teaching sciences UPSTI for innovation

Proposition de corrigé

Concours : Banque PT

Année : 2021

Filière : PT

Épreuve : Informatique et Modélisation

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 (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

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

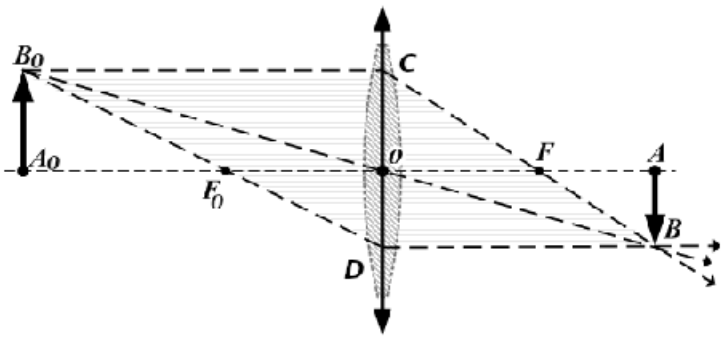
Étude d'un système autofocus d'appareil photo numérique

Corrigé UPSTI

II Modélisation : principe de la méthode de l'auto-focus

II.1 Mise au point

Question 1 À quelle distance d' du centre de la lentille (L) faut-il placer (P) pour avoir une image nette (ceci défini le plan P_0) ? Donner la taille h' de l'image.



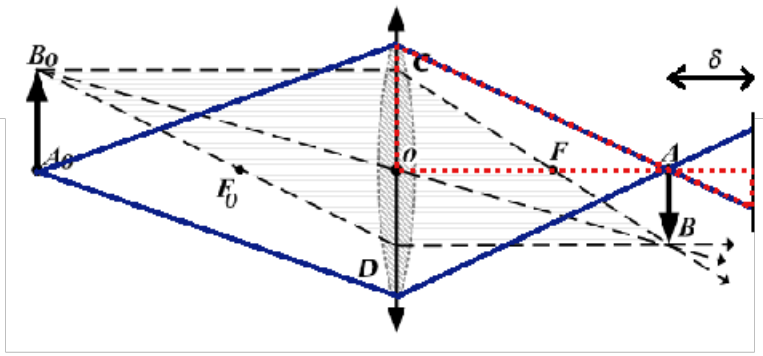
D'après le graphique ci-dessus, il faudra placer le plan récepteur au niveau de l'image AB pour avoir une image nette.

La relation de conjugaison donne : $\frac{1}{d'} - \frac{1}{x_0} = \frac{1}{f'_0}$ soit $\boxed{d' = \frac{x_0 . f'_0}{f'_0 + x_0}}$.

Le grandissement vaut alors : $\frac{\overline{AB}}{\overline{A_0B_0}} = \frac{\overline{OA}}{\overline{OA_0}}$ soit $\frac{-h'}{h} = \frac{d'}{x_0}$. Donc $\boxed{h' = \frac{-f'_0 . h}{f'_0 + x_0}}$

AN : $d' = \frac{-200*10}{10-200} = \frac{2000}{190} \approx 10\,\text{cm}$ et $h' = \frac{-10*10}{10-200} = \frac{100}{190} \approx 0{,}5\,\text{cm}$

Question 2 Faire un schéma et tracer les rayons qui parviennent à l'extrémité de cette tâche. Déterminer le rayon a' de la tâche lumineuse formée sur (P).



On applique Thalès dans le trapèze croisé (tracé en pointillés rouges sur la figure ci-dessus) : $\frac{a}{a'} = \frac{d'}{\delta}$

Donc : $\boxed{a' = \frac{a.\delta}{d'}}$

AN : $a' = \frac{5\times0{,}8}{10{,}5} \approx 0{,}4\,\text{cm}$

Question 3 Après avoir déterminé la taille d'un pixel, supposé carré, donner un critère sur a', puis sur δ pour que l'image transmise par le capteur soit nette.

Le capteur est de taille rectangulaire $20\,\text{mm} \times 30\,\text{mm}$ et comporte 6 mégapixels. Si on note t la taille d'un pixel, on peut écrire :

$20 \times 30 = 6 \times 10^6 \times t^2$ donc $t = \sqrt{\frac{600}{6\times10^6}} = \sqrt{10^{-4}} = 0{,}01\,\text{mm}$

Pour que l'image soit nette, il faut que le point A_0 forme une image au point A plus petite qu'un pixel. On doit donc avoir $2 \cdot a' \leq t$, soit $a' \leq \frac{t}{2}$ ou encore $\boxed{\delta \leq \frac{t\cdot d'}{2\cdot a}}$.

AN :$\delta \leq \frac{0{,}001\times10{,}5}{2\times5} \approx 0{,}001\,\text{cm} = 0{,}01\,\text{mm}$

Note : On aurait aimé ici une discussion sur la valeur (faible) obtenue.

II.2 Principe simplifié de l'autofocus

Question 4 Exprimer d en fonction de d_0 et f'.

$$\frac{1}{d} - \frac{1}{-d_0} = \frac{1}{f'} \quad \Rightarrow \quad \boxed{d = \frac{f' \cdot d_0}{d_0 - f'}}$$

Question 5 On se place dans le cas où A est sur (P). On note A_1 son image par (L_1). Faire un schéma représentant A, A_1, F_1 (le foyer image de (L_1)) et les rayons issus de A passant par les bords inférieurs et supérieurs de (L_1).

Question 6 Déterminer l'ordonnée y_1 de A_1 en prenant l'origine O' située sur l'axe (Ox) (on pourra préalablement déterminer l'ordonnée par rapport à l'axe optique de la lentille (L_1)). En déduire l'expression de y_2 l'ordonnée de A_2 l'image de A par (L_2) en prenant également l'origine en O'.

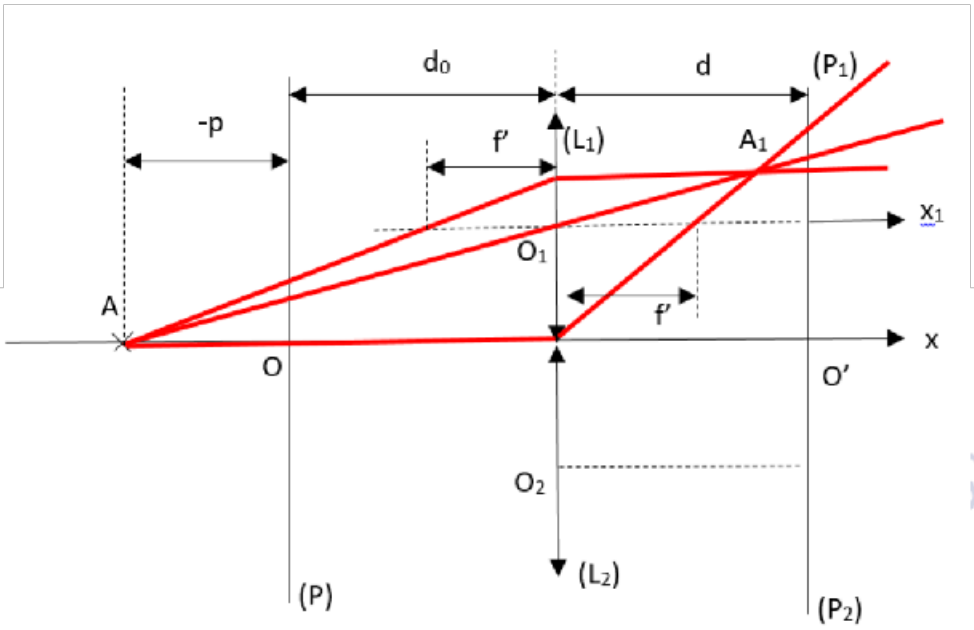
$$\frac{y_1 - a}{a} = \frac{d}{d_0} \quad \Rightarrow \quad y_1 = a + \frac{d \cdot a}{d_0} = a + \frac{f' \cdot a}{d_0 - f'}$$

Par symétrie $y_2 = -y_1$.

Question 7 Calculer $\Delta\Phi_0 = y_1 - y_2$ (appelé différence de phase, même si c'est une longueur) que l'on exprimera en fonction de d_0, d et a.

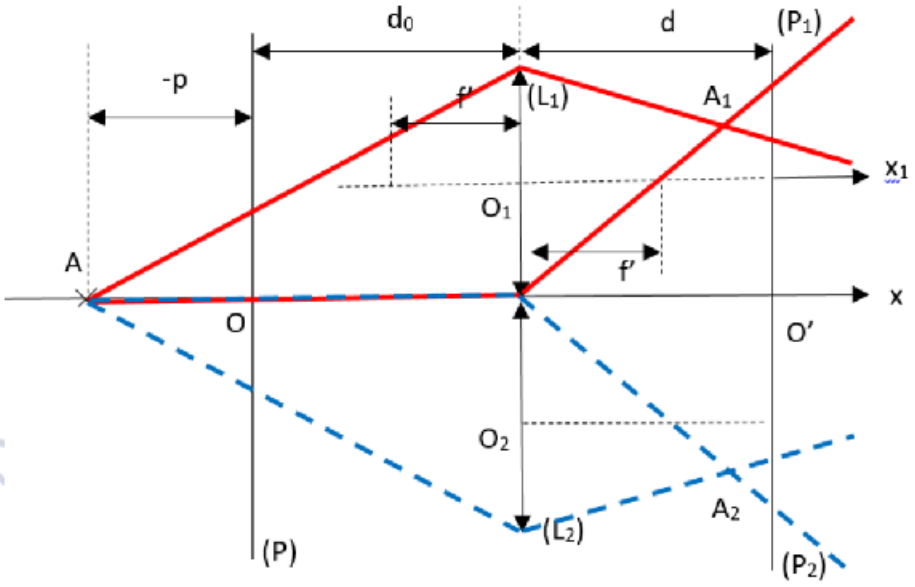
$$\Delta\Phi_0 = 2 \cdot a \left(1 + \frac{d}{d_0}\right)$$

Question 8 Construire A_1 sur le document réponse. On note x_1 l'abcsisse de A_1 mesure sur l'axe (O_1x_1) que l'on ne cherchera pas à exprimer et qui sera donc considérée comme une donnée. Déterminer son ordonnée y_1 mesurée à partir de l'axe (Ox) en fonction de a, x_1, d_0 et p.



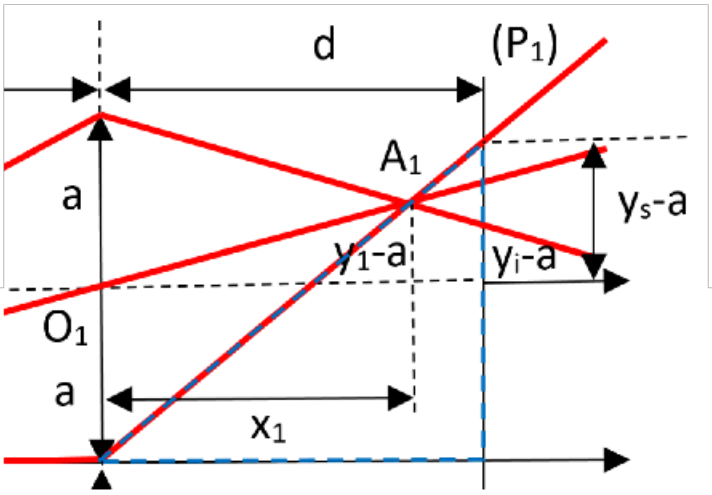
Le grandissement s'écrit : $\frac{y_1 - a}{-a} = \frac{x_1}{p - d_0}$. Donc $\boxed{y_1 = a + \frac{a.x_1}{d_0 - p}}$

Question 9 Tracer les rayons issus de A passant par les extrémités des lentilles (L_1) et (L_2).



Question 10 Exprimer y_s en fonction de y_1, x_1 et d et montrer que $y_i = 2 \cdot a \left(1 - \frac{d}{x_1}\right) + \frac{d \cdot y_1}{x_1}$.

On zoome sur la lentille (L_1) :

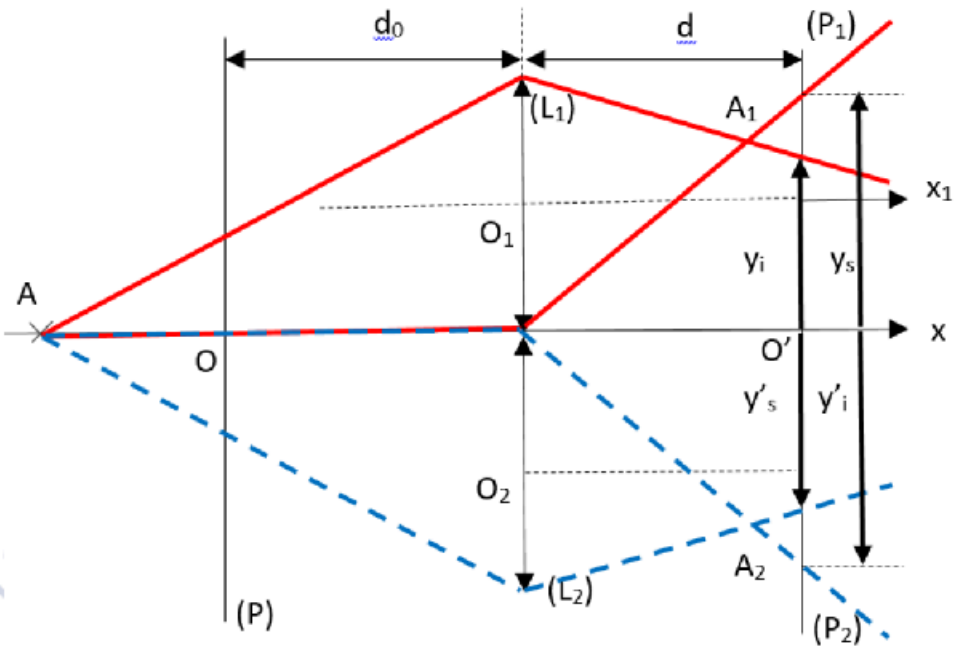


En appliquant Thalès dans le triangle pointillé bleu, on a : $\frac{y_1}{y_s} = \frac{x_1}{d}$, soit $\boxed{y_s = \frac{d \cdot y_1}{x_1}}$

De même, on trouve $\frac{a-(y_i-a)}{d} = \frac{a-(y_1-a)}{x_1}$. Ce qui donne : $y_i = 2 \cdot a - \frac{d(a-(y_1-a))}{x_1}$

Au final : $\boxed{y_i = 2 \cdot a \left(1 - \frac{d}{x_1}\right) + \frac{d \cdot y_1}{x_1}}$

Question 11 Pourquoi a-t-on $y_i = -y'_s$? Montrer que $\Delta\Phi = 2a + \frac{2ad}{d_0 - p}$.



Par symétrie, on voit que y_i et y'_s sont directement opposés. On montrerait de même que $y_s = -y'_i$.

$\Delta\Phi = y_s + y_i = \frac{d \cdot y_1}{x_1} + 2a\left(1 - \frac{d}{x_1}\right) + \frac{d \cdot y_1}{x_1} = 2a + 2d\frac{y_1 - a}{x_1}$

Or d'après le résultat de la question 8, $\frac{y_1 - a}{x_1} = \frac{a}{d_0 - p}$

Au final : $\boxed{\Delta\Phi = 2a + 2d\frac{a}{d_0 - p}}$

Question 12 Evaluer la différence de phase entre le cas où la mise au point n'est pas réalisée et celui où

elle l'est. Soit $\Delta^2\Phi = \Delta\Phi - \Delta\Phi_0$.

$$\Delta^2\Phi = 2a + 2d\frac{a}{d_0 - p} - 2a\left(1 + \frac{d}{d_0}\right)$$

$$\Delta^2\Phi = 2ad\left(\frac{1}{d_0 - p} - \frac{1}{d_0}\right)$$

Question 13 On mesure $\Delta^2\Phi = 0{,}66\,\text{cm}$. Donner la distance de laquelle on doit translater (P) pourobtenir une image nette. On précisera la direction de la translation. On a $d = 2f'$ et $d_0 = 2f'$ avec $f' = 10\,\text{cm}$ et $a = 3\,\text{cm}$.

On extrait p de la relation trouvée à la question précédente : $p = d_0 - \frac{1}{\frac{\Delta^2\Phi}{2ad} + \frac{1}{d_0}}$

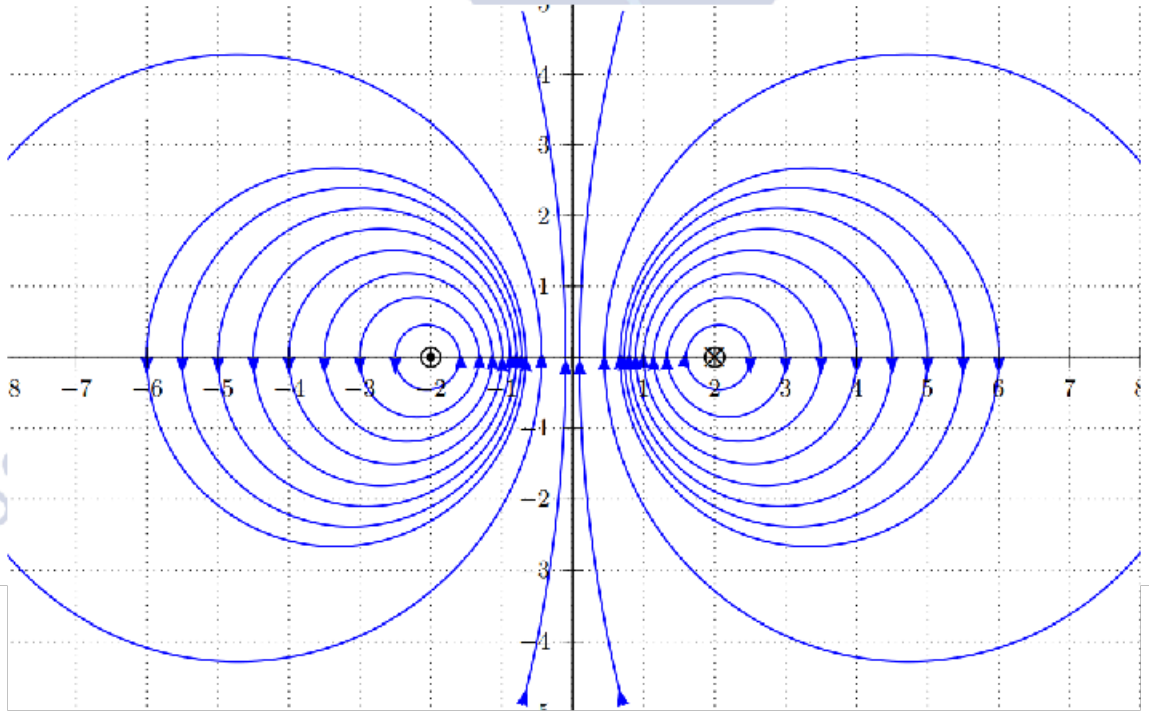
AN : $p = 2 \times 10 - \frac{1}{\frac{0{,}66}{2\times3\times(2\times10)} + \frac{1}{2\times10}} = 20\left(1 - \frac{1}{\frac{0{,}66}{6}+1}\right) = 20\left(1 - \frac{1}{1{,}11}\right) = 20\frac{0{,}11}{1{,}11} \approx 2\,\text{cm}$

Le calcul de p étant algébrique, le point A se situe à droite du plan (P). Il faut donc déplacer le plan (P) vers la droite.

II.3 Moteur pas à pas

Question 14 Déterminer la direction et sens du champ magnétique créé par la bobine B_1 en un point de son plan. Donner la direction d'un moment magnétique à l'équilibre dans un champ magnétique uniforme (à justifier). La bobine B_1 est alimentée seule ; on coupe l'alimentation et on alimente la bobine B_2. Représenter les positions successives du moment magnétique en se limitant aux deux bobines considérées.

Le champ magnétique créé par une spire parcourue par un courant a l'allure suivante :



Ainsi, en tout point du plan de la spire (correspondant à l'axe des abscisses sur l'image ci-dessus), on peut considérer que le champ magnétique créé par la bobine B_1 est dirigé dans la direction $+\vec{x_1}$.

Le moment magnétique $\vec{m}$ fera tourner le rotor du moteur pas à pas. Si on néglige le diamètre du rotor, le champ magnétique peut être considéré de direction $+\vec{x_1}$ constante et d'intensité constante. Le moment magnétique $\vec{m}$ va chercher à s'aligner avec le champ B_1, donc suivant $+\vec{x_1}$.

Si on coupe l'alimentation de la bobine B_1 puis qu'on alimente la bobine B_2, le moment magnétique qui était orienté suivant $+\vec{x_1}$ va devoir s'orienter dans la direction du nouveau champ B_2 créé. Le rotor va donc tourner d'un angle $\frac{\pi}{6}$.

Question 15 On considère une spire plane rectangulaire $ABCD$ de cotés a et b, parcourue par un courant i. Donner l'expression de son moment magnétique. A un aimant on associe également un moment magnétique. Pourquoi ? Quel est l'ordre de grandeur d'un moment magnétique associé à un aimant ?

Le moment magnétique s'exprime par la relation : $\vec{m} = i \cdot S \cdot \vec{n} = i \cdot a \cdot b \cdot \vec{n}$ avec $\vec{n}$ la normale au plan du fil.

Tout comme un courant dans un conducteur, un aimant va créer un champ magnétique. On a alors l'apparition d'un moment magnétique qui crée une action mécanique.

L'ordre de grandeur pour un aimant usuel est : $m \approx 1$ à $10\,\mathrm{A \cdot m^2}$.

Question 16 Déterminer le moment des forces qui s'exercent sur la spire par rapport à l'axe (Δ). Retrouver ce résultat à partir de la forme vectorielle du moment donnée en cours.

Les forces de Laplace s'expriment par la relation : $\overrightarrow{dF_{\text{Laplace}}} = i \cdot d\vec{l} \wedge \vec{B}$.

On additionne les forces sur chaque coté du rectangle $ABCD$:

$\overrightarrow{F_{\text{Laplace}_{AB}}} = -i \cdot a \cdot \vec{z} \wedge (B_r \cdot \overrightarrow{e_r} + B_\theta \cdot \overrightarrow{e_\theta}) = -i \cdot a \cdot (B_r \cdot \overrightarrow{e_\theta} - B_\theta \cdot \overrightarrow{e_r})$

$\overrightarrow{F_{\text{Laplace}_{BC}}} = -i \cdot b \cdot \overrightarrow{e_r} \wedge (B_r \cdot \overrightarrow{e_r} + B_\theta \cdot \overrightarrow{e_\theta}) = -i \cdot b \cdot B_\theta \cdot \overrightarrow{z}$

$\overrightarrow{F_{\text{Laplace}_{CD}}} = i \cdot a \cdot \vec{z} \wedge (B_r \cdot \overrightarrow{e_r} + B_\theta \cdot \overrightarrow{e_\theta}) = i \cdot a \cdot (B_r \cdot \overrightarrow{e_\theta} - B_\theta \cdot \overrightarrow{e_r})$

$\overrightarrow{F_{\text{Laplace}_{DA}}} = i \cdot b \cdot \overrightarrow{e_r} \wedge (B_r \cdot \overrightarrow{e_r} + B_\theta \cdot \overrightarrow{e_\theta}) = i \cdot b \cdot B_\theta \cdot \overrightarrow{z}$

Seules les composantes suivant $\overrightarrow{e_\theta}$ de ces forces vont créer un moment autour de l'axe (Oz).

$\boxed{\overrightarrow{M_{O,\text{Laplace}}} = -2 \cdot i \cdot a \cdot B_r \cdot \frac{b}{2}\vec{z} = -i \cdot a \cdot b \cdot B_r \cdot \vec{z}}$

La formule vectorielle (vue en cours de physique) est : $\overrightarrow{M_{O,\text{Laplace}}} = \vec{m} \wedge \overrightarrow{B} = i \cdot S \cdot \vec{n} \wedge (B_r \cdot \overrightarrow{e_r} + B_\theta \cdot \overrightarrow{e_\theta})$

$\overrightarrow{M_{O,\text{Laplace}}} = i \cdot S \cdot \vec{e_\theta} \wedge (B_r \cdot \overrightarrow{e_r} + B_\theta \cdot \overrightarrow{e_\theta}) = -i \cdot a \cdot b \cdot B_r \cdot \overrightarrow{z}$

Question 17 En utilisant la conversion de puissance $P_{\text{Laplace}} + P_{\text{fem induite}} = 0$, déterminer la force électro-motrice induite dans le cadre lors de son mouvement. Le cadre de résistance R est parcouru par un courant constant i_0. Déterminer la force électro-motrice du générateur qui doit alimenter le cadre.

On considère maintenant les N enroulements de la bobine. Le cadre tourne à une vitesse de rotation $\dot{\theta}$.

$P_{\text{Laplace}} = P_{laplace_{AB}} + P_{\text{Laplace}_{BC}} + P_{\text{Laplace}_{CD}} + P_{\text{Laplace}_{DA}} = N \cdot i \cdot (a \cdot \frac{b}{2} \cdot \dot{\theta} \cdot B_r + 0 + a \cdot \frac{b}{2} \cdot \dot{\theta} \cdot B_r + 0)$

$P_{\text{Laplace}} = N \cdot i \cdot a \cdot b \cdot \dot{\theta} \cdot B_r$

Calcul de la fem induite (notée e) : $P_{\text{Laplace}} + P_{\text{fem induite}} = 0 \Rightarrow N \cdot i \cdot a \cdot b \cdot \dot{\theta} \cdot B_r + e \cdot i = 0$

Donc $\boxed{e = -N \cdot a \cdot b \cdot \dot{\theta} \cdot B_r}$

III Partie informatique

III.1 Mesure du contraste

Question 18 Préciser l'espace mémoire nécessaire pour stocker la valeur d'une composante, puis celle d'un pixel et enfin celle d'une image en Mo (= 1000 ko) ou Mio (= 1024 kio).

Une composante de couleur est un nombre compris entre 0 et 255. Il se stocke sur 1 octet.

Un pixel stocké au format RVB est composé de 3 nombres compris entre 0 et 255. Un pixel nécessite donc 3 octets de stockage.

Une image de 48 MPixels utilisera donc $3 \times 48 \times 10^6$ octets, c'est à dire 144 Mo.

Question 19 Ecrire une fonction `Clinear(val)`, qui prend en argument une valeur de l'espace non linéaire et qui renvoie la valeur linéarisée.

```
def Clinear(val):
    if val<=0.04045:
        Clin=val/12.92
    else:
        Clin=((val+0.055)/1.055)**2.4
    return Clin
```

Question 20 Ecrire une fonction `Y(pix)` qui prend en argument une liste de trois valeurs correspondant à un pixel au format RVB et qui renvoie la valeur Y du niveau de gris dans l'espace non linéaire.

```
def Y(pix):
    Ylin = 0.2126*pix[0] + 0.7152*pix[1] + 0.0722*pix[2]
    if Ylin<=0.0031308:
        Y=12.92*Ylin
    else:
        Y=1.055*Ylin**(1/2.4)-0.055
    return Y
```

Question 21 Ecrire une fonction `NiveauxGris(I)` qui prend en argument une image I au format RVB et qui renvoie une image de même dimension en niveau de gris.

```
def NiveauxGris(I):
    taillex,tailley,nb_couleurs = np.shape(I)   # taille de l'image initiale
    Igris=np.zeros(taillex,tailley) # on crée l'image en niveaux de gris
    for x in range(taillex):
        for y in range(tailley):
            Rlin=Clinear(I[x,y,0])
            Vlin=Clinear(I[x,y,1])
            Blin=Clinear(I[x,y,2])
            Igris[x,y]=Y([Rlin,Vlin,Blin])
    return Igris
```

Question 22 Ecrire une fonction `convolution(A,B)` prenant en argument deux matrices de taille 3x3 et qui renvoie la valeur du produit de convolution.

```
def convolution(A,B):
    conv=0                  # on initialise
    for i in range(3):
        for j in range(3):
            conv = conv + A[i,j]*B[i,j]
    return conv
```

Question 23 Ecrire une fonction `contraste_pixel(I,i,j)` prenant en argument une image I au format niveaux de gris et les coordonnées du pixel (i, j) qui envoie la valeur du contraste défini précédemment par la quantité c.

```
def contraste_pixel(I,i,j):
    Gx=np.array([[ -1, 0, 1],
                 [ -2, 0, 2],
                 [ -1, 0, 1]])
    Gy=np.array([[ -1, -2, -1],
                 [ 0, 0, 0],
                 [ 1, 2, 1]])
    conv1=convolution(I,Gx)
    conv2=convolution(I,Gy)
    c=int(np.sqrt(conv1**2+conv2**2))
    return c
```

Question 24 Ecrire une fonction `contraste(I)` prenant en argument une image I au format niveaux de gris et qui renvoie la valeur du contraste de référence c_{ref}.

```
def contraste(I):
    taillex,tailley = np.shape(I)              # taille de l'image
    cref=0
    for i in range(1,taillex-1):
        for j in range(1,tailley-1):           # on retire les pixels des bords
            cref = cref + contraste_pixel(I,i,j)
    cref = cref / ((taillex-2)*(tailley-2))    # on fait la moyenne
    return cref
```

Question 25 Ecrire une fonction **reglage**, dont les arguments et les valeurs de retour sont à définir, répondant au comportement décrit en partant de la position 0 en pas. On supposera que le maximum de contraste existe.

```
def reglage(I,val):
    '''
    I : image initiale
    val : position de l'objectif '''
    cref=contraste(I)   # contraste de l'image initiale
    sens=1 # sens de déplacement de l'objectif (1 : sens + ; -1 : sens -)
    position_objectif(val+sens) # première itération pour savoir dans quel sens il faut avancer
    Img1=prise()
    c=contraste(Img1)
    if c<cref:  # si on est dans le mauvais sens, on repart de I
        sens = -sens
        c=cref


    while c<=cref:
        cref=c
        val = val+sens
        position_objectif(val)
        Img1=prise()
        c=contraste(Img1)

    return cref
```

Remarque : Cet algorithme est simpliste car il ne cherche le sens de déplacement de l'objectif qu'une fois, et uniquement entre 2 prises. Pour l'améliorer, il faudrait faire plusieurs mesures pour s'assurer du bon sens de déplacement.

III.2 Détection de phase

Question 26 Ecrire une fonction `extraction(L1,L2,dec)` prenant en argument deux listes `L1` et `L2` ainsi qu'une valeur entière de décalage et qui renvoie deux sous-listes à comparer de longueur `len(L1)-dec`, conformément aux règles présentées ci-dessus. On supposera que les deux listes `L1` et `L2` sont de même taille.

```
def extraction(L1,L2,dec):
    if dec==0:
        return L1,L2
    elif dec>0:
        return L1[:-dec],L2[dec:]
    else:
        return L1[-dec:],L2[:dec]
```

Question 27 Ecrire une fonction `comparaison(L1,L2)` (n'utilisant pas la comparaison interne de Python entre les listes) prenant en argument deux listes `L1` et `L2` qui renvoie `True` si les listes sont identiques et `False` sinon.

On peut prévoir le cas où les deux listes ne sont pas de même longueur (elle sont dans ce cas là forcément différentes), même si cette fonction sera utilisée dans ce problème avec des listes de même longueur.

```
def comparaison(L1,L2):
    # Cas des longueurs différentes
    if len(L1) != len(L2):
        return False
    # Cas des longueurs identiques
    s=True
    for i in range(len(L1)):
        if L1[i] != L2[i]:
            s=False
    return s
```

Question 28 Ecrire une fonction `recherche_decalage(L1,L2)` prenant en argument deux listes `L1` et `L2` et qui renvoie la valeur du décalage ou `None` s'il n'existe pas.

```
def recherche_decalage(L1,L2):
    dec=-80
    while dec <= 80: # décalage compris entre -80 et 80 inclus
        L1e, L2e = extraction(L1,L2,dec)
        if comparaison(L1e, L2e):
            return dec
        dec = dec+1
    return None
```

Question 29 Evaluer la complexité de la fonction `recherche_decalage(L1,L2)` en prenant en compte le nombre de comparaisons en fonction de n la taille des listes et de m le nombre de décalages maximal à prendre en compte (161 dans notre exemple) dans le meilleur et dans le pire des cas.

L'extraction se fait en $O(n - |dec|)$ opérations élémentaires, tout comme la comparaison des listes extraites. Chaque itération de la boucle `while` a donc une complexité en $O(n - |dec|)$ opérations élémentaires, soit en $O(n)$ opérations élémentaires (en négligeant dec devant n).

Dans le pire cas, il n'existe pas de décalage. La boucle `while` sera donc itérée m fois : la complexité totale sera en $O(n \times m)$ opérations élémentaires. Dans le meilleur cas, le décalage est de -80, une seule itération de la boucle `while` sera nécessaire. La complexité totale sera en $O(n)$ opérations élémentaires.

Question 30 Ecrire une fonction `erreur(L1,L2)` qui retourne l'erreur quadratique définie ci-dessus.

```python
def erreur(L1,L2):
    err=0
    n=len(L1)
    for i in range(n):
        err = err + (L1[i]-L2[i])**2
    err = err/n
    return err
```

Question 31 Ecrire une fonction `recherche_decalage_2(L1,L2)` qui cherche le minimum de l'erreur pour des décalages de -80 à 80 et qui retourne la valeur de ce décalage.

```python
def recherche_decalage_2(L1,L2):
    # Initialisation de la recherche du minimum
    decalage_min = 0
    valeur_min = -1
    # Valeur aberrante pour l'initialisation (l'erreur quadratique ne peut pas être négative)

    for dec in range (-80, 80+1): # de -80 à +80 inclus
        L1e, L2e = extraction(L1,L2,dec)
        err=erreur(L1e,L2e)
        if valeur_min > err or valeur_min < 0: # On retient le premier minimum
            decalage_min = dec
            valeur_min = err
        dec = dec+1

    return decalage_min
```

III.3 Comparaison des deux méthodes

Question 32 Décrire en 5 lignes maximum les avantages et les inconvénients de ces deux méthodes et laquelle vous semble la plus pertinente.

	Mesure du contraste
Avantages	Simplicité de programmation.
Inconvénients	Calculs itératifs donc position finale plus lente à trouver. On suppose que le contraste possède un minimum.

	Détection de phase
Avantages	Détermination directe du décalage nécessaire.
Inconvénients	Risque d'instabilité plus grand.

III.4 Commande du moteur pas à pas

Question 33 Ecrire une fonction `faire_un_pas_positif(pas_actuel)` qui prend en argument la valeur du pas courant qui modifie l'état des sorties `IN1` à `IN4` et qui renvoie la nouvelle valeur du pas. On veillera à ne changer l'état que des sorties nécessaires.

```python
def faire_un_pas_positif(pas_actuel):
    pas=(pas_actuel+1) % 8
    if pas==0:
        modif_sortie(IN4,False)
```

```
    elif pas==1:
        modif_sortie(IN3,True)
    elif pas==2:
        modif_sortie(IN1,False)
    elif pas==3:
        modif_sortie(IN2,True)
    elif pas==4:
        modif_sortie(IN3,False)
    elif pas==5:
        modif_sortie(IN4,True)
    elif pas==6:
        modif_sortie(IN2,False)
    else:
        modif_sortie(IN1,True)
    return pas
```

On peut compacter ce code en relevant quelques règles :

- Si le pas à réaliser est pair, alors il faut passer une sortie à 0, sinon à 1.
- L'ordre des sorties concernées par des pas est : 4,3,1,2,3,4,2,1.

```
def faire_un_pas_positif(pas_actuel):
    pas=(pas_actuel+1) % 8
    valeur = (pas % 2 == 1)
    sortie = [IN4,IN3,IN1,IN2,IN3,IN4,IN2,IN1][pas]
    modif_sortie(sortie,valeur)
    return pas
```

Question 34 Ecrire une fonction `position_objectif(pas)` qui prend en argument la position en pas à atteindre à partir de la position courante réalisant le déplacement demandé pas à pas. Il peut y avoir plus d'un pas à effectuer. On vérifiera que le pas demandé est atteignable (compris entre 0 et 1000).

```
def position_objectif(pas):
    if pas>=0 and pas<=1000:
        if pas>pas_courant:
            for i in range(pas-pas_courant):
                faire_un_pas_positif(pas_courant+i)
        else:
            for i in range(pas_courant-pas):
                faire_un_pas_negatif(pas_courant-i)
```

IV Gestion des photographies

Question 35 Donner la définition d'une clé primaire.

Une clé primaire est un attribut (en SQL, une colonne) qui permet d'identifier de manière unique une valeur/un enregistrement dans une relation (en SQL : une table).

Question 36 Donner une requête en SQL permettant de sélectionner les identifiants de toutes les photos prises le 11 novembre 2018 .

```
SELECT id FROM PHOTOS WHERE date='20181111';
```

Question 37 Donner une requête en SQL permettant de sélectionner les noms et prénoms des photographes ayant pris des photographies le 11 novembre 2018.

```
SELECT nom,prenom FROM PHOTOGRAPHES
```

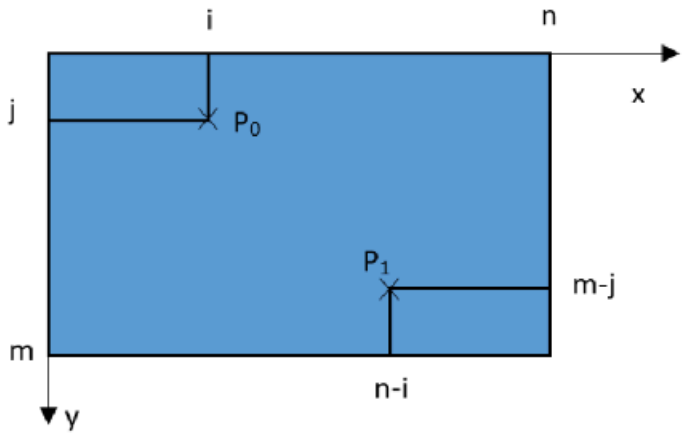
```
                JOIN PHOTOS ON PHOTOGRAPHE.id = PHOTOS.idp
                WHERE date='20181111';
```

Question 38 Donner une requête en SQL permettant de sélectionner les noms, prénoms des photographes et heure de la prise de vue des photographies prises le 11 novembre 2018.

```
SELECT nom,prenom,heure FROM PHOTOGRAPHES
                        JOIN PHOTOS ON PHOTOGRAPHE.id = PHOTOS.idp
                        WHERE date='20181111';
```

Question 39 Illustrer sur un schéma les nouvelles coordonnées d'un pixel de coordonnées (i, j) après une rotation de 180 degrés. Ecrire une fonction `rotation_180(image)` qui prend en argument une image au format RVB et qui renvoie une nouvelle image pivotée de 180°.

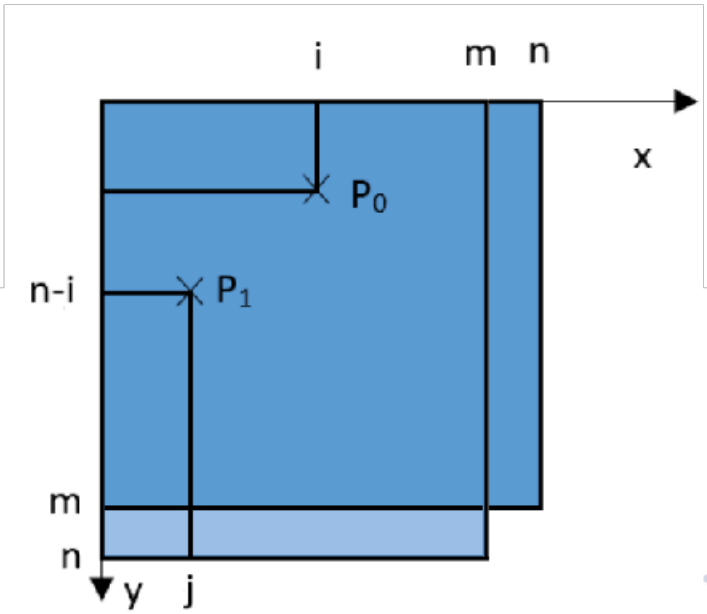
Soit une image de dimensions $n \times m$. Un point de coordonnées (i, j) se retrouvera en position $(n - i, m - j)$.




```
def rotation_180(image):
    taillex,tailley,nb_couleurs = np.shape(image)
    nouvelle_image=np.zeros(taillex,tailley,nb_couleurs)
    for x in range(taillex):
        for y in range(tailley):
            for c in range(nb_couleurs):
                nouvelle_image[x,y,c]=image[taillex-x,tailley-y,c]
    return nouvelle_image
```

Question 40 Illustrer sur un schéma les nouvelles coordonnées d'un pixel de coordonnées (i, j) après une rotation de 90 degrés. Ecrire une fonction `rotation_90(image)` qui prend en argument une image au format RVB et qui renvoie une nouvelle image pivotée de 90° dans le sens trigonométrique.

Soit une image de dimensions $n \times m$. Un point de coordonnées (i, j) se retrouvera en position $(j, n - i)$.




```python
def rotation_90(image):
    taillex,tailley,nb_couleurs = np.shape(image)
    nouvelle_image=np.zeros(tailley,taillex,nb_couleurs)
    for x in range(taillex):
        for y in range(tailley):
            for c in range(nb_couleurs):
                nouvelle_image[x,y,c]=image[y,taillex-x,c]
    return nouvelle_image
```