Automatic Quantization for Physics-Based Simulation – Supplementary Materials

JIAFENG LIU[*], State Key Laboratory of CAD&CG, Zhejiang University, China
HAOYANG SHI[*], State Key Laboratory of CAD&CG, Zhejiang University, China
SIYUAN ZHANG, State Key Laboratory of CAD&CG, Zhejiang University, China
YIN YANG, Clemson University & University of Utah, USA
CHONGYANG MA, Kuaishou Technology, China
WEIWEI XU[†], State Key Laboratory of CAD&CG, Zhejiang University, China

1 QUANTIZATION SCHEMES OF THE EXPERIMENTS

We show the quantization schemes used in each experiment. Note that the ranges shown in the tables are the maximum distance to zero for the signed fix-point numbers. In the tables that contain the computed quantization schemes, We use X, V, C, F, and J to denote the position, velocity, local affine velocity field, deformation gradient, and the determinant of F, respectively.

Error-bound experiments. The quantization schemes of the error-bound experiments in Section 8.1 of our main paper can be found in Tables 1 and 2.

Memory-bound experiments. The quantization schemes of the memory-bound experiments in Section 8.1 can be found in Table 4.

2 OPTIMALITY CHECK, DITHERING, AND PERFORMANCE BENCHMARK ON MPM

The quantization schemes of the optimality check, dithering, and performance benchmark on 2D MPM experiments in Sections 8.1 and 8.2 of our main text can be found in Table 6. For the optimality check experiment, the quantization scheme we present in the table is the base scheme from which all the other testing schemes originate. Besides, the placement of custom data types using *bit struct* or *bit pack* is demonstrated in the following code:

```
particles = ti.root.dense(ti.i, N) # N=8000
# using bit struct
particles.bit_struct(num_bits=64).place(x(0),x(1),v(0))
particles.bit_struct(num_bits=64).place(v(1)),C(0, 0),C
     (0, 1),C(1, 0), C(1, 1))
particles.bit_struct(num_bits=64).place(F(0, 0),F(0, 1),
     F(1, 0))
particles.bit_struct(num_bits=32).place(F(1, 1))

# using bit pack
particles.bit_pack().place(x, v, C, F)
```

As for the 3D MPM performance benchmark, we present the quantization scheme in Table 5 and demonstrate the layout of the custom data types using the following code:

[*]Joint first authors.
[†]Corresponding author: Weiwei Xu (xww@cad.zju.edu.cn)

Authors' addresses: Jiafeng Liu, jiafengliu@zju.edu.cn, State Key Laboratory of CAD&CG, Zhejiang University, China; Haoyang Shi, shay@zju.edu.cn, State Key Laboratory of CAD&CG, Zhejiang University, China; Siyuan Zhang, zhang_sy@zju.edu.cn, State Key Laboratory of CAD&CG, Zhejiang University, China; Yin Yang, yin5@clemson.edu, Clemson University & University of Utah, USA; Chongyang Ma, chongyangma@kuaishou.com, Kuaishou Technology, China; Weiwei Xu, xww@cad.zju.edu.cn, State Key Laboratory of CAD&CG, Zhejiang University, China.

Table 1. Quantization schemes for the error bound experiments of 2D MPM elastic body.

Attributes	Relative error tolerance				Ranges
	0.1	0.01	0.001	0.0001	
X_x	23	26	29	32	1
X_y	22	26	29	32	1
V_x	17	20	23	27	9.398
V_y	17	20	23	27	12.99
$C_{0,0}$	9	12	16	19	7.262×10^2
$C_{0,1}$	11	15	18	21	1.187×10^3
$C_{1,0}$	12	15	18	22	1.452×10^3
$C_{1,1}$	9	12	16	19	8.151×10^2
$F_{0,0}$	18	21	25	28	4.629
$F_{0,1}$	17	21	24	27	4.759
$F_{1,0}$	18	21	24	28	6.725
$F_{1,1}$	18	21	25	28	5.337

Table 2. Bit count and range derived in 2D error-bounded Eulerian fluid simulation. **v** and **p** represent velocity and pressure, respetively.

Relative error tolerance	v Bits	p Bits	v Range	p Range
0.1	7	5	2.03	1.07
0.01	11	8		
0.001	14	12		
0.0001	17	15		

Table 3. Quantization schemes applied in 3D smoke simulation.

v Bits	p Bits	v Range	p Range
17	13	34.05	4.63

```
particles = ti.root.dense(ti.i, N) # N=30,000,000
# using bit struct
particles.bit_struct(num_bits=64).place(x(0),x(1))
particles.bit_struct(num_bits=64).place(x(2),v(0),v(1))
particles.bit_struct(num_bits=64).place(v(2),C(0,0),C
     (0,1),C(0,2),C(1,0))
particles.bit_struct(num_bits=64).place(C(1,1),C(1,2),C
     (2,0),C(2,1),C(2,2))
```

Table 4. Quantization schemes for the experiments about compression rates.

Attributes	Compression rate				Ranges
	0.4	0.5	0.6	0.7	
X_x	18	21	24	27	1
X_y	18	21	25	28	1
V_x	13	16	19	22	8.570
V_y	13	16	20	23	1.140
$C_{0,0}$	9	12	16	19	7.441×10^2
$C_{0,1}$	9	12	15	18	8.479×10^2
$C_{1,0}$	10	13	16	20	8.633×10^2
$C_{1,1}$	9	12	16	19	7.314×10^2
J	11	13	17	20	0.3140

```
particles.bit_struct(num_bits=64).place(F(0,0),F(0,1),F
    (0,2))
particles.bit_struct(num_bits=64).place(F(1,0),F(1,1),F
    (1,2))
particles.bit_struct(num_bits=64).place(F(2,0),F(2,1),F
    (2,2))

# using bit pack
particles.bit_pack().place(x, v, C, F)
```

Comparison with QuanTaichi [Hu et al. 2021]. The quantization scheme which is used in the comparison with QuanTaichi is shown in Table 7.

Scalability. The quantization scheme of the scaling experiment in Section 8.3 of our main text can be found in Table 8.

Large-scale demos. The quantization schemes of the large-scale demo in Section 8.4 of our main text can be found in Tables 3 and 9.

3 ADDITIONAL EXPERIMENTS

3.1 Iterative Solutions

We iteratively solve the optimization for the 2D smoke simulation experiments to test the convergence of our method. Starting from 32 bits for each variable, we restrict the step size of changing bits in each iteration by setting a learning rate of 0.5. All test cases in this experiment converge after no more than six iterations. The results presented in Table 10 show that the final solution is very close to the non-iterative solution, which is obtained from the first iteration.

3.2 Simulation Error of Optimality Check

The simulation error of optimality check in Section 8.1 of our main text can be found in Fig. 1. The error is computed via $\sigma_\theta^2 + (\mu_\theta - z)^2$, where μ_θ and σ_θ denote the mean and standard deviation of the observed distribution θ.

3.3 Preliminary Feasibility Tests

All of the experiments in this section are conducted on 2D MPM elastics simulation with 5000 particles and 128^2 grids. The full-precision reference is `float64`, and the quantized simulator has 19 and 13 bits for position and velocity, respectively. The experiments

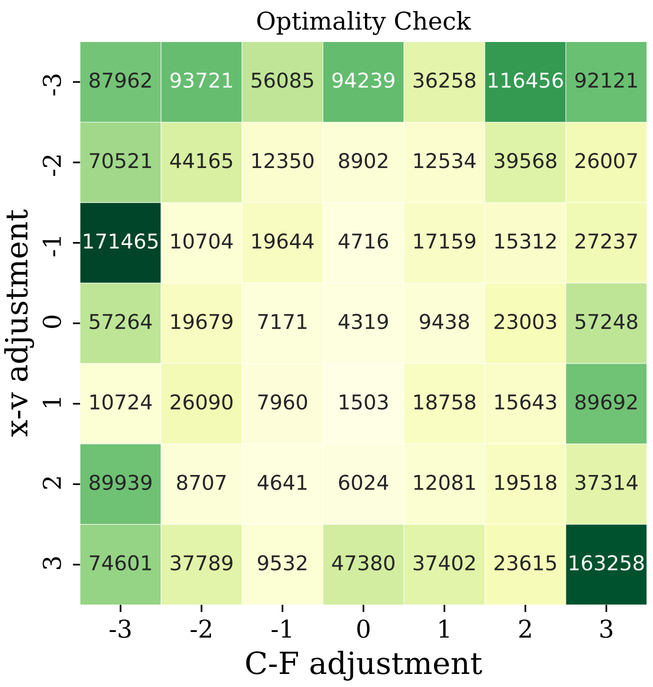

Fig. 1. Optimality check. Error measured in the square sum of standard deviation and bias. Positive x-axis: move bits from x to v; positive y-axis: move bits from C to F.

are implemented using Taichi programming language and run on the CUDA backend. The hardware is an NVIDIA GTX 1080Ti GPU with 11GB memory.

The quantization errors tend to accumulate exponentially with time. We recorded the point-to-point errors, which are shown in Fig. 2, suggesting that mean squared errors might not be an ideal evaluation function. Similarly, gradients also accumulate exponentially with growth in time (shown in Fig. 3), which gives us a hint on the feasibility of using accumulated gradients for error control.

By our linear model, one of our predictions is that with each bit added to all quantized types, the aggregated quantization error is reduced by half. Fig. 4 shows that this is generally true in this experimental setting, as the quantized mean value of evaluation function approaches the reference value roughly in an exponential manner.

3.4 Acceleration of Dithering Operations

The performance statistics of the dithering operation reported in our main paper are obtained using the default random number generator (RNG) algorithm in Taichi programming language [Hu et al. 2019], which includes at least four times global memory access to make the random number reproducible.

To avoid extra memory access, we borrow the idea from the GLSL random number function in Perlin noise. Specifically, we use the following formula: $r = F(\sin(x \times 52.1063) \times 43758.5453123)$, where x is the number to be dithered, r is the random number, and the function F returns the fractional part of the number. However, the

Table 5. Quantization schemes for the 3D MPM performance experiments. The range values of all the $\{C\}$ are scaled by a factor of 10^{-3}.

	X_x	X_y	X_z	V_x	V_y	V_z	$C_{0,0}$	$C_{0,1}$	$C_{0,2}$	$C_{1,0}$	$C_{1,1}$	$C_{1,2}$	$C_{2,0}$	$C_{2,1}$	$C_{2,2}$	$F_{0,0}$	$F_{0,1}$	$F_{0,2}$	$F_{1,0}$	$F_{1,1}$	$F_{1,2}$	$F_{2,0}$	$F_{2,1}$	$F_{2,2}$
Bits	24	23	23	17	17	15	11	14	11	13	11	10	11	12	10	19	18	18	18	18	17	17	17	18
Ranges	4	4	4	3.811	4.299×10^1	3.316×10^1	3.683	9.041	3.286	6.865	5.328	3.070	3.708	6.660	3.490	7.424	7.992	7.768	8.340	8.423	6,283	6.133	5.396	5.734

Table 6. Quantization schemes for optimality check, dithering, and performance benchmarks on MPM.

	Dithering	Bits: Optimality Check Base	MPM Performance	Ranges
X_x	19	26	23	1
X_y	19	26	23	1
V_x	13	20	17	9.398
V_y	13	20	17	1.299
$C_{0,0}$	9	12	10	7.262×10^2
$C_{0,1}$	10	15	12	1.128×10^3
$C_{1,0}$	10	15	12	1.452×10^3
$C_{1,1}$	9	12	10	8.151×10^2
$F_{0,0}$	14	21	19	4.629
$F_{0,1}$	14	21	18	4.759
$F_{1,0}$	14	21	18	6.725
$F_{1,1}$	15	21	19	5.337

Table 7. Quantization schemes used in comparison with QuanTaichi. Note that we only use the fixed-point data format while the velocity is represented by a custom float type (10 fraction bits and 6 exponent bits) in QuanTaichi.

Attributes	Our method		QuanTaichi	
	Bits	Ranges	Bits	Ranges
X_x	18	1.0	20	2.0
X_y	18	1.0	20	2.0
V_x	13	18.153	frac: 10, exp: 6	/
V_y	12	18.810	frac: 10, exp: 6	/
$F_{0,0}$	13	2.913	16	4.0
$F_{0,1}$	12	2.522	16	4.0
$F_{1,0}$	12	2.702	16	4.0
$F_{1,1}$	12	2.465	16	4.0

computation of the sinusoidal function is relatively expensive. So we use a quadratic function S to compute the sine value approximately: $S(x) = \frac{4}{\pi}x - \frac{4}{\pi^2}x|x|, x \in [-\pi, \pi]$. In this way, we can accelerate dithering by up to a factor of 6.12 in the benchmark test. Note that, although the extra global I/O overhead can be alleviated in this way, the quality of the randomness is degraded slightly. The results of the performance benchmark of fast dithering are shown in Table. 11. In most cases, the performance of our fast dithering implementation is very close to the result of the experiment without dithering. However, as the quality of randomness decreases, the simulation errors become larger than the default dithering algorithm. We evaluate the effectiveness by comparing both algorithms in the dithering experiment of Section 8.1 in our main paper. The result of our fast dithering algorithm is shown in the third row in Table 12,

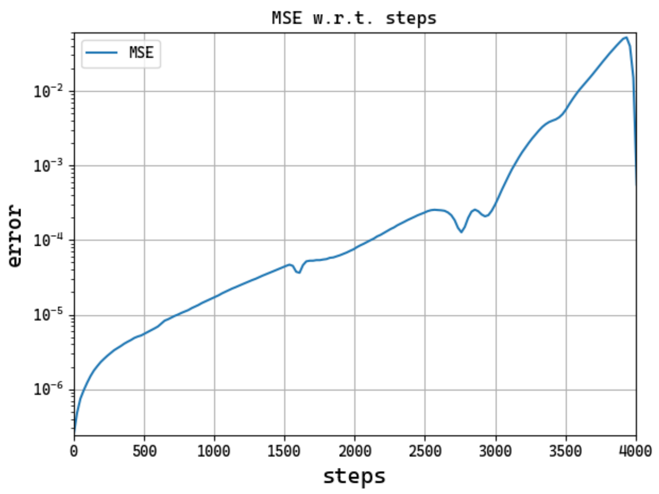

Fig. 2. Mean Square Error is not an ideal canary of the visual quality as false-positive frequents. It undergoes exponential growth with respect to time, but in most cases, large MSE does not directly translate to degraded visual effects.

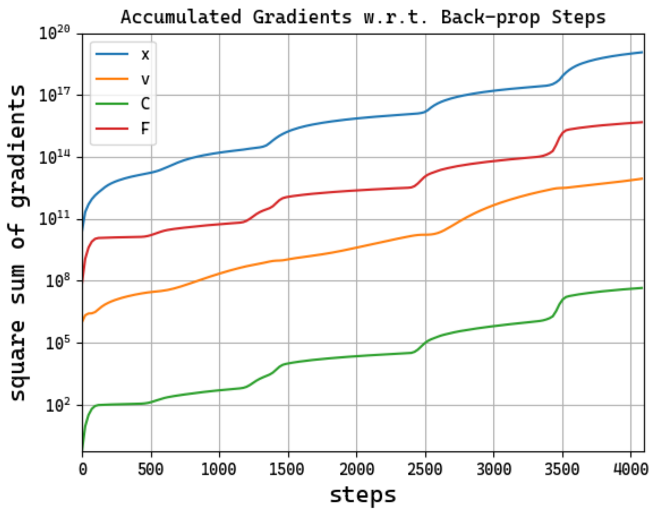

Fig. 3. Accumulated gradients grow exponentially with growth in the number of back-propagation steps. Recorded in 2D MLS-MPM elastics simulation.

and the statistics in the rest rows are copied from Table 5 in the main paper. It verifies that the quality of simulation results based on our accelerated dithering operations is comparable to the default dithering implementation.

Table 8. Quantization schemes for the scaling experiment. The range values of all the $\{C\}$ are scaled by a factor of 10^{-3}.

	X_x	X_y	X_z	V_x	V_y	V_z	$C_{0,0}$	$C_{0,1}$	$C_{0,2}$	$C_{1,0}$	$C_{1,1}$	$C_{1,2}$	$C_{2,0}$	$C_{2,1}$	$C_{2,2}$	J
Bits	22	23	22	17	16	16	14	17	15	14	15	13	14	16	14	16
Ranges	1	1	1	1.103×10^1	1.129×10^1	1.345×10^1	1.793	1.857	1.770	1.901	1.729	1.546	1.921	2.558	1.661	1.403

Table 9. Quantization schemes for the large-scale MPM simulations. The range values of all the $\{C\}$ are scaled by a factor of 10^{-3}.

	X_x	X_y	X_z	V_x	V_y	V_z	$C_{0,0}$	$C_{0,1}$	$C_{0,2}$	$C_{1,0}$	$C_{1,1}$	$C_{1,2}$	$C_{2,0}$	$C_{2,1}$	$C_{2,2}$	$F_{0,0}$	$F_{0,1}$	$F_{0,2}$	$F_{1,0}$	$F_{1,1}$	$F_{1,2}$	$F_{2,0}$	$F_{2,1}$	$F_{2,2}$	J
Bits: Elastic Body	22	23	22	18	18	16	9	14	11	14	10	9	11	11	9	19	18	18	18	19	18	18	18	18	-
Bits: Fluid	21	22	21	15	18	15	12	16	13	14	13	14	14	15	13	-	-	-	-	-	-	-	-	-	17
Ranges: Elastic Body	4	4	4	3.811	4.299×10^1	3.316×10^1	3.683	9.041	3.286	6.865	5.328	3.070	3.708	6.660	3.490	7.424	7.992	7.768	8.340	8.423	6,283	6.133	5.396	5.734	-
Ranges: Fluids	4	1	4	5.141	4.471×10^1	4.915	1.578	6.634	1.414	1.880	1.538	1.986	1.616	4.912	1.230	-	-	-	-	-	-	-	-	-	1.637

Table 10. Results obtained by solving the optimization for the 2D smoke experiments iteratively under the same relative error ϵ_{err}. The results show that the fraction bits converges after at most 6 iterations and the final solution is very close to the non-iterative result (See the last row).

ϵ_{err}	**0.1**	**0.01**	**0.001**	**0.0001**
Iteration 0	[20, 19]	[22, 20]	[23, 22]	[25, 24]
Iteration 1	[13, 12]	[16, 14]	[18, 17]	[20, 19]
Iteration 2	[9, 8]	[13, 11]	[15, 14]	[18, 17]
Iteration 3	[8 ,6]	[11, 9]	[14, 13]	[17, 16]
Iteration 4	[9, 7]	[10, 8]	[13, 12]	[17, 16]
Iteration 5	[8, 6]	-	-	-
Initial solution	[7, 5]	[11, 8]	[14, 12]	[17, 15]

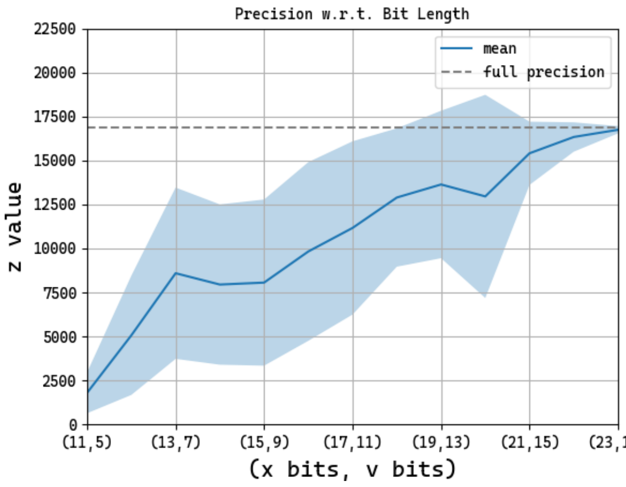

Fig. 4. Blue line and area depicts mean and variance with the number of fraction bits of x and v specified on the x axis. By our error estimation, aggregated error should reduce by half with every bit added to all quantized types. It is shown that the bias roughly follows this postulation as the blue line approximates the reference value (grey) exponentially, while the variance only begins to comply with our prediction after the bias and standard deviation become comparable.

Table 11. Benchmark with accelerated dithering. The time is measured in seconds.

Case	Backend	Fast dithering	No dithering
Store	x64	8.902	8.888
	CUDA	0.060	0.059
MatMul	x64	39.256	16.593
	CUDA	0.268	0.262
MPM 2D	x64	61.592	59.498
	CUDA	25.984	26.651
MPM 3D	CUDA	27.699	27.284

Table 12. The effectiveness of fast dithering. Round-ups/Round-downs indicates the ratio of round-ups operations to round-downs operations.

Data type	**Evaluation function**	**Round-ups/ Round-downs**
`float64`	1.378×10^5	-
Quantization with dithering	1.370×10^5	0.999976
Quantization with fast dithering	1.361×10^5	0.999635
Quantization without dithering	8.885×10^5	0.683994

Users may choose which type of dithering to use according to their preference for performance or quality. If not mentioned in the rest of this supplementary material, we choose the native RNG provided by Taichi to conduct dithering operations for simplicity.

3.5 Failure Cases

After setting a large compression rate of 30%, we re-run the experiment in Fig. 15 in our main paper with two modifications of initial conditions: an initial velocity of 3.0 and 4x times larger pressure. As shown in Fig. 5, noticeable artifacts appear in the simulation results, and the quantization schemes computed in this experiment are shown in Table 13. Comparing to the original quantization scheme listed in Table 8, it can be seen that the bits of each quantity drop significantly with the target compression rate of 30%, which is the cause of the artifacts.

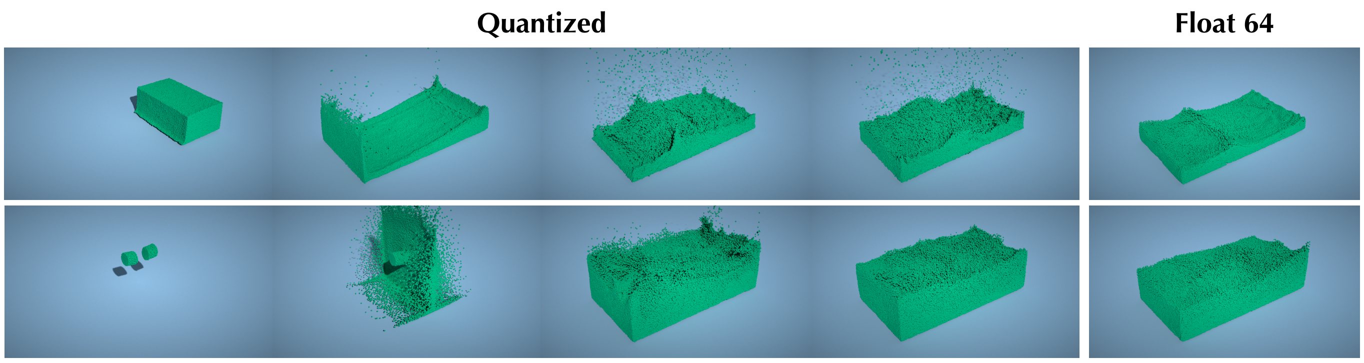

Fig. 5. Failure cases caused by different initial conditions. Top: we increase the pressure coefficient by a factor of 4, and the simulation becomes unstable. Bottom: after we use a large non-zero velocity 3.0, the volume of fluid increases significantly compared to the full-precision reference.

Table 13. Quantization schemes of the failure cases. The range values of all the $\{C\}$ are scaled by a factor of 10^{-3}.

	X_x	X_y	X_z	V_x	V_y	V_z	$C_{0,0}$	$C_{0,1}$	$C_{0,2}$	$C_{1,0}$	$C_{1,1}$	$C_{1,2}$	$C_{2,0}$	$C_{2,1}$	$C_{2,2}$	J
Bits	14	16	15	9	8	9	8	8	8	7	7	6	6	9	7	9
Ranges	1	1	1	1.103×10^1	1.129×10^1	1.345×10^1	1.793	1.857	1.770	1.901	1.729	1.546	1.921	2.558	1.661	1.403

4 ANALYTICAL SOLUTIONS FOR AUTOMATIC QUANTIZATION

4.1 Error-Bounded Quantization

The formulation in Section 4.2 is:

$$\min_{\triangle_h} \sum_{h=1}^{H} -P_h \log_2 \frac{\triangle_h}{R_h},$$
$$s.t. \frac{1}{12}\sum_{h=1}^{H} \triangle_h^2 \left(\sum_{t=0}^{T}\sum_{i=1}^{P_h}\left(\frac{\partial z}{\partial x_{t,h}^i}\right)^2\right) < (z\epsilon_{err})^2, \tag{1}$$
$$\triangle_1, \dots, \triangle_H > 0.$$

With the Lagrange multiplier method, we have:

$$L = \sum_{i=1}^{H} -P_h \log_2 \frac{\triangle_h}{R_h} + \lambda\left(\frac{1}{12}\sum_{h=1}^{H}\triangle_h^2\left(\sum_{t=0}^{T}\sum_{i=1}^{P_h}\left(\frac{\partial z}{\partial x_{t,h}^i}\right)^2\right) - (z\epsilon_{err})^2\right).$$

Setting the partial derivatives to zeros, we have:

$$\frac{\partial L}{\partial \triangle_h} = 0,\ \forall h = 1, \dots, H \tag{2}$$
$$\frac{\partial L}{\partial \lambda} = 0 \tag{3}$$

With algebraic derivation, we can obtain the solution as follows:

$$\lambda = \frac{\sum_{h=1}^{H} P_h}{2\ln 2\, \epsilon_{err}^2 z^2},$$
$$\triangle_h = \epsilon_{err} z \sqrt{\frac{12 P_h}{g_h \sum_{h=1}^{H} P_h}},$$

where $g_h = \sum_{t=0}^{T}\sum_{i=1}^{P_h}\left(\frac{\partial z}{\partial x_{t,h}^i}\right)^2$.

4.2 Memory-Bounded Quantization

The full formulation in Section 4.2 is:

$$\min_{\triangle_h} \frac{1}{12}\sum_{h=1}^{H}\triangle_h^2\left(\sum_{t=0}^{T}\sum_{i=1}^{P_h}\left(\frac{\partial z}{\partial x_{t,h}^i}\right)^2\right),$$
$$s.t. \sum_{h=1}^{H} -P_h \log_2 \frac{\triangle_h}{R_h} < M\epsilon_{mem}, \tag{4}$$
$$\triangle_1, \dots, \triangle_H > 0.$$

Its Lagrangian function can then be constructed as follows:

$$L = \frac{1}{12}\sum_{h=1}^{H}\triangle_h^2\left(\sum_{t=0}^{T}\sum_{i=1}^{P_h}\left(\frac{\partial z}{\partial x_{t,h}^i}\right)^2\right) + \lambda\left(\left(\sum_{h=1}^{H} -P_h \log_2 \frac{\triangle_h}{R_h}\right) - M\epsilon_{mem}\right).$$

The solution can be obtained by setting the partial derivatives to zero:

$$\frac{\partial L}{\partial \triangle_h} = 0,\ \forall h = 1, \dots, H \tag{5}$$
$$\frac{\partial L}{\partial \lambda} = 0 \tag{6}$$

Let $g_h = R_h^2 \sum_{t=0}^{T}\sum_{i=1}^{P_h}\left(\frac{\partial z}{\partial x_{t,h}^i}\right)^2$, and then substitute the variables $\triangle_h$ to $x_h = \frac{\triangle_h^2}{R_h^2}$ in (5), we have:

$$\frac{1}{12} g_h - \lambda \frac{P_h}{2\ln 2\, x_h} = 0,$$
$$x_h = \frac{6\lambda P_h}{\ln 2\, g_h}. \tag{7}$$

Substituting (7) to (6) yields:

$$\ln \lambda = \frac{-\sum_{h=1}^{H} P_h \ln \frac{P_h}{g_h} - 2\epsilon_{mem} \cdot M \ln 2}{\sum_{h=1}^{H} P_h} - \ln \frac{6}{\ln 2},$$

$$\ln x_h = \frac{-\sum_{h=1}^{H} P_h \ln \frac{P_h}{g_h} - 2\epsilon_{mem} \cdot M \ln 2}{\sum_{h=1}^{H} P_h} + \ln \frac{P_h}{g_h}.$$

With algebraic derivation, we can obtain the solution as follows:

$$\triangle_h = R_h \sqrt{\frac{P_h}{g_h}} \exp\left(\frac{-\sum_{h=1}^{H} P_h \ln \frac{P_h}{g_h} - 2\epsilon_{mem} \cdot M \ln 2}{2\sum_{h=1}^{H} P_h} \right).$$

REFERENCES

Yuanming Hu, Tzu-Mao Li, Luke Anderson, Jonathan Ragan-Kelley, and Frédo Durand. 2019. Taichi: a language for high-performance computation on spatially sparse data structures. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–16.

Yuanming Hu, Jiafeng Liu, Xuanda Yang, Mingkuan Xu, Ye Kuang, Weiwei Xu, Qiang Dai, William T. Freeman, and Frédo Durand. 2021. QuanTaichi: A Compiler for Quantized Simulations. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 182:1–182:16.