Devoir Maison 1

Pour le 28/09/2017– Récursivité & Métrologie

Informatique

PSI ⋆

1 Exercice

On considère la fonction suivante :

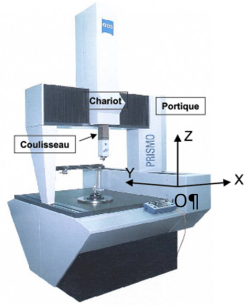
```
def f(x) :
    if x <= 1 :
        return 1
    elif x % 2 == 0 :
        return 2*f(x/2)
    else :
        return 1 + f(x+1)
```

1. Montrer que cette fonction récursive se termine pour toute valeur entière de l'argument `x`.
2. Soient $x \in \mathbb{N}$ et $g(x) = f(x) + x$. À partir du programme précédent, donner un programme python qui calcule $g(x)$.
3. En déduire que pour tout $x \in \mathbb{N}$, $g(x)$ est une puissance de 2.

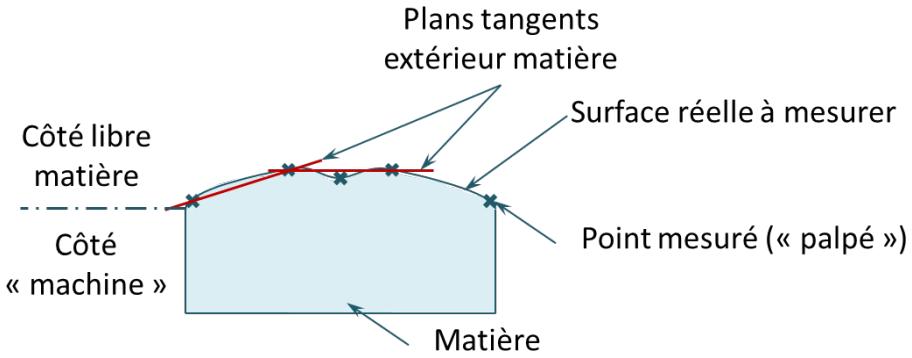
2 Métrologie

2.1 Mise en situation

La métrologie est la « science des mesures ». Dans l'industrie, elle désigne les opération permettant de valider ou non les dimensions et les formes des pièces fabriquées. On peut par exemple vérifier qu'une mesure dimensionnelle vérifie le cahier des charges. On peut aussi vérifier qu'une surface plane fabriquée vérifie le cahier des charges (on parle alors de planéité).
Pour vérifier les pièces, on utilise une machine à mesurer tridimensionnelle (image ci-contre) qui permet de palper des points et de les stocker afin de les analyser.



En métrologie, il peut être est nécessaire de construire un plan idéal à partir d'un nuage de points palpés. Un des critères de construction prévu par la norme et de construire un «plan tangent extérieur à la matière qui minimise l'écart maximum ». Dans la pratique, il est assez difficile de calculer le plan optimal.



Objectif L'objectif de ce travail est de déterminer algorithmiquement, de manière approchée, le plan tangent extérieur matière qui minimise l'écart maximum.

2.2 Mise en équation de détermination du plan des moindre carrés

On suppose que la direction de mesure $\overrightarrow{z}$ est verticale ascendante pour la machine à mesurer tridimensionnelle. En conséquence, on traitera les plans qui ne contiennent pas le vecteur $\overrightarrow{z}$.

Rappel

Dans un repère orthonormé direct, l'équation d'un plan $(\mathscr{P})$ est donnée par $z = a\,x + b\,y + c$.
Le vecteur $\overrightarrow{n}(a, b, 1)$ définit une normale au plan $(\mathscr{P})$.
Le point de coordonnées $(0, 0, c)$ appartient au plan $(\mathscr{P})$.

L'écart e_i d'un point M_i suivant $\overrightarrow{z}$ de coordonnées (x_i, y_i, z_i) au plan $(\mathscr{P})$ est donné par $e_i = z_i - a\,x_i - b\,y_i - c$. On remarquera que e_i est, à une constante multiplicative près, la distance algébrique du point M_i au plan $\mathscr{P}$.

Définition **Écarts** La fonction écart E est définie de la manière suivante :

$$E : (a, b, c) \to \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} \left(z_i - a\,x_i - b\,y_i - c\right)^2$$

Définition **Minimisation des écarts**
Pour minimiser la fonction E, il faut résoudre le système d'équations suivant :

$$\frac{\partial E(a,b,c)}{\partial a} = 0 \quad (1) \qquad \frac{\partial E(a,b,c)}{\partial b} = 0 \quad (2) \qquad \frac{\partial E(a,b,c)}{\partial c} = 0 \quad (3)$$

Rappel

On remarque que la méthode des moindres carrés ne minimise pas la somme des écarts e_i mais celle de leur carré. On pourrait vérifier que minimiser la fonction $E' : (a, b, c) \to \sum_{i=1}^{n} e_i$ à l'aide des relations obtenues par dérivations partielles, ne permettrait pas de déterminer les valeurs de a, b et c.

3 Détermination du défaut de planéité

Objectif L'objectif de cette partie est de rechercher le plan des moindres carrés c'est à dire de trouver les valeurs a, b et c qui minimisent les écarts entre le plan $\mathscr{P}$ et un nuage de points.
Le plan des moindre carrés permettra de déterminer le défaut de planéité.

3.1 Conditionnement du problème

Question 1 *Montrer que la méthode des moindres carrés permet d'aboutir aux 3 équations suivantes :*

$$\sum_{i=1}^{n}\left(a\,x_i^2 + b\,x_i\,y_i + c\,x_i - x_i\,z_i\right) = 0 \quad \sum_{i=1}^{n}\left(a\,x_i\,y_i + b\,y_i^2 + c\,y_i - y_i\,z_i\right) = 0 \quad \sum_{i=1}^{n}\left(a\,x_i + b\,y_i + c - z_i\right) = 0$$

On définit les grandeurs suivantes :
$S_{xx} = \sum_{i=1}^{n} x_i^2$, $S_{yy} = \sum_{i=1}^{n} y_i^2$, $S_{xy} = \sum_{i=1}^{n} x_i\,y_i$, $S_{xz} = \sum_{i=1}^{n} x_i\,z_i$, $S_{yz} = \sum_{i=1}^{n} y_i\,z_i$, $S_x = \sum_{i=1}^{n} x_i$, $S_y = \sum_{i=1}^{n} y_i$ et $S_z = \sum_{i=1}^{n} z_i$.
On note $X = (a, b, c)$ le vecteur solution du problème avec $X \in \mathscr{M}_{1,3}(\mathbb{R})$.

Question 2 *Montrer que le problème peut se mettre sous la forme du système linéaire suivant : $AX + B = 0$ où $A \in \mathscr{M}_{3,3}(\mathbb{R})$ et $B \in \mathscr{M}_{1,3}(\mathbb{R})$. On donnera les expressions de A et de B.*

3.2 Résolution du problème

Méthode Au final, pour déterminer le plan des moindres carrés, la démarche est la suivante :
- mesurer les points(x_i, y_i, z_i) grâce à la machine à mesurer et exporter les données dans un fichier (ici un fichier texte) ;
- importer le fichier de points avec Python ;
- résoudre le système linéaire $AX + B = 0$ pour déterminer a, b et c permettant d'obtenir l'équation du plan.

Question 3 *Donner une méthode (ou le nom d'un algorithme) permettant de résoudre le système linéaire ci-dessus. Préciser les différentes étapes de cet algorithme ainsi que sa complexité.*

Les points mesurés par une machine à mesurer tridimensionnelle sont stockés dans un fichier texte. Dans ce fichier sont inscrits successivement les coordonnés des points puis les coordonnées de la normale de contact entre le palpeur et la surface mesurée. Le fichier est de la forme suivante :

```
-101.88340,-155.21568,-50.30434,0,0,1
-99.21040,-145.54768,-50.304844,0,0,1
-82.43090,-129.69318,-50.292844,0,0,1
-59.72540,-134.12818,-50.301844,0,0,1
...
```

R La fonction `split` permet de séparer les éléments d'une chaîne de caractère suivant un motif et de les stocker dans une liste :

```
>>> a = "-101.88340,-155.21568,-50.30434,0,0,1"
>>> a = a.split(",")
          ['-59.72540', '-134.12818', '-50.301844', '0', '0', '1']
```

Question 4 *Donner l'implémentation de la fonction* `read_file` *permettant de lire un fichier de mesures formaté comme indiqué ci-dessus et permettant de retourner la liste des points mesurés.*

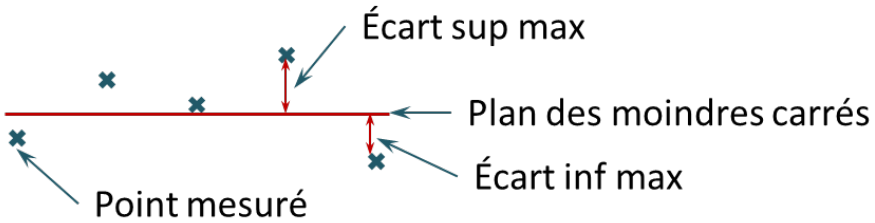
On appelle `plan_moindres_carres` la fonction permettant de trouver la solution du système linéaire à partir d'une liste de points. Ces spécifications sont les suivantes :

▪ Python
```
def plan_moindres_carres(liste_pts):
    """
    Permet de déterminer les caractéristiques du plan des moindres carrés.
    Entrée :
        * liste_pts (list) : liste de points de la forme [[x1,y1,z1],[x2,y2,z2],...]
    Sortie :
        * pl(list) : solution du problème d'optimisation : retourne [a,b,c]
    """
```

Question 5 *Donner l'implémentation du programme principal (main) permettant de lire le fichier de mesure appelé* `mesures.txt` *et de déterminer la liste des paramètres* `[a,b,c]` *correspondant aux paramètres du plan des moindres carrés. Vous utiliserez les fonctions précédemment définies.*

3.3 Application – Détermination du défaut de planéité

Pour déterminer le défaut de planéité d'une forme, il est nécessaire de déterminer la distance maximale entre les points les plus éloignés de part et d'autre du plan des moindres carrés.



Défaut de planéité = Écart sup max + Écart inf max

On définit la distance algébrique d d'un point M de coordonnées (x,y,z) au plan $(\mathscr{P})$ par l'expression suivante : $d = \dfrac{z - ax - by - c}{\sqrt{a^2+b^2+1}}$.

On notera `pt` la liste contenant les coordonnées d'un point et `pl` la liste contenant le triplet `(a,b,c)`.

Question 6 *Donner l'implémentation de la fonction* `dist_pt_plan` *en Python permettant de retourner la distance algébrique entre un point et un plan. Les spécifications de la fonction sont les suivantes :*

▪ Python
```
def dist_pt_plan(pt,plan):
    """
    Permet de calculer une distance point - plan
    Entrées :
        * pt(list) : point de coordonnées [x,y,z]
        * plan(list) : caractéristiques du plan [a,b,c]
    Sortie :
        * d(flt) : distance
    """
```

Question 7 *Donner l'implémentation de la fonction* `defaut_planeite` *en Python permettant de retourner le défaut de planéité. On donne les spécifications de la fonction :*

▪ Python
```
def defaut_planeite(pl,liste_pt):
    """
    Permet de calculer une distance point - plan
    Entrées :
        * pl(list) : caractéristiques du plan [a,b,c]
        * liste_pts (list) : liste de points de la forme [[x1,y1,z1],[x2,y2,z2],...]

    Sortie :
        * d(flt) : distance
    """
```

4 Construction d'une référence spécifiée associée à une surface nominalement plane

Pour associer un plan idéal à un nuage de points, la norme prévoit de lui associé un plan tangent extérieur matière minimisant les écarts. Mathématiquement, ce problème n'est pas simple à résoudre. Afin d'approcher la solution, on se propose d'utiliser, dans le cadre ce travail, la méthode suivante :

Méthode

1. Détermination du plan par la méthode des moindres carrés.
2. Translation du plan au point le plus éloigné côté libre de la matière.
3. Balancement du plan afin de minimiser les défauts : on fait varier son orientation.
4. Vérification que tous les points sont situés sous le plan.
5. Changement au point le plus éloigné situé côté libre de la matière s'il y en a.

Les étapes 4 et 5 peuvent être itératives.

Attention : les points 3, 4 et 5 ne sont qu'une piste envisagée dans le cadre de ce travail et ne constituent pas un algorithme utilisé dans les logiciels de mesure.

On donne la fonction suivante :

```python
def mystere(pl,liste_pt):
    ind=0
    d=dist_pt_plan(liste_pt[ind],pl)
    for i in range(1,len(liste_pt)):
        temp=dist_pt_plan(liste_pt[i],pl)
        if temp>d :
            ind=i
            d=dist_pt_plan(liste_pt[ind],pl)
    return ind
```

Question 8 *Quel est l'objectif de cette fonction ? En déduire le triplet correspondant à l'équation du plan répondant au point **2** de la méthode.*

Le balançage du plan par rapport à un point est réalisé grâce à la fonction suivante.

```python
def balancage(pl, pt):
    npt=25
    pas=0.000003
    dist=100000
    pl2=[None,None,None]
    for k in range(-npt,npt):
        pl2[0]=pl[0]+k*pas
        for j in range(-npt,npt):
            pl2[1]=pl[1]+j*pas
            pl2[2]=pt[2]-pl2[0]*pt[0]-pl2[1]*pt[1]
            temp=defaut_planeite(pl2,liste_pt)
            if temp<dist :
                dist=temp
                f=pl2[0]
                g=pl2[1]
                h=pl2[2]
    pl2=[f,g,h]
    return [pl2,dist]
```

Question 9 *Quel est l'objectif de cette fonction ? Que retourne-t-elle ? Donner un commentaire pour chacune des instructions.*

Question 10 *Quelle est la complexité de cet algorithme si on cherche à améliorer le choix du plan optimal par rapport à un seul des points du nuage de points ? Comment évolue la complexité de ce programme si on cherche à réaliser le balançage en utilisant chacun des points mesurés ?*