

# Le problème à $n$ corps

D'après l'épreuve du concours 1997 de l'École polytechnique et un sujet de Vincent Simonet

Le but de ce problème est l'étude d'une mise en œuvre efficace de la simulation du problème des  $n$  corps dans un univers plan. Le problème des  $n$  corps consiste à calculer les trajectoires de  $n$  corps ou particules qui interagissent entre eux sous l'effet de la gravité. Ce problème n'a pas de solution analytique dans le cas général, on a donc recours à une simulation numérique. Le principe de ce type de simulation est, à un instant  $t$  donné, de calculer les forces qui s'exercent sur chaque corps, puis d'estimer les vitesses et les positions de tous les corps à l'instant  $t + dt$ . Pour calculer les forces, on doit, pour chacun des  $n$  corps, calculer les  $n - 1$  interactions gravitationnelles. Le coût du calcul exact des forces est donc de l'ordre de  $n^2$ . Ce comportement quadratique limite sérieusement l'intérêt de la simulation. Une approximation physique simple va permettre une amélioration de l'efficacité de la simulation. Étant donné un corps  $c$  et un ensemble  $C$  de corps, l'attraction exercée sur  $c$  par les corps de  $C$  peut, pourvu que  $C$  soit relativement éloigné de  $c$ , être approchée par l'attraction exercée sur  $c$  par le centre de masse des corps de  $C$ .

Dans tout le problème,  $\mathcal{U}$  est l'univers, il contient  $n$  corps. Étant donné un corps  $c$ , on note  $\vec{p}_c$  le vecteur position de  $c$ . Étant donné un vecteur  $\vec{u}$ , on note  $u.x$  et  $u.y$  les composantes de  $\vec{u}$ ,  $\|\vec{u}\|_2$  sa norme euclidienne  $\sqrt{u.x^2 + u.y^2}$  et  $\|\vec{u}\|_\infty$  sa norme infinie  $\max(|u.x|, |u.y|)$ .

## 1 Quadrees

Dans la pratique, les ensembles  $C$  de corps sont des cellules carrées et ces ensembles sont organisés hiérarchiquement. L'univers est d'abord assimilé à une grosse cellule carrée de côté  $d_{\mathcal{U}}$ . Chaque cellule qui contient deux corps ou plus est ensuite subdivisée en quatre sous-cellules carrées de même taille, la subdivision s'arrêtant lorsqu'une cellule contient un ou zéro corps. Cette méthode porte le doux nom d'*algorithme de Barnes-Hut*.

Une représentation structurée de la répartition des corps en cellules et sous-cellules est un arbre dont les nœuds internes ont quatre fils et dont les feuilles contiennent zéro ou un corps. La division en sous-cellules n'est poussée qu'autant que nécessaire, c'est-à-dire que chaque nœud qui a des filles représente une cellule qui contient deux corps ou plus. Un tel arbre est un *quadtree* (*adaptif*). La profondeur d'un nœud d'un quadtree est le nombre

d'arcs à parcourir pour rejoindre la racine à partir de ce nœud. La profondeur d'un quadtree est la plus grande profondeur des nœuds de ce quadtree. On étudie dans cette partie quelques propriétés mathématiques des quadtrees.

► **Question 1** Dessiner la division en cellules de deux univers de profondeur 3 contenant le plus et le moins de corps possibles.

► **Question 2** Donner, en la justifiant brièvement, une minoration  $f(n)$ , fonction de  $n$ , de la profondeur d'un quadtree contenant  $n$  corps (ce minorant devra être atteint).

► **Question 3** Montrer qu'il n'existe pas de majoration fonction de  $n$  de la profondeur d'un quadtree.

Soit  $\delta$  la quantité

$$\min\{\|\vec{p}_c - \vec{p}_{c'}\|_\infty \mid c, c' \in \mathcal{U} \text{ et } c \neq c'\}$$

► **Question 4** Donner une borne supérieure de la profondeur d'un quadtree représentant  $\mathcal{U}$  en fonction de  $\delta$  et  $d_{\mathcal{U}}$ . On supposera  $\delta > 0$ .

► **Question 5** On suppose que les représentations en machine des grandeurs homogènes à une distance sont des nombres flottants du même ordre de grandeur. C'est-à-dire que la valeur absolue est assimilable à un entier codé en base deux à l'aide de  $m$  bits multiplié par un ordre de grandeur constant  $2^e$ . En déduire un majorant absolu de la profondeur des quadrees manipulés par notre programme.

## 2 Construction du quadtree

Un vecteur de l'espace plan est représenté par un enregistrement dont le type est le suivant :

```
type vecteur =
  { x: float;
    y: float
  }
;;
```

À titre d'exemple, le vecteur nul est défini comme suit :

```
let vecteur_nul =
  { x = 0.0;
    y = 0.0
  }
;;
```

Un corps  $c$  est la donnée d'une masse  $m_c$ , d'un vecteur position  $\vec{p}_c$ , d'une vitesse  $\vec{v}_c$  et d'une accélération  $\vec{a}_c$ . Le type des corps est un type enregistrement :

```
type corps =
  { mass: float;
    mutable pos: vecteur;
    mutable vel: vecteur;
    mutable acc: vecteur
  }
;;
```

**Rappel.** Les champs position, vitesse et accélération sont mutables car ils changeront au cours du temps. On pourra donc, par exemple, donner une nouvelle valeur  $v$  à l'accélération d'un corps  $c$  par l'opération  $c.\text{acc} \leftarrow v$ .

### 2.1 Opérations sur les vecteurs

On écrit tout d'abord quelques fonctions effectuant des opérations élémentaires sur les vecteurs.

► **Question 6** Écrivez deux fonctions `add` et `sub` prenant deux vecteurs en argument et renvoyant respectivement leur somme et leur différence vectorielle comme résultat.

```
value add: vecteur -> vecteur -> vecteur
value sub: vecteur -> vecteur -> vecteur
```

► **Question 7** Écrivez la fonction `scal` qui prend pour arguments un scalaire  $m$  et un vecteur  $\vec{u}$ , et retourne le produit  $m \cdot \vec{u}$ .

```
value scal: float -> vecteur -> vecteur
```

► **Question 8** Définissez une fonction `carre` qui calcule le carré de la norme euclidienne d'un vecteur.

```
value carre: vecteur -> float
```

### 2.2 Quadrees

En Caml, le type arbre des quadrees est le suivant :

```
type arbre =
  Noeud of cellule
  | Feuille of corps
  | Vide

and cellule =
  { mutable cm_mass: float; (* masse des corps de la cellule *)
    mutable cm_pos: vecteur; (* centre de masse des corps de la cellule *)
    filles: arbres vect (* 4 sous-arbres *)
  }
;;
```

Les sous-cellules d'une cellule sont rangées dans un tableau et sont donc numérotées. Précisez un ordre de numérotation des sous-cellules et définissez les deux fonctions suivantes.

► **Question 9** Étant données une position  $p$  et une cellule  $C$ , centrée en  $p_c$ , l'appel `indice_fille pc_c p` renvoie l'indice de la sous-cellule de  $C$  qui contient la position  $p$ .

```
value indice_fille: vecteur -> vecteur -> int
```

► **Question 10** Étant données une cellule  $C$ , de côté `taille` et centrée en  $p_c$ , l'appel `position_fille p_c taille i` renvoie la position du centre de la sous-cellule d'indice  $i$  de la cellule  $C$ .

```
value position_fille: vecteur -> float -> int -> vecteur
```

### 2.3 L'arbre univers

Nous représenterons un univers  $\mathcal{U}$  par la liste de ses corps. L'arbre-univers, représentation arborescente de  $\mathcal{U}$ , sera construit par insertion successive de tous les corps de l'univers dans une cellule racine dont le centre est à l'origine des coordonnées et dont le côté est  $d_{\mathcal{U}}$ .

► **Question 11** Programmez la fonction `insere_corps` : elle prend en arguments un corps à insérer, un arbre où insérer ce corps, ainsi que la position du centre et la valeur du côté de la cellule qu'il représente ; et renvoie un quadree qui contient le nouveau corps en plus de tous les autres. (Vous ne chercherez pas à positionner correctement les champs `cm_mass` et `cm_pos` des cellules.)

```
value insere_corps: corps -> arbre -> vecteur -> float -> arbre
```

► **Question 12** Prouver que la fonction `insere_corps` termine, à condition que le côté de la cellule racine soit suffisamment grand et que tous les corps occupent des positions distinctes.

► **Question 13** Écrivez une fonction `construit_arbre` qui construit l'arbre représentant un univers. Cette fonction prendra pour arguments le diamètre de l'univers et la liste des corps.

```
value construit_arbre: float -> corps list -> arbre
```

### 3 Calcul des forces

Il faut maintenant positionner correctement les champs `cm_mass` et `cm_pos` dans l'arbre univers. On rappelle que la masse  $m_C$  et la position  $\vec{p}_C$  du centre de masse (ou barycentre) des corps d'une cellule  $C$  sont données par les formules suivantes :

$$m_C = \sum_{c \in C} m_c \quad \vec{p}_C = \frac{1}{m_C} \sum_{c \in C} m_c \cdot \vec{p}_c$$

► **Question 14** *Écrire une fonction `barycentres` calculant les valeurs correctes des champs `cm_mass` et `cm_pos` de chacune des cellules d'un arbre. (La fonction modifiera en place la valeur de ces champs.)*

`value barycentres: arbre -> unit`

Selon les lois de la mécanique, l'accélération du corps  $c$  soumis à l'attraction des autres corps de l'univers est calculée en sommant les contributions de tous les autres corps.

$$\vec{a}_c = \sum_{c' \in \mathcal{U} \setminus \{c\}} \vec{a}_{c' \rightarrow c} \quad \text{avec } \vec{a}_{c' \rightarrow c} = \frac{m_{c'}}{\|\vec{p}_{c'} - \vec{p}_c\|_2^3} \cdot (\vec{p}_{c'} - \vec{p}_c)$$

On considère maintenant un corps  $c$  et une cellule  $C$  de côté  $d$ , dont la masse est  $m_C$  et le centre de masse

est positionné en  $\vec{p}_C$ . Soit  $\vec{r}$  le vecteur  $\vec{p}_C - \vec{p}_c$  et  $r$  sa norme. On estime que la force gravitationnelle exercée par les corps de  $C$  sur  $c$  peut être assimilée à celle exercée par leur centre de masse affecté de la masse  $m_C$  quand le rapport  $d/r$  est strictement inférieur à une constante  $\theta$ . On a alors :

$$\sum_{c' \in C} \vec{a}_{c' \rightarrow c} \approx \frac{m_C}{r^3} \vec{r}$$

L'accélération d'un corps  $c$  sera estimée en sommant les contributions des autres corps au cours d'un parcours partiel de l'arbre univers. Si, au cours de ce parcours, une cellule  $C$  satisfait le critère d'approximation, alors on remplacera la contribution de ses corps par celle de son centre de masse. Dans le cas contraire, on continuera de subdiviser la cellule  $C$ .

► **Question 15** *Écrivez une fonction `grav_approx` qui calcule l'approximation de l'accélération d'un corps soumis à l'attraction d'un ensemble de corps représenté par un quadtree. La fonction prendra pour arguments la valeur de la constante  $\theta$ , la position du corps considéré et le diamètre de la cellule représenté par l'arbre et l'arbre.*

`value grav_approx: float -> vecteur -> float -> arbre -> vecteur`



# Le problème à $n$ corps

## Un corrigé

► **Question 2** Considérons un quadtree de profondeur  $p$  et contenant  $n$  corps. Un quadtree de profondeur  $p$  compte au maximum  $4^p$  cellules. Chaque cellule contenant au plus un corps, on en déduit que  $n \leq 4^p$ , autrement dit que  $\log_4 n \leq p$ .  $p$  étant un entier, cette inégalité donne  $\lceil \log_4 n \rceil \leq p$ . En considérant un univers où les corps sont répartis aux intersections d'un quadrillage régulier, il est facile de vérifier que ce minimum est atteint.

► **Question 3** Dans un quadtree de profondeur  $p$ , le diamètre de chaque cellule est supérieur ou égal à  $d_U/2^p$ . En considérant un univers à deux corps de coordonnées  $(0, -d_U/2^n)$  et  $(0, d_U/2^n)$ , on montre que pour tout  $n$ , il existe un quadtree à deux corps de profondeur supérieure ou égale à  $n$ .

► **Question 4** Considérons un quadtree de profondeur  $p$ . Dans un tel quadtree, il existe deux cellules adjacentes de diamètre  $d_U/2^p$  et contenant chacune un corps. Notons  $c$  et  $c'$  deux tels corps. On a  $\|\vec{p}_c - \vec{p}_{c'}\|_\infty \leq 2 \times d_U/2^p$ . On en déduit que  $\delta \leq d_U/2^{p-1}$ , ce qui donne la majoration  $p \leq 1 + \log_2(d_U/\delta)$ .

► **Question 5** Dans une telle représentation, la distance entre deux nombres flottants distincts est supérieure ou égale à  $2^e$ . On en déduit que, si  $\delta > 0$  alors  $\delta \geq 2^e$ . On obtient donc la majoration  $p \leq 1 + \log_2(d_U/2^e)$ .

► **Question 6** Les composantes des vecteurs étant représentées par des flottants, on utilise les opérateurs  $+$  et  $-$  pour calculer respectivement leur somme et leur différence.

```
let add = fun { x = x1; y = y1 } { x = x2; y = y2 } ->
  {x = x1 +. x2; y = y1 +. y2 }
;;

let sub = fun { x = x1; y = y1 } { x = x2; y = y2 } ->
  {x = x1 -. x2; y = y1 -. y2 }
;;
```

► **Question 7**

```
let scal = fun m { x = x1; y = y1 } ->
  {x = x1 *. m; y = y1 *. m }
;;
```

► **Question 8**

```
let carre = fun { x = x1; y = y1 } ->
  x1 *. x1 +. y1 *. y1
;;
```

► **Question 9** On choisit de numérotter les filles d'une cellule en suivant la convention suivante :

0	1
2	3

```
t indice_fille = fun { x = xp; y = yp } { x = xp.c; y = yp.c } ->
  match ( xp < xp.c, yp < yp.c ) with
  | ( true, true ) -> 0
  | ( true, false ) -> 2
  | ( false, true ) -> 1
  | ( false, false ) -> 3
```

► **Question 10**

```
let position_fille = fun { x = xp.c; y = yp.c } taille i ->
  { x = xp.c +. taille *. (0.5 +. float_of_int (2*(i/2)-i));
    y = yp.c +. taille *. (0.5 -. float_of_int (i/2)) }
;;
```

► **Question 11**

```
let rec insere_corps = fun corps arbre p.c taille ->
  match arbre with
  | Vide -> Feuille corps
  | Feuille corpsbis ->
    (* il y a déjà un corps dans la cellule, il faut donc la couper en 4 *)
    let temp = { cm_mass = 0.0; cm_pos = vecteur_nul;
                 filles = make_vect 4 Vide } in
    let i = indice_fille corps.pos p.c in
    let ibis = indice_fille corpsbis.pos p.c in
    temp.filles.(ibis) <- Feuille corpsbis;
    temp.filles.(i) <- insere_corps corps temp.filles.(i)
      (position_fille p.c taille i) (taille /. 2.);
    Noeud temp
  | Noeud noeud ->
    (* la cellule est déjà coupée en 4 *)
    let i = indice_fille corps.pos p.c in
    noeud.filles.(i) <- insere_corps
      corps noeud.filles.(i) (position_fille p.c taille i)
      (taille /. 2.);
    Noeud noeud
  ;;
```

### ► Question 13

```
let rec construit_arbre = fun taille corps ->
  match corps with
  | [] -> Vide
  | t::q -> insere_corps t (construit_arbre taille q) vecteur_nul taille
;;
```

### ► Question 14

```
let rec aux = fun arbre ->
  match arbre with
  | Vide -> (0.0, vecteur_nul)
  | Feuille corps -> (corps.mass, corps.pos)
  | Noeud noeud -> let mass = ref 0. and pos = ref vecteur_nul in
    for i = 0 to 3 do (
      let (m_i, pos_i) = aux noeud.filles.(i) in
      mass := (!mass) +. m_i;
      pos := add (!pos) (scal m_i pos_i);
    ) done;
  pos := scal (1. /. !mass) !pos;
  noeud.cm_mass <- !mass;
  noeud.cm_pos <- !pos;
  (!mass, !pos)
;;

let barycentre = fun arbre ->
  let a = aux arbre in ()
;;
```

### ► Question 15

```
let rec grav_approx = fun theta pos taille arbre ->
  let accel = fun mass pos posbis ->
    scal (mass /. ((carre (sub posbis pos))* 1.5)) (sub posbis pos)
  in
  match arbre with
  | Vide -> vecteur_nul
  | Feuille corps when corps.pos = pos -> vecteur_nul
  | Feuille corps -> accel corps.mass pos corps.pos
  | Noeud noeud -> let r = sqrt (carre (sub noeud.cm_pos pos)) in
    if taille /. r < theta then accel noeud.cm_mass pos noeud.cm_pos
    else (
      let tmp = ref vecteur_nul in
      for i = 0 to 3 do
        tmp := add (!tmp) (grav_approx theta pos (taille /. 2.) noeud.filles.(i))
      done;
      !tmp
    )
  )
;;
```