Université de Montpellier

M1 - IMAGINE - Modelisation et Geométrie Discrète
Compte Rendu TP7 - Enveloppe convexe

Etudiant :
Guillaume Bataille

Encadrant :
Noura Faraj
Marc HARTLEY

Année 2022-2023

Sommaire

1. Orientation de trois points dans le plan

1.1 Determinant

```
// v1 et v2 sont des vecteurs du plan représentés par leurs coordonnées en 2D
static determinant(v1, v2) {
  // todo
  return v1.x * v2.y - v1.y * v2.x;
}
```

1.2 Det sign

```
static detSign(v1, v2) {
  //todo
  let det = this.determinant(v1, v2);
  let detsign;
  if (det == 0) detsign = 0;
  else det > 0 ? (detsign = 1) : detsign - 1;
  return detsign;
}
```

1.3 Tour

```
static tour(o, p1, p2) {
  //todo
  let v1 = this.vecteur(o, p1);
  let v2 = this.vecteur(o, p2);

  return this.detSign(v1, v2);
}
```

1.4 Test des méthodes précédentes

```
run() {
  this.d.init();
  this.d.setModelSpace(...this.getModelSpace());
  this.d.drawPoints(this.getPoints(), false);

  //todo
  let key = "tour";
  let p = this.getPoints();
  if (key == "tour") {
    let n = this.getPoints().length;
    let i;
    console.log(n);
    for (i = 0; i < n - 2; i += 3) { // 3 par 3 points et on ignore le dernier cas
      this.d.drawTour(
        Coord2D.tour(p[i], p[i + 1], p[i + 2]),
        [p[i], p[i + 1], p[i + 2]],
        true
      );
    }
    if (n != i) {
      // Si on a un nombre non multiple de 3 de points
      this.d.drawTour(
        Coord2D.tour(p[n - 3], p[n - 2], p[n - 1]),
        [p[n - 3], p[n - 2], p[n - 1]],
        true
      );
    }
  }
}

/* VERSION BOUCLE SUR TOUT LES SOMMETS  (utilisé plus tard)
  let n = this.getPoints().length;
    console.log(n);
    for (let i = 0; i < n - 1; i++) {
      for (let j = i; j < n; j++) {
        for (let k = i; k < n; k++) {
          if (k != j) {
            this.d.drawTour(
              Coord2D.tour(p[i], p[j], p[k]),
              [p[i], p[j], p[k]],
              true
            );
          }
        }
      }
    }
*/
```

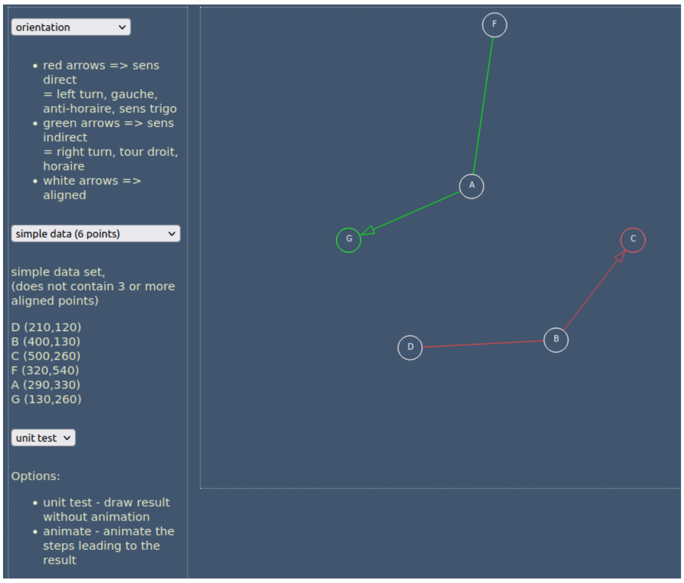

FIGURE 1.1 – Test avec simple data - 6 points

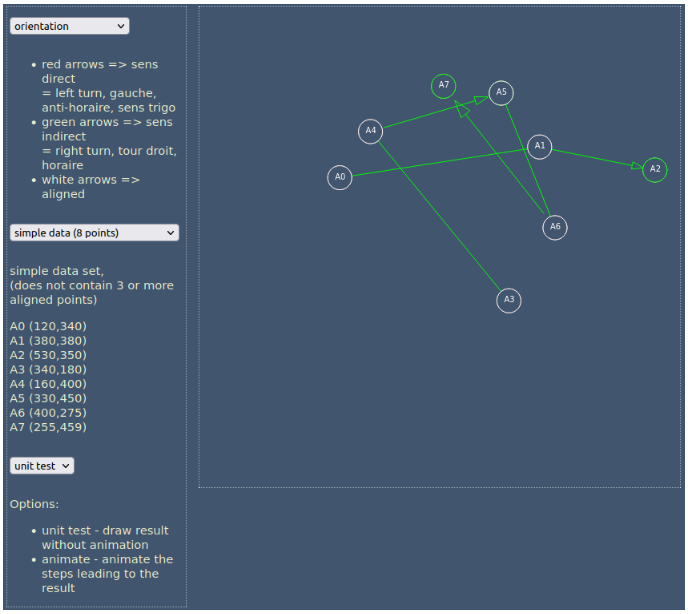

FIGURE 1.2 – Test avec simple data - 8 points

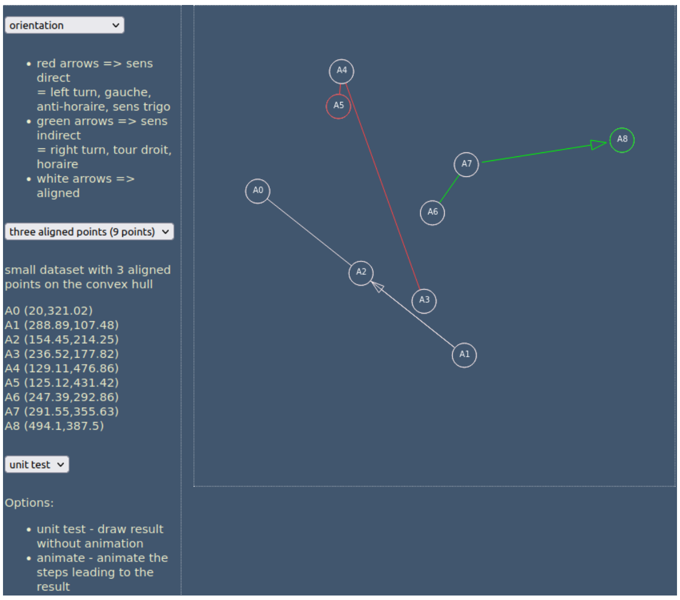

FIGURE 1.3 – Test avec three aligned points - 9 points

2. Algorithme d'enveloppe convexe : Graham

2.1 Pré requis : Compare

```
static compare(origine, p1, p2) { // Permet de comparer deux vec en fct d'un pivot dans le sens trigo
  //todo
  // vec_x est la base avec laquelle on va créer nos angles
  let vec_x = Coord2D.vecteur(new Coord2D(0, 0, "O"), new Coord2D(1, 0, "X"));

  let vec1 = Coord2D.vecteur(origine, p1);
  let vec2 = Coord2D.vecteur(origine, p2);

  let dot_prod1 = vec1.x * vec_x.x + vec1.y * vec_x.y;
  let len_vec1 = Math.sqrt(vec1.x ** 2 + vec1.y ** 2);
  let cos1 = dot_prod1 / len_vec1; // Cos1 de l'angle entre le vecteur x et le vecteur 1

  let dot_prod2 = vec2.x * vec_x.x + vec2.y * vec_x.y;
  let len_vec2 = Math.sqrt(vec2.x ** 2 + vec2.y ** 2);
  let cos2 = dot_prod2 / len_vec2;// Cos2 de l'angle entre le vecteur x et le vecteur 2

  //Recupère les angles1 et 2
  let angle1 = Math.acos(cos1);
  let angle2 = Math.acos(cos2);

  // Si l'angle est négatif on le rends positif en fesant un 2pi tour
  if (angle1 < 0) angle1 += Math.PI * 2;
  if (angle2 < 0) angle2 += Math.PI * 2;

  // Comme l'angle 1 et le 2 ont pour base x :
  if (angle1 < angle2) { // Si angle 1 < angle 2, alors ils se suivent et sont bien ordonnés (return 1)
    return 1;
  } else if (angle2 < angle1) {// Si angle 1 > angle 2, alors ils ne sont pas ordonnés (return -1)
    return -1;
  } else { // S'il sont égaux, alors ils sont alignés (return 0);
    return 0;
  }
}

```

Malheuresement je n'arrive pas a faire le visuel pour tester la fonction compare ;

2.2 Pré requis : MinY

```
static findMinIdx(points) {
  //todo
  let current_minY = Number.MAX_SAFE_INTEGER; // On prends un Y très très grand comme minY de base
  let current_min_id;

  for (var i = 0; i < points.length; i++) { // pour tout les points
    if (points[i].y < current_minY) { // Si le point courant est plus petit que le minY courant
      current_minY = points[i].y; // Il devient le nouveau minY courant
      current_min_id = i; // Et on récupère son indice
    }
  }
  return current_min_id; // On retourne son indice
}
```

2.3 Pré requis : Tri

```
tri(min, points) {
  let triRadial = new TriRadial(points, min); // Appel a triRadial qui fait appel à un tri par tas
  return triRadial.V;
}
```

2.4 Algorithme de Graham (O(n log n))

```
algoGraham(points) {

  // Initialisation
  let min = Coord2D.findMinIdx(points); // Recup l'indice du minY

  let points_copy = points.map((x) => x); // Copie de points
  points_copy.splice(min, 1); // Retire le min de l'ensemble

  let sorted_points = this.tri(points[min], points_copy); // On trie par rapport a min
  let candidates = sorted_points.map((x) => x); // Recopier les points triés
  candidates.push(points[min]); // Ajout de min en fin de liste

  let result = [points[min], candidates[0]];
  let n = candidates.length;
  let m = result.length;

```

```
    for (let k = 1; k <= n - 1; k++) { // Pour chaque élément hors pivot
      let t;
      while (
        m >= 2 &&
        (t = Coord2D.tour(result[m - 2], result[m - 1], candidates[k])) <= 0
      ) {
        if (t == 0) {
          new Error("alignment", result[m - 2], result[m - 1], candidates[k]);
        }
        result.pop();
        m = result.length;
      }
      result.push(candidates[k]);
      m = result.length;
    }
    return result;
  }

```

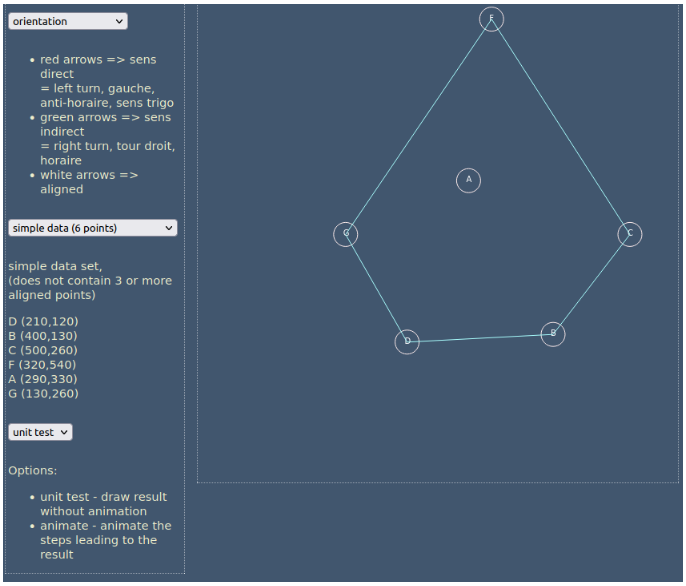

FIGURE 2.1 – Algo Graham - 6 points

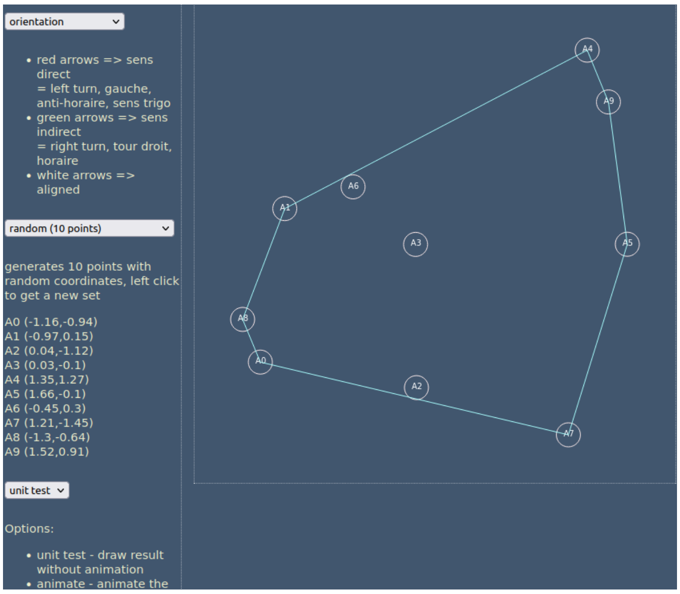

FIGURE 2.2 – Algo Graham - 10 points

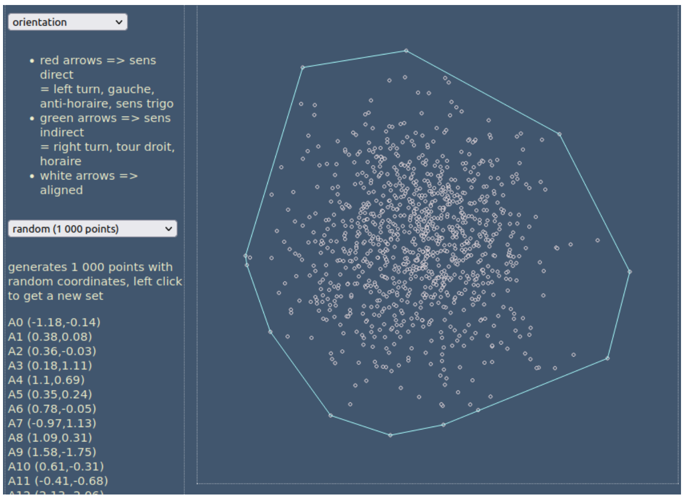

FIGURE 2.3 – Algo Graham - 1000 points

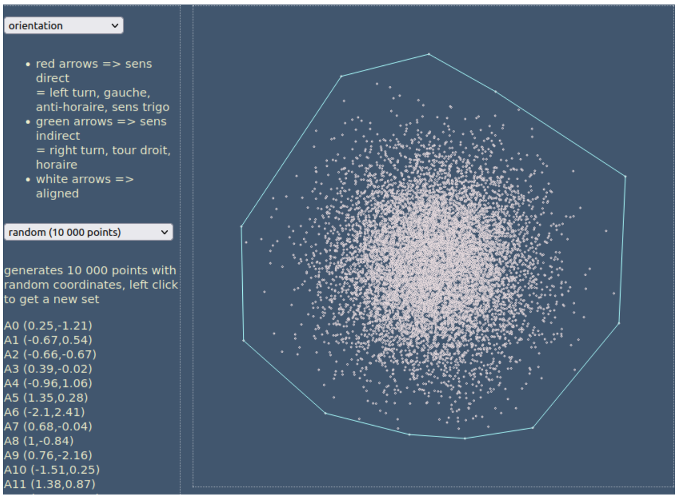

Figure 2.4 – Algo Graham - 10 000 points

3. Mesure Empiriques

```
// POUR ALGO DE COMPLEXITE O(n3) -> AFFICHAGE DANS LA CONSOLE
Tour en O(n3) a pris 14 milliseconds pour 6 elements.
Tour en O(n3) a pris 35 milliseconds pour 8 elements.
Tour en O(n3) a pris 41 milliseconds pour 10 elements.
Tour en O(n3) a pris 25053 milliseconds pour 100 elements.
Pour 1000 elements cela semble prendre une eternité...
```

```
// POUR ALGO GRAHAM O(nlogn) -> AFFICHAGE DANS LA CONSOLE
Graham a pris 3 milliseconds pour 6 elements.
Graham a pris 3 milliseconds pour 8 elements.
Graham a pris 5 milliseconds pour 10 elements.
Graham a pris 7 milliseconds pour 100 elements.
Graham a pris 24 milliseconds pour 1000 elements.
Graham a pris 61 milliseconds pour 10000 elements.
```

Pour n = 100 on a :
n3 = 1 000 000
nlogn = 200
Rapport théorique de 5000 entre 1000000 et 200.
Rapport réel de 3571 entre 25000 et 7.
On est bien dans cet ordre de grandeur avec 3500 et 5000.
Les ordres de grandeurs des complexités sont donc bien réelles.

4. Mise à jour dynamique de l'enveloppe convexe

Soit V un ensemble de points et EC l'ensemble de points dans V formant l'enveloppe convexe. Soit k le nombre de points dans EC et n le nombre de points dans V.

4.1 Update en O(k) après ajout d'un point dans V

Pour tout ki dans EC :
—-On teste entre newpoint et ki+1 lequel conserve EC comme convexe
———Si c'est newpoint ————On fait le nouveau lien
———Fin si
—-Fin balaiement
Fin pour

4.2 Update en O(n) après le retrait d'un point dans EC

Le point retiré était a l'indice kx
On a E[ki] précédant E[kx] et E[kii] le suivant de kx
Soit j = 0
Tant que j != ii j < n :
—-Si Vi et Vj sont toujours convexe
———On ajoute Vj dans EC entre Vi et Vii ;
—-Fin si
Fin tant que

5. ISSUES et ESSAI

J'ai essayé d'implémenter un algo O(n3) (Demi plan) mais je n'ai pas réussi à le faire (et a dégager du temps pour le faire). C'est notamment une difficulté de compréhension de l'algo en lui même qui m'a coincé.
Il me manque donc les exos suivants :
- 2 demi plan
(Mal compris l'algo) - 3 Yarvis
(Manque de temps) - 8 Intersection
(Manque de temps et mal compris ce qui était demandé ici)