(Supplemental Material) MPM Time Step Breakdown for Hybrid Grains

YONGHAO YUE*, The University of Tokyo
BREANNAN SMITH*, Columbia University
PETER YICHEN CHEN*, Columbia University
MAYTEE CHANTHARAYUKHONTHORN*, Massachusetts Institute of Technology
KEN KAMRIN+, Massachusetts Institute of Technology
EITAN GRINSPUN+, Columbia University

CCS Concepts: • **Computing methodologies → Physical simulation**;

Additional Key Words and Phrases: Granular materials, Material point method, Contact dynamics, Constraints, Physical simulation, MPM Pseudocode

ACM Reference format:
Yonghao Yue*, Breannan Smith*, Peter Yichen Chen*, Maytee Chantharayukhonthorn*, Ken Kamrin+, and Eitan Grinspun+. 2018. (Supplemental Material) MPM Time Step Breakdown for Hybrid Grains. *ACM Trans. Graph.* 37, 6, Article 283 (November 2018), 2 pages.
https://doi.org/10.1145/3272127.3275095

1 MPM TIME STEP BREAKDOWN

Algorithm 1 `MPM_Step`

```
MPM_Step_First_Phase                          ▷ Alg. 2
MPM_Step_Second_Phase                         ▷ Alg. 7
```

Algorithm 2 `MPM_Step_First_Phase`

```
Rasterize_Mass_And_Momentum_To_Grid           ▷ Alg. 3
Compute_Stress_At_Points                      ▷ Alg. 4
Compute_Forces_On_Grid                        ▷ Alg. 5
Update_Momentum_On_Grid                       ▷ Alg. 6
```

Algorithm 3 `Rasterize_Mass_And_Momentum_To_Grid`

```
for point ∈ Material_Points do
    for node ∈ Stencil(point) do
        w ← Weight(point, node)
        node.m += w · point.m
        node.p += w · point.m · point.v
    end for
end for
```

Algorithm 4 `Compute_Stress_At_Points`

```
for point ∈ Material_Points do
    τ ← 0
    if point.J ≤ 1 then
        τ ← (κ/2) · (point.J^2 − 1) · I + μ · dev[point.b̄^e]
    end if
    point.σ ← τ / point.J
end for
```

Algorithm 5 `Compute_Forces_On_Grid`

```
for point ∈ Material_Points do
    for node ∈ Stencil(point) do
        ∇w ← Weight_Grad(point, node)
        node.f += −point.V · point.J · point.σ · ∇w
    end for
end for
for node ∈ Grid_Nodes do
    node.f += node.m · g
end for
```

Algorithm 6 `Update_Momentum_On_Grid`

```
for node ∈ Grid_Nodes do
    node.p_new ← node.p + dt · node.f
end for
```

Algorithm 7 `MPM_Step_Second_Phase`

```
Lumped_Mass_Velocity_Update_On_Grid           ▷ Alg. 8
Compute_Velocity_Gradient_At_Points           ▷ Alg. 9
Elastic_Prediction_At_Points                  ▷ Alg. 10
Plastic_Correction_At_Points                  ▷ Alg. 11
Update_Velocities_At_Points                   ▷ Alg. 12
Update_Positions_At_Points                    ▷ Alg. 13
```

Algorithm 8 `Lumped_Mass_Velocity_Update_On_Grid`

```
for node ∈ Grid_Nodes do
    node.v ← node.p_new / node.m
    node.a ← (node.p_new − node.p) /(dt · node.m)
end for
```

*Co-first authors — authors contributed equally.
+Corresponding authors (e-mail: kkamrin@mit.edu, eitan@cs.columbia.edu).

0730-0301/2018/11-ART283 $15.00
https://doi.org/10.1145/3272127.3275095

Algorithm 9 `Compute_Velocity_Gradient_At_Points`

for point ∈ Material_Points **do**
  point.$\nabla \boldsymbol{v} \leftarrow 0$
  for node ∈ Stencil(point) **do**
    $\nabla w \leftarrow$ Weight_Grad(point, node)
    point.$\nabla \boldsymbol{v}$ += point.$\boldsymbol{v} \cdot \nabla w^T$
  end for
end for

Algorithm 10 `Elastic_Prediction_At_Points`

for point ∈ Material_Points **do**
  $\boldsymbol{b}^e \leftarrow$ point.$J \cdot$ point.$\bar{\boldsymbol{b}}^e$
  $\boldsymbol{b}^{e*} \leftarrow \boldsymbol{b}^e + \mathrm{dt} \cdot \left(\text{point.}\nabla \boldsymbol{v} \cdot \boldsymbol{b}^e + \boldsymbol{b}^e \cdot \text{point.}\nabla \boldsymbol{v}^T\right)$
  point.$J \leftarrow \sqrt{\det\left(\boldsymbol{b}^{e*}\right)}$
  point.$\bar{\boldsymbol{b}}^e \leftarrow \boldsymbol{b}^{e*}$ / point.J ▷ if 2D
  point.$\bar{\boldsymbol{b}}^e \leftarrow \boldsymbol{b}^{e*}$ / (point.$J)^{2/3}$ ▷ if 3D
end for

Algorithm 11 `Plastic_Correction_At_Points`

for point ∈ Material_Points **do**
  yield_threshold $\leftarrow -\alpha \cdot \frac{\kappa}{2} \cdot (\text{point.}J^2 - 1)$
  $\mathrm{dev}[\boldsymbol{b}^e] \leftarrow \text{point.}\boldsymbol{b}^e - \frac{1}{2} \cdot \mathrm{Tr}[\text{point.}\boldsymbol{b}^e] \cdot \boldsymbol{I}$ ▷ if 2D
  $\mathrm{dev}[\boldsymbol{b}^e] \leftarrow \text{point.}\boldsymbol{b}^e - \frac{1}{3} \cdot \mathrm{Tr}[\text{point.}\boldsymbol{b}^e] \cdot \boldsymbol{I}$ ▷ if 3D
  $\mathrm{dev}[\bar{\boldsymbol{b}}^e] \leftarrow \mathrm{dev}[\boldsymbol{b}^e]$ / point.J ▷ if 2D
  $\mathrm{dev}[\bar{\boldsymbol{b}}^e] \leftarrow \mathrm{dev}[\boldsymbol{b}^e]$ / (point.$J)^{2/3}$ ▷ if 3D
  if $\mu \cdot \| \mathrm{dev}[\bar{\boldsymbol{b}}^e] \|_F >$ yield_threshold **then**
    $\lambda_2 \leftarrow$ yield_threshold $/ \left(\mu \cdot \| \mathrm{dev}[\bar{\boldsymbol{b}}^e] \|_F\right)$
    $\lambda_1 \leftarrow \sqrt{\det[\text{point.}\boldsymbol{b}^e] - \lambda_2^2 \cdot \det[\mathrm{dev}[\boldsymbol{b}^e]]}$ ▷ if 2D
    Solve (17) for λ_1 using Cardano's method ▷ if 3D
    point.$\boldsymbol{b}^e \leftarrow \lambda_1 \cdot \boldsymbol{I} + \lambda_2 \cdot \mathrm{dev}[\boldsymbol{b}^e]$
  end if
end for

Algorithm 12 `Update_Velocities_At_Points`

for point ∈ Material_Points **do**
  $\boldsymbol{v}_{pic} \leftarrow 0$
  $\boldsymbol{a}_{flip} \leftarrow 0$
  for node ∈ Stencil(point) **do**
    $w \leftarrow$ Weight(point, node)
    $\boldsymbol{v}_{pic}$ += $w \cdot$ node.$\boldsymbol{v}$
    $\boldsymbol{a}_{flip}$ += $w \cdot$ node.$\boldsymbol{a}$
  end for
  $\boldsymbol{v}_{flip} \leftarrow$ point.$\boldsymbol{v} + \mathrm{dt} \cdot \boldsymbol{a}_{flip}$
  point.$\boldsymbol{v} \leftarrow (1-\beta) \cdot \boldsymbol{v}_{pic} + \beta \cdot \boldsymbol{v}_{flip}$
end for

Algorithm 13 `Update_Positions_At_Points`

for point ∈ Material_Points **do**
  $\boldsymbol{v}_{pic} \leftarrow 0$
  for node ∈ Stencil(point) **do**
    $w \leftarrow$ Weight(point, node)
    $\boldsymbol{v}_{pic}$ += $w \cdot$ node.$\boldsymbol{v}$
  end for
  point.$\boldsymbol{x}$ += $\mathrm{dt} \cdot \boldsymbol{v}_{pic}$
end for