

Rapport Hebdo

Viet Anh Quach

3SR

5 septembre 2025

Changer la vitesse

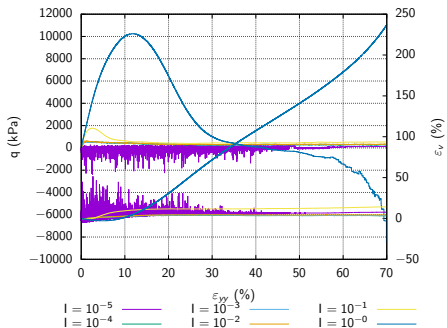


Figure 1 – Contrainte - Déformation DEM (changer la vitesse)

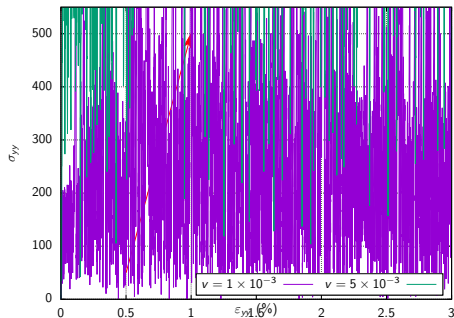


Figure 2 – Bruyant concernant pas de temps MPM (avant)

$$\dot{x}(t) = \frac{x(t + \varepsilon) - x(t - \varepsilon)}{2\varepsilon}$$

Problème de arrondir ?

- Un type flottant ne représente qu'un nombre limité de chiffres significatifs ; au-delà, la valeur devient inexacte.
- Manipulation délicate à cause des différences entre binaire et décimal.
- Les opérations mathématiques amplifient les erreurs d'arrondi (e.g : + et ×).

```

1 #include <iomanip>
2 #include <iostream>
3 #include <algorithm>
4
5 int main()
6 {
7     std::cout << std::setprecision(17);
8
9     constexpr double d1{1.0};
10    constexpr double d2{0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1};
11    std::cout << std::boolalpha << (d2 == 1.0) << '\n'; // true
12
13    constexpr double d3{100.0 - 99.99}; // 0.01
14    constexpr double d4{10.0 - 9.99}; // 0.01
15
16    if (d3 == d4)
17        std::cout << "d3 == d4" << '\n';
18    else if (d3 > d4)
19        std::cout << "d3 > d4" << '\n';
20    else if (d3 < d4)
21        std::cout << "d3 < d4" << '\n';
22
23    std::cout << std::boolalpha << (0.3 == 0.2 + 0.1) << '\n'; // true
24
25    return 0;
26 }

```

Revient chez DEM - cellule cube

Problème d'arrondir (Standard IEEE 754)

- The Art of Computer Programming, Volume II : Seminumerical Algorithms (Addison-Wesley, 1969)”
- Toujours contrôler la tolérance d'erreur (ϵ) au lieu de compter sur la “précision absolue” de l'ordinateur.

```

1 #include <algorithm> // for std::max
2 #include <cmath>    // for std::abs
3 #include <iostream>
4
5 // Return true if the difference between a and b is within epsilon percent of the larger of a and b
6 bool approximatelyEqualRel(double a, double b, double relEpsilon)
7 {
8     return (std::abs(a - b) <= (std::max(std::abs(a), std::abs(b)) * relEpsilon));
9 }
10
11 // Return true if the difference between a and b is less than or equal to absEpsilon, or within relEpsilon percent of the l
12 bool approximatelyEqualAbsRel(double a, double b, double absEpsilon, double relEpsilon)
13 {
14     // Check if the numbers are really close -- needed when comparing numbers near zero.
15     if (std::abs(a - b) <= absEpsilon)
16         return true;
17     // Otherwise fall back to Knuth's algorithm
18     return approximatelyEqualRel(a, b, relEpsilon);
19 }
20
21 int main()
22 {
23     // a is really close to 1.0, but has rounding errors
24     constexpr double a(0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1);
25
26     constexpr double relEps(1e-8);
27     constexpr double absEps(1e-12);
28
29     std::cout << std::boolalpha; // print true or false instead of 1 or 0
30
31     std::cout << approximatelyEqualRel(a, 1.0, relEps) << "\n"; // compare "almost 1.0" to 1.0
32     std::cout << approximatelyEqualRel(a - 1.0, 0.0, relEps) << "\n"; // compare "almost 0.0" to 0.0
33
34     std::cout << approximatelyEqualAbsRel(a, 1.0, absEps, relEps) << "\n"; // compare "almost 1.0" to 1.0
35     std::cout << approximatelyEqualAbsRel(a - 1.0, 0.0, absEps, relEps) << "\n"; // compare "almost 0.0" to 0.0
36
37     return 0;
38 }

```

PROBLEME OUTPUT DEBUG CONSOLE TERMINAL PORTS SPELL CHECKER

```

+ vscode@DESKTOP-GW050:~/Documents/rapporthese$ cd "/home/35-LAB/vquach/Documents/rapporthese/" && g++ -std=c++23 roundingError.cpp
true
false
true
true
+ vscode@DESKTOP-GW050:~/Documents/rapporthese$

```

Code : ajouter tous les parties dynamiques

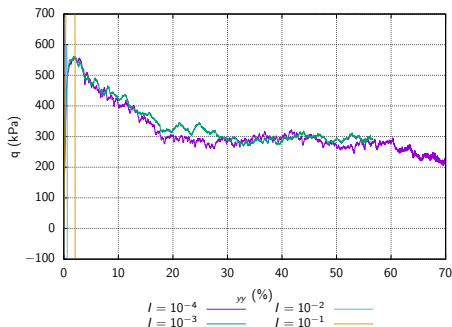


Figure 5 – Courbe Contrainte ($\sigma_3 = 300kPa$)

- l dans le régime quasi-statique : normal
- $l > 10^{-2}$: calcul erroné

⇒ Les résultats varient de manière très sensible avec l

Comparer entre les versions du code

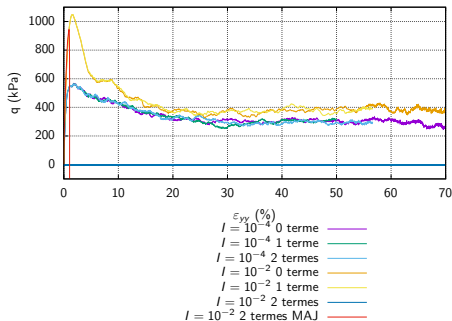


Figure 6 – Courbe Contrainte ($\sigma_3 = 300\text{kPa}$)

- 0 terme : $\ddot{s} = h^{-1} \cdot (F/m) \rightarrow$ ancienne version
- 1 terme : $\ddot{s} = h^{-1} \cdot (F/m - 2\dot{h}\dot{r}) \rightarrow$ presque inchangé
- 2 termes : $\ddot{s} = h^{-1} \cdot (F/m - 2\dot{h}\dot{r} - \ddot{h}r) \rightarrow$ calcul erroné

Essayer de changer la masse de la périodic

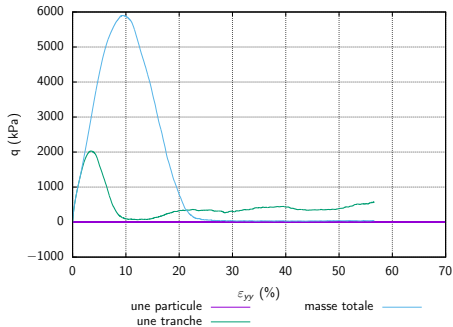


Figure 7 – Courbe Contrainte ($I = 10^{-2}$)

$$\ddot{h}_{xx} = \frac{V_{\text{cell}} (\sigma_{xx} - p)}{h_{xx} h_{\text{mass}}}$$

$$\ddot{h}_{yy} = \frac{V_{\text{cell}} (\sigma_{yy} - p)}{h_{yy} h_{\text{mass}}}$$

Les résultats varient de manière très sensible avec l

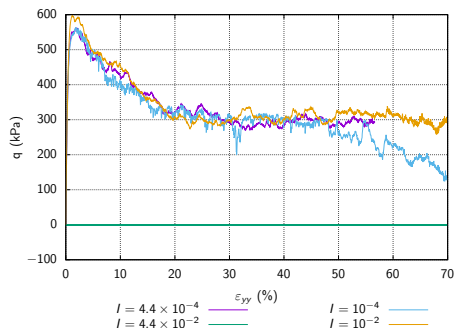


Figure 8 – Recalcul l exact ($\sigma_3 = 300$ kPa, $R = 0.005$ m)

ν	l
0.1	4.4×10^{-4}
10	4.4×10^{-2}

ν	l
0.0227	10^{-4}
2.27	10^{-2}

Table 1 – Calcul précédent approximatif

Table 2 – Calcul exact

Les résultats varient de manière très sensible avec l

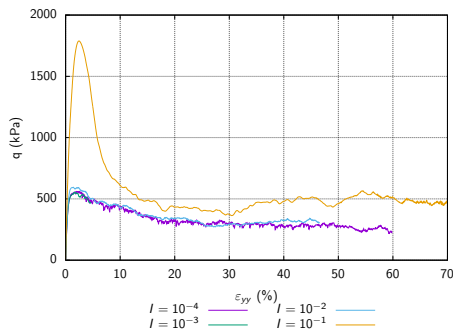


Figure 9 – Contrainte - Déformation DEM (changer la vitesse)

$l = 10^{-2}$ se marche maintenant

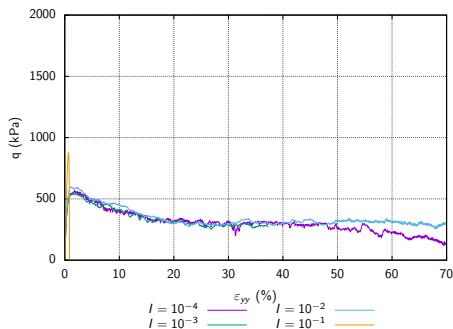


Figure 10 – Bruyant concernant pas de temps MPM (avant)

Souci concernant demi-vélocité

- Si Δt est très petit et $a(t) \approx a(t + \Delta t)$, on “perd” peut-être la moitié de l’incrément de $v(t + \Delta t)$.
- Pour des essais triaxiaux à chargement rapide, cette différence pourrait devenir critique.
- Lors du calcul des contraintes avec le terme dynamique (mv^2), l’erreur peut s’amplifier, possiblement proportionnellement au carré de la vitesse.

Formules de référence du Verlet Velocity :

$$\text{Demi-pas vitesse : } v(t + 0.5\Delta t) = v(t) + 0.5 \Delta t a(t)$$

$$\text{Vitesse au pas entier : } v(t + \Delta t) = v(t) + 0.5 \Delta t [a(t) + a(t + \Delta t)]$$

```

for (loop_i = 0; i < Particles.size(); i++) {
    Particles[i].pos += dt * Particles[i].vel + dt2 * 2 * Particles[i].acc;
    Particles[i].vel += dt * 2 * Particles[i].acc;

    // Periodicity in position (can be useful in the sample preparation)
    if (enablePeriodicity == 0) {
        // wrap around
        // (this recomputes velocity = false)
        for (loop_j = 0; j < 3; j++) {
            while (Particles[i].pos[j] < 0.0) {
                Particles[i].pos[j] += 1.0;
                // normalize
                // recompute velocity = true;
            }
            while (Particles[i].pos[j] > 1.0) {
                Particles[i].pos[j] -= 1.0;
                // normalize
                // recompute velocity = true;
            }
        }
    }
    // if (recompute velocity == true) {
    // Particles[i].vel = (cell.xh * Particles[i].pos + cell.b * Particles[i].vel);
    // }
    // }

    // Rotation: Q = Q * (dt/dt0) * dt * (2*Q0/dt2) * dt/dt0
    // It reads like this with quaternions
    double omega2 = 0.5 * Particles[i].vrot * Particles[i].vrot;
    Particles[i].Q *= quat(
        0.5 * (dt * (Particles[i].Q.x * Particles[i].vrot + cross(Particles[i].Q.v, Particles[i].Q.v)) +
        dt2 * (Particles[i].Q.x * Particles[i].vrot + cross(Particles[i].Q.v, Particles[i].Q.v)) -
        omega2 * 2 * Particles[i].Q.v),
        0.5 * (Particles[i].vrot * Particles[i].Q.v + dt * (Particles[i].vrot * Particles[i].Q.v + omega2 * 2 * dt2 * 2) *
        Particles[i].Q.normalized());
    Particles[i].vrot += dt * 2 * Particles[i].vrot;
}

```

Figure 12 – Cours de master

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ↺ 🔍 ↻