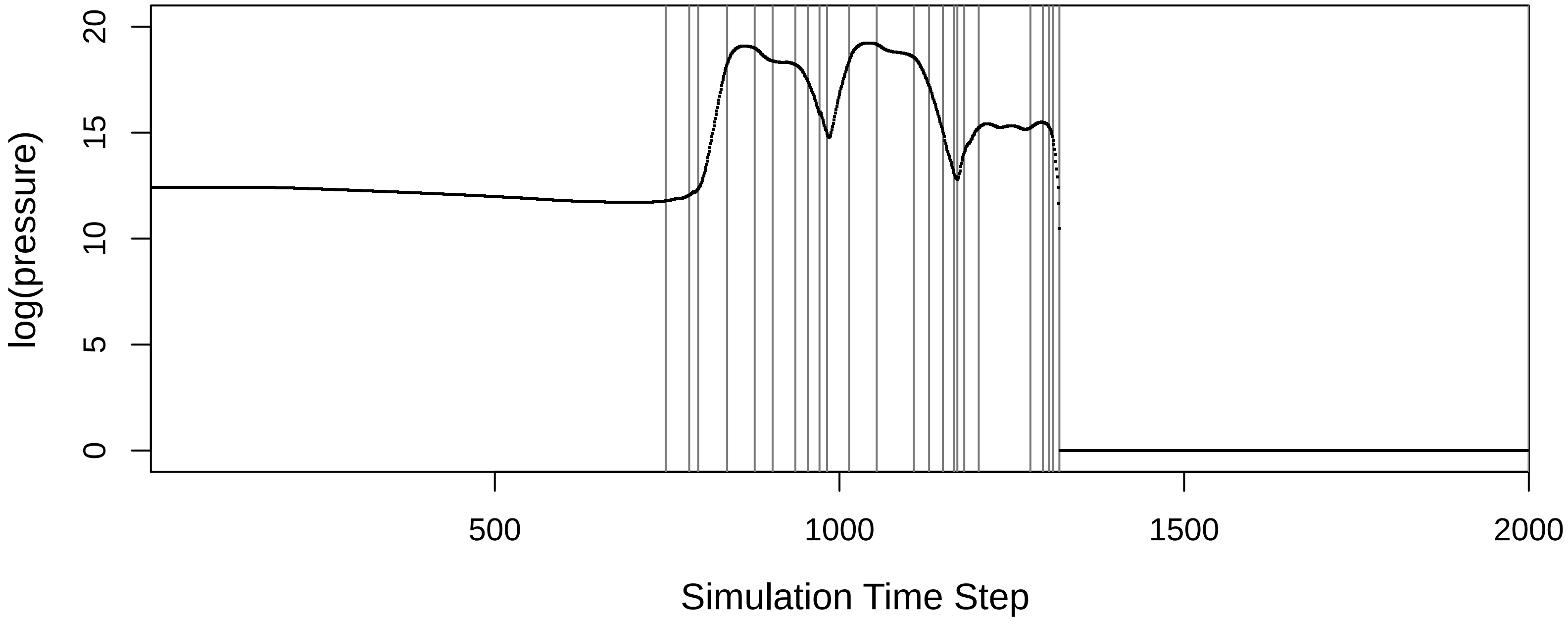


Total RSS: 1140.15

Assuming linear interpolation.

Demonstration with LCROSS simulation (single pixel)

Our approach: 25 partitions selected with $\alpha = 0.001$, $\delta^2 = 0.001$, $B = 5$.



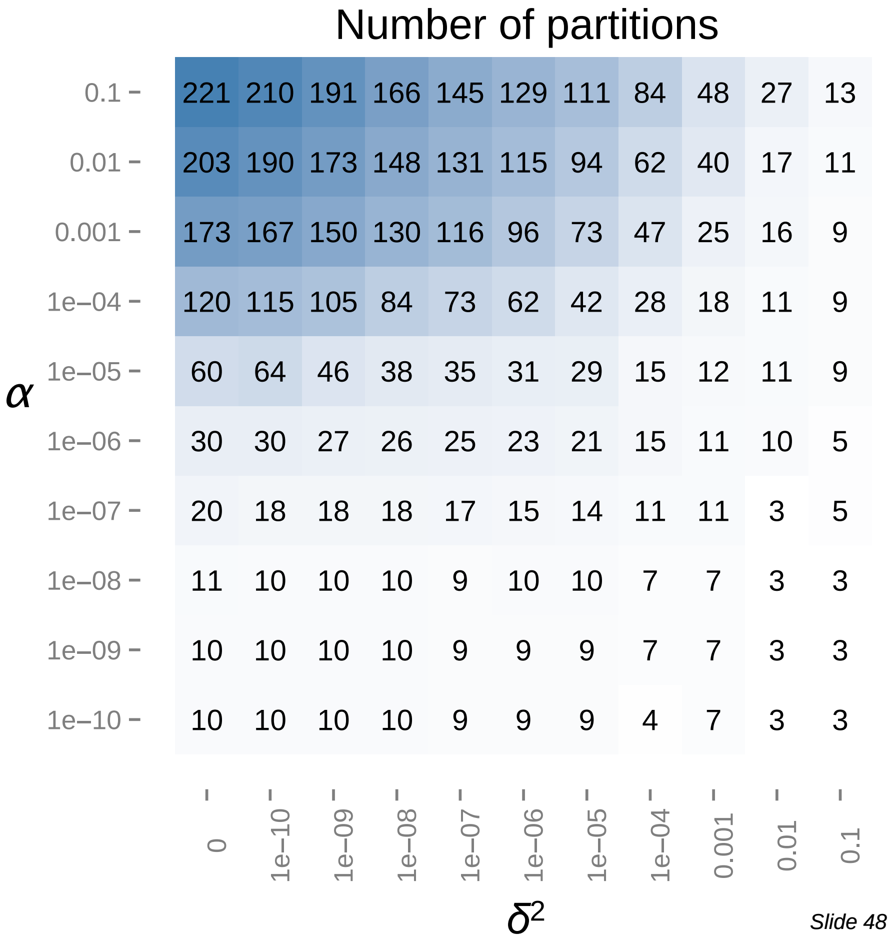
Total RSS: 6.40

But how to choose those tuning parameters?

We’ve got α, nugget δ^2, and buffer size B.

- B we have little control over.

- We can explore α and δ^2 in terms of their impact on the **number of partitions** and the **total RSS**.



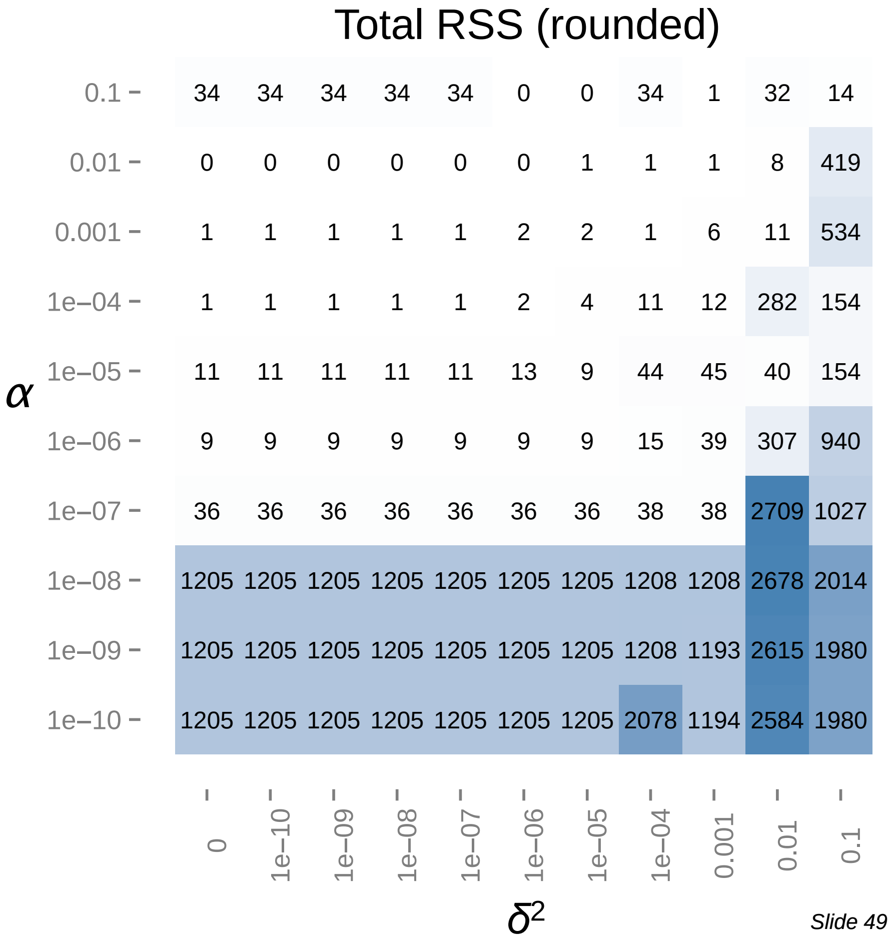
Number of partitions

α \ δ^2	0	1e–10	1e–09	1e–08	1e–07	1e–06	1e–05	1e–04	0.001	0.01	0.1
0.1	221	210	191	166	145	129	111	84	48	27	13
0.01	203	190	173	148	131	115	94	62	40	17	11
0.001	173	167	150	130	116	96	73	47	25	16	9
1e–04	120	115	105	84	73	62	42	28	18	11	9
1e–05	60	64	46	38	35	31	29	15	12	11	9
1e–06	30	30	27	26	25	23	21	15	11	10	5
1e–07	20	18	18	18	17	15	14	11	11	3	5
1e–08	11	10	10	10	9	10	10	7	7	3	3
1e–09	10	10	10	10	9	9	9	7	7	3	3
1e–10	10	10	10	10	9	9	9	4	7	3	3

But how to choose those tuning parameters?

We’ve got α, nugget δ^2, and buffer size B.

- B we have little control over.
- We can explore α and δ^2 in terms of their impact on the **number of partitions** and the **total RSS**.



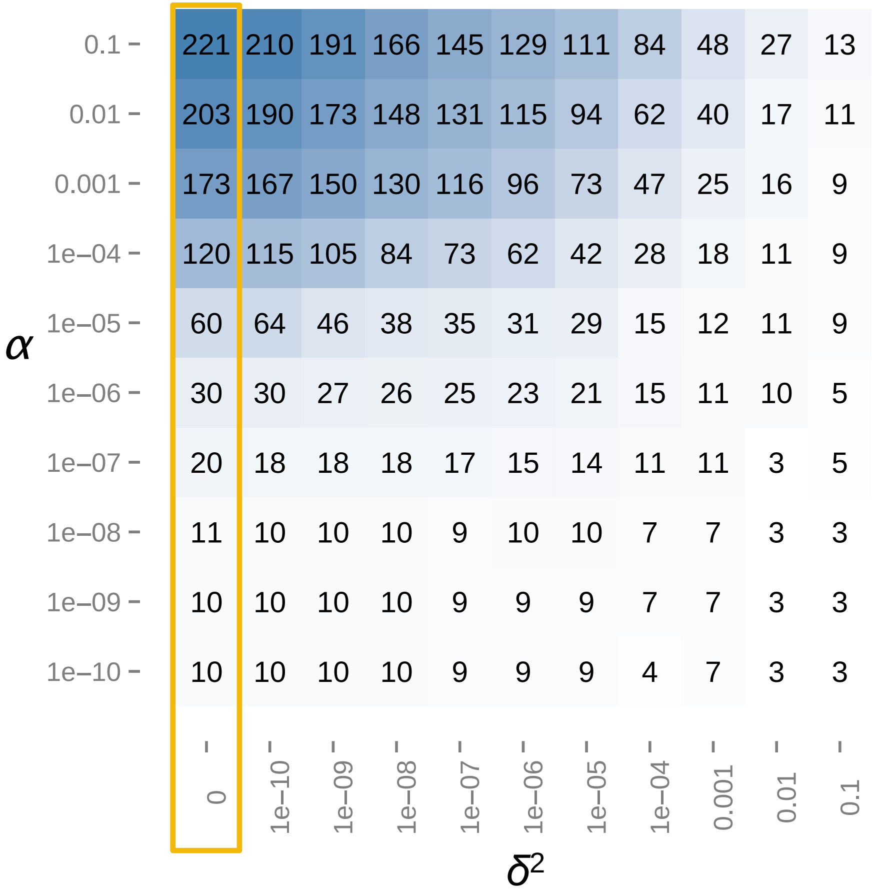
Total RSS (rounded)

α \ δ^2	0	1e−10	1e−09	1e−08	1e−07	1e−06	1e−05	1e−04	0.001	0.01	0.1
0.1	34	34	34	34	34	0	0	34	1	32	14
0.01	0	0	0	0	0	0	1	1	1	8	419
0.001	1	1	1	1	1	2	2	1	6	11	534
1e−04	1	1	1	1	1	2	4	11	12	282	154
1e−05	11	11	11	11	11	13	9	44	45	40	154
1e−06	9	9	9	9	9	9	9	15	39	307	940
1e−07	36	36	36	36	36	36	36	38	38	2709	1027
1e−08	1205	1205	1205	1205	1205	1205	1205	1208	1208	2678	2014
1e−09	1205	1205	1205	1205	1205	1205	1205	1208	1193	2615	1980
1e−10	1205	1205	1205	1205	1205	1205	1205	2078	1194	2584	1980

Start by understanding the $\delta^2 = 0$ case

You might think we could just turn the α knob to reject less often.

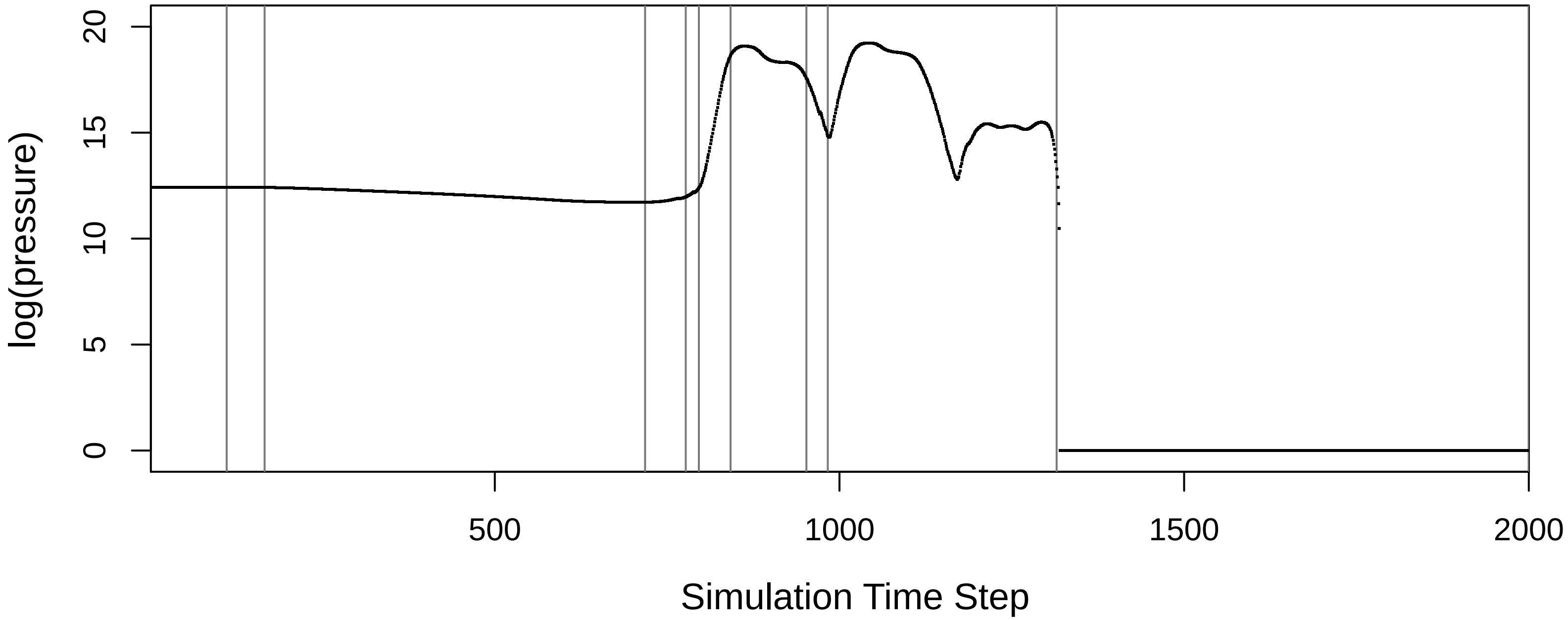
Number of partitions



	0	1e−10	1e−09	1e−08	1e−07	1e−06	1e−05	1e−04	0.001	0.01	0.1
0.1	221	210	191	166	145	129	111	84	48	27	13
0.01	203	190	173	148	131	115	94	62	40	17	11
0.001	173	167	150	130	116	96	73	47	25	16	9
1e−04	120	115	105	84	73	62	42	28	18	11	9
1e−05	60	64	46	38	35	31	29	15	12	11	9
1e−06	30	30	27	26	25	23	21	15	11	10	5
1e−07	20	18	18	18	17	15	14	11	11	3	5
1e−08	11	10	10	10	9	10	10	7	7	3	3
1e−09	10	10	10	10	9	9	9	7	7	3	3
1e−10	10	10	10	10	9	9	9	4	7	3	3

What can we accomplish by adjusting *α* alone?

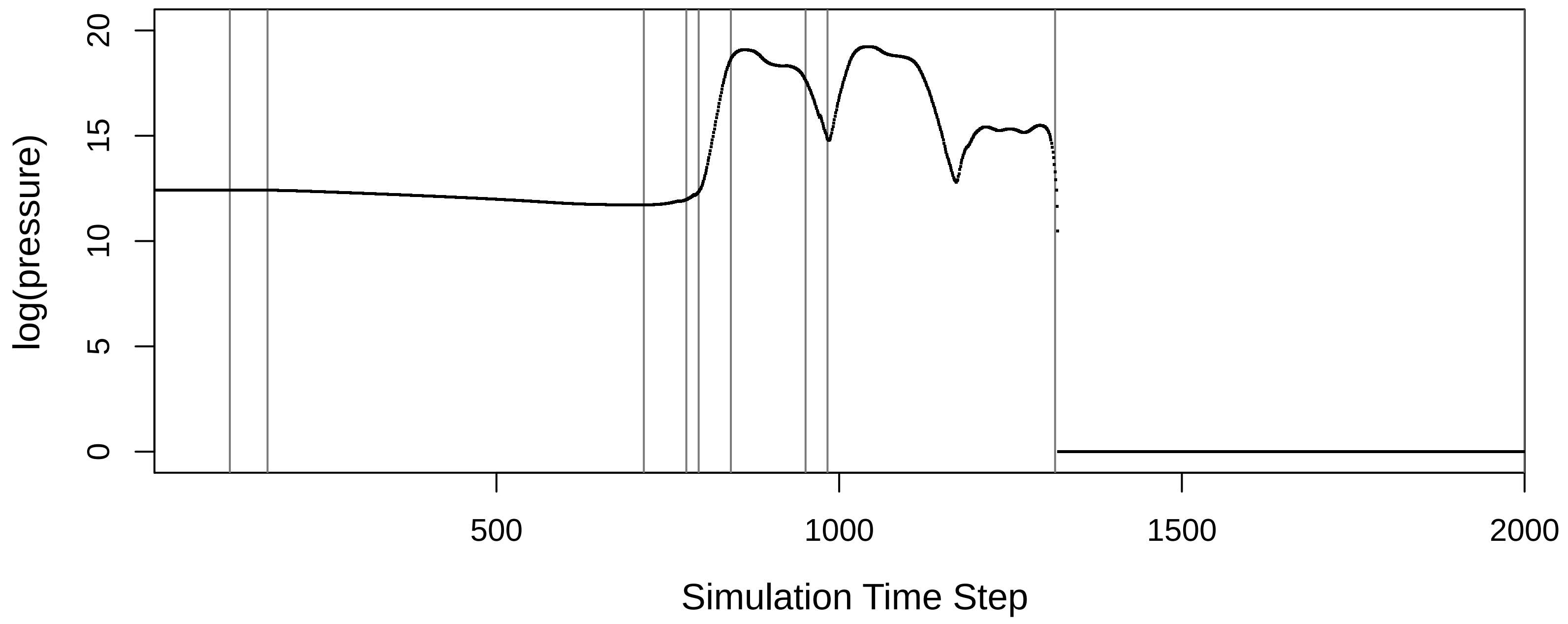
With $\delta^2 = 0$, the hypothesis test gets fooled when **curr** and **buff** both have extremely low error.



0.1	221
0.01	203
0.001	173
1e–04	120
1e–05	60
1e–06	30
1e–07	20
1e–08	11
1e–09	10
1e–10	10
0	

What can we accomplish by adjusting *α* alone?

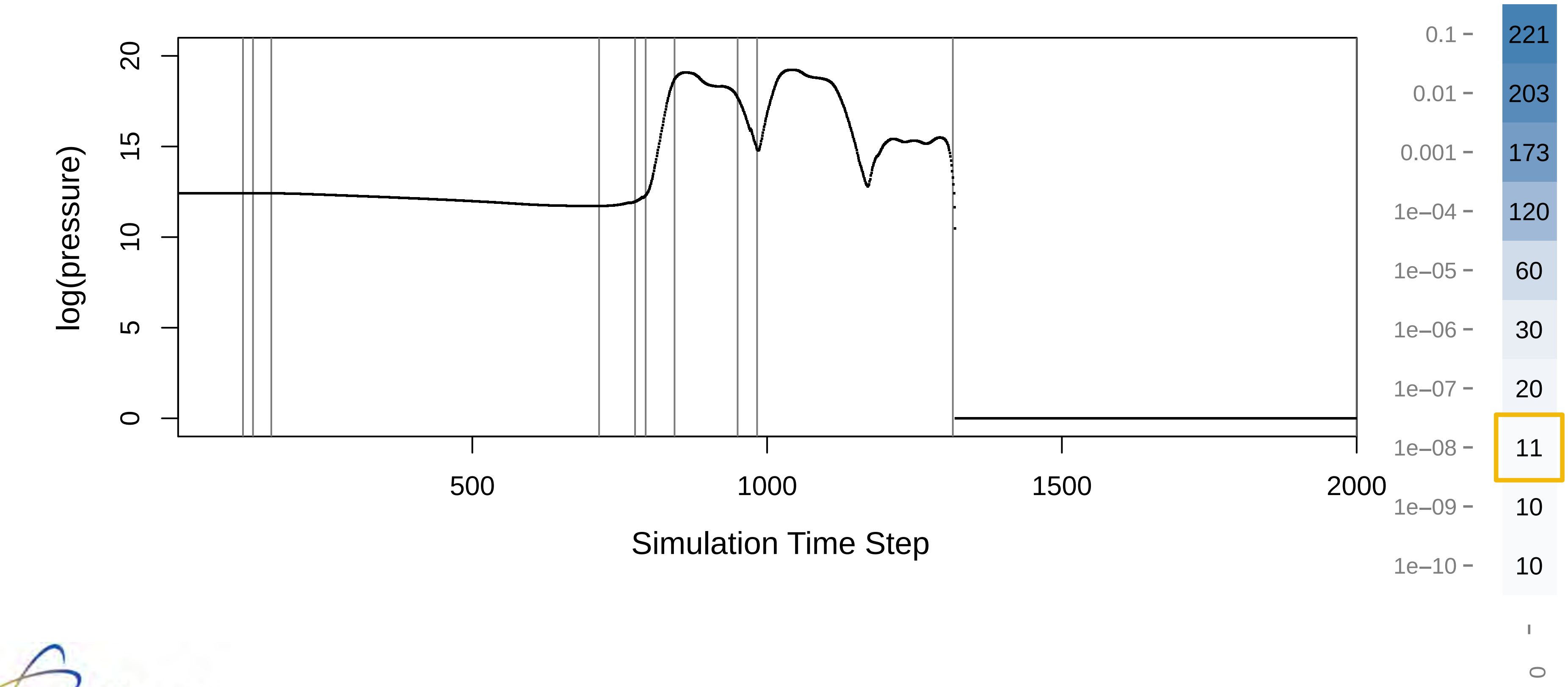
With $\delta^2 = 0$, the hypothesis test gets fooled when **curr** and **buff** both have extremely low error.



0.1	221
0.01	203
0.001	173
1e–04	120
1e–05	60
1e–06	30
1e–07	20
1e–08	11
1e–09	10
1e–10	10
0	

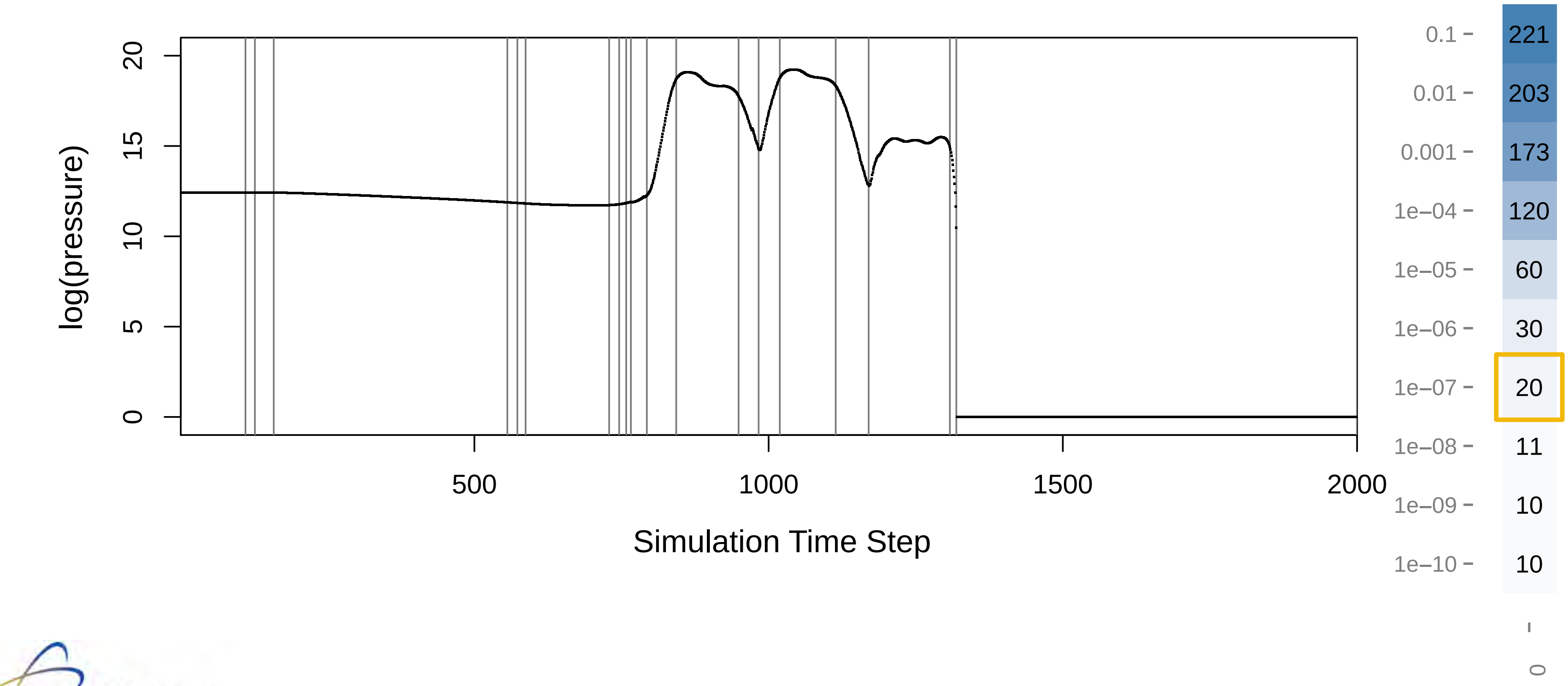
What can we accomplish by adjusting *α* alone?

With $\delta^2 = 0$, the hypothesis test gets fooled when **curr** and **buff** both have extremely low error.



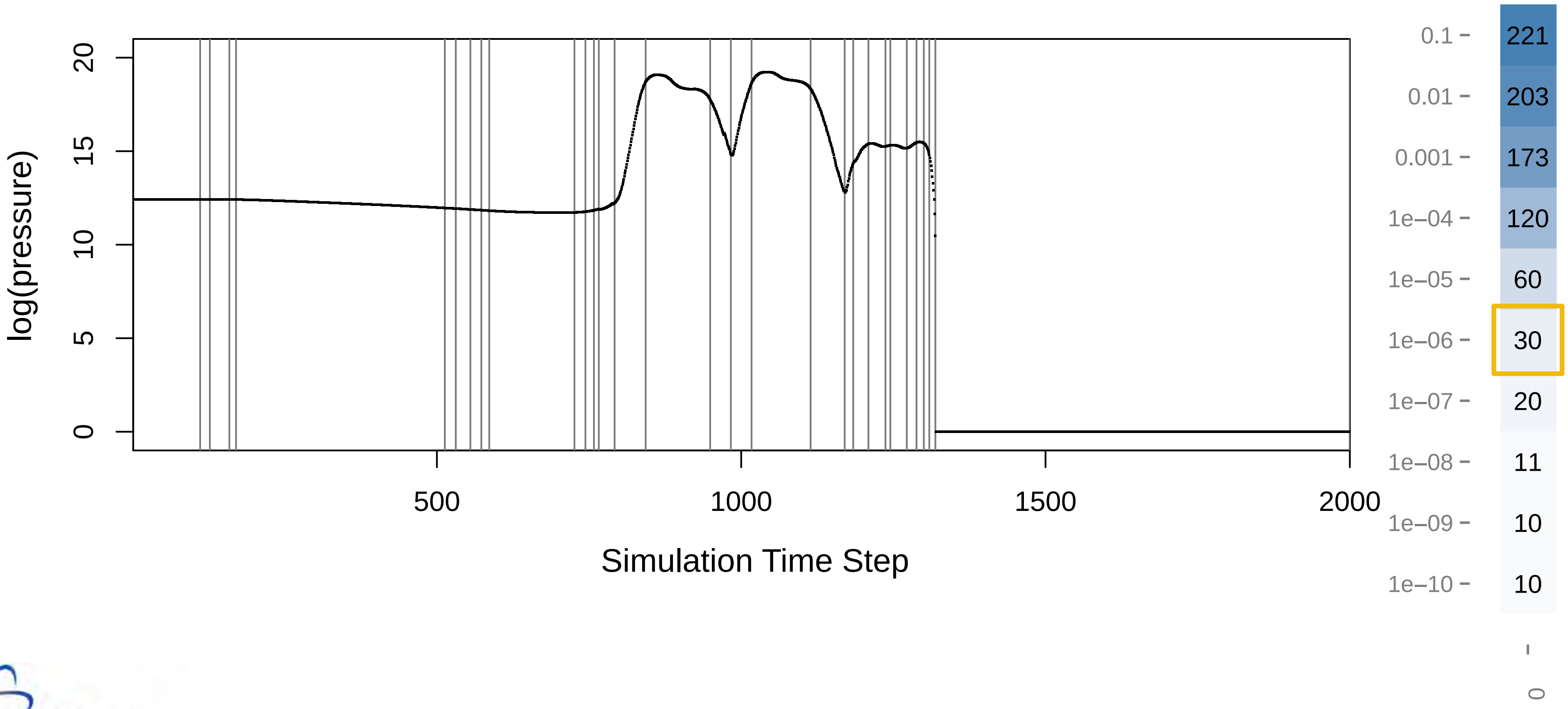
What can we accomplish by adjusting *α* alone?

With $\delta^2 = 0$, the hypothesis test gets fooled when **curr** and **buff** both have extremely low error.



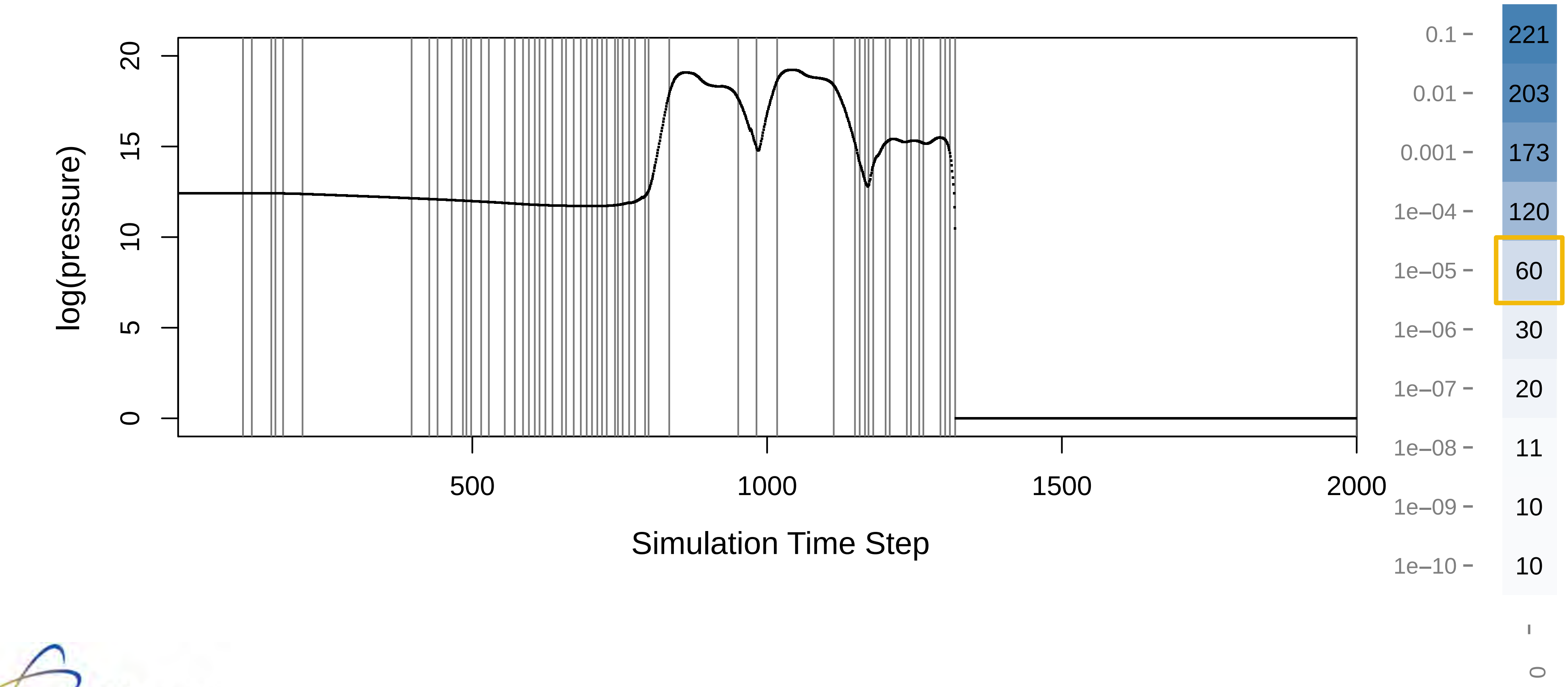
What can we accomplish by adjusting *α* alone?

With $\delta^2 = 0$, the hypothesis test gets fooled when **curr** and **buff** both have extremely low error.



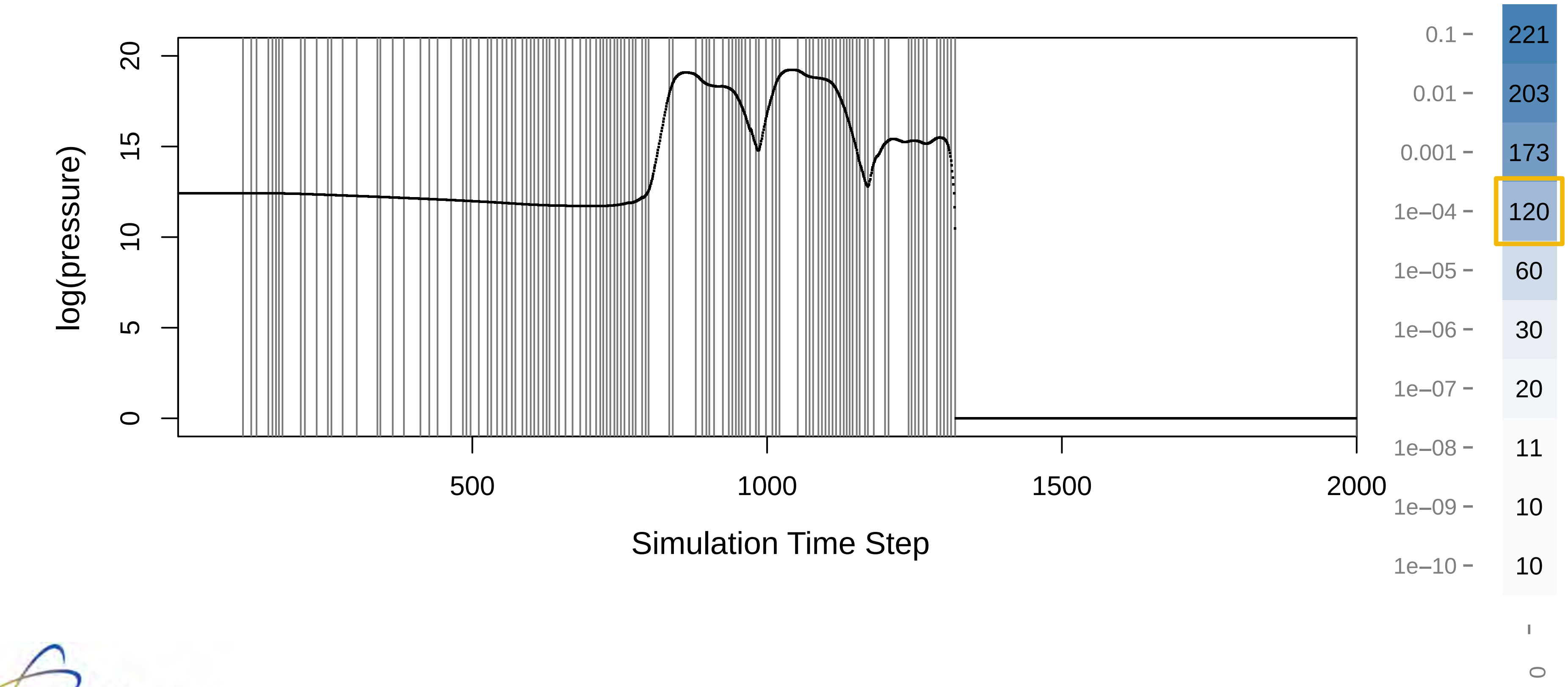
What can we accomplish by adjusting *α* alone?

With $\delta^2 = 0$, the hypothesis test gets fooled when **curr** and **buff** both have extremely low error.



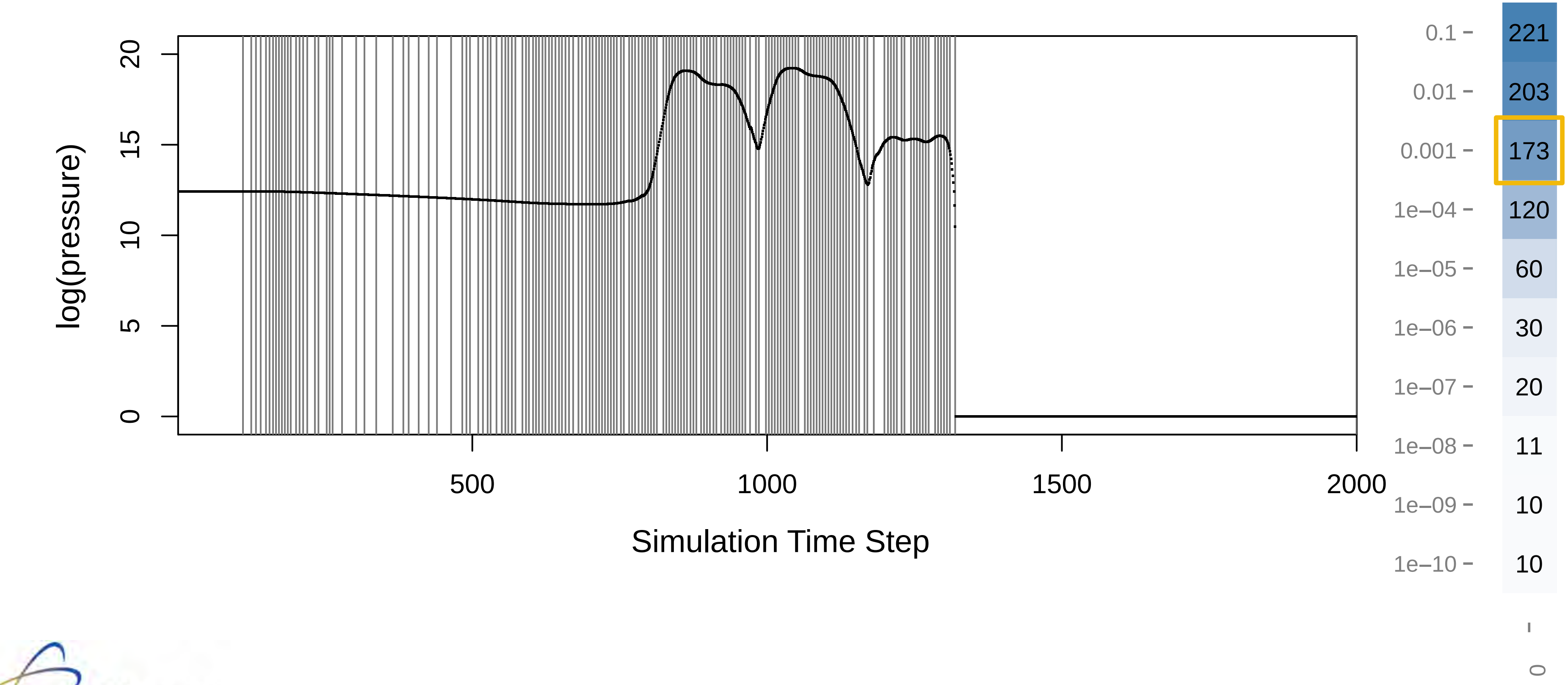
What can we accomplish by adjusting *α* alone?

With $\delta^2 = 0$, the hypothesis test gets fooled when **curr** and **buff** both have extremely low error.



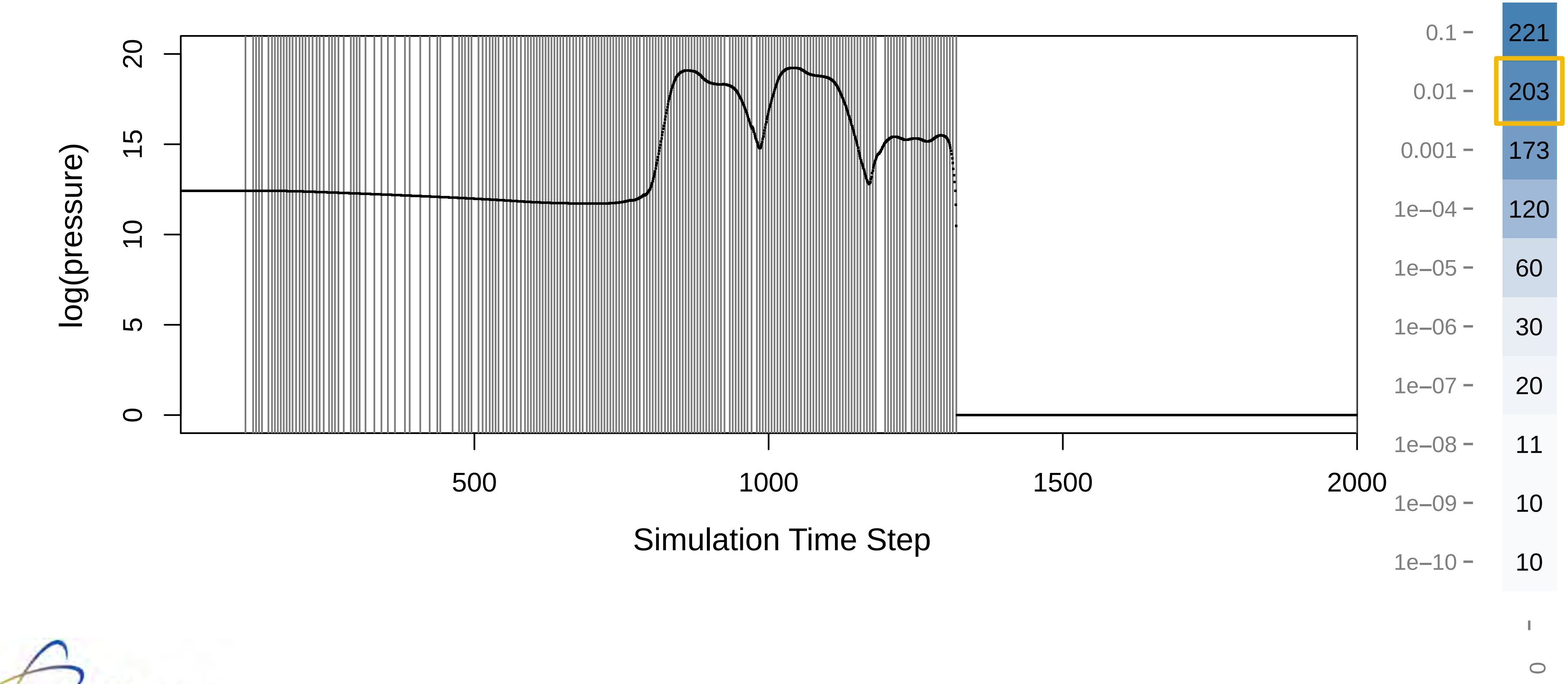
What can we accomplish by adjusting *α* alone?

With $\delta^2 = 0$, the hypothesis test gets fooled when **curr** and **buff** both have extremely low error.



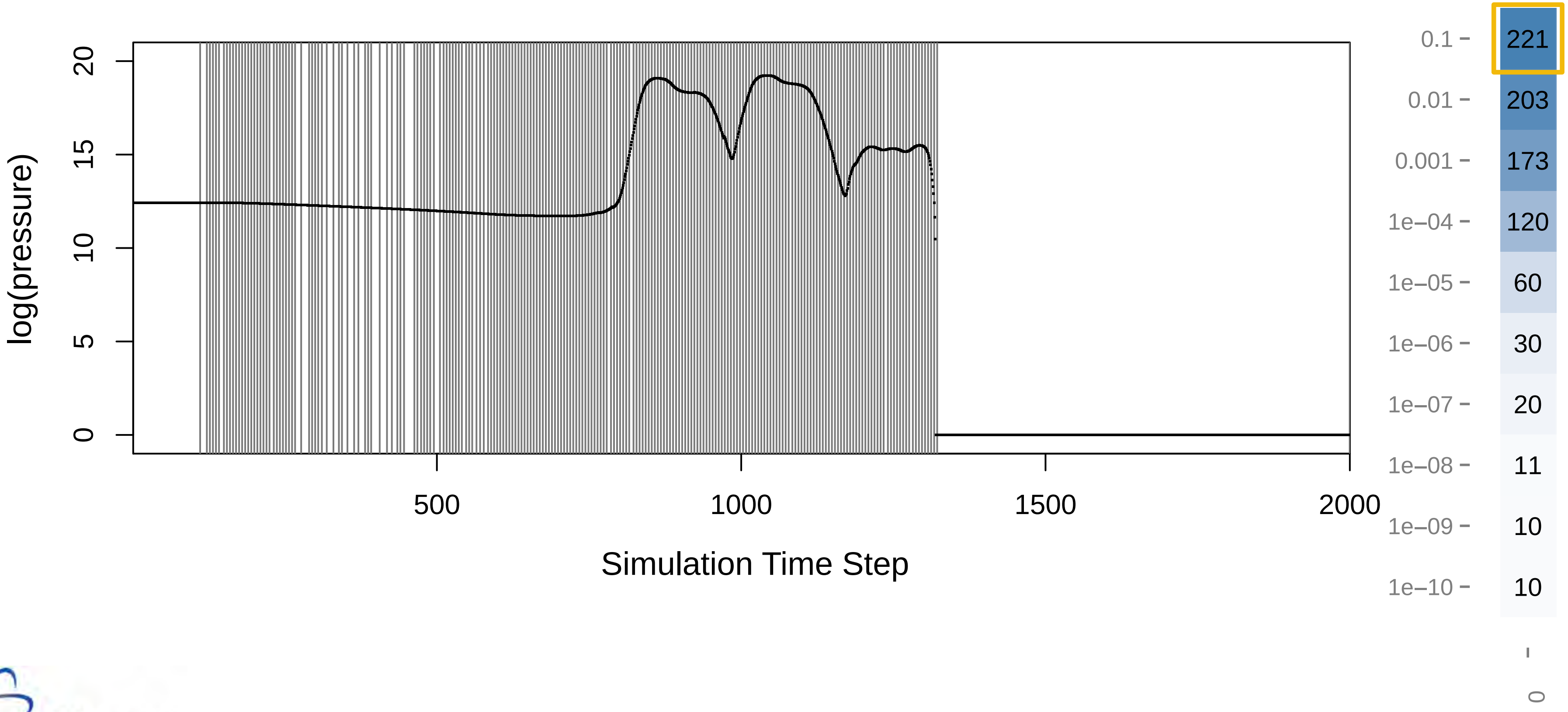
What can we accomplish by adjusting α alone?

With $\delta^2 = 0$, the hypothesis test gets fooled when **curr** and **buff** both have extremely low error.



What can we accomplish by adjusting *α* alone?

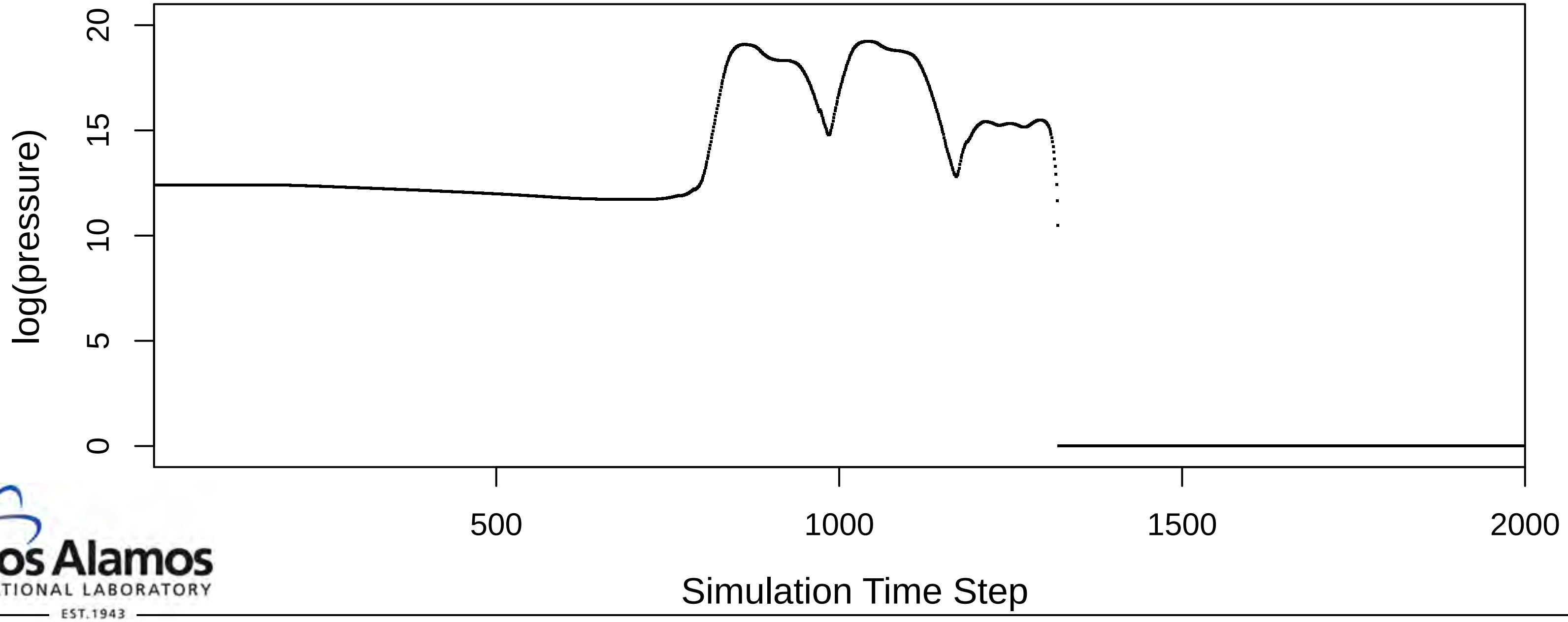
With $\delta^2 = 0$, the hypothesis test gets fooled when **curr** and **buff** both have extremely low error.



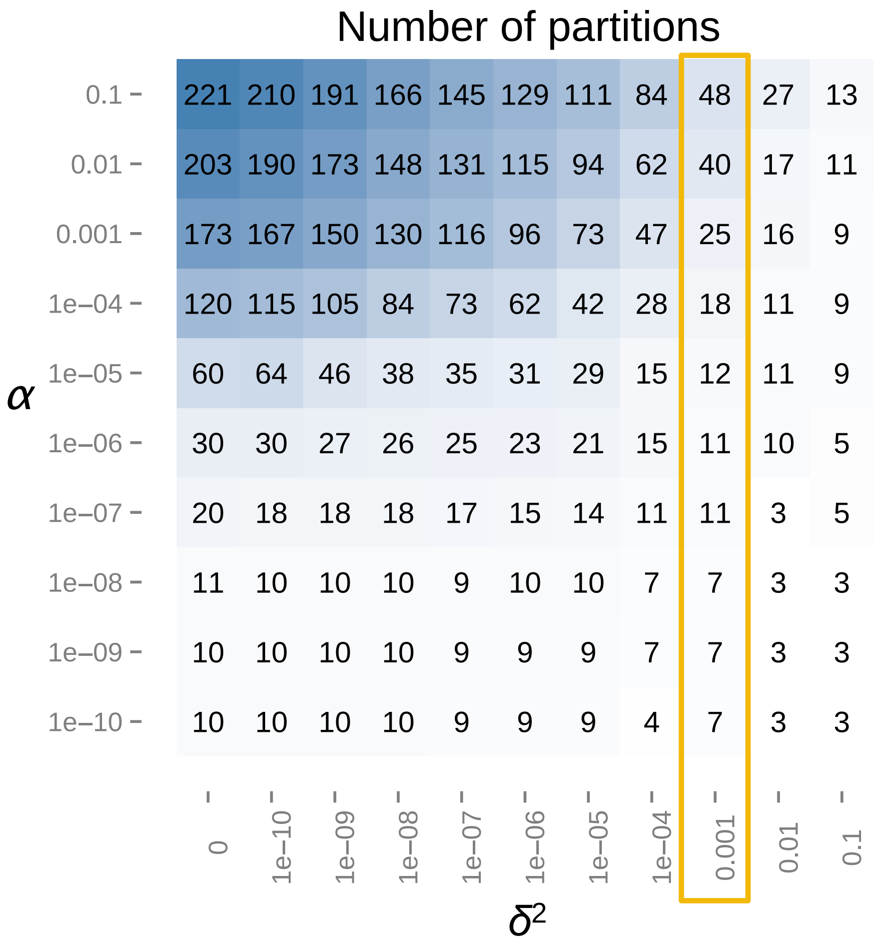
Q: What's going wrong? A: What isn't going wrong?

These deterministic computer codes violate pretty much every statistical assumption we typically like to make:

- Samples aren't i.i.d. but rather come from a smooth process.
- Error isn't Gaussian.
- Variances of **curr** and **buff** aren't necessarily equal.



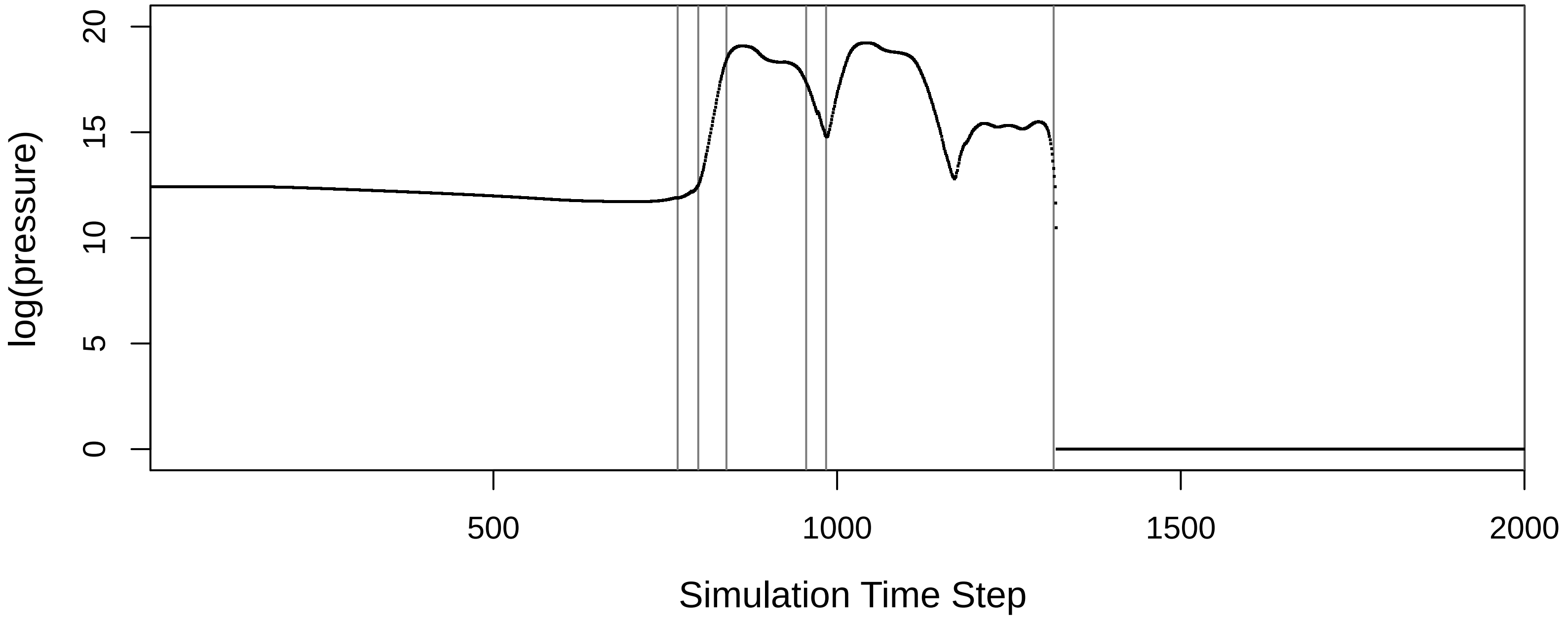
Take a look at a positive δ^2 case



α \ δ^2	0	1e−10	1e−09	1e−08	1e−07	1e−06	1e−05	1e−04	0.001	0.01	0.1
0.1	221	210	191	166	145	129	111	84	48	27	13
0.01	203	190	173	148	131	115	94	62	40	17	11
0.001	173	167	150	130	116	96	73	47	25	16	9
1e−04	120	115	105	84	73	62	42	28	18	11	9
1e−05	60	64	46	38	35	31	29	15	12	11	9
1e−06	30	30	27	26	25	23	21	15	11	10	5
1e−07	20	18	18	18	17	15	14	11	11	3	5
1e−08	11	10	10	10	9	10	10	7	7	3	3
1e−09	10	10	10	10	9	9	9	7	7	3	3
1e−10	10	10	10	10	9	9	9	4	7	3	3

What does a positive δ^2 buy us?

It's like adding white noise with variance δ^2, providing a global effect on the kinds of changes that can be ignored.

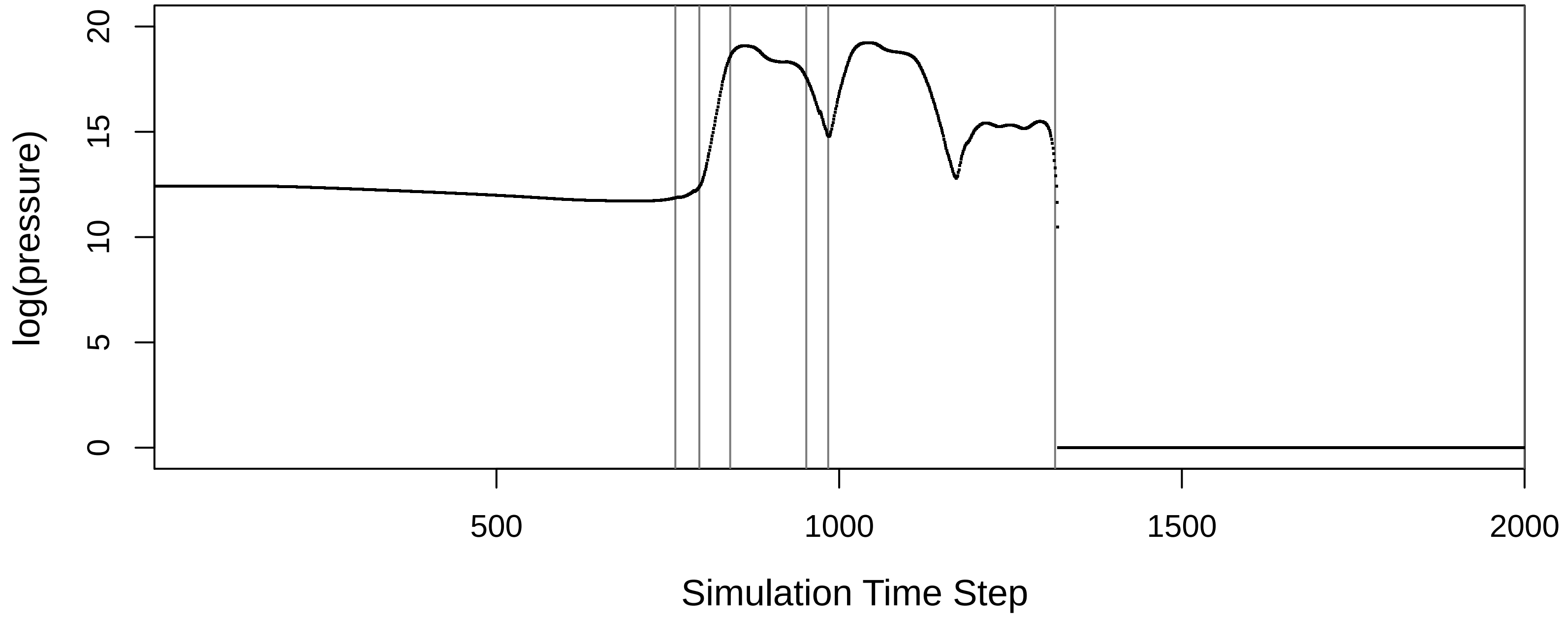


0.1	48
0.01	40
0.001	25
1e–04	18
1e–05	12
1e–06	11
1e–07	11
1e–08	7
1e–09	7
1e–10	7

0.001

What does a positive δ^2 buy us?

It’s like adding white noise with variance δ^2, providing a global effect on the kinds of changes that can be ignored.

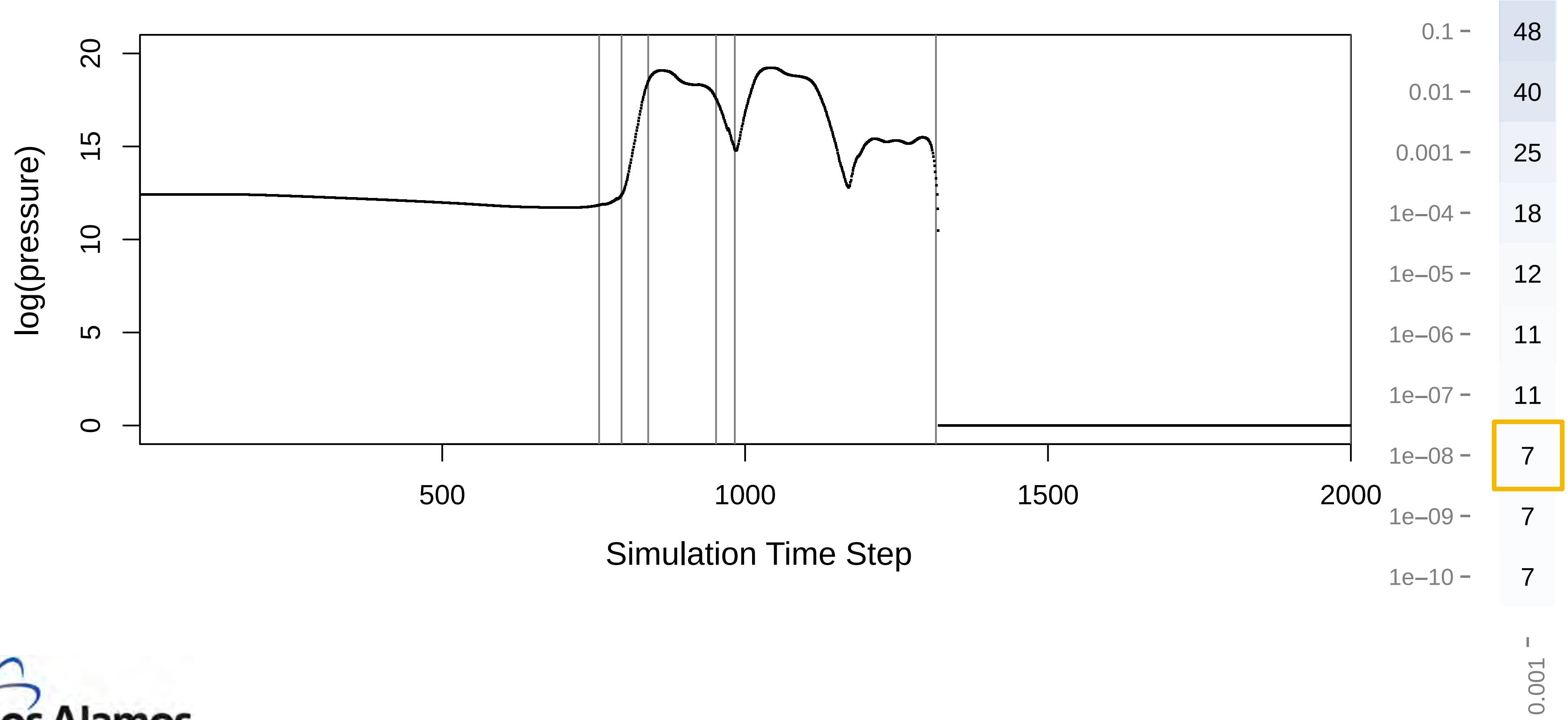


0.1	48
0.01	40
0.001	25
1e–04	18
1e–05	12
1e–06	11
1e–07	11
1e–08	7
1e–09	7
1e–10	7

0.001

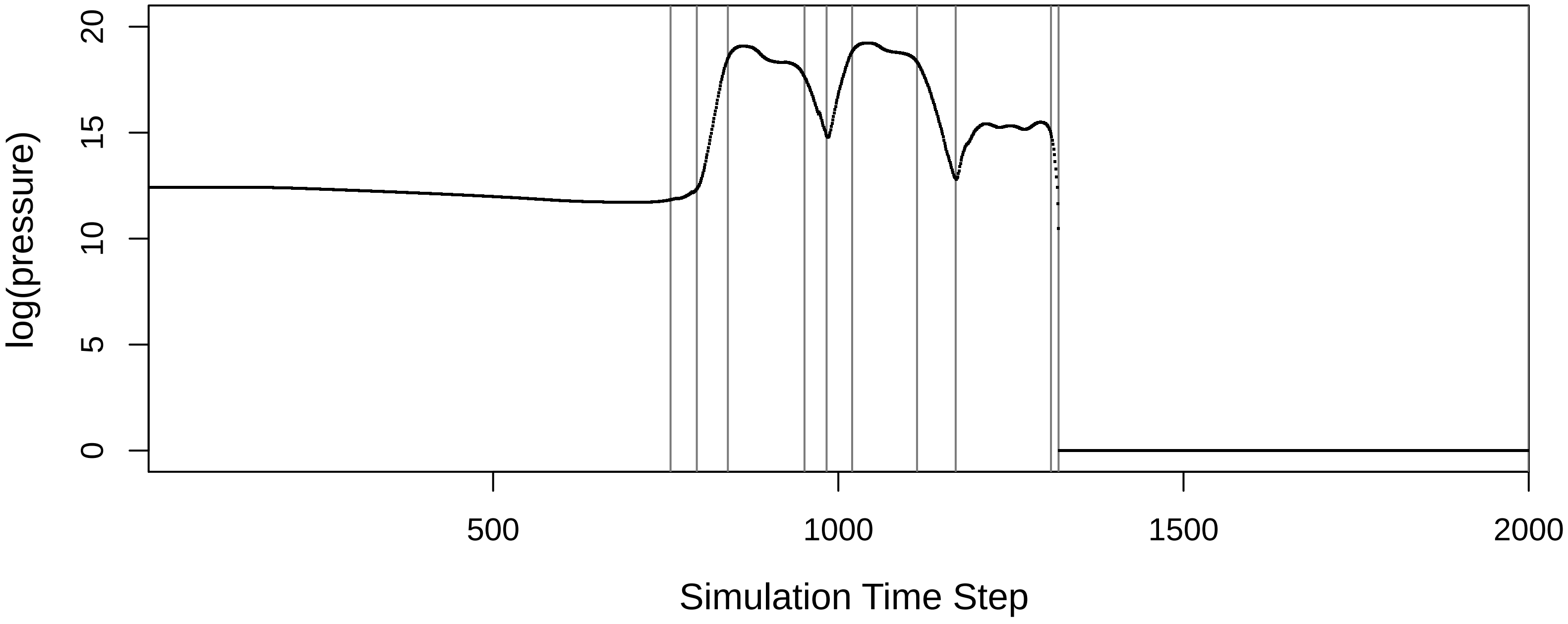
What does a positive δ^2 buy us?

It's like adding white noise with variance δ^2, providing a global effect on the kinds of changes that can be ignored.



What does a positive δ^2 buy us?

It's like adding white noise with variance δ^2, providing a global effect on the kinds of changes that can be ignored.

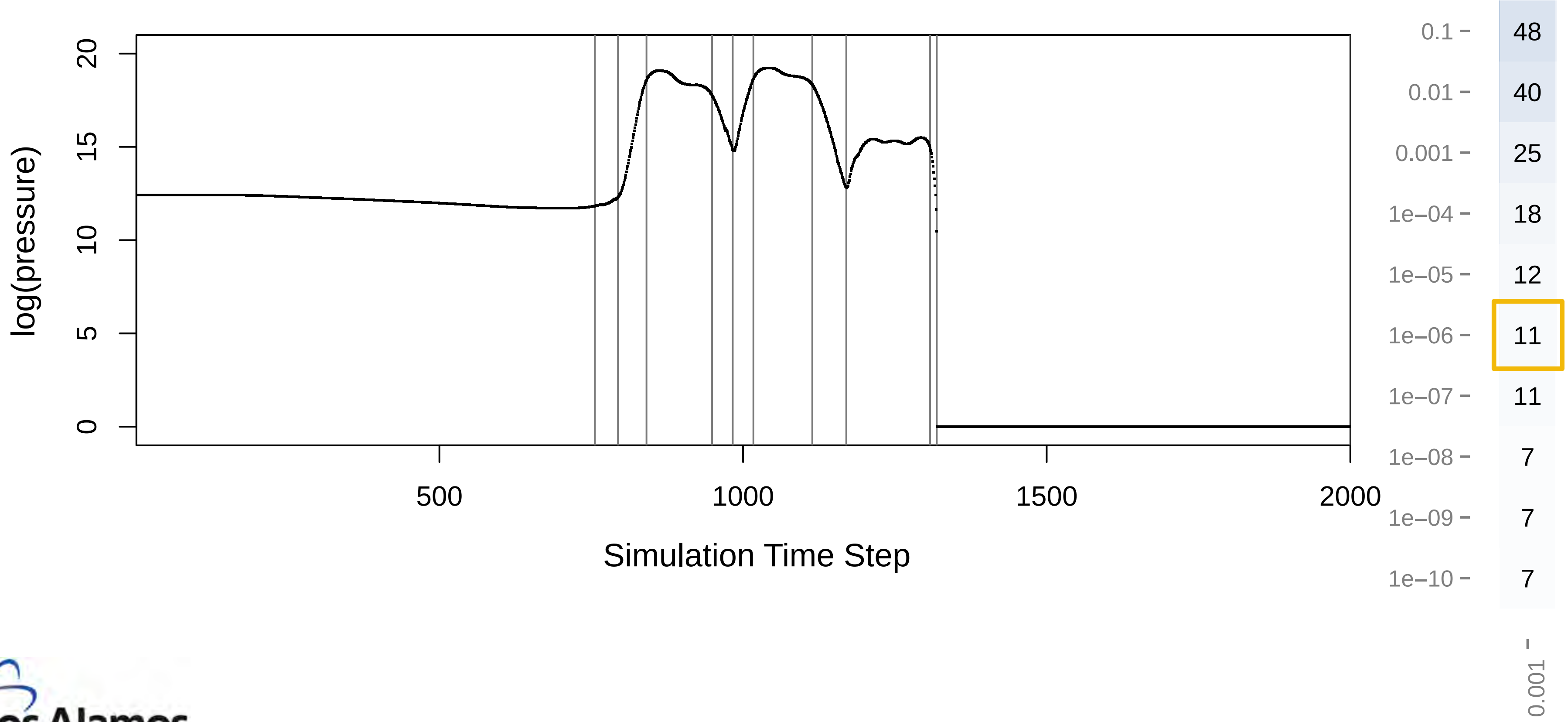


0.1	48
0.01	40
0.001	25
1e–04	18
1e–05	12
1e–06	11
1e–07	11
1e–08	7
1e–09	7
1e–10	7

0.001

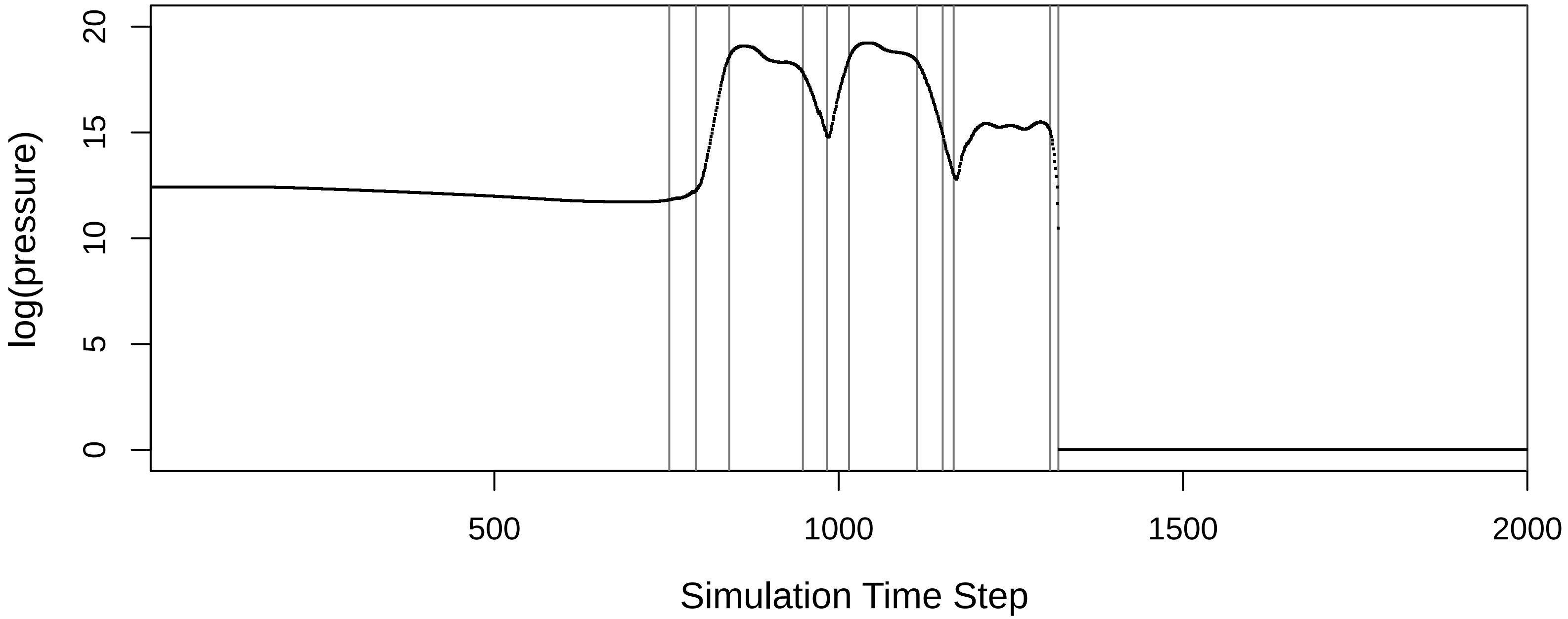
What does a positive δ^2 buy us?

It's like adding white noise with variance δ^2, providing a global effect on the kinds of changes that can be ignored.



What does a positive δ^2 buy us?

It's like adding white noise with variance δ^2, providing a global effect on the kinds of changes that can be ignored.

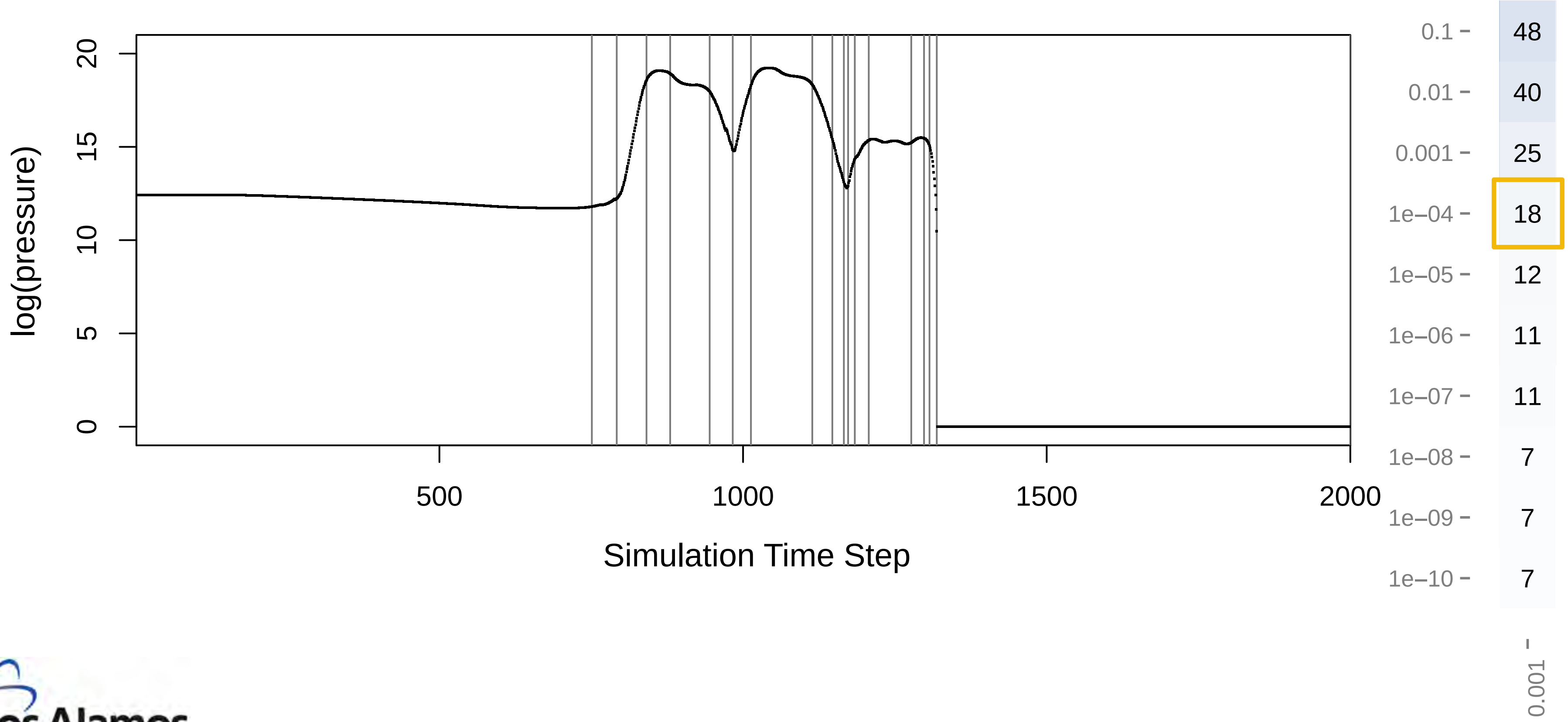


0.1	48
0.01	40
0.001	25
1e−04	18
1e−05	12
1e−06	11
1e−07	11
1e−08	7
1e−09	7
1e−10	7

0.001

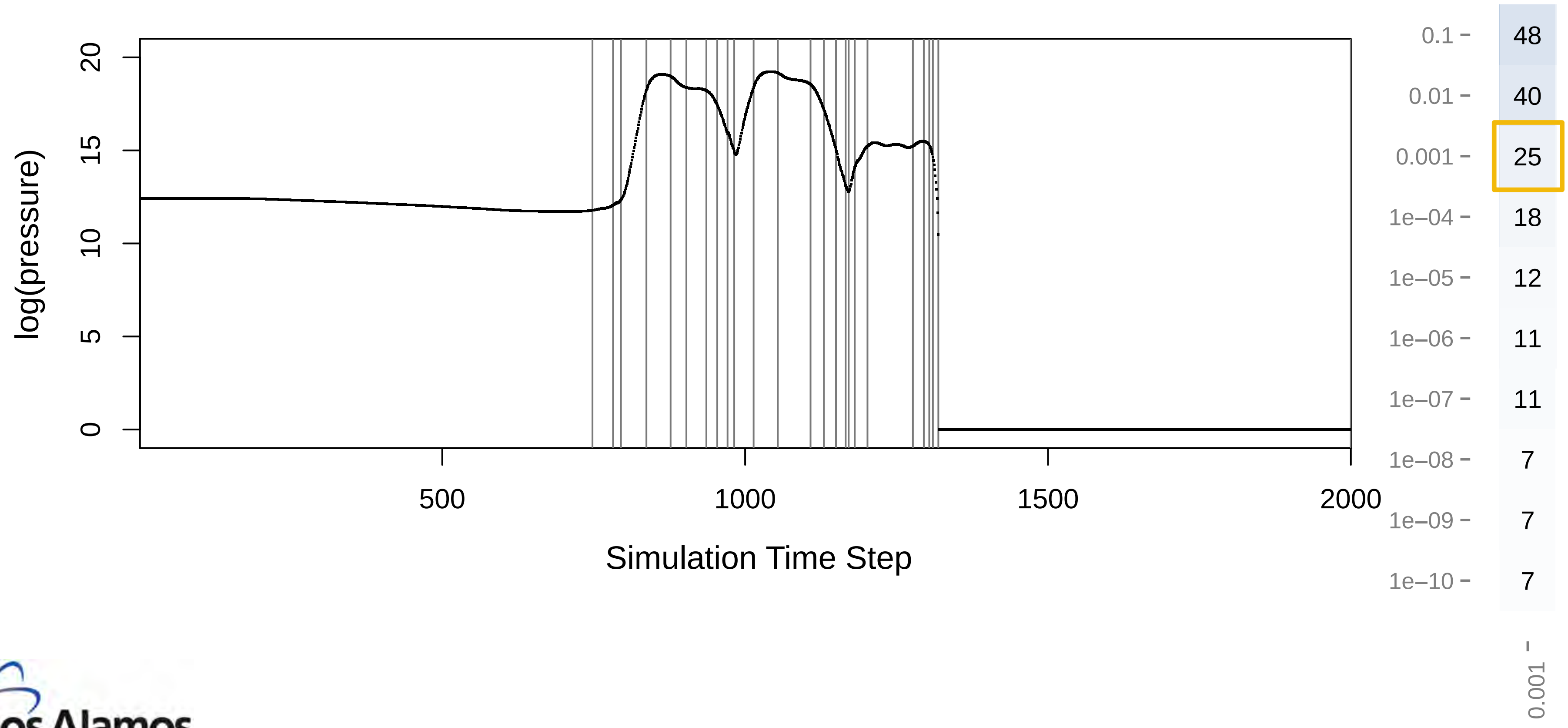
What does a positive δ^2 buy us?

It's like adding white noise with variance δ^2, providing a global effect on the kinds of changes that can be ignored.



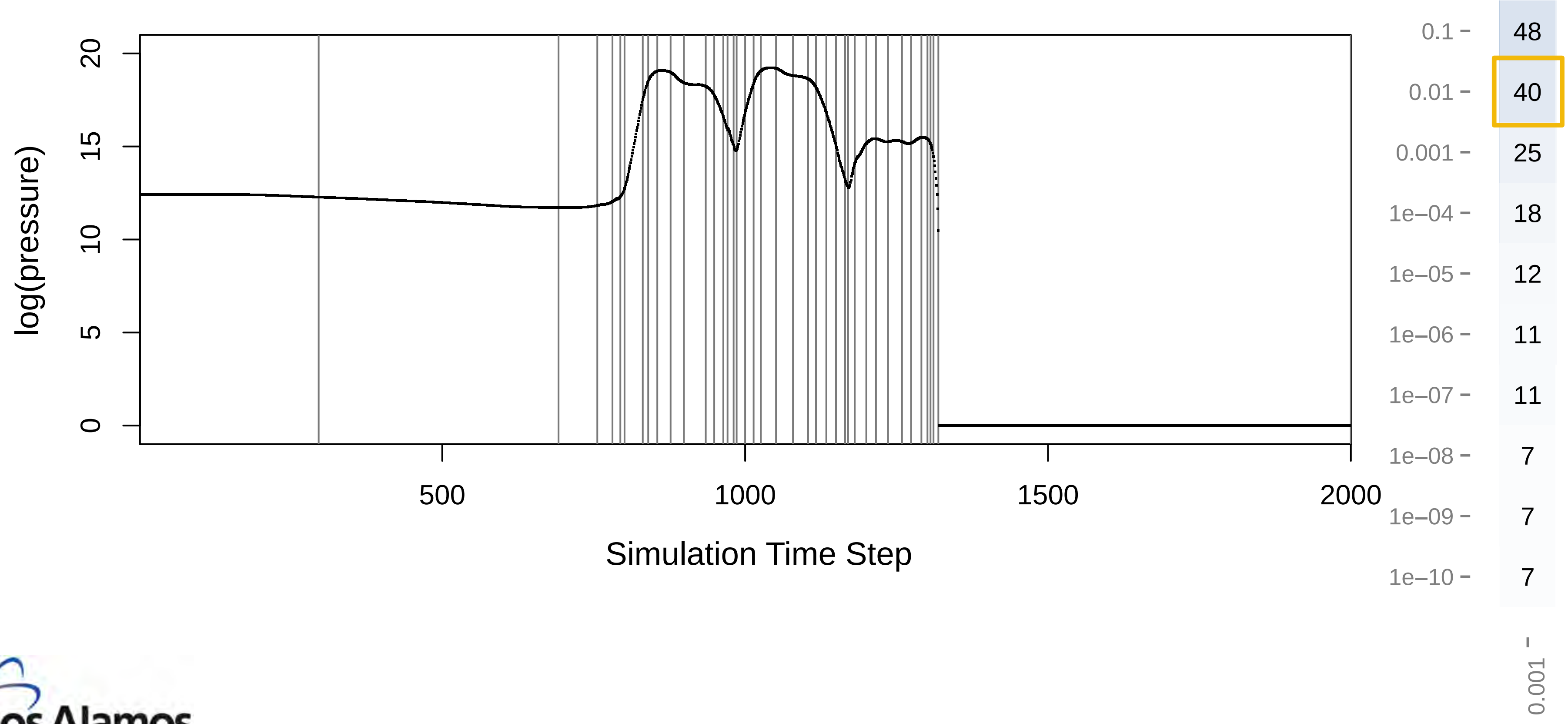
What does a positive δ^2 buy us?

It's like adding white noise with variance δ^2, providing a global effect on the kinds of changes that can be ignored.



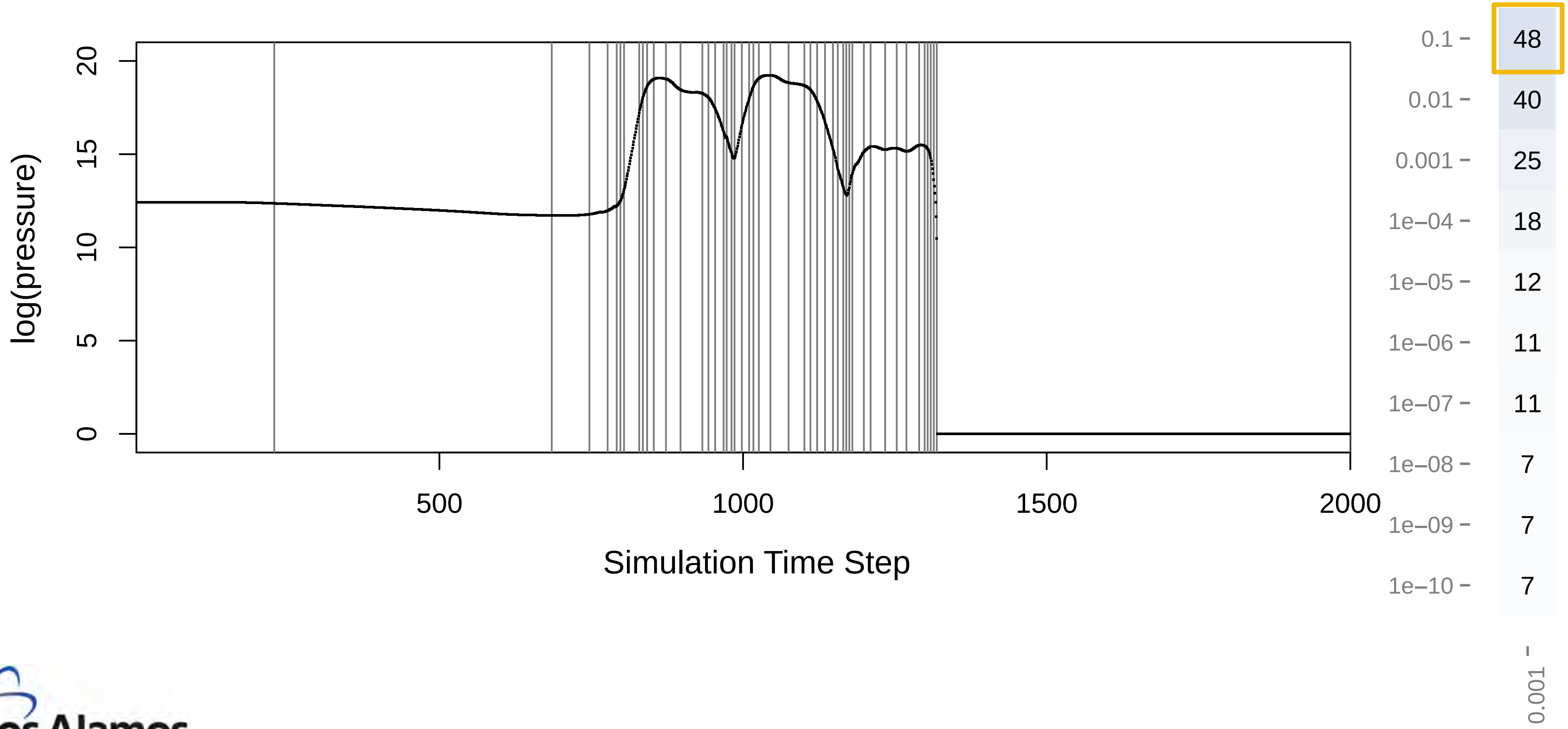
What does a positive δ^2 buy us?

It’s like adding white noise with variance δ^2, providing a global effect on the kinds of changes that can be ignored.



What does a positive δ^2 buy us?

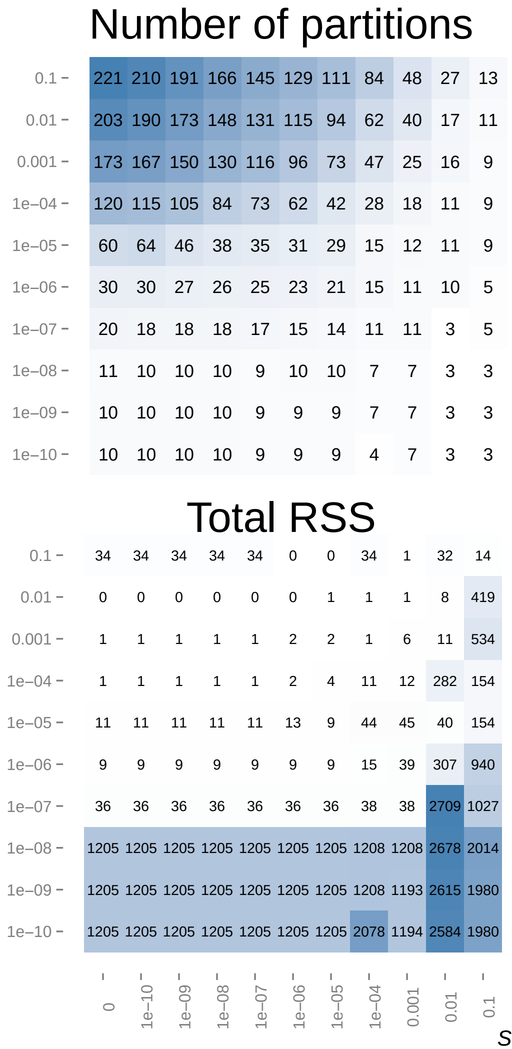
It’s like adding white noise with variance δ^2, providing a global effect on the kinds of changes that can be ignored.



Ok, but still: How to choose those tuning parameters?

Here’s what we’ve learned:

- α governs local choices about partitioning **curr** and **buff**. Increasing α fills in areas that already have partitions, making the fit more detailed.
- δ^2 provides a global effect about the kinds of changes to ignore.
- For now we recommend doing a few “scanning” runs of the simulation to build small versions of tables like these.



Number of partitions

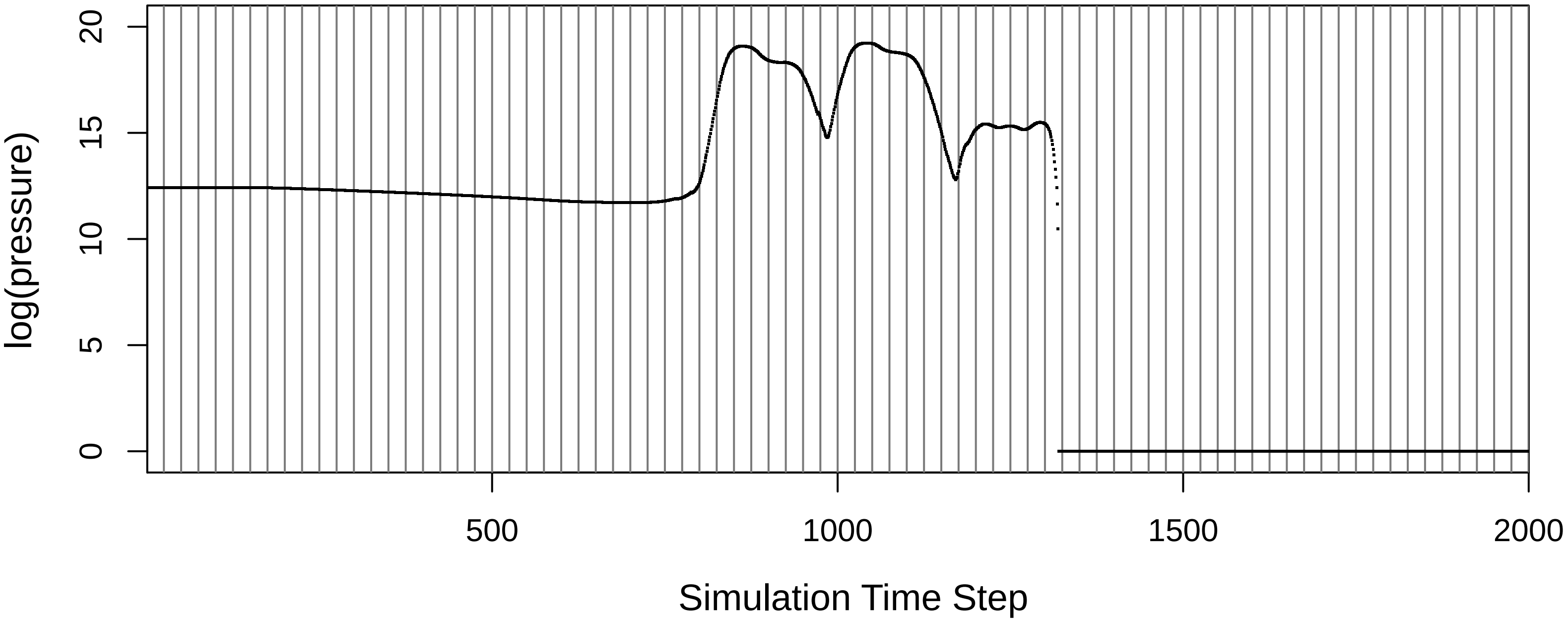
0.1	221	210	191	166	145	129	111	84	48	27	13
0.01	203	190	173	148	131	115	94	62	40	17	11
0.001	173	167	150	130	116	96	73	47	25	16	9
1e–04	120	115	105	84	73	62	42	28	18	11	9
1e–05	60	64	46	38	35	31	29	15	12	11	9
1e–06	30	30	27	26	25	23	21	15	11	10	5
1e–07	20	18	18	18	17	15	14	11	11	3	5
1e–08	11	10	10	10	9	10	10	7	7	3	3
1e–09	10	10	10	10	9	9	9	7	7	3	3
1e–10	10	10	10	10	9	9	9	4	7	3	3

Total RSS

	0	1e–10	1e–09	1e–08	1e–07	1e–06	1e–05	1e–04	0.001	0.01	0.1
0.1	34	34	34	34	34	0	0	34	1	32	14
0.01	0	0	0	0	0	0	1	1	1	8	419
0.001	1	1	1	1	1	2	2	1	6	11	534
1e–04	1	1	1	1	1	2	4	11	12	282	154
1e–05	11	11	11	11	11	13	9	44	45	40	154
1e–06	9	9	9	9	9	9	9	15	39	307	940
1e–07	36	36	36	36	36	36	36	38	38	2709	1027
1e–08	1205	1205	1205	1205	1205	1205	1205	1208	1208	2678	2014
1e–09	1205	1205	1205	1205	1205	1205	1205	1208	1193	2615	1980
1e–10	1205	1205	1205	1205	1205	1205	1205	2078	1194	2584	1980

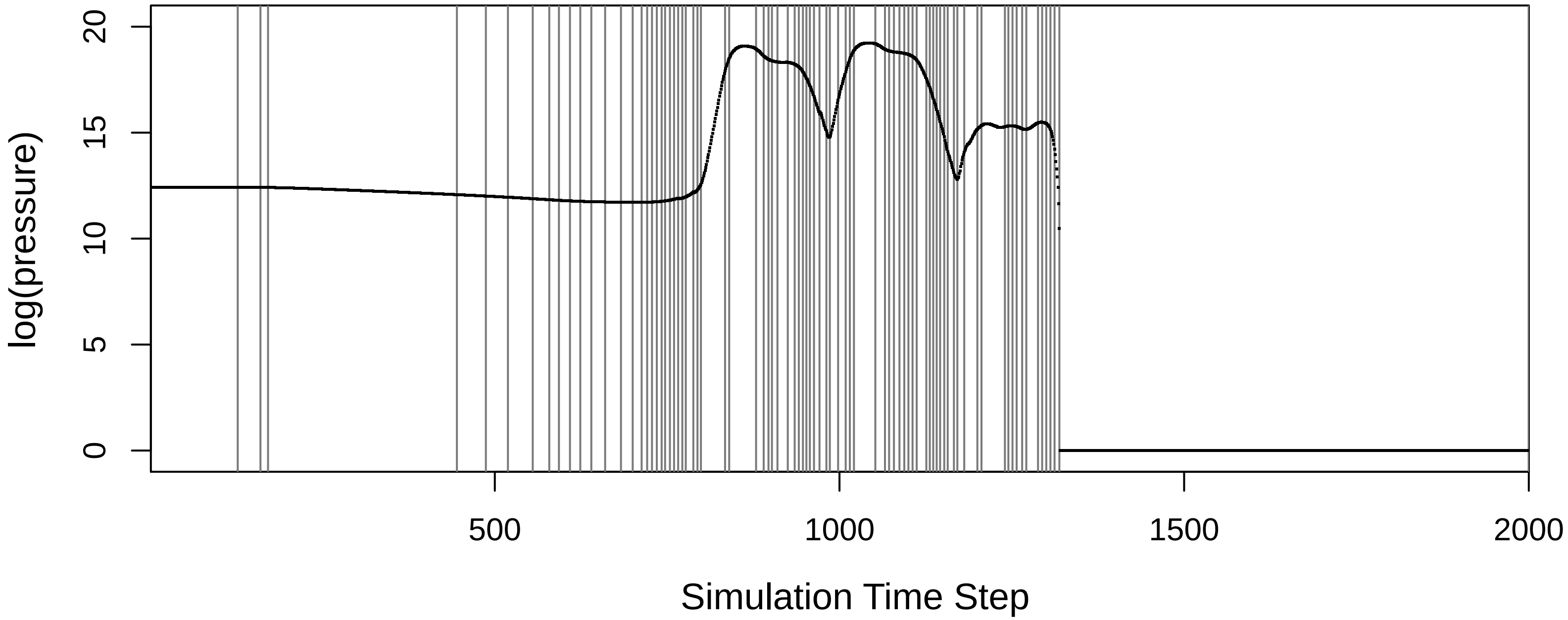
We argue: It’s worth it to do a few scanning runs

Standard practice: 80 partitions, total RSS 280.43.



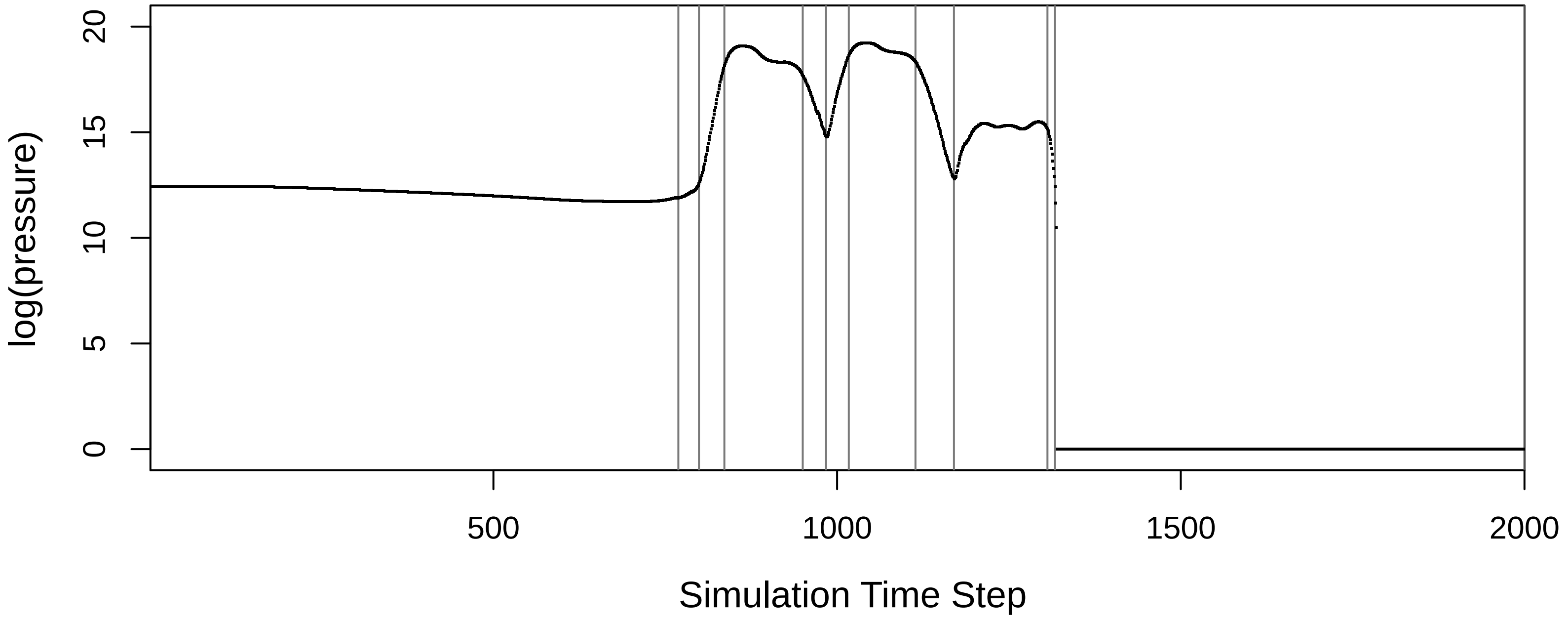
We argue: It’s worth it to do a few extra runs

Standard practice: 80 partitions, total RSS 280.43.
Our approach: 84 partitions, total RSS 1.30.



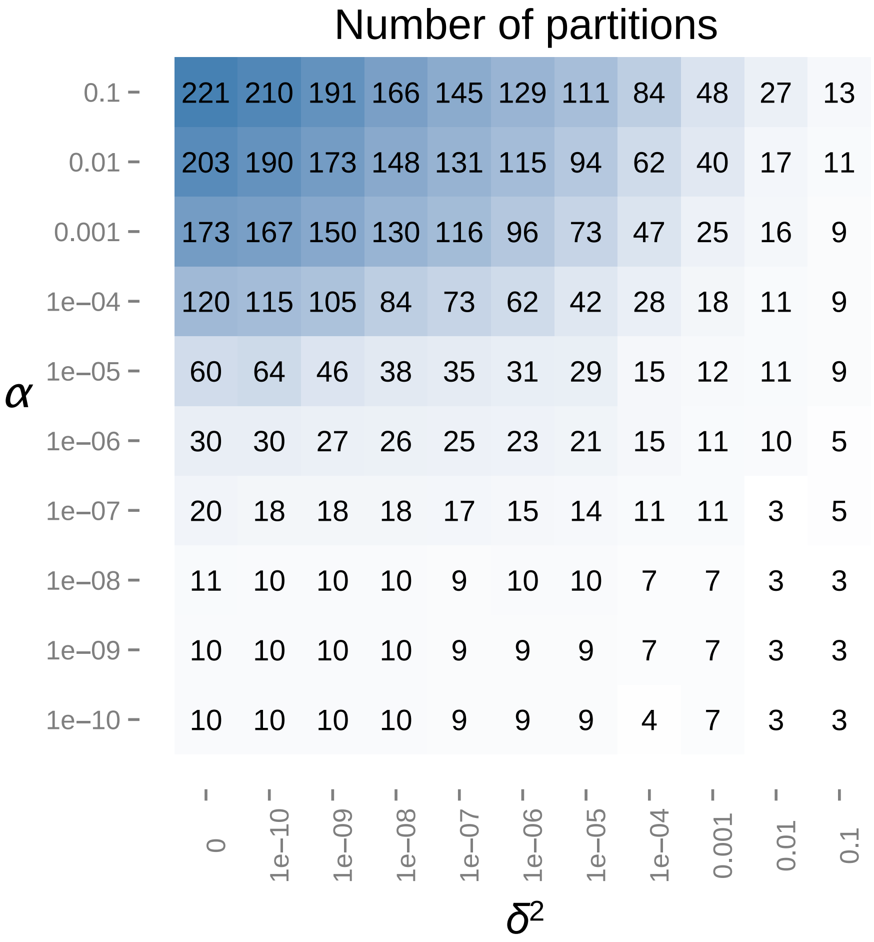
We argue: It’s worth it to do a few extra runs

Standard practice: 80 partitions, total RSS 280.43.
Our approach: 11 partitions, total RSS 281.61.



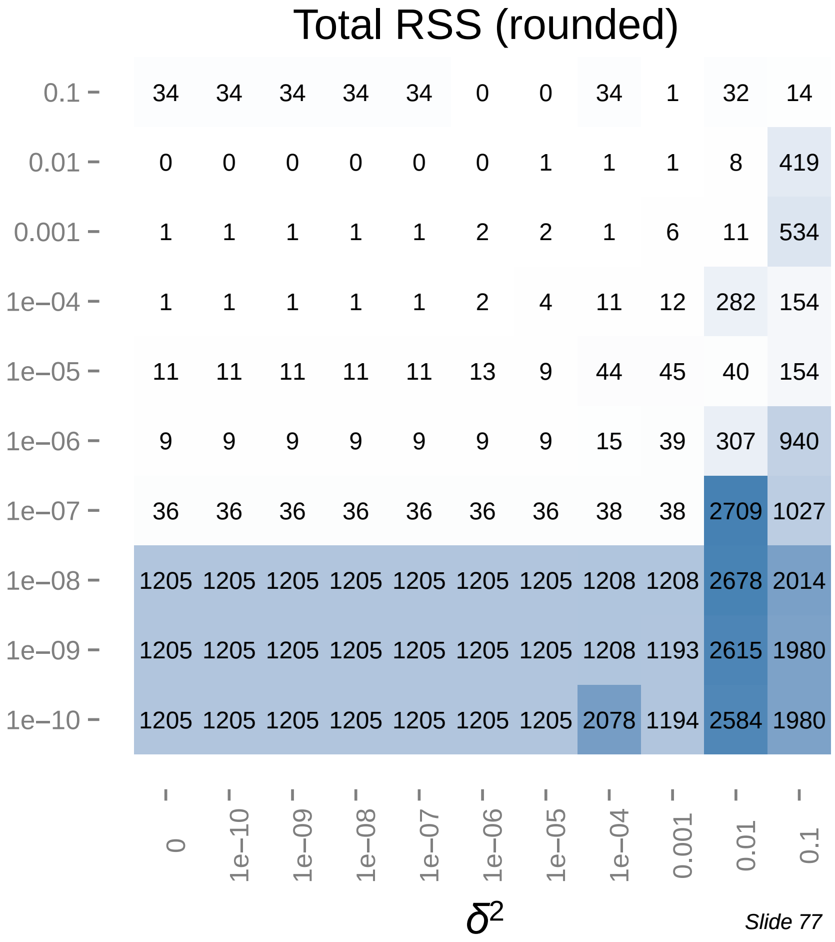
Tradeoff between number of partitions and total RSS

Ultimately would like to find an AIC-like criterion to balance this.



Number of partitions

α \ δ²	0	1e−10	1e−09	1e−08	1e−07	1e−06	1e−05	1e−04	0.001	0.01	0.1
0.1	221	210	191	166	145	129	111	84	48	27	13
0.01	203	190	173	148	131	115	94	62	40	17	11
0.001	173	167	150	130	116	96	73	47	25	16	9
1e−04	120	115	105	84	73	62	42	28	18	11	9
1e−05	60	64	46	38	35	31	29	15	12	11	9
1e−06	30	30	27	26	25	23	21	15	11	10	5
1e−07	20	18	18	18	17	15	14	11	11	3	5
1e−08	11	10	10	10	9	10	10	7	7	3	3
1e−09	10	10	10	10	9	9	9	7	7	3	3
1e−10	10	10	10	10	9	9	9	4	7	3	3



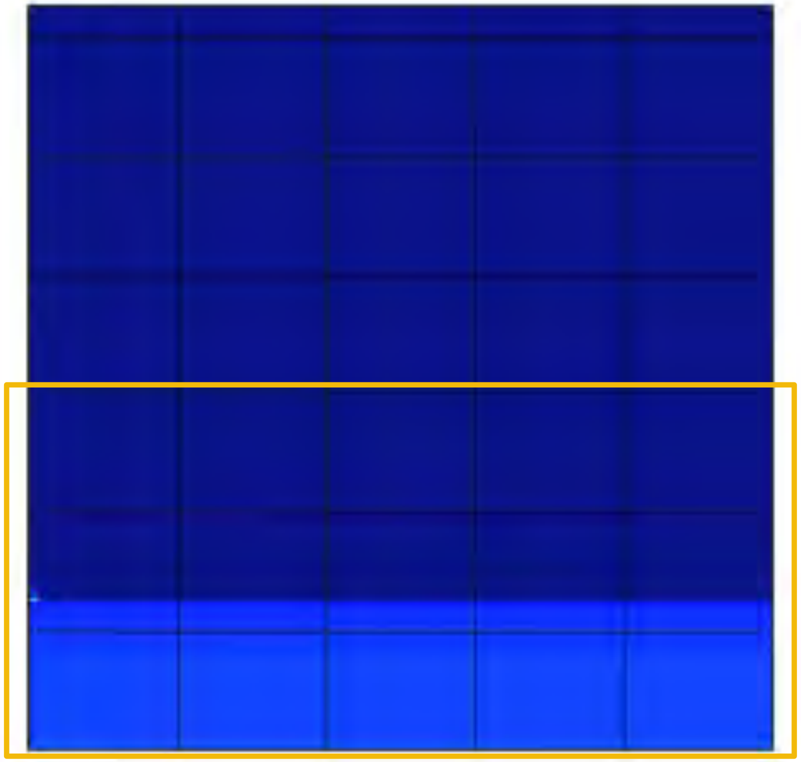
Total RSS (rounded)

α \ δ²	0	1e−10	1e−09	1e−08	1e−07	1e−06	1e−05	1e−04	0.001	0.01	0.1
0.1	34	34	34	34	34	0	0	34	1	32	14
0.01	0	0	0	0	0	0	1	1	1	8	419
0.001	1	1	1	1	1	2	2	1	6	11	534
1e−04	1	1	1	1	1	2	4	11	12	282	154
1e−05	11	11	11	11	11	13	9	44	45	40	154
1e−06	9	9	9	9	9	9	9	15	39	307	940
1e−07	36	36	36	36	36	36	36	38	38	2709	1027
1e−08	1205	1205	1205	1205	1205	1205	1205	1208	1208	2678	2014
1e−09	1205	1205	1205	1205	1205	1205	1205	1208	1193	2615	1980
1e−10	1205	1205	1205	1205	1205	1205	1205	2078	1194	2584	1980

Incorporating spatial characteristics of the simulation

A simple initial approach with the LCROSS simulation:

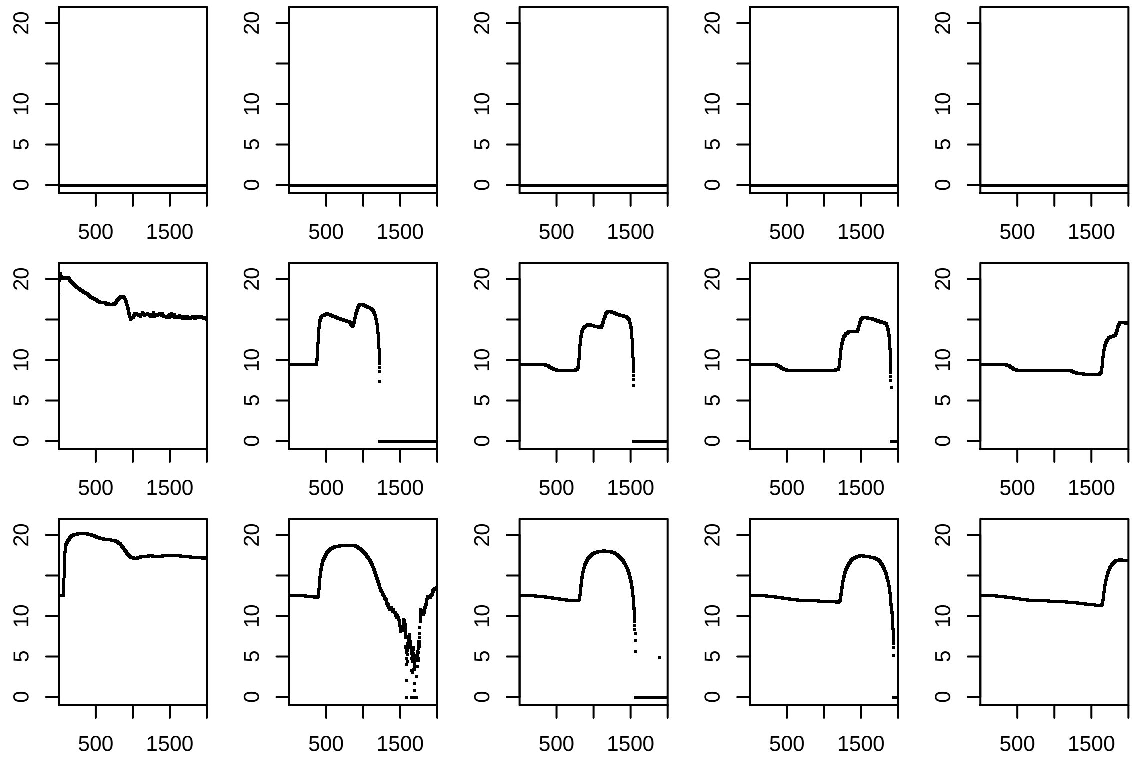
- Split the simulation frames into blocks.
- For each block and each time step, compute the mean over pixels.
- Apply the method to the trace of each block mean.



Pressure

Incorporating spatial characteristics of the simulation

Applying our approach to pixel means for different regions of the simulation with $\alpha = 0.001$, $\delta^2 = 0.001$, $B = 5$.



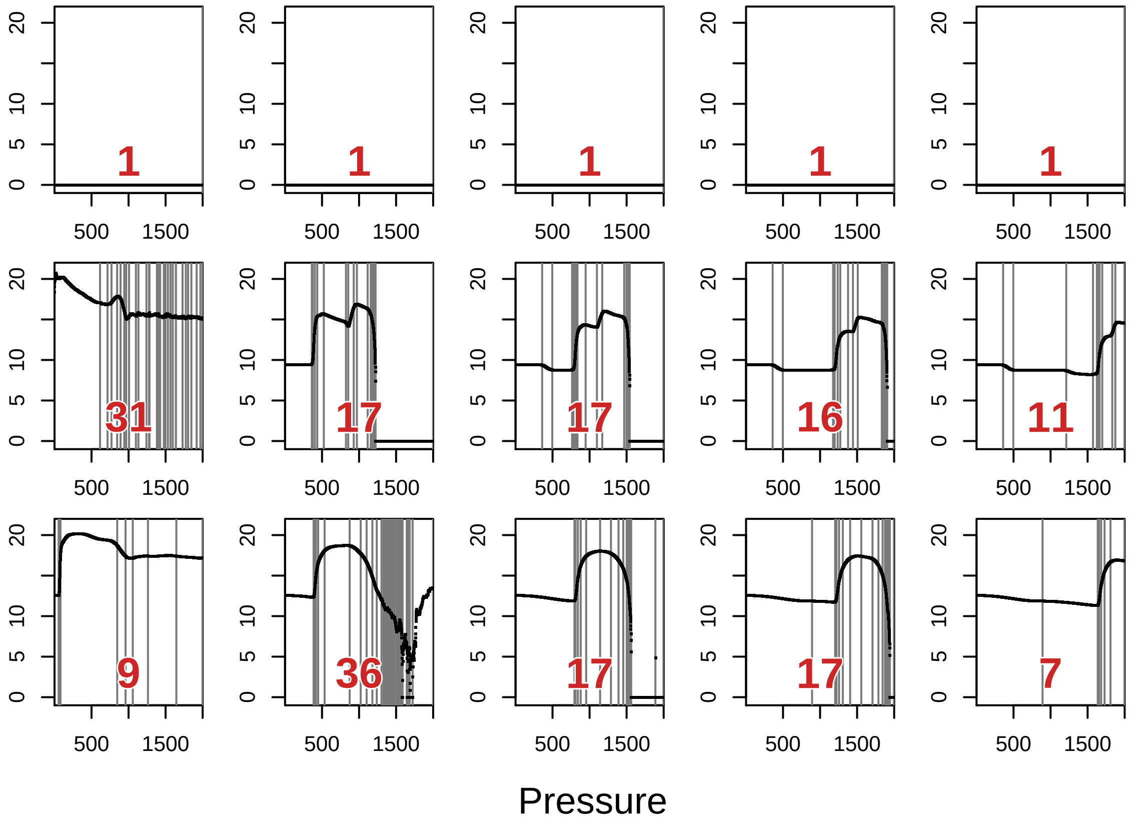
Pressure

Atmosphere blocks

Doomed rocket stage lands in this block.

Incorporating spatial characteristics of the simulation

Applying our approach to pixel means for different regions of the simulation with $\alpha = 0.001$, $\delta^2 = 0.001$, $B = 5$.



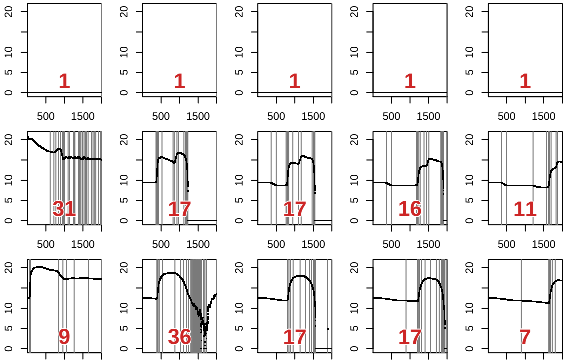
Atmosphere blocks

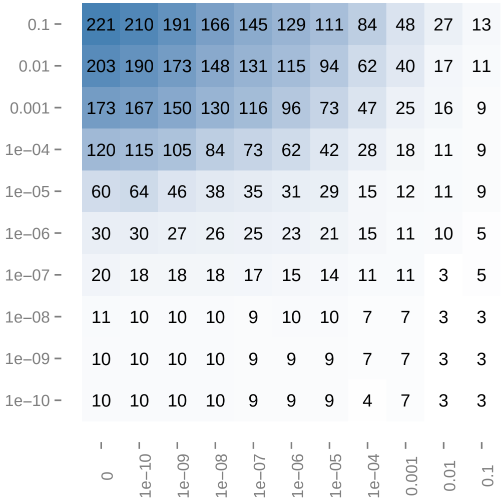
Doomed rocket stage lands in this block.

Lots of potential next directions

To name just a few:

- Using our partitioning approach to define spatial regions as the simulation evolves.
- Identifying a mathematical criterion to guide selection of α and δ^2.
- Incorporating other types of fits that can be cheaply computed and updated.
- Handling multivariate trajectories.





	0	1e−10	1e−09	1e−08	1e−07	1e−06	1e−05	1e−04	0.001	0.01	0.1
0.1	221	210	191	166	145	129	111	84	48	27	13
0.01	203	190	173	148	131	115	94	62	40	17	11
0.001	173	167	150	130	116	96	73	47	25	16	9
1e−04	120	115	105	84	73	62	42	28	18	11	9
1e−05	60	64	46	38	35	31	29	15	12	11	9
1e−06	30	30	27	26	25	23	21	15	11	10	5
1e−07	20	18	18	18	17	15	14	11	11	3	5
1e−08	11	10	10	10	9	10	10	7	7	3	3
1e−09	10	10	10	10	9	9	9	7	7	3	3
1e−10	10	10	10	10	9	9	9	4	7	3	3

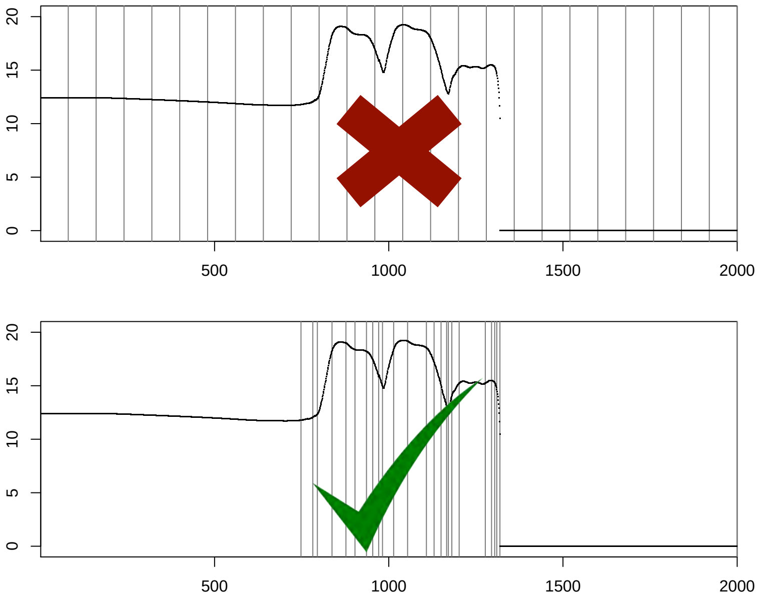
But for now…

…our *in situ* approach is cheap to compute and update, and it provides:

- Substantial memory savings over storing the full output of the simulation.

- Improved fidelity to the simulation over selecting evenly spaced partitions.

- Ability to reconstruct a linear approximation of the simulation with known error.



The end

More details: arxiv.org/abs/1409.0909

Or: The end!

More details: arxiv.org/abs/1409.0909

Also: Statistics and Beer Day
June 13

Some math

In a typical simulation setting, a scalar response y_i will be an unknown deterministic function of time t_i:

$$y_i = \mathcal{F}(t_i),\ i = 1, \ldots, T$$

where T is the total number of time steps in the simulation. Our goal is to approximate this function and locate interesting changes:

$$y_i = f(t_i) + \epsilon_i,\ i = 1, \ldots, T$$

Let P_0, P_1, …, P_m be a set of breakpoints of the sequence 1, …, T, with $P_0 = 0$ and P_m=T. The function f can be written as a sum over the partitions defined by the breakpoints:

$$f(t_i) = \sum_{j=1}^{m} (\beta_{j,0} + \beta_{j,1} t_i)\, I\{P_{j-1} < i \leq P_j\}$$

To fit the model, we need to estimate the number of partitions, the breakpoints, and the regression coefficients.

Sufficient statistics

$$
\begin{aligned}
\theta &= \sum t_i \\
\Theta &= \sum t_i^2 \\
\psi &= \sum y_i \\
\Psi &= \sum y_i^2 \\
\tau &= \sum t_i y_i \\
T_\bullet &
\end{aligned}
$$

Compute the residual sum of squares (RSS) and the slope and intercept:

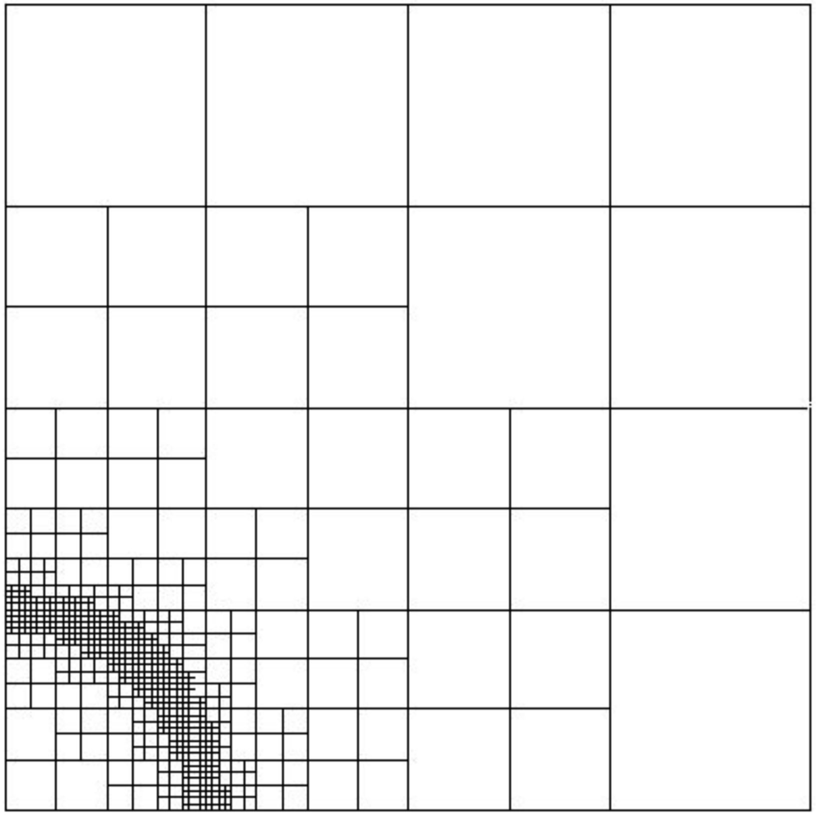
$$
\begin{aligned}
RSS &= \Psi - \frac{1}{T_\bullet}\psi^2 - \frac{(\tau - \theta\psi/T_\bullet)^2}{\Theta - \theta^2/T_\bullet} \\
\hat{\beta}_0 &= \frac{1}{T_\bullet}(\psi - \hat{\beta}_1\theta) \\
\hat{\beta}_1 &= \frac{\tau - \theta\psi/T_\bullet}{\Theta - \theta^2/T_\bullet}
\end{aligned}
$$

RAGE uses adaptive mesh refinement (AMR)

- Considers **spatial variation** in each variable to choose cell size.

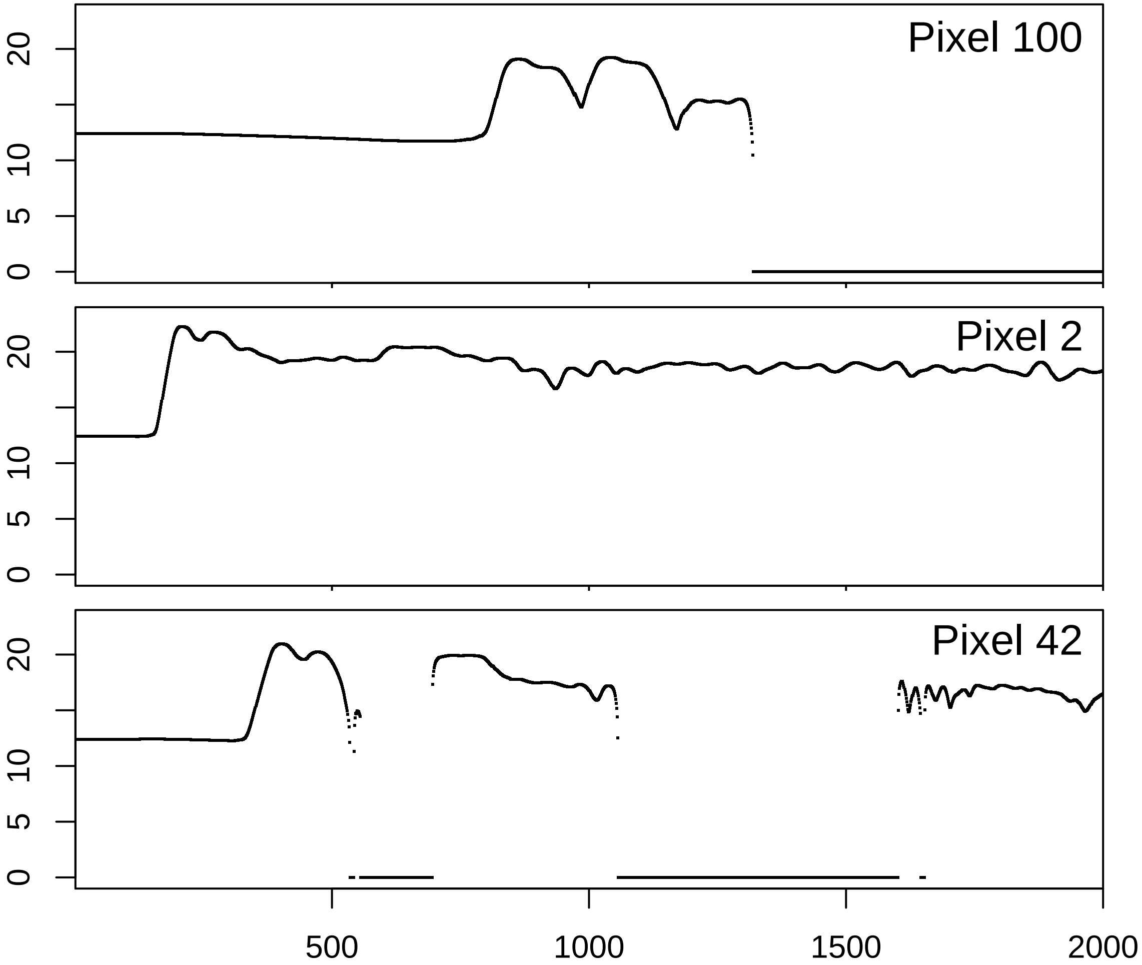
- Makes decisions to split / merge cells **at each time step**.

- **Constrains splits and merges** so adjacent cells are within 1 level of each other.



Gittings et al. 2008

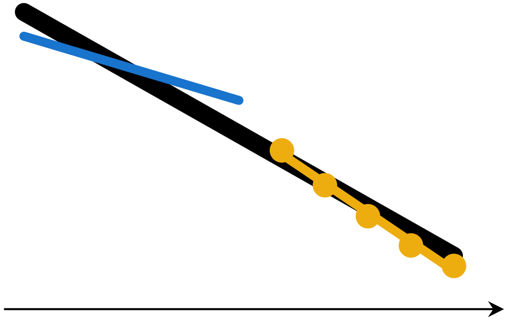
Other examples of pixel trajectories



We describe capturing the lines with sufficient statistics

But in practice, these sums can get too large to be computationally stable.

$$T_\bullet, \sum t_i, \sum t_i^2, \sum y_i, \sum y_i^2, \sum t_i y_i$$



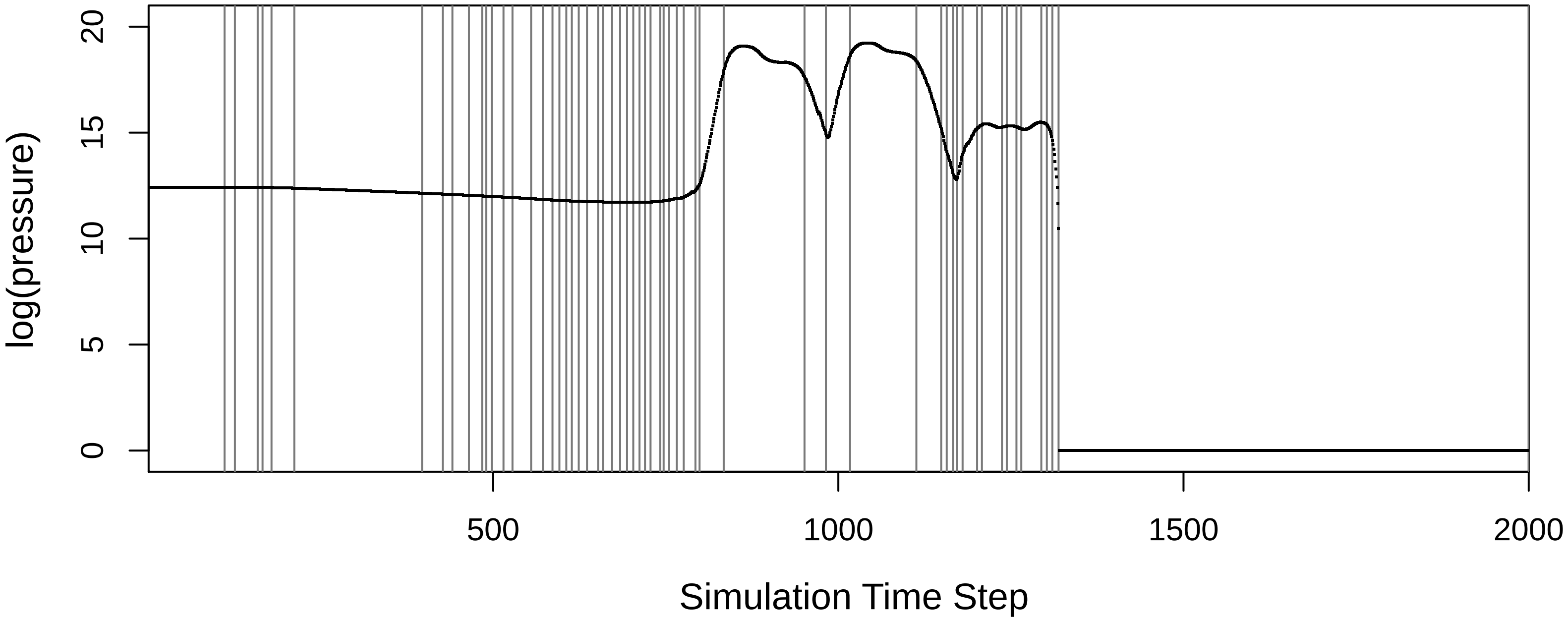
An alternative: incremental QR decomposition:

Miller (1992). Algorithm AS 274: Least Squares Routines to Supplement Those of Gentleman, *Journal of the Royal Statistical Society. Series C (Applied Statistics),* Vol. 41, No. 2, pp. 458-478

This is implemented in the R package `biglm`.

Adjusting α alone doesn't make the decisions we want

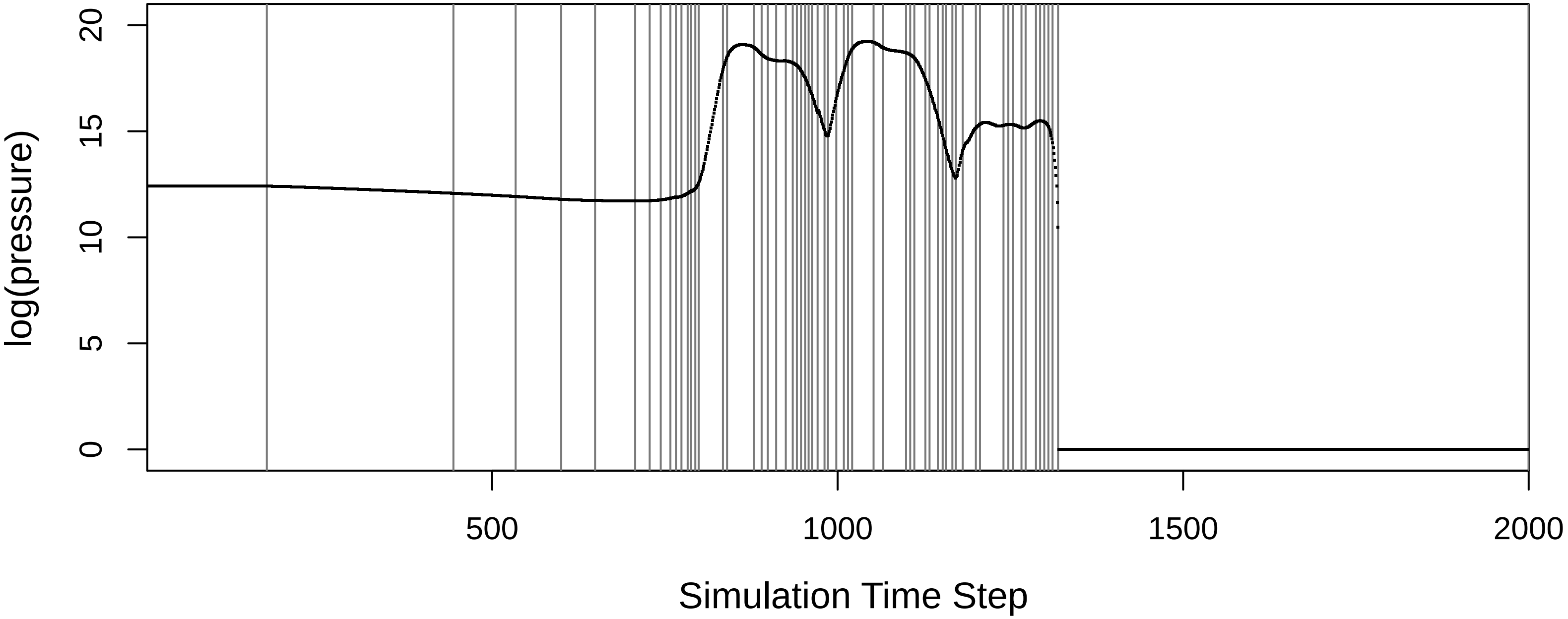
60 partitions via $\alpha = 1 \times 10^{-5}$, $\delta^2 = 0$.



Total RSS: 11.05

Adjusting α alone doesn't make the decisions we want

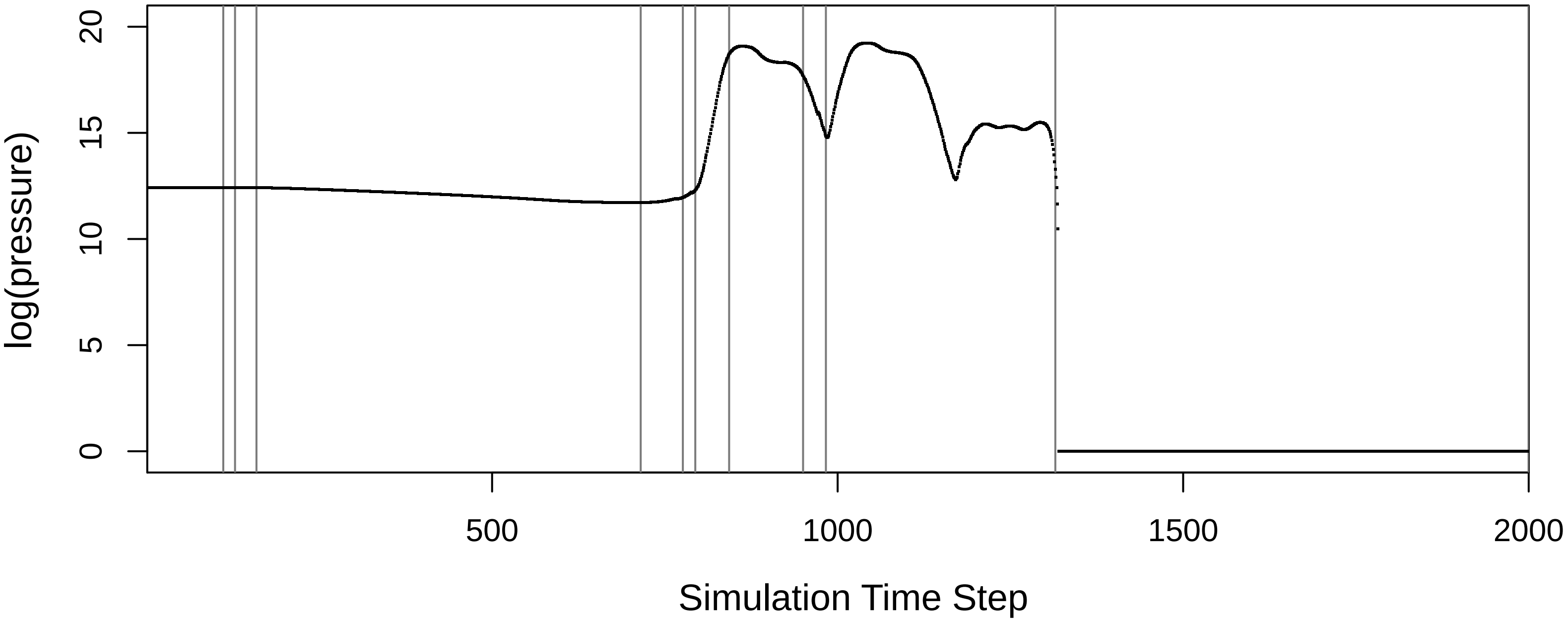
62 partitions via $\alpha = 1 \times 10^{-4}$, $\delta^2 = 1 \times 10^{-6}$.



Total RSS: 1.51

Adjusting α alone doesn’t make the decisions we want

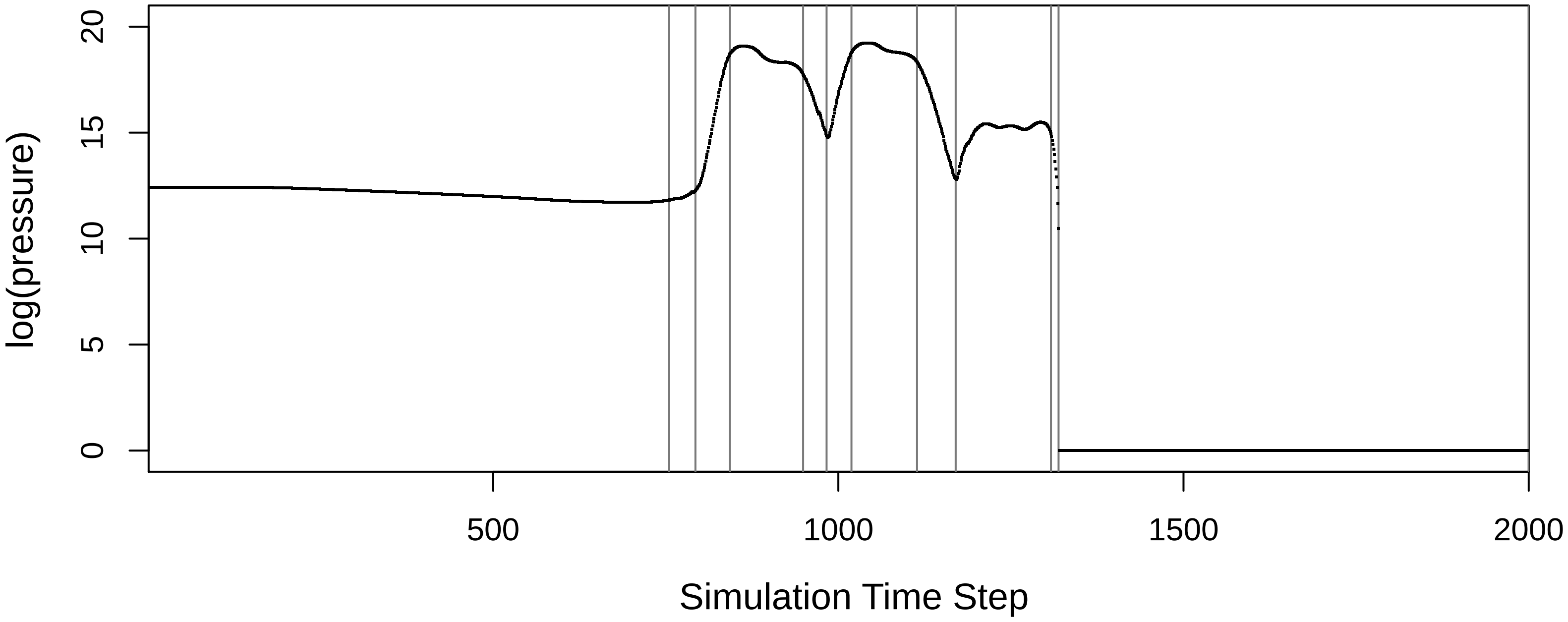
11 partitions via $\alpha = 1 \times 10^{-8}$, $\delta^2 = 0$.



Total RSS: 1205.31

Adjusting *α* alone doesn't make the decisions we want

11 partitions via $\alpha = 1 \times 10^{-7}$, $\delta^2 = 1 \times 10^{-4}$.



Total RSS: 38.33