

Proposition de corrigé

Concours : Concours Centrale-Supélec

Année : 2022

Filière : MP - PC - PSI - TSI

Épreuve : Informatique

Ceci est une proposition de corrigé des concours de CPGE, réalisée bénévolement par des enseignants de Sciences Industrielles de l'Ingénieur et d'Informatique, membres de l'[UPSTI](https://www.upsti.fr) (Union des Professeurs de Sciences et Techniques Industrielles), et publiée sur le site de l'association :

<https://www.upsti.fr/espace-etudiants/annales-de-concours>

A l'attention des étudiants

Ce document vous apportera des éléments de corrections pour le sujet traité, mais n'est ni un corrigé officiel du concours, ni un corrigé détaillé ou exhaustif de l'épreuve en question.

L'UPSTI ne répondra pas directement aux questions que peuvent soulever ces corrigés : nous vous invitons à vous rapprocher de vos enseignants si vous souhaitez des compléments d'information, et à vous adresser à eux pour nous faire remonter vos éventuelles remarques.

Licence et Copyright

Toute représentation ou reproduction (même partielle) de ce document faite sans l'accord de l'UPSTI est **interdite**. Seuls le téléchargement et la copie privée à usage personnel sont autorisés (protection au titre des [droits d'auteur](#)).

En cas de doute, n'hésitez pas à nous contacter à : corrigesconcours@upsti.fr.

Informez-vous !

Retrouvez plus d'information sur les [Sciences de l'Ingénieur](#), l'[orientation](#), les [Grandes Ecoles](#) ainsi que sur les [Olympiades de Sciences de l'Ingénieur](#) et sur les [Sciences de l'Ingénieur au Féminin](#) sur notre site : www.upsti.fr

L'équipe UPSTI

Modélisations autour de la Formule 1

Corrigé UPSTI

I Modélisation sommaire d'un circuit

I.A – Validité de la représentation d'un circuit

Question 1

```
1 def longueur1(c, d):
2     return c.count("A")*d
```

Question 2

Les étapes liées à la boucle for sont les suivantes, et le resultat final est ["A","A","D","A"].

Etape	e	res	nbg
1	"A"	["A"]	0
2	"A"	["A","A"]	0
3	"G"	["A","A"]	1
4	"D"	["A","A"]	0
5	"G"	["A","A"]	1
6	"G"	["A","A"]	2
7	"G"	["A","A"]	3
8	"A"	["A","A","D","A"]	0

Question 3 Le but de ce code est de simplifier les successions de virage non significatives. En effet pour passer d'une orientation à une autre il ne peut pas y avoir plus de 2 instructions de virage (90° à gauche, 180° à gauche, 270° à gauche (ce qui revient en fait à faire 90° à droite)), On additionne donc les virages demandés pour simplifier!

Question 4

```
1 def contient_demi_tour1(c):
2     cmin = representation_minimale(c)
3     for i in range(len(cmin)-1):
4         # Test de la présence de demi-tour
5         if cmin[i:i+2] == ["G","G"] :
6             return True
7     return False
```

Question 5

```

1 def est_ferme1(c):
2     # definition des coordonnees initiales
3     x,y,ori = 0,0,0
4     # Variation de position en fonction de l'orientation
5     dx = [1,0,-1,0]
6     dy = [0,1,0,-1]
7     for elt in c :
8         if elt == "A" : # On avance
9             x+=dx[ori]
10            y+=dy[ori]
11        elif elt == "G" : # On change l'orientation
12            ori = (ori + 1)%4
13        else : # Virage à droite
14            ori = (ori - 1)%4
15    return (x,y,ori) == (0,0,0) # On regarde si on est dans la configuration d'origine

```

Question 6

```

1 def circuit_convenable1(c):
2     # Sur la même base que la question précédente
3     if contient_demi_tour1(c) or not est_ferme1(c) :
4         return False
5     # definition des coordonnees initiales
6     x,y,ori = 0,0,0
7     # Variation de position en fonction de l'orientation
8     position=[]
9     dx = [1,0,-1,0]
10    dy = [0,1,0,-1]
11    for elt in c :
12        if elt == "A" : # On avance
13            x+=dx[ori]
14            y+=dy[ori]
15            if [x,y] in position : # On est deja passé par ce point
16                return False
17            position.append([x,y])
18        elif elt == "G" : # On change l'orientation
19            ori = (ori + 1)%4
20        else : # Virage à droite
21            ori = (ori - 1)%4
22    return True

```

I.B – Tracé d'un circuit

Question 7

```

1 def dessine_circuit1():
2     for elt in c :
3         if elt == "A" :
4             turtle.forward(d)
5         elif elt == "G" :
6             turtle.left(90)
7         else :
8             turtle.right(90)

```

II Modélisation plus réaliste d'un circuit

II.A – Validation

Question 8

```
1 def element_valide2(e):
2     # L'element est un entier
3     if isinstance(elt,int) :
4         return elt > 0 :
5     # L'element est une liste (virage)
6     elif isinstance(elt,tuple) :
7         r, a = elt
8         if isinstance(r,int) and isinstance(a,int) :
9             return (r > 0) and (-360 < a < 360)
10    return False
```

II.B – Première méthode de tracé à l'écran

Question 9

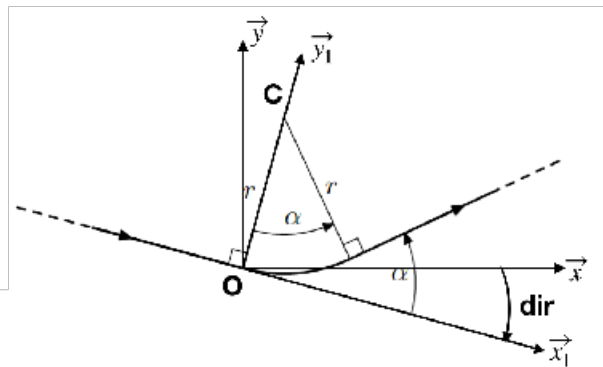
```
1 def dessine_circuit2(c, echelle):
2     for elt in c :
3         if isinstance(elt,int) : # Tout droit
4             turtle.forward(echelle*elt)
5         elif isinstance(elt,tuple) :
6             if elt[1] < 0 : # A droite
7                 r = -elt[0]
8             else :
9                 r = elt[0]
10            turtle.circle(r*echelle,abs(elt[1]))
```

II.C – Tracé pixel par pixel

Question 10

```
1 def matrot(t) :
2     tr = t*np.pi/180 # Conversion en radians
3     return np.array([[np.cos(tr),-np.sin(tr)],[np.sin(tr),np.cos(tr)])]
```

Question 11



Les entrées sont :

- **pos** est un array numpy représentant la position du point d'origine de la courbe (O);
- **dir** est un nombre représentant un angle (en degrés) représentant l'inclinaison d'incidence;
- **r** est un nombre représentant le rayon de courbure;
- **alpha** est un nombre signé représentant l'angle de la portion de courbe.

La sortie renvoie alors la position du centre de courbure (C) de la courbe de rayon r et d'angle α .

$\vec{OC} = r \cdot \vec{y}_1$. Pour trouver les coordonnées du point C , il faut donc ajouter aux coordonnées du point O , le vecteur $r \cdot \vec{y}_1$ écrit dans la base (\vec{x}, \vec{y}) . On applique donc la matrice rotation (d'angle dir) au vecteur \vec{y}_1 . La position du centre dépend du sens de rotation d'où la présence du signe.

Question 12

```

1 def dessine_circuit3(s, c, echelle):
2     position = np.array([len(s)//2, len(s[0])//2]) # Position du centre de l'écran
3     dir = 0
4     for elt in c :
5
6         if isinstance(elt, int) : # Tout droit
7             delta = matrot(dir)@np.array([elt*echelle, 0.]) # Projection dans la base x,y
8             ligne(s, position, position+delta*echelle)
9             position+=delta*echelle # Mise à jour de l'extrémité
10        elif isinstance(elt, tuple) :
11            centre = cc(position, dir, echelle*elt[0], elt[1]) # calcul du centre
12            arc(s, position, centre, elt[1]) #dessin de l'arc
13            dir+=elt[1] # Ajout de l'angle
14            vecteur = centre - position
15            position = centre - matrot(elt[1]) @ vecteur # Rotation du segment point d'origine ->
16            centre

```

III Le parcours d'une voiture

III.A – Formules de calcul des vitesses et temps de parcours

III.A.1)

Question 13 L'expression temporelle de la vitesse lors d'une accélération constante est de la forme $v(t) = a_{max} \cdot t$. Le temps nécessaire est alors $t_{vmax} = \frac{v_{max}}{a_{max}} = \frac{100}{10} = 10 \text{ s}$.

Question 14 Pour une accélération on a $v_2 = a_{max} \cdot t_{min} + v_1$ donc $t_{min} = \frac{v_2 - v_1}{a_{max}}$.

Pour une décélération on a alors $v_2 = f_{max} \cdot t_{min} + v_1$ donc $t_{min} = \frac{v_2 - v_1}{f_{max}}$.

Application numérique : $t_{min} = \frac{120 - 300}{3,6 \times (-20)} = 2,5 \text{ s}$

Question 15 En intégrant la relation de la question précédente : $d = a_{max} \cdot \frac{t_{min}^2}{2} + v_1 \cdot t_{min}$.

$$\begin{aligned} \text{On remplace alors l'expression de } t_{min} \text{ d'où } d &= a_{max} \cdot \frac{(v_2 - v_1)^2}{2 a_{max}^2} + v_1 \cdot \frac{v_2 - v_1}{a_{max}} \\ &= \frac{v_2^2 - 2 v_2 v_1 + v_1^2 + 2 v_1 v_2 - 2 v_1^2}{2 a_{max}} = \frac{v_2^2 - v_1^2}{2 a_{max}}. \end{aligned}$$

Question 16

De la même manière : $d = \frac{v_2^2 - v_1^2}{2 f_{max}} = \frac{120^2 - 300^2}{3,6^2 \times 2 \times (-20)} \simeq 146 \text{ m}$

III.A.2)

Question 17 Il faut le temps d'accélérer jusqu'à v_{max} puis de décélérer pour atteindre v_2 :

$$d_{min} = \frac{v_{max}^2 - v_1^2}{2 a_{max}} + \frac{v_2^2 - v_{max}^2}{2 f_{max}}$$

Question 18 On est sur un profil en trapèze :

- accélération : en $t_a = \frac{v_{max} - v_1}{a_{max}}$ on parcourt $\frac{v_{max}^2 - v_1^2}{2 a_{max}}$;
- décélération : en $t_d = \frac{v_2 - v_{max}}{f_{max}}$ on parcourt $\frac{v_2^2 - v_{max}^2}{2 f_{max}}$;
- à vitesse constante v_{max} il reste à parcourir : $d - d_{min}$ en $t_c = \frac{d - d_{min}}{v_{max}}$.

Le temps nécessaire est alors $t = t_a + t_c + t_d$

Question 19

- Accélération : $t_a = \frac{v_3 - v_1}{a_{max}}$;
- décélération : $t_d = \frac{v_2 - v_3}{f_{max}}$.

Le temps nécessaire est alors $t = t_a + t_d$

III.B – Implantation en Python

III.B.1) Temps pour un tour

Question 20

```

1 def vitesses_entree_max(c,vf) :
2 # Il faut parcourir la liste à l'inverse pour tenir compte de la remarque sur la vitesse ultérieure
3 # En entrée de ligne droite il n'y a pas de limitation de vitesse
4     v= [vf]
5     for elt in c[::-1] :
6         if isinstance(elt,int) :
7             # En ligne droite on utilise la formule de la distance de freinage pour trouver la vitesse max
8             v.append(min(VMAX ,2*FMAX*elt+v[-1]**2))
9         else :
10            # En virage, vitesse minimale entre la vitesse en sortie de virage et celle autorisée par le virage
11            v.append(min(v[-1],vr(elt[0])))
12     return v[:0:-1] # On retourne la liste

```

Question 21

```

1 def temps_droite (d,v1,v2) :
2     # On utilise les notations de la Q19
3     v3 = vmax_droite(d,v1,v2)
4     ta = (v3-v1)/AMAX
5     da = (v3**2-v1**2)/(2*AMAX)
6     if v3 < v2 : # Acceleration pure
7         return ta, v3
8     td = (v2-v3)/FMAX
9     dd = (v2**2-v3**2)/(2*FMAX)
10    if dd > d : # Deceleration trop longue
11        raise ValueError
12    return ta+td+(d-da-dd)/VMAX ,v2

```

Question 22

```

1 def temps_tour (c,v0,vf) :
2     # On calcule les vitesses d'entrée :
3     lv = vitesses_entree_max(c,vf)
4     lv[0] = v0 # la vitesse de depart est égale à v0
5     lv.append(vf)# Pour le calcul de dernier segment
6     t = 0
7     for i,elt in enumerate(c) :
8         if isinstance(elt,int) :
9             #Si on est en ligne droite
10            td,vd = temps_droite(elt, lv[i], lv[i+1])
11            t+=td
12            lv[i+1] = vd # Mise à jour de la vitesse en bout de droite
13        else :
14            # distance = rayon * angle (en radians)
15            t+= elt[0]*np.radians(elt[1])/lv[i]
16    return t

```

