MINISTÈRES
ÉDUCATION
JEUNESSE
SPORTS
ENSEIGNEMENT
SUPÉRIEUR
RECHERCHE
Liberté
Égalité
Fraternité

EAE SIN 2

SESSION 2026

AGREGATION
CONCOURS EXTERNE

Section : SCIENCES INDUSTRIELLES DE L'INGÉNIEUR

Option : SCIENCES INDUSTRIELLES DE L'INGÉNIEUR ET INGÉNIERIE INFORMATIQUE

MODÉLISATION D'UN SYSTÈME, D'UN PROCÉDÉ OU D'UNE ORGANISATION

Durée : 6 heures

Calculatrice autorisée selon les modalités de la circulaire du 17 juin 2021 publiée au BOEN du 29 juillet 2021.

L'usage de tout ouvrage de référence, de tout dictionnaire et de tout autre matériel électronique est rigoureusement interdit.

Il appartient au candidat de vérifier qu'il a reçu un sujet complet et correspondant à l'épreuve à laquelle il se présente.

Si vous repérez ce qui vous semble être une erreur d'énoncé, vous devez le signaler très lisiblement sur votre copie, en proposer la correction et poursuivre l'épreuve en conséquence. De même, si cela vous conduit à formuler une ou plusieurs hypothèses, vous devez la (ou les) mentionner explicitement.

NB : Conformément au principe d'anonymat, votre copie ne doit comporter aucun signe distinctif, tel que nom, signature, origine, etc. Si le travail qui vous est demandé consiste notamment en la rédaction d'un projet ou d'une note, vous devrez impérativement vous abstenir de la signer ou de l'identifier.
Le fait de rendre une copie blanche est éliminatoire

Tournez la page S.V.P.

INFORMATION AUX CANDIDATS

Vous trouverez ci-après les codes nécessaires vous permettant de compléter les rubriques figurant en en-tête de votre copie.

Ces codes doivent être reportés sur chacune des copies que vous remettrez.

Concours	Section/option	Epreuve	Matière
EAE	1417A	102	2680

Systèmes de surveillance de la qualité de l'air

Ce sujet comporte 46 pages, annexes incluses. Le travail demandé, à rédiger exclusivement sur copies anonymisées, est divisé en **trois parties**, formant un total de **47 questions**. Les trois parties sont indépendantes et relativement équilibrées. Un parcours attentif de l'ensemble du sujet est recommandé avant de composer.

Table des matières

Conventions de notation mathématique

Les formules mathématiques adoptent les conventions d'écriture suivantes :

- Les grandeurs vectorielles spatiales sont écrites en caractères ***italiques gras*** ; implicitement, si $\boldsymbol{v}$ est un vecteur, alors v représente sa norme ; quant au produit vectoriel, il est noté $\times$.
- Les grandeurs matricielles sont simplement écrites en caractères **gras** (non italiques).
- Les grandeurs complexes, c'est-à-dire à valeurs dans $\mathbb{C}$, sont écrites en caractères $\underline{\text{soulignés}}$. L'unité imaginaire est notée $\imath$ (sans point).

IMPRIMERIE NATIONALE – 26 0105 – D'après documents fournis

Présentation générale

La qualité de l'air est une question sociétale qui prend de plus en plus d'ampleur dans le monde. En 2014, l'*Organisation mondiale de la santé* (OMS) estimait, sur la base d'études scientifiques, que la pollution de l'air était la cinquième cause de mortalité humaine, avec environ 7 millions de décès prématurés par an à l'échelle du Globe [1]. Notre pays n'est pas épargné :

- En 2021, l'agence nationale *Santé publique France* a évalué les conséquences de l'exposition chronique aux particules fines à 40 000 décès annuels [2], soit environ 10 fois plus que le chiffre de la mortalité routière.
- En 2015, une commission d'enquête du *Sénat* a chiffré le coût économique et financier annuel de la pollution de l'air à environ 100 milliards d'euros [3], soit environ 4 % du PIB français.

En France, la surveillance de la qualité de l'air est essentiellement décentralisée et confiée, dans chaque région, à une AASQA, c'est-à-dire une **association agréée de surveillance de la qualité de l'air** : *Airparif* en Île-de-France, *Air Breizh* en Bretagne, *Atmo Sud* en Provence Alpes Côte d'Azur, etc. À l'échelle de sa région, chaque AASQA met en place et gère :

- un réseau de **stations de mesure** des principaux polluants aériens, placées sur des sites représentatifs de différents milieux de vie (urbain, rural, etc.) ;
- un **système d'information** permettant d'estimer en temps-réel avec une bonne résolution spatiale les niveaux de concentration des polluants et de prédire leur évolution à brève échéance.

Par exemple, *Airparif* dispose de 70 stations de mesure dont 50 permanentes (fixes) et 20 semi-permanentes, c'est-à-dire pouvant si besoin être relocalisées à proximité des grands axes du trafic routier (principal émetteur de polluants dans la région) – cf. la carte en figure 1 ci-dessous.

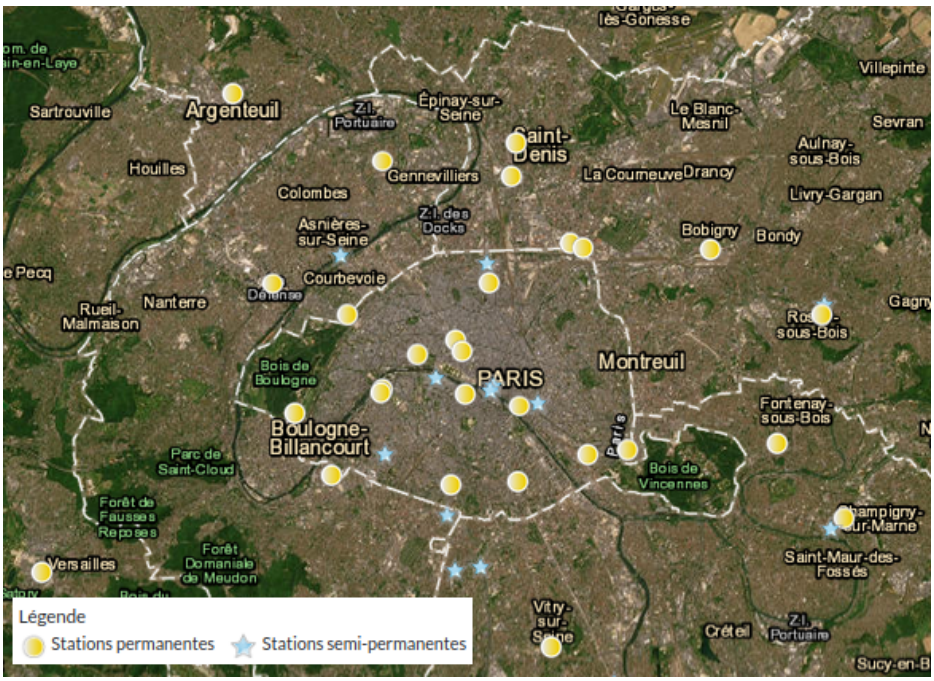


FIGURE 1 – Carte (partielle) des stations de mesure en Île-de-France
www.airparif.fr/carte-des-stations

Typiquement de la taille d'un module préfabriqué, une station de mesure embarque des chaînes d'acquisition complexes, chacune spécifique à un type de polluant (oxydes d'azote, ozone, particules fines…) et entièrement automatisée – cf. les photographies en figure 2, page suivante.

B

Tournez la page S.V.P.

FIGURE 2 – Aspect extérieur et intérieur d'une station de mesure (source *Airparif*)

Chaque station effectue des prélèvements de l'air extérieur et des analyses tous les quarts d'heure afin d'établir une moyenne horaire des polluants pour lesquels elle est équipée. Ces moyennes sont immédiatement envoyées au système informatique de l'AASQA pour être intégrées aux bases de données et aux modèles numériques de cartographie temps-réel de la concentration des polluants. Les résultats sont alors publiés sur un site dédié à l'information du grand public – cf. à titre d'exemple la capture d'écran en figure 3 ci-dessous.

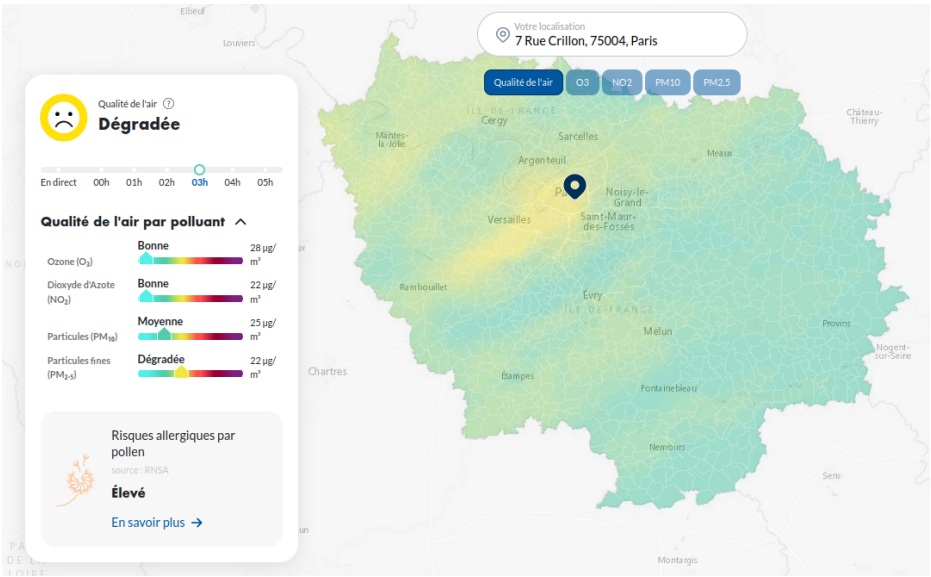


FIGURE 3 – Carte interactive de la qualité de l'air en Île-de-France (www.airparif.fr)

Cas des particules fines

Les particules d'aérosol dites « **particules fines** » de façon simpliste forment une famille de polluants aériens étroitement surveillés en raison de leur nocivité (depuis 2013, elles sont classées *cancérogène certain* par le *Centre international de la recherche sur le cancer* [4]). Elles sont définies par catégories gigognes de taille, chacune désignée par le sigle PM (de l'anglais *particulate matter*) suivi d'un nombre en indice qui précise la valeur supérieure du *diamètre aérodynamique*[1] des particules de la catégorie, exprimée en microns. Ainsi, le sigle PM_{10} fait référence aux particules de diamètre inférieur ou égal à 10 µm. Car la taille est un paramètre déterminant, non seulement en termes d'effets délétères (les particules pénètrent d'autant plus profondément dans l'organisme qu'elles sont petites), mais aussi en termes de détection et de protection.

1. La notion de *diamètre aérodynamique équivalent* d'une particule de matière est celui d'une sphère de densité 1 (celle de l'eau) qui aurait la même vitesse de chute dans l'air que celle de la particule considérée. Elle fait abstraction des variations de forme d'une particule à l'autre (comme le fait aussi la notion de diamètre volumique équivalent) sur la base du comportement aérodynamique des particules (l'aptitude à rester en suspension dans l'air) [5].

Les particules fines sont des polluants très répandus, avec des sources d'émission naturelles (sables, sels de mer, poussières issues de la croûte terrestre soulevées par le vent, fumées volcaniques, pollens et spores, etc.) mais surtout anthropiques : fumées de combustion (moteurs, appareils de chauffage, industries métallurgiques et pétrochimiques), usure des pneumatiques et des dispositifs de freinage des véhicules, poussières générées par les activités minières, agricoles, etc. Elles sont quantifiées par leur **concentration massique** dans l'air, exprimée en $\mu g\,m^{-3}$ conformément aux normes et recommandations sanitaires internationales, dont la table ci-dessous rappelle les valeurs maximales recommandées, actualisées par l'*Organisation mondiale de la Santé* en 2021 [6].

	$PM_{2.5}$	PM_{10}
moyenne annuelle	5	15
moyenne journalière sur 3 à 4 jours excédentaires	15	45

TABLE 1 – Valeurs maximales recommandées de concentration en $\mu g\,m^{-3}$

Un autre aspect crucial est la **composition chimique** des particules fines, qui a un impact majeur sur leur nocivité. Sans surprise, des analyses menées en 2011-2012 par *Airparif* et le *LSCE*[2] sur les échantillons prélevées dans des stations de mesure ont mis en évidence :

- une grande diversité de composition des particules – qu'on présente typiquement dans les catégories « carbone suie » (EC pour *elementary carbon*, aussi appelé *black carbon*), « matière organique » (OM), « aérosols inorganiques secondaires » (SIA), « poussières crustales » (CD), « sels de mers » (SS) et autres (dont les oxydes métalliques d'origine anthropique…) ;
- des proportions significativement différentes des types chimiques de particules selon que la station de mesure est en milieu rural, urbain ou à proximité d'un axe à fort trafic routier ;

comme l'illustre l'infographie en figure 4 ci-dessous [7] (reformatée pour une meilleure lisibilité).

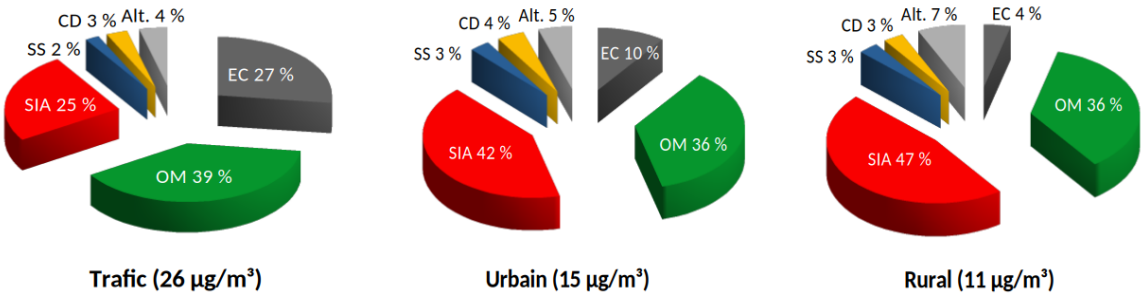


FIGURE 4 – Composition d'échantillons de particules $PM_{2.5}$ prélevées en Île-de-France sur 3 sites (moyennes annuelles de concentration indiquées entre parenthèses)

Les **chaînes de mesure de concentration des particules fines** installées dans les stations de mesure des AASQA sont des équipements complexes et volumineux (cf. celui représenté en fig. 2, p. 3). Ils comportent un préleveur séquentiel de poussières pour maîtriser précisément le volume d'air prélevé, associé à l'un des deux instruments de mesure suivants :

- une microbalance à cône oscillant (*tapered element oscillating microbalance* – TEOM [9]), dont la fréquence d'oscillation varie avec la masse déposée sur son élément conique ;
- un analyseur de masse à atténuation bêta (*beta attenuation monitoring* – BAM [10]), qui détermine, par absorption de rayonnement, la masse des particules piégées sur un filtre ruban.

2. *Laboratoire des sciences du climat et de l'environnement*, CNRS-CEA, Gif-sur-Yvette, France

Tournez la page S.V.P.

Ces technologies entièrement automatisées ont l'une comme l'autre l'avantage de fournir des valeurs précises de concentration massique des particules fines par catégories de tailles, et avec une résolution temporelle satisfaisante (plusieurs mesures par heure). Mais elles présentent l'inconvénient d'être onéreuses, tant à l'acquisition (une chaîne de mesure coûte plusieurs dizaines de milliers d'euros) qu'en maintenance périodique. Elles ne sont donc pas installées dans toutes les stations de mesure.

Et c'est pourquoi depuis plus d'une dizaine d'années, les AASQA s'intéressent de près aux technologies alternatives à bas coût, notamment celle de **comptage optique** qui est mise en œuvre dans les moniteurs de qualité de l'air à destination du grand public, ainsi que pour la surveillance du niveau de poussières dans les « salles blanches » [8].

Cas du dioxyde d'azote

Le dioxyde d'azote (NO_2) est un gaz polluant nocif pour le système respiratoire, avec des effets avérés aussi bien à court qu'à long terme. Il est principalement émis par les processus de combustion et en particulier par les moteurs thermiques, ce qui en fait l'un des principaux polluants atmosphériques dans les zones urbaines. Ce polluant peut également se former dans l'atmosphère à partir du monoxyde d'azote (NO), sous l'effet de réactions chimiques impliquant l'ozone et d'autres composés oxydants.

Bien que le NO_2 ne soit pas un gaz à effet de serre, il influence le changement climatique en participant à la production d'ozone troposphérique et en réduisant la durée de vie du méthane. Historiquement associé aux pluies acides, ce phénomène est aujourd'hui rare dans les régions développées, mais le NO_2 continue d'avoir un impact environnemental notable, notamment via la formation de particules organiques secondaires comme le nitrate d'ammonium. Malgré les progrès réalisés en matière de qualité de l'air, de nombreuses populations restent exposées à des concentrations dépassant les seuils recommandés par l'OMS, fixés à $10\,\mu g\,m^{-3}$ en moyenne annuelle et $25\,\mu g\,m^{-3}$ en moyenne horaire. Une exposition prolongée au NO_2 aggrave les symptômes respiratoires, notamment chez les personnes asthmatiques et les enfants, et peut entraîner une diminution de la fonction pulmonaire. À des concentrations très élevées, ce gaz provoque une inflammation aiguë des voies respiratoires.

De plus, le NO_2 joue un rôle indirect dans la dégradation de la qualité de l'air en favorisant la formation de particules fines. Les niveaux de pollution au NO_2 varient fortement selon les zones géographiques, avec des concentrations plus élevées dans les agglomérations urbaines, en particulier à proximité des axes routiers majeurs.

1 – Technologie des compteurs optiques de particules

Mise en situation

Techniquement, un **compteur optique de particules** (*optical particle counter*, abrégé OPC) est un capteur de petite dimension capable de compter les PM_1, $PM_{2.5}$ et PM_{10} – cf. par exemple le modèle *NextPM* représenté en figure 5 ci-contre, développé et fabriqué par la société française *Tera Sensor* [11] (ses caractéristiques seront utilisées pour les applications numériques). À titre de comparaison avec les technologies de référence onéreuses et encombrantes mentionnées supra (TEOM ou BAM), il ne mesure que quelques centimètres de côté et est commercialisé sur le marché européen à un prix public inférieur à cent euros (2025).

FIG. 5 – La sonde *NextPM*

Un tel capteur est constitué d'une chambre noire (blindée contre toute lumière externe) dans laquelle on trouve les éléments suivants (cf. l'infographie en figure 6 ci-dessous).

- Un système de drainage à débit connu (typiquement, un ventilateur dont la fréquence de rotation est asservie) véhicule un flux d'air ambiant entre deux orifices, l'un d'admission et l'autre d'échappement.
- Une source de lumière (typiquement, un tube à diode laser) engendre un faisceau qui traverse le flux d'air dans une zone appelée chambre de diffusion (*scattering chamber*) ; au delà, ce faisceau est absorbé dans une chambre d'extinction (*light dump*).
- Une cellule photosensible (typiquement, une photodiode) positionnée à quelques mm de la chambre de diffusion mesure la lumière diffusée (*scattered light*) par les particules dans le flux d'air et éclairées par la source.

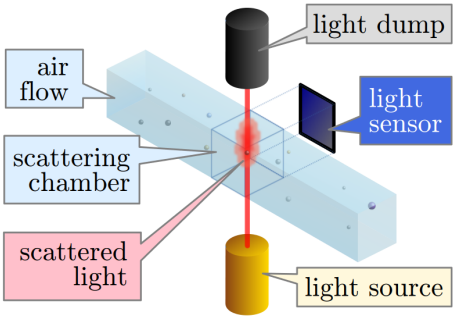


FIG. 6 – Technologie d'un OPC

- Une carte électronique (non représentée sur la figure 6) compte les impulsions de lumière captées et, via une chaîne d'amplification, quantifie leur amplitude afin d'en déduire la taille des particules ; moyennant certaines hypothèses, elle détermine alors leur concentration massique.

L'objectif de cette partie du travail demandé est justement d'étudier dans quelle mesure un OPC peut établir la concentration massique des particules fines dans l'air ambiant.

Tournez la page S.V.P.

1.1 Caractérisation de l'onde diffusée par une particule

■ La diffusion lumineuse (*light scattering*) est le phénomène optique qui donne au ciel sa couleur bleue et aux nuages leur couleur blanche, ou encore qui permet de voir de côté le faisceau d'un phare dans la nuit. En effet, lorsque qu'une onde lumineuse incidente rencontre une particule de matière, cette dernière la *diffuse* dans toutes les directions de l'espace, de façon anisotrope, par le jeu de multiples interactions (réflexion, diffraction, réfraction – cf. la figure 7 ci-contre). Ce phénomène a été théorisé en 1908 par le physicien allemand Gustav Mie comme une solution exacte particulière des équations de Maxwell, d'où le nom de **théorie** ou **diffusion de Mie**.

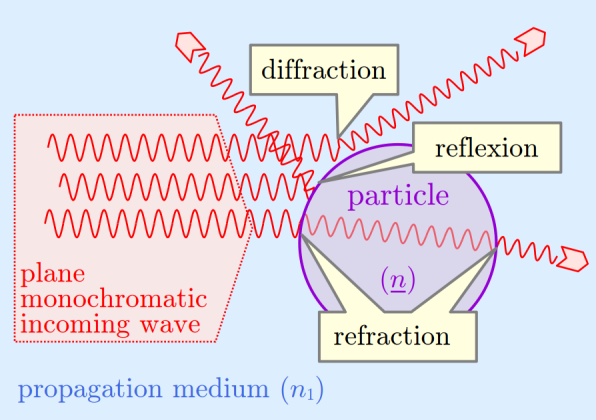


FIG. 7 – Principe de la diffusion de Mie

Cette théorie s'applique de façon pertinente au cadre d'étude d'un OPC, détaillé ci-dessous.

- La source de lumière est caractérisée par sa longueur d'onde $\lambda = 650\,\text{nm}$ dans le vide (valeur typique d'un laser rouge) et une polarisation linéaire constante d'amplitude électrique maximale notée E_0.
- Le milieu de propagation est l'air, réputé linéaire, homogène, isotrope, non absorbant et non dispersif, avec un indice de réfraction $n \approx 1{,}0003$ pour la longueur d'onde λ. On peut donc raisonnablement prendre $n \approx 1$ dans tous les calculs qui suivent.
- On associe à ce milieu un repère cartésien $(O, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$, en plaçant l'origine O au centre de la particule de matière et en considérant l'axe z comme la direction de propagation de l'onde incidente, l'axe x étant la direction de polarisation.
- Pour simplifier les expressions, on pose $\psi = 0$ où ψ est le retard de phase à l'origine.

Q1 En s'aidant de l'annexe A.1, donner l'expression littérale de la fonction d'onde incidente $\underline{\boldsymbol{E}}_{\text{inc}}$. Préciser aussi l'expression de son nombre d'onde k et de sa pulsation ω.

■ La **particule** de matière est supposée sphérique de diamètre d compris entre 0,1 et 10 µm, et constituée d'un matériau homogène, isotrope, d'indice de réfraction complexe $\underline{n}_1$.

Pour caractériser la fonction d'onde diffusée $\underline{\boldsymbol{E}}_{\text{sca}}$ en un point arbitraire M de l'espace défini par ses coordonnées polaires (r, θ, φ), on définit la base vectorielle d'observation $(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w})$ telle que :

$$\boldsymbol{w} = \frac{\boldsymbol{OM}}{OM}, \qquad \boldsymbol{v} = \boldsymbol{z} \times \boldsymbol{w}, \qquad \boldsymbol{u} = \boldsymbol{v} \times \boldsymbol{w}$$

où le vecteur $\boldsymbol{w}$, orienté par l'angle polaire $\theta = (\widehat{\boldsymbol{z}, \boldsymbol{w}})$, donne la direction de propagation de l'onde diffusée au point M. On définit également le vecteur directeur $\boldsymbol{n} = \boldsymbol{v} \times \boldsymbol{z}$, orienté par l'angle azimutal $\varphi = (\widehat{\boldsymbol{x}, \boldsymbol{n}})$. Enfin, on note $\hat{\boldsymbol{u}}$ le vecteur qui donne la direction de polarisation de l'onde diffusée, défini par un angle $\alpha = (\widehat{\boldsymbol{u}, \hat{\boldsymbol{u}}})$ dans le plan $(O, \boldsymbol{u}, \boldsymbol{v})$. Tous ces éléments sont représentés sur le schéma en figure 8 p. 8.

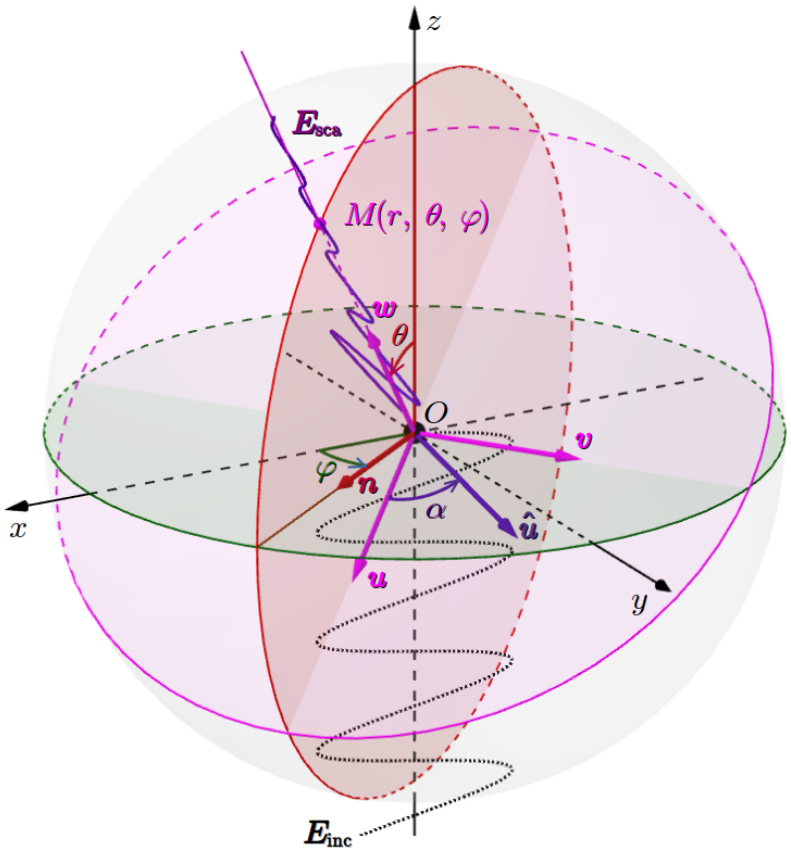


FIGURE 8 – Géométrie de la diffusion de Mie
(amplitudes non réalistes pour une bonne visualisation des ondes)

Q2 En illustrant le raisonnement par une figure, montrer que la fonction d'onde incidente $\underline{\boldsymbol{E}}_{\text{inc}}$ peut s'exprimer dans la base vectorielle $(\boldsymbol{n}, \boldsymbol{v})$ du plan $(O, \boldsymbol{x}, \boldsymbol{y})$ comme ci-dessous :

$$\underline{\boldsymbol{E}}_{\text{inc}}(z,t) = E_0\, e^{\imath(kz-\omega t)} \begin{pmatrix} \cos\varphi \\ -\sin\varphi \end{pmatrix}_{(\boldsymbol{n},\boldsymbol{v})} \tag{1}$$

■ La particule de matière peut être vue comme un système optique qui transforme l'onde incidente en une onde diffusée. Compte tenu des hypothèses précédentes, la théorie de Mie établit la relation entre les deux fonctions onde $\underline{\boldsymbol{E}}_{\text{inc}}$ et $\underline{\boldsymbol{E}}_{\text{sca}}$ sous la forme de l'équation vectorielle complexe suivante :

$$\underline{\boldsymbol{E}}_{\text{sca}\,(\boldsymbol{u},\boldsymbol{v})}(r,\theta,\varphi,t) \;=\; -\frac{1}{\imath k}\,\frac{e^{\imath k(r-z)}}{r}\,\underline{\mathbf{S}}(\theta)\cdot\underline{\boldsymbol{E}}_{\text{inc}\,(\boldsymbol{n},\boldsymbol{v})}(z,t) \tag{2}$$

où $\underline{\mathbf{S}}$ est une matrice carrée diagonale de dimension 2, appelée **matrice d'amplitude de diffusion**, dont les coefficients ne dépendent que de l'angle polaire θ.

Q3 Montrer que l'expression de la fonction d'onde diffusée $\underline{\boldsymbol{E}}_{\text{sca}}$ dans la base $\boldsymbol{u}, \boldsymbol{v}$ est indépendante de la coordonnée spatiale z. Expliquer à quoi correspond le facteur restant qui est fonction des coordonnées de propagation r et t.

Q4 Calculer dans le pire cas la distance d'observation à partir de laquelle le modèle de la fonction d'onde diffusée $\underline{\boldsymbol{E}}_{\text{sca}}$ serait a priori valide. Conclure.

Tournez la page S.V.P.

Q5 En faisant abstraction de la matrice $\underline{\mathbf{S}}$, calculer l'ordre de grandeur du rapport d'amplitudes $\mathcal{H} = |\underline{\boldsymbol{E}}_{\text{sca}}| \,/\, |\underline{\boldsymbol{E}}_{\text{inc}}|$ de l'onde diffusée comparée à l'onde incidente à une distance $r = 5\,\text{mm}$ de la particule (typiquement là où est située la cellule de l'OPC). Conclure brièvement.

1.2 Modélisation du mesurande d'un OPC

■ Dans les formules de la théorie de Mie intervient fréquemment le **paramètre de taille**, noté x et défini par $x = \pi d/\lambda$. C'est donc le rapport entre le périmètre de la section transversale de la particule et la longueur d'onde de l'onde incidente dans son milieu de propagation. Au delà de son utilité théorique, le paramètre de taille x est aussi, dans la pratique, un indicateur dimensionnel pertinent pour juger de la bonne adéquation de la théorie de Mie au problème considéré. En effet, pour obtenir des approximations numériques satisfaisantes de l'éclairement diffusé par la particule, il est réputé nécessaire de calculer au moins $N = x + 4x^{1/3} + 2$ coefficients de Mie [3].

Q6 Avec toutes les hypothèses précédentes, calculer dans le pire cas le nombre N de coefficients de Mie nécessaires pour modéliser l'éclairement reçu par le capteur. Conclure brièvement.

■ La théorie de Mie met également en évidence le fait que la zone d'influence de la particule sur l'onde incidente est plus grande que la particule elle-même, notamment à cause des phénomènes bien connus de diffraction se produisant aux abords de la matière. Cet aspect est caractérisé par la notion de **pouvoir de diffusion** (en anglais, *scattering efficiency*). C'est un nombre réel positif sans dimension qui est défini comme le quotient $Q_{\text{sca}} = P_{\text{sca}}/P_{\text{inc}}$ où :

- P_{sca} est la puissance de l'onde diffusée par la particule ;
- P_{inc} est la puissance reçue par la particule en étant éclairée par l'onde incidente.

De manière analogue, la théorie de Mie définit aussi :

- le pouvoir d'absorption $Q_{\text{abs}} = P_{\text{abs}}/P_{\text{inc}}$ où P_{abs} est la puissance absorbée par la particule ;
- le pouvoir d'extinction (ou de dissipation) $Q_{\text{ext}} = Q_{\text{sca}} + Q_{\text{abs}}$;

Ces grandeurs sont calculables sur la base des coefficients de Mie à l'aide de moyens informatiques. Elles peuvent prendre des valeurs supérieures à 1.

On peut alors définir respectivement les **sections transversales** de diffusion, d'absorption et d'extinction, par les formules suivantes :

$$\sigma_{\text{sca}} = Q_{\text{sca}}.\sigma \ , \qquad \sigma_{\text{abs}} = Q_{\text{abs}}.\sigma \ , \qquad \sigma_{\text{ext}} = Q_{\text{ext}}.\sigma$$

où $\sigma = \pi(d/2)^2$ est la *surface apparente* de la particule éclairée par l'onde incidente (en anglais, *particle cross section*). Homogènes à des surfaces – et typiquement exprimées en μm^2 – ces quantités n'ont pas d'interprétation physico-géométrique. Elles sont surtout utilisées pour pondérer des calculs d'intégration de densités de puissance de l'onde respectivement diffusée, absorbée et éteinte (dissipée) par la particule, comme on le verra par la suite.

3. Les coefficients de Mie sont les termes de suites complexes notées $(\underline{a}_n)$ et $(\underline{b}_n)$, composées de fonctions de Riccati-Bessel et Hankel [14]. Il n'est pas opportun d'en détailler ici les formules et l'exploitation calculatoire

À titre académique, le programme suivant (incomplet) a pour but d'afficher dans la console d'exécution les valeurs respectives de Q_{sca}, σ_{sca}, Q_{abs} et σ_{abs} d'une particule de diamètre $d = 2{,}5\,\mu\text{m}$, éclairée par une lumière incidente de longueur d'onde $\lambda = 650\,\text{nm}$, pour un échantillon de **trois matériaux** pouvant la constituer : quartz (sable), suie de moteur diesel (en anglais, *diesel soot*) et oxydes de fer (composants majoritaires des particules émises par les système de freinage) [4].

```
1  import numpy             as np
2  import miepython         as mie
3  import matplotlib.pyplot as plt
4  import scipy.integrate   as spi
5
6  ### Particle CRI taken at wl = 650 nm from https://refractiveindex.info/
7  materials = [
8    {
9      'name' : 'quartz',       # SiO2
10     'cri'  : 1.4565 + 0.0j,
11     'color': 'goldenrod'
12   },
13   {
14     'name' : 'diesel soot', # Elementary carbon
15     'cri'  : 1.649 + 0.288j,
16     'color': 'darkgreen'
17   },
18   {
19     'name' : 'iron oxydes', # Fe2O3 (50%), Fe3O4 (50%)
20     'cri'  : 2.72 + 0.0975j,
21     'color': 'dodgerblue'
22   }
23 ]
24
25 ### Geometrical parameters
26 wl = 650.0  # incident wavelength in the vacuum (in nm)
27 d  = 2.5E3  # particule diameter           (also in nm)
28 x  = ???    # size parameter            (dimensionless)
29 s  = ???    # particle cross section          (in um^2)
30
31 ### Efficiencies and cross sections display
32 print(f"Size parameter x = {x:.2f}   particle cross section s = {s:.2f} um^2")
33 print(f"{'MATERIAL':>12}{'Qsca':>8}{'Ssca':>8}{'Qabs':>8}{'Sabs':>8}")
34 ???         # one line per material to be displayed
```

Q7 À l'aide des annexes A.2 et A.3, finaliser le codage des lignes n$^{\text{os}}$ 28 & 29 (champs marqués « ??? ») et à partir de la ligne n$^{\text{o}}$ 34, afin d'obtenir un affichage formaté des valeurs de Q_{sca}, σ_{sca}, Q_{abs} et σ_{abs} à 10^{-2} comme ci-dessous.

```
Size parameter x = 12.08   particle cross section s = 4.91 um^2
    MATERIAL    Qsca    Ssca    Qabs    Sabs
      quartz    2.93   14.40    0.00    0.00
 diesel soot    1.20    5.88    1.15    5.64
 iron oxydes    1.36    6.69    0.98    4.81
```

Q8 En procédant par comparaisons et en s'appuyant sur des connaissances générales, pour chacun des trois matériaux étudiés, commenter brièvement les résultats numériques obtenus dans l'affichage donné à la question précédente.

4. Dans ce programme, les lignes n$^{\text{o}}$ 3 & 4 ainsi que l'attribut `'color'` des éléments de la liste `materials` seront exploités ultérieurement pour tracer des courbes.

Tournez la page S.V.P.

■ La théorie de Mie quantifie l'éclairement énergétique $E_{\text{e sca}}$ de l'onde diffusée tout autour de la particule, dans un angle solide Ω donné [5]. Avec les hypothèses précédentes, elle établit que $E_{\text{e sca}}(\Omega)$ est invariant par rotation selon l'angle azimutal φ, mais en revanche très dépendant de l'angle polaire θ. Pour caractériser cette dépendance, elle définit la **fonction de phase** :

$$p(\theta) = \frac{1}{f} \times \frac{|\underline{\boldsymbol{S}}_1(\theta)|^2 + |\underline{\boldsymbol{S}}_2(\theta)|^2}{2} \tag{3}$$

où $\underline{S}_1$ et $\underline{S}_2$ sont les valeurs propres de la matrice d'amplitude de diffusion $\underline{\mathbf{S}}$ – introduite dans l'équation (2) p. 8 – et où f est le **facteur de normalisation**, un coefficient scalaire qui ajuste l'échelle des valeurs de p en vue de son intégration. La fonction p étant paire, il suffit de calculer ses valeurs dans l'intervalle $\theta \in [0;\pi]$ pour la caractériser complètement.

Dans la continuité du programme donné en page 10, les instructions suivantes (incomplètes) ont pour but de tracer en coordonnées polaires semi-logarithmiques la courbe représentative de p pour chacun des trois matériaux détaillés dans la liste `materials`.

```
40 ### Phase functions display
41 fig1, ax1 = plt.subplots(subplot_kw={'projection': 'polar'})
42 ax1.set_yscale('log') # Set logarithmic radial scale
43 p_max = ???           # initial absolute max value of all phase functions
44 p_min = ???           # initial absolute min value of all phase functions
45 theta_range = ???     # [0; pi] with a 0.2 degree resolution
46 mu_range    = ???     # mu = cos(theta)
47 for material in materials:
48   s1_range, s2_range = mie.S1_S2(material['cri'], x, mu_range, 'one')
49   p_range = ???       # values of the current phase function
50   p_min   = ???       # conditional update
51   p_max   = ???       # conditional update
52   ax.plot(theta_range, p_range, material['color'], linewidth=1.5)
53 ax1.set_rmin(???)     # Set the 10^k_min radius to be displayed on the grid
54 ax1.set_rmax(???)     # Set the 10^k_max radius to be displayed on the grid
55 # ...
```

Attention, la fonction de haut niveau `S1_S2` (cf. annexe A.2 p. 33) présente deux particularités.

- Elle prend pour paramètre géométrique principal non pas θ mais $\mu = \cos\theta$, car c'est cette expression qu'on trouve directement dans le calcul des coefficients de Mie.
- Elle distribue le facteur $\sqrt{f}$ dans chacune des valeurs de retour `S1` et `S2`, sachant que les valeurs « brutes » de $\underline{S}_1$ et $\underline{S}_2$ sont calculées par la fonction de bas niveau `_S1_S2`.

Q9 Toujours à l'aide des annexes A.2 et A.3, compléter les lignes n$^{\text{os}}$ 45, 46 et 49 (champs marqués « `???` ») pour coder les listes de valeurs représentant respectivement les grandeurs θ, μ et p calculées pour $\theta \in [0;\pi]$ avec une résolution de 0,2°.

Q10 Compléter les lignes n$^{\text{os}}$ 43 & 44, 50 & 51, 53 & 54 pour calculer, en puissances entières de 10, les valeurs optimales des bornes respectivement haute et basse de l'échelle radiale logarithmique du diagramme, en fonction des valeurs minimales et maximales mémorisées de p.

5. Dans l'espace tridimensionnel, un **angle solide** Ω mesure en *stéradian* (symbole sr) l'« ampleur » d'un cône de sommet O par le rapport s/r^2 où s est l'aire de la calotte sphérique de centre O et de rayon r interceptée par le cône. Dans le système de coordonnées polaires (r,θ,φ), un angle solide Ω à section rectangulaire est défini par le produit d'intervalles $[\theta_1;\theta_2]\times[\varphi_1;\varphi_2]$. Un élément infinitésimal dΩ s'exprime alors par la formule $\mathrm{d}^2\Omega = \sin\theta\,\mathrm{d}\theta\,\mathrm{d}\varphi$.

Correctement paramétré et complété, le programme en page 11 produit le diagramme représenté sur la figure 10 ci-dessous.

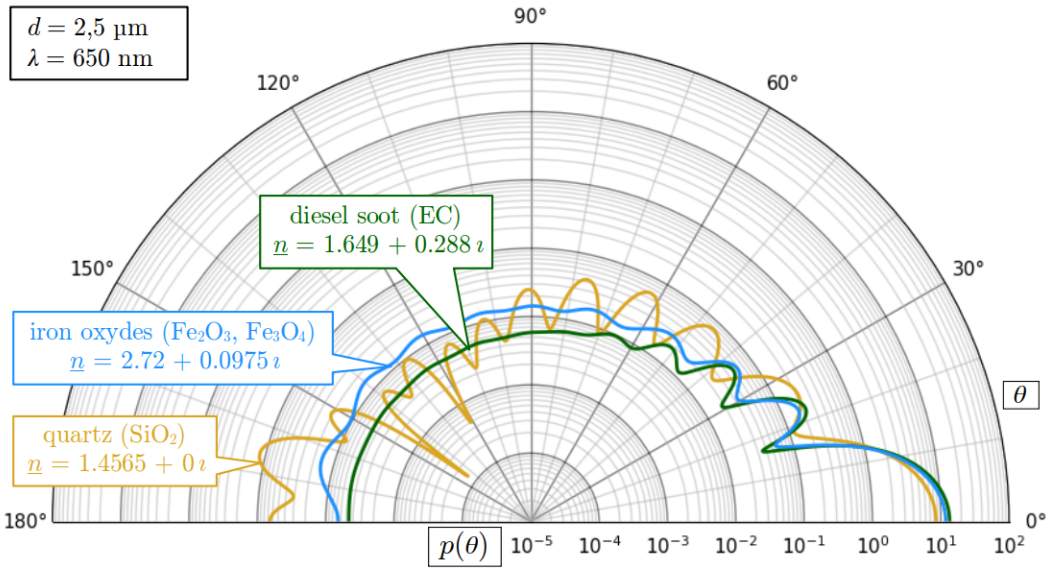


FIGURE 10 – Courbes représentatives de la fonction de phase de trois matériaux

Q11 En termes de proportions relatives, sans distinction de matériaux, commenter brièvement l'allure globale des courbes de la figure 10 selon la valeur de l'angle polaire θ.

■ Dans le programme donné en page 11, à l'appel de la fonction `S1_S2`, la valeur de l'argument effectif du facteur de normalisation codée `'one'` modifie le calcul des valeurs de retour `S1` et `S2` de telle sorte que la somme intégrale de la fonction de phase p soit unitaire sur l'angle solide plein 4π sr, c'est-à-dire :

$$\int_{4\pi\ \mathrm{sr}} p(\theta)\ \mathrm{d}\Omega \ = \ 1$$

La fonction de phase p peut alors être interprétée comme celle d'une *densité de probabilité* d'émission de photons dans une direction (θ, φ) de l'espace, sachant que ses valeurs sont indépendantes de l'angle azimutal φ. Ensuite, on peut quantifier l'**éclairement énergétique** $E_{\mathrm{e\ sca}}$ de l'onde diffusée dans un angle solide donné $\Omega = [\theta_1; \theta_2] \times [\varphi_1; \varphi_2]$ par la relation de proportionnalité [6] :

$$E_{\mathrm{e\ sca}}(\Omega) \ \propto \ \sigma_{\mathrm{sca}} \int_{\varphi_1}^{\varphi_2} \int_{\theta_1}^{\theta_2} p(\theta) \sin\theta \,\mathrm{d}\theta \,\mathrm{d}\varphi \ = \ \sigma_{\mathrm{sca}} \cdot (\varphi_2 - \varphi_1) \int_{\theta_1}^{\theta_2} p(\theta) \sin\theta \,\mathrm{d}\theta \qquad (4)$$

La sonde *NextPM* présentée à la page 6 est équipée d'une photodiode principale dont la surface sensible peut être modélisée par un carré d'aire $a = 7{,}5\,\mathrm{mm}^2$. Dans le référentiel polaire (O, r, θ, φ) de la chambre de diffusion, elle est positionnée à $r = 4{,}5\,\mathrm{mm}, \theta = 90°$ et $\varphi = 0°$, le centre O étant considéré comme la position moyenne des particules éclairées par l'onde incidente.

Q12 En illustrant le calcul par une figure plane, déterminer au degré près les valeurs des bornes du cône $[\theta_1\,;\,\theta_2] \times [\varphi_1\,;\,\varphi_2]$ centré sur la direction $(90°, 0°)$ formant l'angle solide Ω dans lequel la photodiode principale « voit » la lumière diffusée par une particule éclairée en O.

6. Le symbole $\propto$ se lit : « est proportionnel à ».

Tournez la page S.V.P.

Q13 En considérant `delta_deg` comme une donnée déjà déclarée représentant la mesure en degrés du demi-angle au sommet du cône pyramidal constituant Ω, compléter le code des lignes du programme ci-dessous (champs marqués « `???` ») pour déterminer par des expressions les valeurs des variables représentant ses bornes (θ_1, θ_2) ainsi que la différence $\varphi_2 - \varphi_1$. Quant à la liste `theta_range` des valeurs de θ en radians, elle est à définir avec une résolution de 0,2°.

```
93  delta           = ???  # in rad
94  theta1, theta2  = ???  # also in rad
95  phi_width       = ???  # phi_2 - phi_1
96  theta_range     = ???  # [theta1; theta2] with a 0.2 deg resolution
97  mu_range        = ???  # u = cos(theta)
```

Et finalement, pour l'échantillon de matériaux codés dans le programme donné en p. 10, les lignes de code (incomplètes) ci-dessous ont pour but d'afficher, en échelles logarithmiques (abscisses et ordonnées), les courbes théoriques de **mesurande** de l'OPC, c'est-à-dire de l'éclairement énergétique $E_{\mathrm{e\,sca}}(\Omega)$ diffusé dans l'angle solide Ω de la photodiode principale par les particules en fonction de leur diamètre $d \in [0{,}1\,;\ 10]\,\mu\mathrm{m}$, et ceci avec une résolution de 0,002 par décade.

```
100 fig2, ax2 = plt.subplots()
101 d_range   = ???                 # diameters from 0.1 to 10 um (in nm)
102 for material in materials:
103   mesurand_range = []           # Initialize results array
104   for d in d_range:
105     x = ???                     # size parameter   (dimensionless)
106     s = ???                     # particle cross section (in um^2)
107     qext, qsca, _, _    = ???   # efficiencies
108     s1_range, s2_range = ???    # scattering amplitude coefficients
109     # phase function weighted before integration over theta and phi
110     wp_range     = ???
111     # phase function integrated, scalled by Ssca
112     wp_integral = ???
113     mesurand_range.append(wp_integral)
114   ax2.plot(???,                 # diameters in um
115            ???,                 # mesurands
116            color=material['color'],
117            linewidth=2)
118 # ...
```

Q14 Compte tenu des spécifications qui viennent d'être décrites, en exploitant les éléments de code pertinents et les résultats des questions précédentes, compléter toutes les lignes de code du programme ci-dessus comportant un champ marqué « `???` ».

Correctement paramétré et complété, le programme ci-dessus produit le diagramme représenté sur la figure 12 en page 14.

Q15 Dans une perspective de modélisation pour exploitation opérationnelle dans le micrologiciel embarqué d'un OPC, commenter les courbes de la figure 12, notamment en termes de linéarité, de monotonie et de positions relatives. Conclure.

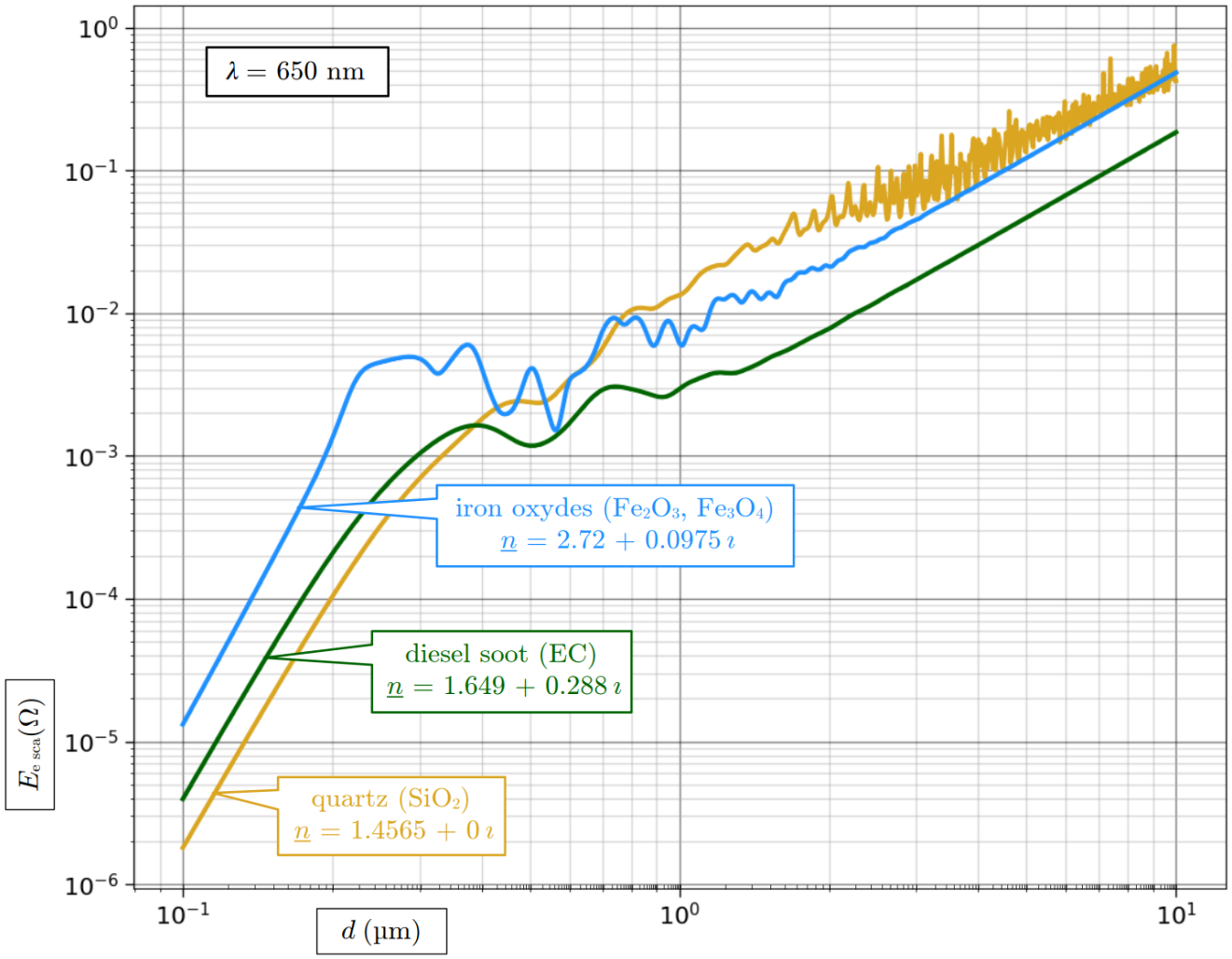


FIGURE 12 – Courbes théoriques de mesurande pour trois matériaux

1.3 Principe de mesure d'un OPC

■ Dans la pratique, le système de mesure d'un OPC repose sur une partition de la pleine échelle des diamètres d des particules en une série de n intervalles D_n, qu'on appelle des **canaux** (en anglais, *channels* ou *bins*). Ces canaux sont mis en correspondance biunivoque (autrement dit, en bijection) avec une partition en n intervalles de l'excursion de la tension U lue aux bornes de la photodiode (après amplification). La correspondance est établie lors d'une phase d'**étalonnage** en soumettant l'OPC à des *aérosols monodispersés* (c'est-à-dire contenant des particules sphériques de diamètre calibré – typiquement du latex de polystyrène ou du verre sodocalcique).

La tension U étant une fonction monotone du mesurande $E_{e\,sca}(\Omega)$, on peut raisonner en théorie sur l'excursion de cette grandeur abstraite pour évaluer les potentielles erreurs de mesure théorique de l'appareil. Ainsi, le tableau ci-dessous propose une correspondance biunivoque des 5 canaux de la sonde *NextPM* avec l'excursion établie par la simulation précédente (cf. la figure 12).

Une fois l'étalonnage effectué, le principe de mesure de l'OPC est le suivant : sur une période donnée (typiquement, 10 secondes), il compte les pics de tension aux bornes de la photodiode en incrémentant, pour chaque pic, un compteur c_i associé au canal de mesure D_i correspondant à l'amplitude mesurée du pic. Le débit d'air traversant la chambre de diffusion étant régulé et connu, les valeurs de concentration massique en particules fines sont établies comme ci-dessous.

Tournez la page S.V.P.

d (µm)	$D_1 = [0{,}3\,;\ 0{,}5[$	$D_2 = [0{,}5\,;\ 1[$	$D_3 = [1\,;\ 2{,}5[$
$E_{\text{e sca}}(\Omega)$	$[7 \times 10^{-4}\,;\ 3 \times 10^{-3}[$	$[3 \times 10^{-3}\,;\ 1 \times 10^{-2}[$	$[1 \times 10^{-2}\,;\ 4 \times 10^{-2}[$
d (µm)	$D_4 = [2{,}5\,;\ 5[$	$D_5 = [5\,;\ 10]$	
$E_{\text{e sca}}(\Omega)$	$[4 \times 10^{-2}\,;\ 2 \times 10^{-1}[$	$[2 \times 10^{-1}\,;\ 7 \times 10^{-1}]$	

TABLE 2 – Correspondance avec $E_{\text{e sca}}(\Omega)$ des canaux de mesure de la sonde *NextPM*

– Dans chaque intervale D_i, on calcule le volume V_i des particules comptées par c_i en les considérant toutes sphériques [7] de même diamètre aérodynamique égal à la médiane (ou milieu) d_i des bornes de l'intervalle.

– Puis on somme dans un total V les volumes V_i pertinents pour chaque catégorie normalisée de particules fines (par exemple, $V = V_1 + V_2 + V_3$ pour les $PM_{2.5}$ – cf. p. 3).

– Enfin, on multiplie ce volume total V par une valeur de masse volumique considérée comme moyenne dans l'environnement [8] avec une mise à l'échelle en fonction des unités employées.

Q16 En exploitant la figure 12 et les explications ci-dessus, pour la valeur particulière du mesurande $E_{\text{e sca}}(\Omega) = 2 \times 10^{-2}$, déterminer en cm^3 : **a)** le volume V_c qui serait calculé par l'OPC, **b)** le volume théorique V_{th1} s'il s'agissait d'une particule sphérique d'oxydes de fer, **c)** le volume théorique V_{th2} s'il s'agissait d'une particule sphérique de suie de diesel.

■ Pour déterminer la concentration massique en particules fines – a priori de toutes formes et potentiellement poreuses – à partir d'un volume calculé sur la base d'enveloppes sphériques, la **masse volumique apparente** [15] (en anglais, *bulk density*) est une grandeur répertoriée qui fournit une approximation convenable (en tout cas meilleure que la masse volumique absolue), puisqu'elle inclut un volume d'air interstitiel dans le calcul. Elle peut être estimée expérimentalement par la masse d'échantillons prélevés sur des filtres, rapportée à leur volume enveloppe sans tassement, donc incluant un volume d'espaces interstitiels. Dans la pratique, la valeur $\rho_m = 1{,}65\,\text{g}\,\text{cm}^{-3}$ est usuellement retenue comme moyenne de celle des particules d'aérosols en environnement urbain [16]. Par ailleurs, on trouve dans la littérature scientifique les estimations $\rho_1 = 2{,}2\,\text{g}\,\text{cm}^{-3}$ et $\rho_2 = 1{,}2\,\text{g}\,\text{cm}^{-3}$ respectivement pour les particules d'usure de systèmes de freinage (oxydes de fer) [17] et de suie de diesel [18].

Q17 À partir des valeurs des volumes V_c, V_{th1} et V_{th2} déterminées à la question précédente, calculer respectivement en µg les masses m_c, m_{th1} et m_{th2} de particules correspondantes. Pour les deux dernières, en déduire à chaque fois l'erreur relative théorique de mesure de l'OPC.

Q18 Conclure quant à l'aptitude de l'OPC à fournir des valeurs de concentration massique réalistes sur la base du principe de mesure qui est exposé dans cette étude. Proposer éventuellement des solutions qui permettraient d'améliorer la précision de l'appareil.

7. On rappelle la formule $V = \frac{1}{6}\pi d^3$ du volume d'une sphère de diamètre d.
8. En effet, l'appareil n'a aucun moyen de connaître la nature physico-chimique des particules qu'il détecte.

2 – Calibrage de capteurs électrochimiques de NO_2

Dans des environnements urbains denses, il est nécessaire pour suivre la qualité de l'air d'avoir un maillage étroit de stations de collecte de mesures. Cependant, pour des raisons économiques, ces infrastructures sont dans la pratique assez dispersées. Les nouvelles technologies de capteurs à faible coût offrent la possibilité d'étendre considérablement le réseau de surveillance officiel et d'améliorer l'analyse et la modélisation de la pollution atmosphérique urbaine au niveau local. Des capteurs miniaturisés et abordables permettent potentiellement au grand public de mesurer leur environnement direct dans l'espace et le temps. Un exemple de système abordable, développé par la société *Waag* est présenté en figure 14 a).

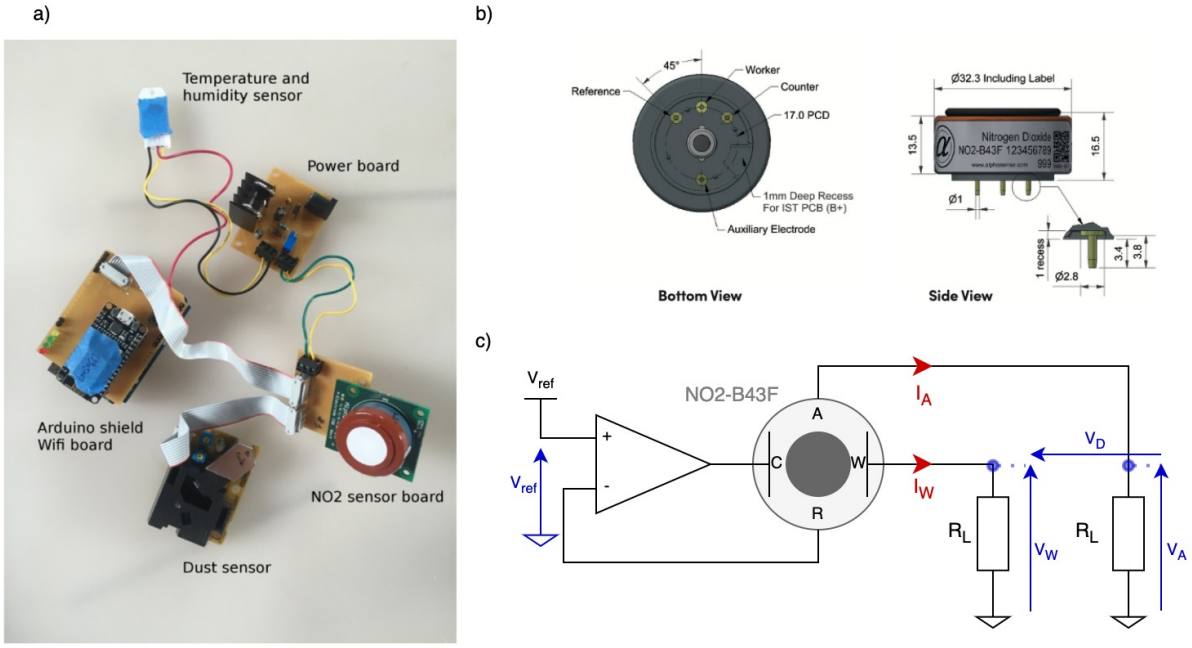


FIGURE 14 – a) Système embarqué de mesure de qualité de l'air développé par la société *Waag* b) Vues de dessus et latérale du capteur NO2-B43F c) Schéma de principe d'utilisation du capteur NO2-B43F

2.1 Principe du capteur et récupération des données

Ce système possède en particulier un capteur de NO_2, le NO2-B43F, qui est étudié dans cette partie. Ce capteur à faible coût (voir figure 14 b) repose sur l'utilisation de deux cellules électrochimiques similaires dont l'une seulement est soumise à l'air ambiant. Le composant comporte quatre électrodes (voir figure 14 c) :

- les électrodes de référence (R) et la contre-électrode (C). Ces deux électrodes sont nécessaires pour polariser les jonctions électro-chimiques et une différence de potentiels contrôlée et égale à une tension nommée V_{ref} y est appliquée,
- l'électrode de travail (W pour *working electrode*) est située dans la cellule électrochimique en contact avec le gaz mesuré,
- l'électrode auxiliaire (A) est située dans la cellule électrochimique qui n'est pas en contact avec le gaz mesuré mais dans un environnement de contrôle (la concentration en NO_2 à son contact peut être considérée nulle en première hypothèse).

Les deux dernières électrodes ont un courant proportionnel à la concentration en NO_2 à travers

Tournez la page S.V.P.

une constante nommée S qui est la sensibilité du capteur. La concentration est mesurée par différence entre l'électrode de travail et l'électrode auxiliaire. Ce type de capteur a deux principaux inconvénients :

1. il est nécessaire de le calibrer en utilisant une mesure de référence,
2. il est potentiellement sensible mais dans une moindre mesure à d'autres quantités physico-chimiques.

La documentation technique du capteur NO2-B43F est donnée en document annexe A.4.

Q19 À l'aide de la documentation constructeur, donner l'ordre de grandeur ainsi que l'unité de la sensibilité du capteur. Que peut-on déduire de la mesure réalisée et de sa reproductibilité ?

Les courants sur les électrodes auxiliaires (A) et de travail (W) peuvent se décomposer en une valeur constante d'*offset* (annotée 0) et une partie variable dans le temps (annotée E). On peut donc noter :

$$\begin{cases} I_A = I_{A0} + I_{AE} \\ I_W = I_{W0} + I_{WE} \end{cases}$$

Cette même notation (*offset* plus partie variable) peut être reprise pour d'autres quantités.

Q20 À partir du schéma présenté en figure 14 c, montrer que la concentration en NO_2 est une fonction linéaire de la différence de potentiel entre les électrodes notée V_D. Expliciter le coefficient directeur et l'ordonnée à l'origine.

En pratique, les deux électrodes de travail (W) et auxiliaires (A) ne sont pas rigoureusement identiques et ont chacune leur propre sensibilité notées respectivement S_W et S_A.

Q21 En déduire le lien entre la concentration en NO_2 et les tensions des électrodes W et A. Quelle est la conséquence pour le calibrage de la mesure ?

Le système étudié a été utilisé, en parallèle d'instruments plus onéreux de stations de mesure classique (nommé également *airboxes* par la suite), sur une campagne de mesures ayant impliqué le grand public (projet *Urban AirQ*) en 2016 à Amsterdam. La campagne s'est déroulée en deux phases :

– une phase de calibrage a permis de récolter des données utilisées pour identifier les modèles des capteurs à bas cout. Pour ce faire l'ensemble des systèmes ont été placés à proximité immédiate d'une station de mesure classique. Ce phase a été répétée deux fois : du 2 au 10 juin 2016 puis du 18 au 29 août 2016 ;
– une phase d'utilisation, pendant laquelle des particuliers se sont vu confier un système faible cout, déployé librement, et dont les emplacements ont été enregistrés (latitude/longitude). Cette phase s'est déroulée entre le 11 juin et le 17 août 2016.

La figure 15 a) montre les positions des capteurs pendant la phase d'utilisation, la figure 15 b) montre le placement des système faible cout autour de la station de mesure classique de référence pour le calibrage.

L'ensemble des données de la campagne, des systèmes *Waag* et de la station classique, sont en accès libre (partagés sur le *repository* de *Waag*). Le script *Python* suivant permet de charger les

a) b)

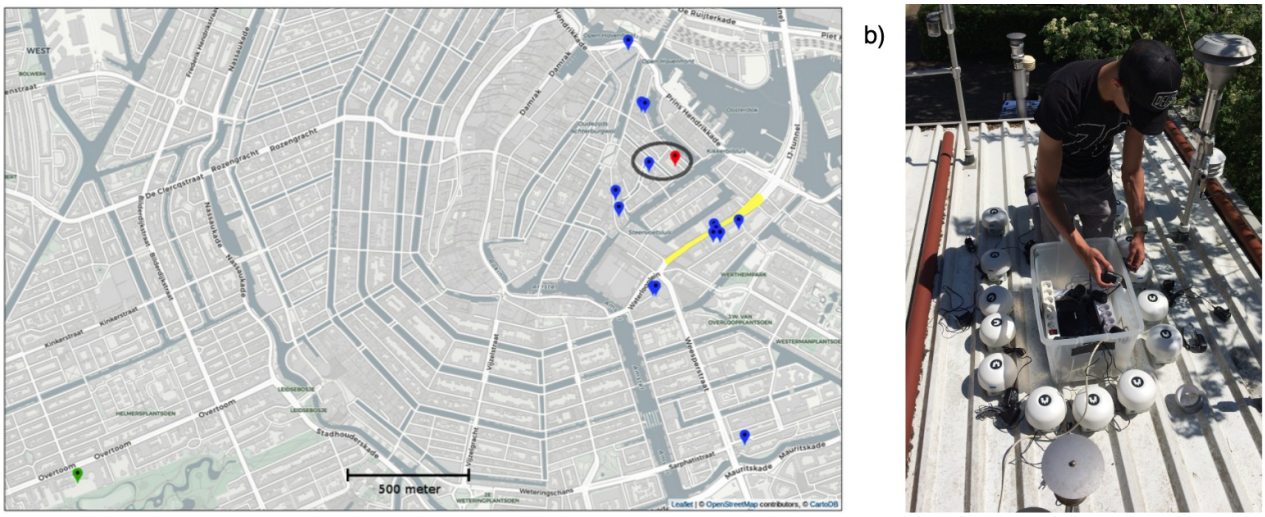


FIGURE 15 – a) Position des capteurs pendant la phase de mesure b) Installation des systèmes à bas coût à la station de mesure pour la calibrage.

données récoltées :

```python
import pandas as pd

file_waag = '../making-sensor-master/data/sensormeasures.csv.zip'
file_station = '../making-sensor-master/data/AMS_hourly_airboxdata_total.csv'


if __name__ == "__main__":
    ##########################
    ## Sensor data loading ##
    ##########################
    print("Data collected with sensors")
    data_waag = pd.read_csv(file_waag, compression='zip')
    print(data_waag.head())

    ##########################
    ## Station data loading ##
    ##########################
    print("Data collected with Airboxes")
    data_station = pd.read_csv(file_station)
    print(data_station.head())
```

Des aides pour les API des modules `pandas` et `matplotlib` sont données en annexes A.3, ainsi que les informations fournies par la société *Waag* sur les données enregistrées. L'exécution de ce script renvoie le prompt suivant :

Q22 En utilisant la sortie de code donnée en figure 16, évaluer l'ordre de grandeur de l'échantillonage des systèmes à bas coût et de la station utilisée pour la phase de calibrage. Proposer deux solutions possibles pour faire coïncider temporellement les données.

Tournez la page S.V.P.

```
Data collected with sensors
(1593993, 11)
        id                      srv_ts                topic  rssi  temp   pm10    pm25    no2a    no2b  humidity          message
0    55303   2016-06-25 21:14:06.58+02    sensor/55303/data -90.0  18.6   0.62    0.00  1231.0  1362.0      68.2              NaN
1    54911  2016-06-25 21:14:10.966+02    sensor/54911/data -79.0  19.7  73.45  348.10  1249.0  1191.0      62.7              NaN
2    55300  2016-06-25 21:14:15.572+02    sensor/55300/data   NaN   NaN    NaN     NaN     NaN     NaN       NaN  Sensor startup
3    54200  2016-06-25 21:14:15.849+02    sensor/54200/data -89.0  20.2  97.81  299.30  1245.0  1209.0      64.8              NaN
4  1184453  2016-06-25 21:14:21.106+02  sensor/1184453/data -64.0  19.4   0.62  356.97  1251.0  1254.0      67.8              NaN
Data collected with Airboxes
(2235, 9)
             Time  AB46_NO2  AB46_PM1  AB46_PM2.5  AB46_PM10  AB47_NO2  AB47_PM1  AB47_PM2.5  AB47_PM10
0  06-06-16 22:00   13.6694    4.6189      8.8686    22.6504       NaN       NaN         NaN        NaN
1  06-06-16 23:00   17.0299    4.9692      9.1241    22.4980       NaN       NaN         NaN        NaN
2  07-06-16 00:00   18.0006    5.4216      9.8942    25.0329   0.67579    5.5812     11.1193    30.4342
3  07-06-16 01:00   12.0413    5.5281     10.4413    28.1565   2.96940    5.7964     12.0206    34.1666
4  07-06-16 02:00   12.7530    5.7873     10.5704    26.8454   5.22820    5.9380     11.6046    34.8098
```

FIGURE 16 – Sortie standard du terminal après exécution du script précédent

Q23 Proposer le code de la fonction `extract_sensor_data` dont le prototype est donné ci-dessous et qui renvoie un nouveau *DataFrame* qui :

- contient les données correspondant au numéro de capteur choisi,
- contient une nouvelle colonne avec les dates converties en '*pd.TimeStamp*'
- ne contient plus les données des colonnes de l'identifiant du capteur, du format de date initial, du topic, du rssi et du message.

```python
def extract_sensor_data(df, sensor_No):
    """
    Load sensor data from the main dataframe.

    Parameters:
    df : DataFrame
        The main DataFrame containing sensor data.
    sensor_No : int
        The sensor number to filter the data.
    Returns:
    DataFrame
        Filtered and processed sensor data with a 'TimeStamp' column.
    """
```

L'alignement temporel des données entre les relevés des capteurs et de la station nécessite d'avantages d'opérations sur les *DataFrames* et requiert une certaine maîtrise de l'API *pandas*. Une intelligence artificielle générative propose le code suivant pour accomplir cette tâche :

```python
def align_data(sensor_data, station_data, cal_start, cal_end):
    """
    Align sensor data for calibration.
    """
    # bloc de code n1
    sensor_data_cal = sensor_data.query('TimeStamp == @cal_start or TimeStamp
        == @cal_end')

    # bloc de code n2
    station_data.insert(0, 'TimeStamp', pd.to_datetime(station_data['Time'],
        format='mixed', dayfirst=True).map(lambda x: x.tz_localize(tz='Europe/
        Amsterdam')))
    station_data = station_data.sort_values(by='TimeStamp')
    station_data = station_data.drop(columns=['Time'])
    station_data_cal = station_data.query('TimeStamp == @cal_start or TimeStamp
        == @cal_end')
```

```
13
14      # bloc de code n3
15      sensor_data_cal['TimeStamp'] = sensor_data_cal['TimeStamp'].dt.round('h')
16      sensor_data_cal['TimeStamp'] = sensor_data_cal['TimeStamp'].dt.tz_convert('
            Europe/Amsterdam')
17      sensor_averaged_data_cal = sensor_data_cal.groupby('TimeStamp').mean()
18
19      # bloc de code n4
20      aligned_data = pd.merge_asof(station_data_cal, sensor_averaged_data_cal, on
            ='TimeStamp')
21
22      return aligned_data
```

Q24 Spécifier ce que réalise le code en détaillant les opérations réalisées par chaque bloc de code. Cette proposition est elle adaptée ? Dans le cas contraire, proposer les corrections nécessaires permettant d'utiliser cette fonction.

2.2 Modélisation linéaire et modélisation multivariée du capteur

On souhaite dans un premier temps prendre un modèle en première approximation du capteur donné par une équation affine (comme obtenu en question 20) :

$$\tilde{\mathrm{Y}} = a\mathrm{X} + b$$

où $\tilde{\mathrm{Y}} = (\tilde{y}_1, \cdots \tilde{y}_N)^T \in \mathbb{R}^N$ est l'estimation de $\mathrm{Y} = (y_1, \cdots y_N)^T \in \mathbb{R}^N$ la concentration et $\mathrm{X} = (x_1, \cdots x_N)^T \in \mathbb{R}^N$ la mesure de tension différentielle provenant du capteur, où $N \in \mathbb{N}^*$ est le nombre de mesures. Il s'agit donc pour modéliser le capteur d'obtenir le couple de valeurs $(a, b) \in \mathbb{R}^2$. Une solution simple consiste à faire une régression linéaire par moindres carrés : on minimise l'erreur quadratique entre les valeurs vraies y (obtenue dans la station) et la valeur estimée $\tilde{y}$. La quantité à minimiser est :

$$Q = \sum_{i=1}^{N} (y_i - \tilde{y}_i)^2$$

où Q, le résidu, dépend du couple de valeurs (a, b) recherché. Un minimum peut être atteint en annulant ses dérivées partielles, soit :

$$\begin{cases} \dfrac{\partial Q}{\partial a} = 0 \\ \dfrac{\partial Q}{\partial b} = 0 \end{cases}$$

Q25 Montrer que ce système a pour solution :

$$\begin{cases} a = \dfrac{\sum_{i=1}^{N} \left(x_i - \overline{\mathrm{X}}\right)\left(y_k - \overline{\mathrm{Y}}\right)}{\sum_{i=1}^{N} \left(x_k - \overline{\mathrm{X}}\right)^2} \\ b = \overline{\mathrm{Y}} - a\overline{\mathrm{X}} \end{cases}$$

où l'opérateur $\bar{\cdot}$ est la moyenne des éléments.

Le nombre de valeurs étant assez important, on souhaite avoir une implémentation efficace de la régression linéaire. Pour ce faire on dispose des fonctions `mean` et `var` et `cov` de la bibliothèque

Tournez la page S.V.P.

`numpy` (dont les aides sont données en annexe A.3).

Q26 En justifiant le raisonnement, compléter le code proposé ci dessous pour calculer le couple de valeur (a, b) en lieu et place des points d'interrogations sur les lignes 11 et 12.

```
1  import numpy as np
2
3  def regression_lineaire(X, Y):
4      """
5      Calculate the slope and intercept of the linear regression line
6      for the given data points X and Y.
7      """
8      X = np.asarray(X)
9      Y = np.asarray(Y)
10
11     a = ???
12     b = ???
13     R2 = 1 - np.sum((Y - (a * X + b)) ** 2) / np.sum((Y - np.mean(Y)) ** 2)
14
15     return a, b, R2
```

Q27 À partir du code donné dans la question précédente, expliciter mathématiquement le terme R^2 renvoyé par la fonction. Vers quelle valeur doit tendre ce terme dans l'idéal, c'est à dire si le modèle permet d'estimer correctement les valeurs vraies ?

Q28 En supposant les fonctions demandées justes, proposer une structure de script python permettant :

- d'isoler les données du capteur 55303 sur la durée de la première phase de calibrage,
- d'isoler les données de la station sur la durée de la première phase de calibrage,
- d'uniformiser les données capteur et de la station,
- d'identifier le modèle univarié (fonction linéaire) du capteur.

L'exécution du code demandé permet d'obtenir le retour présenté en figure 17.

```
Slope (a): -0.31905627197832465, Intercept (b): -5.609677177178369, R^2: 0.05880959169465405
```

FIGURE 17 – retour terminal de la regression linéaire

Q29 Commenter les résultats obtenus en figure 17 quant à l'adéquation du modèle aux données.

La régression linéaire mise en place ne fonctionne que si les données en entrée sont de dimension 1 (un scalaire mesuré N fois). Pour des données à M dimensions (avec $M \in \mathbb{N}^* \setminus \{1\}$ le nombre de valeurs explicatives, soit M scalaires mesurés simultanément N fois), on parle de régression linéaire multiple ou *multilinear regression* en anglais. On peut écrire le modèle sous la forme :

$$\tilde{y}_i = \beta_0 + \sum_{j=1}^{M} \beta_j x_{ij}$$

Trouver le modèle revient donc à identifier les valeurs numériques du vecteur $\beta = (\beta_0, \beta_1, \cdots \beta_M)^T \in$

$\mathbb{R}^{M+1}$. Afin de simplifier les calculs et l'écriture du code, on pose la notation matricielle suivante :

$$\tilde{\mathrm{Y}} = \mathrm{Z}\beta$$

Q30 Expliciter la matrice Z d'entrée du modèle et en préciser la taille.

On peut remarquer que le terme de résidu peut s'écrire :

$$Q = \left(\mathrm{Y} - \tilde{\mathrm{Y}}\right)^T \left(\mathrm{Y} - \tilde{\mathrm{Y}}\right)$$

Q31 Par un raisonnement similaire à celui effectué pour la régression linéaire simple, montrer que Q admet un minimum pour $\beta = \left(\mathrm{Z}^T\mathrm{Z}\right)^{-1}\mathrm{Z}^T\mathrm{Y}$

Q32 En utilisant les fonctions de calcul matriciel de la librairie `numpy`, proposer le code de la fonction suivante renvoyant le vecteur β et la valeur de R^2

```
def multilinear_regression(Y, X):
    """
    Perform multilinear regression

    Parameters:
    Y : array-like
        Dependent variable.
    X : array-like
        Independent variables, should be a 2D array with shape (n_samples,
            n_features).

    Returns:
    arraylike
        Coefficients of the regression model, including the intercept
    float
        R-squared value of the model.
    """
```

À partir des données collectées et de modèles multilinéaires multiples uniquement, les résultats présentés en figure 18 ont été obtenus.

Q33 À partir des résultats obtenus, préciser pour les modèles A et B le nombre de variables d'observations et les colonnes du *Dataframe* utilisées, et analyser de manière quantitative l'adéquation par rapport au capteur utilisé.

On s'intéresse maintenant à l'implémentation dans le dispositif embarqué du modèle multivarié le plus fiable. Le programme *Arduino* utilisant le capteur et un convertisseur analogique-numérique 16 bits (ADS1115) pour mesurer les tensions des électrodes du capteur de NO2-B4F3, ainsi qu'un capteur de température/humidité (DHT11), est donné en annexe A.5.

Q34 Proposer le code d'une fonction `calculate_ NO2_ concentration` (appelée à la ligne 108) qui implémente le modèle multivarié, et préciser les unités de chaque élément utilisé dans la structure `ModelParameters`.

Tournez la page S.V.P.

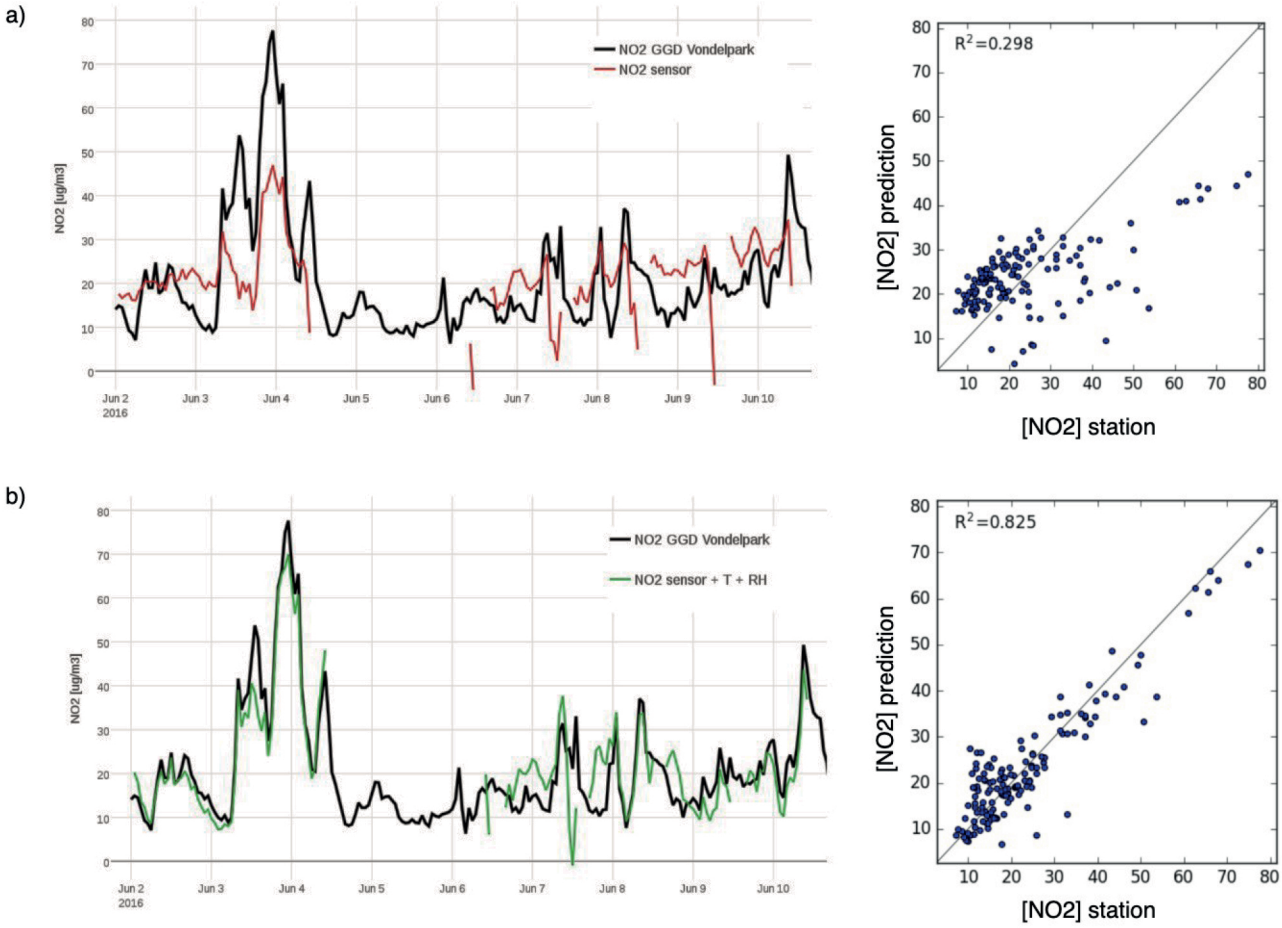


FIGURE 18 – Résultats au cours de la phase de calibrage de suivi en station et de prédiction à partir de capteurs à faible coût de la concentration de NO_2 pour : a) le modèle A, b) le modèle B.

Q35 Quelle est la périodicité de l'appel à la fonction `calculate_NO2_concentration` ? Proposer une stratégie permettant d'avoir un flux de données cohérent par rapport aux données du projet *Urban AirQ*, et en préciser son éventuel intérêt.

3 – Cartographie du taux de particules fines dans une zone

Le but de cette partie est d'étudier et implanter plusieurs méthodes pour cartographier le taux de particules fines $PM_{2.5}$ dans une zone délimitée plane définie comme un rectangle de $X_{\max}$ mètres de longueur, et $Y_{\max}$ mètres de hauteur. Les points sont définis de manière classique par des couples $(x, y) \in \mathbb{N}$ avec $x \in \{0, \ldots, X_{\max}\}$ et $y \in \{0, \ldots, Y_{\max}\}$. Cependant, pour faciliter le lien avec l'écriture matricielle, l'origine $(0, 0)$ est en haut à gauche et l'axe des y orienté vers le bas. La figure 19 représente la zone considérée pour $X_{\max} = 6$, $Y_{\max} = 5$. On a représenté les 3 points $s_1 = (1, 4)$, $s_2 = (4, 3)$ et $s_3 = (2, 1)$.

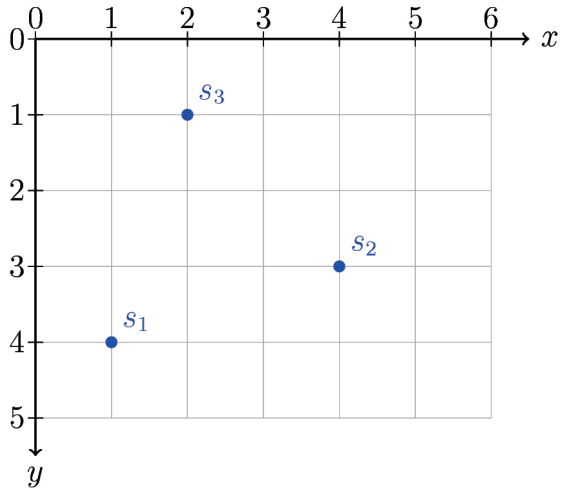


FIGURE 19 – Zone du plan considérée pour $X_{\max} = 6$ et $Y_{\max} = 5$.

La matrice $\mathcal{M}$ associée en *Python* qui permet de représenter cette zone (en supposant que l'on ne peut avoir qu'un point pour un couple de coordonnées) est constituée de $Y_{\max}+1$ listes de $X_{\max}+1$ éléments. Pour l'exemple, $\mathcal{M}$ est une liste de $Y_{\max}+1 = 6$ listes de $X_{\max}+1 = 7$ éléments définie par :

$$\begin{bmatrix} [\cdot, & \cdot, & \cdot, & \cdot, & \cdot, & \cdot, & \cdot], \\ [\cdot, & \cdot, & s_3, & \cdot, & \cdot, & \cdot, & \cdot], \\ [\cdot, & \cdot, & \cdot, & \cdot, & \cdot, & \cdot, & \cdot], \\ [\cdot, & \cdot, & \cdot, & \cdot, & s_2, & \cdot, & \cdot], \\ [\cdot, & s_1, & \cdot, & \cdot, & \cdot, & \cdot, & \cdot], \\ [\cdot, & \cdot, & \cdot, & \cdot, & \cdot, & \cdot, & \cdot] \end{bmatrix}$$

On observe que le sommet de coordonnées (x, y) est accessible par la commande $\mathcal{M}[y][x]$.

Plusieurs capteurs ont été placés dans cette zone. Chaque capteur possède des coordonnées entières dans la zone considérée, et la valeur mesurée du taux de particules fines $PM_{2.5}$. Le problème consiste à évaluer pour tout couple $(x, y) \in \{0, \ldots, X_{\max}\} \times \{0, \ldots, Y_{\max}\}$, un taux de particules fines $PM_{2.5}$ de manière réaliste et de le représenter par une zone de couleur.

3.1 Interpolation par pondération inverse de la distance

L'interpolation par pondération inverse de la distance (IDW en anglais pour *inverse distance weighting*) est une méthode d'interpolation spatiale utilisée pour estimer une valeur à un point inconnu à partir de valeurs connues à des points voisins. L'idée principale de cette méthode est que les points proches ont plus d'influence que les points éloignés. Autrement dit, la valeur d'un point inconnu est une moyenne pondérée des valeurs connues, où les poids dépendent de la distance.

Concrètement, supposons que l'on dispose d'un ensemble $\mathcal{C}$ de N capteurs numérotés de 1 à N. Pour tout $i \in \{1, \ldots, N\}$, le capteur $c_i = (x_i, y_i, m_i)$ où $(x_i, y_i) \in \{0, \ldots, X_{\max}\} \times \{0, \ldots, Y_{\max}\}$

Tournez la page S.V.P.

sont les coordonnées du capteur c_i et $m_i \in \mathbb{R}$ le taux de particules fines $PM_{2.5}$ qu'il a mesuré.

La notation $d(s, s')$ désigne la distance euclidienne entre les deux sommets s et s' : si leurs coordonnées dans le plan sont respectivement (x, y) et (x', y'), $d(s, s') = \sqrt{(x - x')^2 + (y - y')^2}$.

Pour tout sommet $s = (x, y) \in \{0, \ldots, X_{\max}\} \times \{0, \ldots, Y_{\max}\}$, on peut obtenir son taux de particules fines $PM_{2.5}$:

- si s possède un capteur c_i, alors $\hat{v}(s) = m_i$,
- si s ne possède pas de capteur, par estimation avec la formule :

$$\hat{v}(s) = \frac{\sum_{i=1}^{N} w_i(s, c_i) \times m_i}{\sum_{i=1}^{N} w_i(s, c_i)} \quad \text{où} \quad w_i(s, c_i) = \frac{1}{d(s, c_i)^2}$$

Q36 Pour cette question, on suppose que l'on a 3 capteurs $c_1 = (1, 4, 20)$, $c_2 = (4, 3, 40)$ et $c_3 = (2, 1, 50)$. Que vaut $\hat{v}(s)$ pour $s = (2, 3)$? Préciser les calculs intermédiaires.

Par la suite, on considère la déclaration *Python* suivante de la classe `Capteur` :

```
import math
class Capteur:
  def __init__(self, x, y, mesure=None):
      self.x = x
      self.y = y
      self.mesure = mesure

  @property
  def __repr__(self):
      return f"Capteur(x={self.x}, y={self.y}, mesure={self.mesure})"
```

De plus, les capteurs sont placés dans le dictionnaire `dico_capteurs` avec pour clef le couple d'entiers (x, y).

Q37 Donner le code de la méthode `distance_to(self, x, y)` qui retourne la distance euclidienne du capteur considéré avec le sommet s de coordonnées (x, y).

Q38 On considère un ensemble de capteurs stockés dans `dico_capteurs`.

1. Définir la fonction `Calcul_vchapeau(dico_capteur, x, y)` qui retourne $\hat{v}(s)$ pour un sommet $s \in \{(x, y) \in \{0, \ldots, X_{\max}\} \times \{0, \ldots, Y_{\max}\}\}$.
2. Compléter la fonction `IDW_1(dico_capteur, Xmax, Ymax)` qui retourne dans `grille` les valeurs $\hat{v}(s)$ pour $s \in \{(x, y) \in \{0, \ldots, X_{\max}\} \times \{0, \ldots, Y_{\max}\}\}$.

```
def IDW_1(dico_capteur, Xmax, Ymax):
  grille = [[0 for x in range(Xmax+1)] for y in range(Ymax+1)]

  # A COMPLETER

  return grille
```

3. Évaluer la complexité de ces deux fonctions. Justifier les réponses.

TABLE 3 – Couleurs en fonction des mesures m de $PM_{2.5}$, exprimées en $\mu g\,m^{-3}$

Intervalle	Commentaire	Couleur	Code RGB
$m = -1$	absence de mesure	blanc	$(255, 255, 255)$
$0 \leq m < 10$	minimum de l'OMS	bleu	$(0, 0, 255)$
$10 \leq m < 25$	minimum de directive européenne	vert	$(0, 255, 0)$
$25 \leq m < 50$	inférieur au pic de pollution de l'AASQA	rouge	$(255, 0, 0)$
$m \geq 50$	pic de pollution atteint et dépassé	noir	$(0, 0, 0)$

Q39 On souhaite dans cette question choisir la couleur d'un sommet s en fonction de la valeur de sa mesure m. Pour cela, on fixe les seuils dans la table 3 en fonction des valeurs proposées par l'OMS en 2021, la directive européenne 2008/50/CE et l'AASQA.
Définir la fonction `couleur(m)` qui renvoie le code RGB associé à la mesure m du taux de $PM_{2.5}$, selon l'intervalle auquel m appartient.

Q40 On souhaite visualiser les valeurs mesurées et estimées sous la forme d'une carte de couleurs. Pour cela, on crée une image `img` sous la forme d'une matrice de triplets de taille $(echelle \times Y_{\max}) \times (echelle \times X_{\max})$. Le paramètre *echelle* est un entier qui permet d'associer un carré de surface $(echelle)^2$ à chaque point dans le but d'agrandir l'image. L'image sera convertie en fichier `png` dont le nom est également passé en paramètre de la fonction. La figure 20 présente un exemple de figure obtenue.

1. Compléter la fonction `creation_image` dont le code partiel suit et qui effectue ce traitement.
2. Quelle est la complexité de cette fonction ? Justifier la réponse.

```
1  def creation_image(grille, Xmax, Ymax, echelle, nom):
2
3    img = np.zeros((echelle * Ymax, echelle * Xmax, 3), dtype=np.uint8)
4
5    for y in range(0, Ymax):
6      for x in range(0, Xmax):
7        # A COMPLETER
8    Image.fromarray(img).save(nom)
```

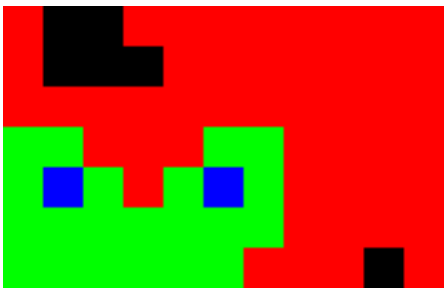


FIGURE 20 – Exemple de carte obtenue avec $X_{\max} = 10$, $Y_{\max} = 6$, et les capteurs $c_1 = (1, 4, 8)$, $c_2 = (4, 3, 40)$, $c_3 = (2, 1, 60)$, $c_4 = (5, 4, 5)$, $c_5 = (8, 2, 40)$, $c_6 = (9, 6, 50)$.

3.2 Étude d'une zone dense

Le nombre de capteurs est en général assez limité sur le territoire. Cependant, grâce à des méthodes basées sur la modélisation des transports urbains et de la dispersion des composants chimiques, il est possible d'obtenir, dans certaines zones, des estimations fiables des niveaux de pollution. Pour simplifier cette partie, on ne distingue pas les valeurs mesurées par un capteur de celles obtenues par modélisation : elles sont toutes considérées comme des valeurs issues de capteurs.

Tournez la page S.V.P.

On considère dans cette partie que le nombre de capteurs N est important, c'est-à-dire non négligeable par rapport à $X_{\max} \times Y_{\max}$. L'estimation du taux de particules fines d'un sommet sans capteur dépend surtout des capteurs proches ; l'idée est de limiter les capteurs considérés à ceux qui sont à une distance inférieure ou égale à R pour estimer ce taux.

Q41 1. Décrire un algorithme qui calcule la valeur minimale $R_{\min} \in \mathbb{R}$ telle que, pour tout $s = (x, y) \in \{0, \dots, X_{\max}\} \times \{0, \dots, Y_{\max}\}$ il existe au moins un capteur $c_i = (x_i, y_i, m_i)$ tel que $d(s, c_i) \leq R_{\min}$.

2. Quelle est la complexité de cet algorithme dans le pire des cas ? Justifier la réponse.

Par la suite, on suppose que $R \in \mathbb{N}$ avec $R \geq R_{\min}$. On rappelle que la zone étudiée est de longueur $X_{\max}$ et de hauteur $Y_{\max}$. On souhaite la découper en carrés de côté R. Le but est de limiter l'exploration pour accélérer le calcul de $\mathcal{C}(s) = \{c_i = (x_i, y_i, m_i) \in \mathcal{C}, d(s, c_i) \leq R\}$.

On découpe alors la zone considérée en carrés de côté R de la manière suivante :

- Soit $\bar{\imath} = \lceil \frac{Y_{\max}}{R} \rceil$ et la suite $y_i = i \times R$ pour $i \in \{0, \dots, \bar{\imath}\}$; soit également $\bar{\jmath} = \lceil \frac{X_{\max}}{R} \rceil$ et la suite $x_j = j \times R$ pour $j \in \{0, \dots, \bar{\jmath}\}$.
- Pour tout couple $(i, j) \in \{0, \dots, \bar{\imath}\} \times \{0, \dots, \bar{\jmath}\}$, $\mathcal{P}_{ij}$ désigne les couples $(x, y) \in \{0, \dots, X_{\max}\} \times \{0, \dots, Y_{\max}\}$ tels que $x_j \leq x \leq x_{j+1}$ et $y_i \leq y \leq y_{i+1}$.

Pour les applications numériques, on considère $X_{\max} = 9$ et $Y_{\max} = 7$. L'emplacement des capteurs est représenté par la figure 21.

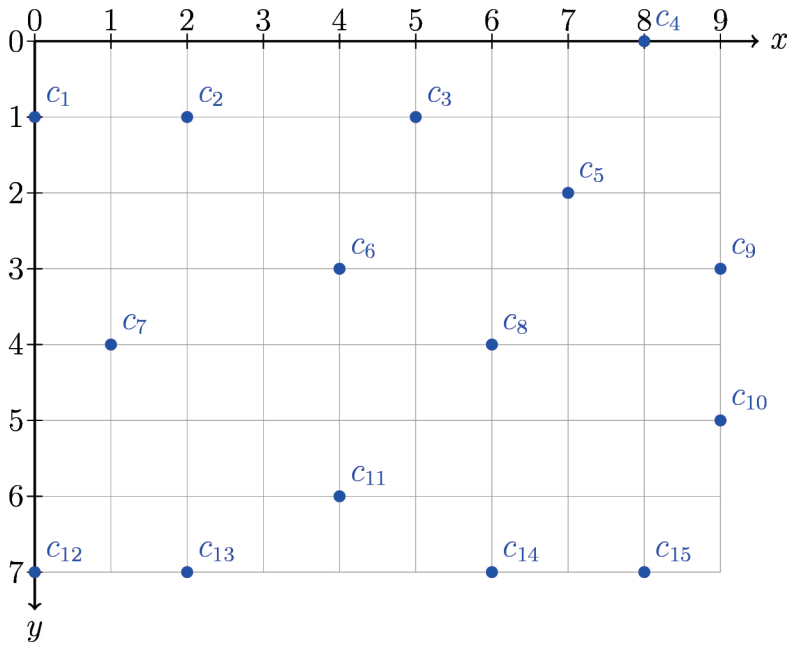


FIGURE 21 – Zone du plan considérée pour $X_{\max} = 9$ et $Y_{\max} = 7$. On a représenté la position d'un ensemble de capteurs $c_1, \dots, c_{15}$.

Q42 Pour cette question, on considère les capteurs dont les emplacements sont représentés par la figure 21.

1. Donner sans explication la valeur de $R_{\min}$. En déduire que $R = 2$, considérée par la suite, est une valeur réalisable.
2. Donner les valeurs $\bar{\jmath}$ et $\bar{\imath}$. Donner explicitement les ensembles $\mathcal{P}_{1,2}$ et $\mathcal{P}_{2,4}$.

On souhaite construire une matrice PC de taille $\bar{\imath} \times \bar{\jmath}$ telle que ses coefficients PC_{ij} stockent

les éléments de $\mathcal{P}_{i,j} \cap \mathcal{C}$, autrement dit les capteurs qui se trouvent dans $\mathcal{P}_{i,j}$, avec les indices $(i,j) \in \{0,\dots,\bar{\imath}-1\} \times \{0,\dots,\bar{\jmath}-1\}$.

Q43 Les éléments de $\mathcal{P}_{i,j} \cap \mathcal{C}$ peuvent être stockés sous la forme d'une liste ou d'un dictionnaire. Quels sont les avantages à choisir un dictionnaire par rapport à une liste ? On rappelle que les dictionnaires sont stockés en *Python* sous la forme d'une table de hachage, tandis que les listes sont des tableaux.

Q44 Le but de cette question est de programmer le calcul de la matrice PC.

1. Définir la fonction `is_inside (c, i, j, R)` qui teste si le capteur c est dans $\mathcal{P}_{ij}$.
2. En déduire la définition de la fonction `calcul_PC(dico_capteur, i, j, R)` qui retourne un ensemble contenant les éléments de $\mathcal{P}_{i,j} \cap \mathcal{C}$.
3. En déduire la définition de la fonction `matrice_PC(dico_capteur, R, Xmax, Ymax)` qui retourne la matrice PC telle que, pour tout couple $(i,j) \in \{0,\dots,\bar{\imath}\} \times \{0,\dots,\bar{\jmath}\}$, PC_{ij} est l'ensemble des éléments de $\mathcal{P}_{i,j} \cap \mathcal{C}$.

Pour tout sommet $s = (x,y) \in \{0,\dots,X_{\max}\} \times \{0,\dots,Y_{\max}\}$, on rappelle que :

$$\mathcal{C}(s) = \{c_i = (x_i, y_i, m_i) \in \mathcal{C}, d(s,c_i) \leq R\}$$

On définit également l'ensemble $\mathcal{D}(s) \subseteq \{(i,j) \in \{0,\dots,\bar{\imath}\} \times \{0,\dots,\bar{\jmath}\}\}$ le plus petit possible tel que $\mathcal{C}(s) \subseteq \bigcup_{(i,j)\in\mathcal{D}(s)} \mathcal{P}_{ij}$.

Q45 Soit $A = \{(i,j) \in \{0,\dots,\bar{\imath}\} \times \{0,\dots,\bar{\jmath}\}\}$ l'ensemble des couples d'indices réalisables (i,j) de $\mathcal{P}_{i,j}$. Soit $s = (x,y)$ un élément de la zone étudiée.

1. On suppose que x et y sont tous les deux divisibles par R. Définir $\mathcal{D}(s)$.
2. Même question pour x divisible par R et y non divisible par R.
3. Même question pour x et y non divisibles par R.
4. En déduire que, pour tout sommet $s = (x,y) \in \mathcal{P}_{i,j}$, $\mathcal{D}(s)$ est inclus dans un sous-ensemble de A à préciser et qui comporte au plus 9 éléments. On nomme cet ensemble $\mathcal{D}^\star_{ij}$. Donner l'expression de $\mathcal{D}^\star_{ij}$.

Q46 Donner la complexité totale du calcul de la matrice $\mathcal{D}^\star_{ij}$. Préciser pour cela les différentes étapes de ce calcul ainsi que leur complexité.

Pour tout sommet $s = (x,y) \in \{0,\dots,X_{\max}\} \times \{0,\dots,Y_{\max}\}$ qui ne possède pas de capteur, on souhaite estimer son taux de particules fines $PM_{2.5}$ par la formule :

$$\tilde{v}(s) = \frac{\sum_{c_i \in \mathcal{C}(s)} w_i(s,i) \times m_i}{\sum_{c_i \in \mathcal{C}(s)} w_i(s,c_i)} \quad \text{avec} \quad c_i = (x_i, y_i, m_i) \quad \text{et} \quad w(s,c_i) = \frac{1}{d(s,c_i)^2}$$

Sinon, si s possède un capteur $c = (x,y,m)$, alors $\tilde{v}(s) = m$.

Q47 On suppose dans cette question que la matrice $\mathcal{D}^\star$ a été calculée dans une phase de pré-traitement.

1. Expliciter comment vous pouvez utiliser la matrice $\mathcal{D}^\star$ pour calculer $\tilde{v}(s)$;
2. En déduire la complexité du calcul de $\tilde{v}(s)$, si s est un sommet de $\mathcal{P}_{ij}$. On suppose que les capteurs sont uniformément répartis sur la zone : dans une case de $\mathcal{P}$, le nombre de capteurs est approximativement égale à $\frac{N}{i \times j}$. Justifier la réponse.
3. En déduire la complexité du calcul de $\tilde{v}(s)$ pour l'ensemble des sommets. Justifier la réponse.
4. Comparer les deux méthodes, c'est-à-dire pour calculer $\hat{v}$ et $\tilde{v}$.

A – Annexes

A.1 Rappels sur les ondes électromagnétiques

Une **onde électromagnétique** consiste en des variations d'amplitude sinusoïdales et synchrones des champs de force électrique $\boldsymbol{E}$ et magnétique $\boldsymbol{B}$, qui se propagent dans le vide ou un milieu matériel. Pour simplifier, on ne représente que les variations du champ électrique, décrites par la fonction d'onde vectorielle complexe $\underline{\boldsymbol{E}}(M,t)$ où M est un point courant de l'espace et t l'abscisse temporelle. Toutefois, seule la partie réelle $\boldsymbol{E}(M,t) = \Re(\underline{\boldsymbol{E}}(M,t))$ trouve une interprétation physique, la notation complexe étant adoptée pour faciliter les calculs.

On appelle **éclairement énergétique** de l'onde sur une période T – ou **intensité moyenne** (en anglais, *irradiance*) – au point M la grandeur notée E_e et exprimée en $\mathrm{W\,m^{-2}}$ telle que :

$$E_\mathrm{e}(M) = \frac{1}{T}\int_0^T \boldsymbol{E}(M,t) \times \boldsymbol{H}(M,t)\,dt \qquad \text{avec} \qquad \boldsymbol{H} = \boldsymbol{B}/\mu_0 \quad \text{(excitation magnétique}\,[9]\text{)}$$

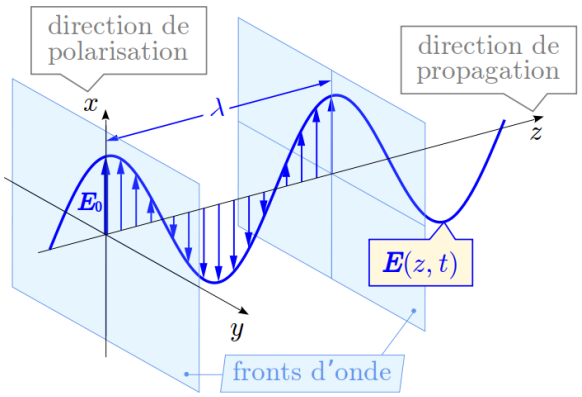
Onde plane monochromatique polarisée linéairement

Une **onde plane** (c'est-à-dire, à fronts d'onde plans et tous parallèles), **monochromatique** (présentant une seule fréquence) et **polarisée linéairement** (à variations d'amplitude selon une direction constante orthogonale à la direction de propagation) est un modèle qui ne dépend que d'une seule coordonnée spatiale, selon l'axe de propagation, par exemple z. Sa fonction s'écrit :

$$\underline{\boldsymbol{E}}(z,t) = \boldsymbol{E_0}\,\mathrm{e}^{\imath(kz-\omega t+\psi)} \qquad \text{avec} \qquad k = \frac{2\pi}{\lambda} \quad \text{(nombre d'onde)} \qquad \text{et} \qquad \omega = \frac{2\pi c}{\lambda} \quad \text{(pulsation)}$$

où $\boldsymbol{E_0}$ est son vecteur d'amplitude, λ sa longueur d'onde dans le milieu de propagation, c la vitesse de la lumière dans le vide et ψ un retard de phase à l'origine – cf. la figure ci-contre où x est la direction de polarisation de $\boldsymbol{E}$. Sa **partie réelle** peut donc s'écrire :

$$\boldsymbol{E}(z,t) = E_0\,\cos(kz-\omega t+\psi)\,\boldsymbol{x}$$



Son **intensité moyenne** est constante, elle vaut $E_\mathrm{e} = \frac{1}{2}c\varepsilon_0 {E_0}^2$ en tout point de l'espace[9]. Ce modèle décrit bien une source de lumière laser.

Onde sphérique monochromatique polarisée

Une **onde sphérique** (formant des fronts d'onde sphériques, tous concentriques), **monochromatique** (cf. supra) est un modèle qui ne dépend, en termes d'amplitude de variation, que de la coordonnée radiale r du point courant M considéré, exprimé en coordonnées polaires (r,θ,φ) dans un référentiel dont l'origine O est le centre des fronts d'onde. Quant à sa **polarisation**, elle peut être orientée par un vecteur $\hat{\boldsymbol{u}}(\theta,\varphi)$ orthogonal à l'axe de propagation $w(\theta,\phi)$, donc tangent à la surface du front d'onde. Sa fonction d'onde s'écrit alors :

$$\underline{\boldsymbol{E}}(r,\theta,\varphi,t) = \frac{1}{r}\,E_0\,\mathrm{e}^{\imath(kr-\omega t+\psi)}\,\hat{\boldsymbol{u}}(\theta,\varphi)$$

9. Les constantes μ_0 et ε_0 sont respectivement la perméabilité magnétique et la permittivité diélectrique du vide. Elles sont liées par la relation $c = 1/\sqrt{\varepsilon_0\mu_0}$ où c est la vitesse de la lumière dans le vide.

C

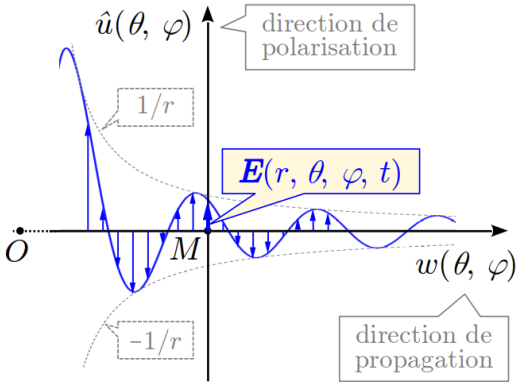
Tournez la page S.V.P.

où k, λ et ψ sont définis comme pour une onde plane (cf. la figure suivante). Ce modèle correspond typiquement à l'onde émise par l'agitation thermique d'une particule de matière.

L'**intensité moyenne** de l'onde au point M décroît en fonction de r pour compenser l'augmentation de surface du front d'onde, elle vaut [9] :

$$E_e(r) = \frac{c\varepsilon_0 {E_0}^2}{2r^2}$$

Toutefois, le modèle n'est réaliste que dans ce qu'on appelle le **champ lointain**, défini par la condition $r \gg d_F$ où $d_F = 2a^2/\lambda$ est appelée *distance de Fraunhofer* [13] (a étant le diamètre de la particule [10]).



Indice de réfraction d'un milieu de propagation non absorbant

Dans le **vide**, une onde électromagnétique monochromatique de longueur d'onde λ_0 et de fréquence f se propage à la vitesse de la lumière $c = \lambda_0 \cdot f$.

Dans un **milieu matériel non absorbant** (autrement dit « transparent »), la fréquence de l'onde reste inchangée mais sa longueur d'onde λ est diminuée. Il en résulte que sa vitesse de propagation $v = \lambda \cdot f$, dite *vitesse de phase*, est également diminuée.

On définit l'**indice réel de réfraction** du milieu par le quotient sans dimension $n = c/v = \lambda_0/\lambda$, sachant qu'il peut dépendre fortement de la longueur d'onde λ_0 considérée. L'indice n intervient dans la loi de Snell-Descartes décrivant le phénomène de réfraction ($n_1 \sin \alpha_1 = n_2 \sin \alpha_2$).

À titre de référence, l'indice de l'**air** dans des conditions usuelles de température, de pression et d'humidité vaut $n \approx 1{,}0003$ pour la longueur d'onde d'un laser rouge $\lambda_0 = 650\,\text{nm}$.

Indice de réfraction complexe d'un milieu absorbant

La notion d'indice de réfraction peut être généralisée à tout **milieu matériel absorbant** en adoptant la définition $\underline{n}^2 = \underline{\varepsilon}/\varepsilon_0$, la permittivité diélectrique $\underline{\varepsilon}$ d'un milieu pouvant prendre des valeurs complexes. Il en résulte que $\underline{n}$ est aussi à valeurs complexes.

En pratique, on note $\underline{n} = n + \imath\kappa$ l'indice de réfraction complexe (en anglais, *complex refractive index*, abrégé CRI) d'un milieu matériel absorbant, sachant que :

- sa partie réelle n caractérise le phénomène de réfraction ;
- sa partie imaginaire κ quantifie l'ampleur du phénomène d'absorption (cette valeur dépendant également de la longueur d'onde considérée).

Enfin, si le milieu matériel est lui-même immergé dans un milieu de propagation non vide d'indice de réfraction n_1, il est alors pertinent de considérer son **indice de réfraction complexe relatif** $\underline{m} = \underline{n}/n_1$.

10. La formule est surtout utilisée en radioélectricité où la source émettrice est une antenne, d'où la notation a.

A.2 Bibliothèque `miepython` (extraits)

La bibliothèque `miepython` [11], développée par Scott Prahl [12], permet de calculer les paramètres essentiels de la diffusion de lumière par des particules sphériques conformément à la théorie de Mie. Elle a pour dépendance le module de calcul `numpy` du langage *Python*.

Seules les fonctions de haut niveau utiles pour la présente étude sont données ci-après.

```python
def normalization_factor(m, x, norm_str):
    # Figure out scattering function normalization.
    # Args:
    #     m: complex index of refraction of sphere
    #     x: dimensionless sphere size
    #     norm_str: string describing type of normalization
    # Returns:
    #     scaling factor needed for scattering function
    factor = None
    norm = norm_str.lower()
    if norm in ["bohren"]:
        factor = 1 / 2
    elif norm in ["wiscombe"]:
        factor = 1
    elif norm in ["qsca", "scattering_efficiency"]:
        factor = x * np.sqrt(np.pi)
    else:
        qext, qsca, _, _ = _mie_scalar(m, x, 0)
        if norm in ["a", "albedo"]:
            factor = x * np.sqrt(np.pi * qext)
        if norm in ["1", "one", "unity"]:
            factor = x * np.sqrt(qsca * np.pi)
        if norm in ["four_pi", "4pi"]:
            factor = x * np.sqrt(qsca / 4)
        if norm in ["qext", "extinction_efficiency"]:
            factor = x * np.sqrt(qsca * np.pi / qext)
    if factor is None:
        raise ValueError(
            "normalization must be one of 'albedo' (default), 'one'"
            "'4pi', 'qext', 'qsca', 'bohren', or 'wiscombe'"
        )
    return factor
```

```python
def efficiencies(m, d, lambda0, n_env=1.0):
    # Calculate the efficiencies of a sphere.
    # Args:
    #    m: the complex index of refraction of the sphere       [-]
    #    d: the diameter of the sphere      [same units as lambda0]
    #    lambda0: wavelength in a vacuum          [same units as d]
    #    n_env: real index of medium around sphere, optional.
    # Returns (as a tupple, via the lower level function efficiencies_mx):
    #    qext: the total extinction efficiency                   [-]
    #    qsca: the scattering efficiency                         [-]
    #    qback: the backscatter efficiency                       [-]
    #    g: the average cosine of the scattering phase function [-]
    m_env = m / n_env
    x_env = np.pi * d / (lambda0 / n_env)
    return efficiencies_mx(m_env, x_env)
```

11. https://miepython.readthedocs.io/en/latest/index.html
12. Pr. d'optique biomédicale à l'*Institut de technologie de l'Orégon* (https://www.oit.edu/directory/scott-prahl)

Tournez la page S.V.P.

```
def S1_S2(m, x, mu, norm="albedo", n_pole=0):
    # Calculate the scattering amplitude functions for spheres.
    #
    # The amplitude functions have been normalized so that when integrated
    # over all 4*pi solid angles, the integral will be qext*pi*x**2.
    #
    # The normalization is controlled by `norm` and should be one of
    # ['albedo', 'one', '4pi', 'qext', 'qsca', 'bohren', or 'wiscombe']
    # The normalization describes the integral of the scattering phase
    # function over all 4 pi steradians.
    #
    # The units are weird, sr**(-0.5)
    #
    # Args:
    #     m: the complex index of refraction of the sphere
    #     x: the size parameter of the sphere
    #     mu: the angles, cos(theta), to calculate scattering amplitudes
    #     norm: (optional) string describing scattering function normalization
    #     n_pole: return n_pole term from series (default=0 include all terms)
    # Returns:
    #     S1, S2: the scattering amplitudes at each angle mu [sr**(-0.5)]
    if np.imag(m) > 0:  # ensure imaginary part of refractive index is negative
        m = np.conj(m)

    if np.isscalar(mu):
        mu_array = np.array([mu], dtype=float)
        S1, S2 = _S1_S2(m, x, mu_array, n_pole)
    else:
        S1, S2 = _S1_S2(m, x, mu, n_pole)

    normalization = normalization_factor(m, x, norm)

    S1 /= normalization
    S2 /= normalization

    return S1, S2
```

A.3 Mémento du langage *Python*

Modules numpy & scipy

`import numpy as np` — Aide `np.info(np.nom)` — Réf. numpy.org/doc/stable/reference/

Éléments mathématiques usuels

`uintn` `intn` (*n* = 8, 16, 32, 64) `floatn` (*n* = 16, 32, 64) `complexn` (*n* = 64, 128, 256)

`pi e inf nan abs fabs trunc round rint fix floor ceil min max square sqrt cbrt exp log log10 log2`
`sin cos tan sinh cosh tanh asin acos atan atan2 asinh acosh atanh degrees(rad2deg) radians(deg2rad)`
`real imag conj angle sign`

toutes les fonctions sont vectorisées

Tableaux multidimensionnels homogènes (ndarray)

■ **Constructeur et routines de création**

`ndarray(shape, dtype=float, buffer=None, offset=0, strides=None, order=None)` constructeur de bas niveau
`asarray(A, dtype=None, order=None, device=None, copy=None, like=None)` convertisseur général (*A* `array_like`)
`empty(shape,dtype=float,...)` `zeros(shape,dtype=float,...)` `ones(shape,dtype=float,...)` `array(object,dtype=None, ...)`
`empty_like(array,dtype=float,...)` `zeros_like(array,dtype=float,...)` `ones_like(array,dtype=float,...)`
`column_stack(tup)` → tableau 2D (*tup* tuple de tableaux 1D ou 2D)

■ **Attributs :** `dtype` → type `ndim` → nbre de dim. `shape` → tuple des tailles des dim. `size` → nbre total d'élém. `T` → tableau transposé

■ **Opérations surchargés :** *tous les opérateurs du noyau (* `+ - * /` *etc.)*

■ **Méthodes générales :** `item(indices)` `tolist()` `tobytes()` `fill(value)` `reshape(shape)` `flatten()` `squeeze()`
`take(indices)` `put(indices,values)` `choose(indices)` `sort()` `argsort()` `min()` `argmin()` `max()` `argmax()`
`sum()` `cumsum()` `mean()` `var()` `std()` `prod()` `cumprod()`

■ **Méthodes vectorielles (sur tableaux de dimension 1) :** • produit scalaire *V1*`.dot(`*V2*`)` ou `np.dot(`*V1*`,`*V2*`)` ou *V1* `@` *V2*
• générateurs `linspace(a,b,n)` → *n* valeurs de *a* à *b* (bornes incluses) `logspace(a,b,n)` → idem en $\log_{10}$ par défaut
• produit vectoriel (vecteurs de taille 3) `cross(`*V1*`,`*V2*`)`

■ **Méthodes matricielles (sur tableaux de dimension 2) :** • produit matriciel *M1*`.dot(`*M2*`)` ou `np.dot(`*M1*`,`*M2*`)` ou *M1* `@` *M2*
• générateurs `np.eye(n,k=d)` → matrice identité carrée de taille *n* [**1** décalés de `k` vers la droite] `np.diag(V)` → matrice diagonale de vecteur *V*
• méthodes générales *M*`.transpose()` → M^T *M*`.trace()` → tr(*M*)
• méthodes sur matrices carrées (algèbre linéaire) `import numpy.linalg as la`
`la.det(M)` → $|M|$ `la.inv(M)` → M^{-1} `la.matrix_rank(M)` → rang de *M* `la.matrix_power(M,p)` → M^p
`la.eigvals(M)` → vect. des val. propres `la.eig(M)` → vect. des val. propres, mat. de passage `la.solve(M,V)` → sol. *X* de *M X* = *V*

Tirages pseudo-aléatoires

`import numpy.random as rd` *toutes les fonctions sont vectorisables avec l'argument optionnel* `size`

■ **Loi uniforme :** `random()` → décimal dans [0 ; 1[`uniform(a,b)` → décimal dans [*a* ; *b*[
`randint(a,b)` → entier dans {*a*, … , *b* − 1} `choice(C)` → élément dans *C* (conteneur) `shuffle(C)` mélange les éléments de *C* (mutable)

■ **Autres lois :** `normal(m,σ)` `binomial(n,p)` `exponential(β)` ($\beta = 1/\lambda$) `beta(α,β)` `poisson(λ)` `chisquare(k)`

Statistiques

Arguments principaux : *A*, *B* *tableaux des données,* *axis*=`None` *dimension(s) de calcul,* *ddof*=`0` *delta degrees of freedom*

`ptp(A)` → étendue `median(A)` → médiane `quantile(A,q)` → *q*-ième quantile `percentile(A,q)` → *q*-ième centile
`mean(A)` → moyenne arithmétique `average(A,weights=None)` → moyenne algébrique [pondérée] `std(A)` → écart-type
`var(A)` → variance `corrcoef(A,B)` → coefficient de correlation *R* (*A* et *B* de même forme)
`cov(A,B, bias=False,ddof=None)` → covariance (*A* et *B* de même forme, par défaut *ddof* = `1` si *bias* = `False`)

Intégration numérique

`import scipy.integrate as spi`

■ **Méthode pour une fonction définie :** `quad(f,a,b)` → $\int_a^b f(t)dt$, ε estimation de l'erreur (algorithme adaptatif de *Fortran Quadpack*)

■ **Méthode pour tableaux de points :** `simpson(y,x)` → $\int_{x_{\min}}^{x_{\max}} y(x)dx$ (méthode de Simpson)

Module matplotlib

`import matplotlib.pyplot as plt` — Réf. matplotlib.org/stable/api/index.html

API implicite

`subplots(nrows=1,ncols=1,sharex=False,sharey=False)` → *figure* , *axes* crée une figure et un ensemble de tracés
`tight_layout(pad=1.08,h_pad=None,w_pad=None)` ajuste les marges autour des figures `draw()` redessine la figure courante
`show()` affiche toutes les figures `savefig(path)` enregistre la figure courante dans un fichier `close(fig=None)` ferme la figure courante

API explicite (orientée objet)

fig`,`*ax* `= plt.subplots()` *kwargs* *dictionnaire de mots-clefs attributs*

ax`.plot(`*x*`,`*y*`,**`*kwargs*`)` trace les points *y*(*x*), attributs de la classe `Line2D` : `color alpha marker markersize linestyle linewidth`...
ax`.grid(`*visible*`=None,`*which*`='major',`*axis*`='both',**`*kwargs*`)` configure la grille, mêmes attributs que `plot`
ax`.tick_params(`*axis*`='both',`*which*`='major',**`*kwargs*`)` configure les marques de grille, attributs `direction length width pad color labelsize labelcolor labelfontfamily labelrotation`
ax`.set_title(`*title*`,`*loc*`=None,`*pad*`=None,**`*kwargs*`)` définit le titre, attributs de la classe `Text` : `color family fontsize fontstyle`...
ax`.set_xscale(`*base*`,**`*kwargs*`)` définit l'échelle de l'axe `x` : `linear log function`... *ax*`.set_yscale` idem pour l'axe `y`

Tournez la page S.V.P.

Data Wrangling

with pandas Cheat Sheet

http://pandas.pydata.org

Pandas API Reference Pandas User Guide

Tidy Data – A foundation for wrangling in pandas

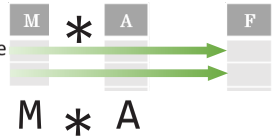
In a tidy data set:

Each **variable** is saved in its own **column**

&

Each **observation** is saved in its own **row**

Tidy data complements pandas's **vectorized operations**. pandas will automatically preserve observations as you manipulate variables. No other format works as intuitively with pandas.



Creating DataFrames

	a	b	c
1	4	7	10
2	5	8	11
3	6	9	12

```
df = pd.DataFrame(
        {"a" : [4, 5, 6],
         "b" : [7, 8, 9],
         "c" : [10, 11, 12]},
       index = [1, 2, 3])
```
Specify values for each column.

```
df = pd.DataFrame(
    [[4, 7, 10],
     [5, 8, 11],
     [6, 9, 12]],
    index=[1, 2, 3],
    columns=['a', 'b', 'c'])
```
Specify values for each row.

		a	b	c
N	v			
D	1	4	7	10
	2	5	8	11
e	2	6	9	12

```
df = pd.DataFrame(
        {"a" : [4 ,5, 6],
         "b" : [7, 8, 9],
         "c" : [10, 11, 12]},
index = pd.MultiIndex.from_tuples(
        [('d', 1), ('d', 2),
         ('e', 2)], names=['n', 'v']))
```
Create DataFrame with a MultiIndex

Method Chaining

Most pandas methods return a DataFrame so that another pandas method can be applied to the result. This improves readability of code.

```
df = (pd.melt(df)
        .rename(columns={
                'variable':'var',
                'value':'val'})
        .query('val >= 200')
    )
```

Reshaping Data – Change layout, sorting, reindexing, renaming

`pd.melt(df)`
Gather columns into rows.

`df.pivot(columns='var', values='val')`
Spread rows into columns.

`pd.concat([df1,df2])`
Append rows of DataFrames

`pd.concat([df1,df2], axis=1)`
Append columns of DataFrames

`df.sort_values('mpg')`
Order rows by values of a column (low to high).

`df.sort_values('mpg', ascending=False)`
Order rows by values of a column (high to low).

`df.rename(columns = {'y':'year'})`
Rename the columns of a DataFrame

`df.sort_index()`
Sort the index of a DataFrame

`df.reset_index()`
Reset index of DataFrame to row numbers, moving index to columns.

`df.drop(columns=['Length', 'Height'])`
Drop columns from DataFrame

Subset Observations - rows

`df[df.Length > 7]`
Extract rows that meet logical criteria.

`df.drop_duplicates()`
Remove duplicate rows (only considers columns).

`df.sample(frac=0.5)`
Randomly select fraction of rows.

`df.sample(n=10)` Randomly select n rows.

`df.nlargest(n, 'value')`
Select and order top n entries.

`df.nsmallest(n, 'value')`
Select and order bottom n entries.

`df.head(n)`
Select first n rows.

`df.tail(n)`
Select last n rows.

Subset Variables - columns

`df[['width', 'length', 'species']]`
Select multiple columns with specific names.

`df['width']` *or* `df.width`
Select single column with specific name.

`df.filter(regex='regex')`
Select columns whose name matches regular expression *regex*.

Using query

query() allows Boolean expressions for filtering rows.

```
df.query('Length > 7')
df.query('Length > 7 and Width < 8')
df.query('Name.str.startswith("abc")',
         engine="python")
```

Subsets - rows and columns

Use `df.loc[]` and `df.iloc[]` to select only rows, only columns or both.
Use `df.at[]` and `df.iat[]` to access a single value by row and column.
First index selects rows, second index columns.

`df.iloc[10:20]`
Select rows 10-20.

`df.iloc[:, [1, 2, 5]]`
Select columns in positions 1, 2 and 5 (first column is 0).

`df.loc[:, 'x2':'x4']`
Select all columns between x2 and x4 (inclusive).

`df.loc[df['a'] > 10, ['a', 'c']]`
Select rows meeting logical condition, and only the specific columns .

`df.iat[1, 2]` Access single value by index

`df.at[4, 'A']` Access single value by label

Logic in Python (and pandas)

<	Less than	!=	Not equal to
>	Greater than	`df.column.isin(values)`	Group membership
==	Equals	`pd.isnull(obj)`	Is NaN
<=	Less than or equals	`pd.notnull(obj)`	Is not NaN
>=	Greater than or equals	`&,|,~,^,df.any(),df.all()`	Logical and, or, not, xor, any, all

regex (Regular Expressions) Examples

`'\.'`	Matches strings containing a period '.'
`'Length$'`	Matches strings ending with word 'Length'
`'^Sepal'`	Matches strings beginning with the word 'Sepal'
`'^x[1-5]$'`	Matches strings beginning with 'x' and ending with 1,2,3,4,5
`'^(?!Species$).*'`	Matches strings except the string 'Species'

Cheatsheet for pandas (http://pandas.pydata.org/) originally written by Irv Lustig, Princeton Consultants, inspired by Rstudio Data Wrangling Cheatsheet

Group Data

`df.groupby(by="col")`
Return a GroupBy object, grouped by values in column named "col".

`df.groupby(level="ind")`
Return a GroupBy object, grouped by values in index level named "ind".

All of the summary functions listed above can be applied to a group. Additional GroupBy functions:

`size()`
Size of each group.

`agg(function)`
Aggregate group using function.

The examples below can also be applied to groups. In this case, the function is applied on a per-group basis, and the returned vectors are of the length of the original DataFrame.

`shift(1)`
Copy with values shifted by 1.

`rank(method='dense')`
Ranks with no gaps.

`rank(method='min')`
Ranks. Ties get min rank.

`rank(pct=True)`
Ranks rescaled to interval [0, 1].

`rank(method='first')`
Ranks. Ties go to first value.

`shift(-1)`
Copy with values lagged by 1.

`cumsum()`
Cumulative sum.

`cummax()`
Cumulative max.

`cummin()`
Cumulative min.

`cumprod()`
Cumulative product.

Summarize Data

`df['w'].value_counts()`
Count number of rows with each unique value of variable

`len(df)`
# of rows in DataFrame.

`df.shape`
Tuple of # of rows, # of columns in DataFrame.

`df['w'].nunique()`
# of distinct values in a column.

`df.describe()`
Basic descriptive and statistics for each column (or GroupBy).

`df.info()`
Prints a concise summary of the DataFrame.

`df.memory_usage()`
Prints the memory usage of each column in the DataFrame.

`df.dtypes()`
Prints a Series with the dtype of each column in the DataFrame.

pandas provides a large set of **summary functions** that operate on different kinds of pandas objects (DataFrame columns, Series, GroupBy, Expanding and Rolling (see below)) and produce single values for each of the groups. When applied to a DataFrame, the result is returned as a pandas Series for each column. Examples:

`sum()`
Sum values of each object.

`count()`
Count non-NA/null values of each object.

`median()`
Median value of each object.

`quantile([0.25,0.75])`
Quantiles of each object.

`apply(function)`
Apply function to each object.

`min()`
Minimum value in each object.

`max()`
Maximum value in each object.

`mean()`
Mean value of each object.

`var()`
Variance of each object.

`std()`
Standard deviation of each object.

Handling Missing Data

`df.dropna()`
Drop rows with any column having NA/null data.

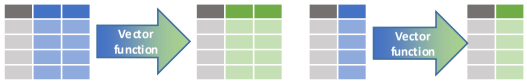
`df.fillna(value)`
Replace all NA/null data with value.

Make New Columns

`df.assign(Area=lambda df: df.Length*df.Height)`
Compute and append one or more new columns.

`df['Volume'] = df.Length*df.Height*df.Depth`
Add single column.

`pd.qcut(df.col, n, labels=False)`
Bin column into n buckets.



pandas provides a large set of **vector functions** that operate on all columns of a DataFrame or a single selected column (a pandas Series). These functions produce vectors of values for each of the columns, or a single Series for the individual Series. Examples:

`max(axis=1)`
Element-wise max.

`clip(lower=-10,upper=10)`
Trim values at input thresholds

`min(axis=1)`
Element-wise min.

`abs()`
Absolute value.

Windows

`df.expanding()`
Return an Expanding object allowing summary functions to be applied cumulatively.

`df.rolling(n)`
Return a Rolling object allowing summary functions to be applied to windows of length n.

Combine Data Sets

adf

x1	x2
A	1
B	2
C	3

+

bdf

x1	x3
A	T
B	F
D	T

=

Standard Joins

x1	x2	x3
A	1	T
B	2	F
C	3	NaN

`pd.merge(adf, bdf, how='left', on='x1')`
Join matching rows from bdf to adf.

x1	x2	x3
A	1.0	T
B	2.0	F
D	NaN	T

`pd.merge(adf, bdf, how='right', on='x1')`
Join matching rows from adf to bdf.

x1	x2	x3
A	1	T
B	2	F

`pd.merge(adf, bdf, how='inner', on='x1')`
Join data. Retain only rows in both sets.

x1	x2	x3
A	1	T
B	2	F
C	3	NaN
D	NaN	T

`pd.merge(adf, bdf, how='outer', on='x1')`
Join data. Retain all values, all rows.

Filtering Joins

x1	x2
A	1
B	2

`adf[adf.x1.isin(bdf.x1)]`
All rows in adf that have a match in bdf.

x1	x2
C	3

`adf[~adf.x1.isin(bdf.x1)]`
All rows in adf that do not have a match in bdf.

ydf

x1	x2
A	1
B	2
C	3

+

zdf

x1	x2
B	2
C	3
D	4

=

Set-like Operations

x1	x2
B	2
C	3

`pd.merge(ydf, zdf)`
Rows that appear in both ydf and zdf (Intersection).

x1	x2
A	1
B	2
C	3
D	4

`pd.merge(ydf, zdf, how='outer')`
Rows that appear in either or both ydf and zdf (Union).

x1	x2
A	1

```
pd.merge(ydf, zdf, how='outer',
         indicator=True)
.query('_merge == "left_only"')
.drop(columns=['_merge'])
```
Rows that appear in ydf but not zdf (Setdiff).

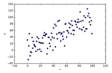
Cheatsheet for pandas (http://pandas.pydata.org/) originally written by Irv Lustig, Princeton Consultants, inspired by Rstudio Data Wrangling Cheatsheet

Tournez la page S.V.P.

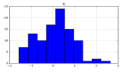
Plotting

df.plot()
Plot a line graph for the DataFrame.

df.plot.scatter(x='w', y='h')
Plot a scatter graph of the DataFrame.



df.plot.hist()
Plot a histogram of the DataFrame.



df.plot.pie()
Plot a pie chart of the DataFrame.

df.plot.bar()
Plot a line graph for the DataFrame.

df.plot.boxplot()
Plot a scatter graph of the DataFrame.

df.plot.area()
Plot an area graph of the DataFrame.

df.plot.hexbin()
Plot a hexbin graph of the DataFrame.

df.plot(subplots=True)
Separate into different graphs for each column in the DataFrame.

df.plot(title="Graph of A against B")
Sets the title of the graph.

df.plot(cumulative=True)
Creates a cumulative plot

df.plot(bins=30)
Set the number of bins into which data is grouped (histograms)

df.plot(stacked=True)
Stacks the data for the columns on top of each other. (bar, barh and area only)

df.plot(alpha=0.5)
Sets the transparency of the plot to 50%.

df.plot(subplots=True, title=['col1', 'col2', 'col3'])
Arguments can be combined for more flexibility when graphing, this would plot a separate line graph for of column of a 3-columned DataFrame. The first string in the list of titles applies to the graph of the left-most column.

Changing Type

pd.to_numeric(data)
Convert non-numeric types to numeric.

pd.to_datetime(data)
Convert non-datetime types to datetime type

pd.to_timedelta(data)
Convert non- timedelta types to timedelta

df.astype(type)
Convert data to (almost) any given type including categorical

df.infer_objects()
Attempts to infer a better type for object type data.

df.convert_dtypes()
Convert columns to best possible dtypes

Datetime

With a Series containing data of type datetime, the dt accessor is used to get various components of the datetime values:

s.dt.year
Extract the year

s.dt.month
Extract the month as an integer.

s.dt.day
Extract the day (int) from the date.

s.dt.quarter
Find which quarter the date lies in.

s.dt.hour
Extract the hour.

s.dt.minute
Extract the minute.

s.dt.second
Extract the second.

Mapping

Apply a mapping to every element in a DataFrame or Series, useful for recategorizing or transforming data.

s.map(lambda x: 2*x)
Returns a copy of the series where every entry is doubled

df.apply(lambda s: s.max() - s.min(), axis=1)
Returns a Series with the difference of the maximum and minimum values of each row of the DataFrame

Series String Operations

Similar to python string operations, except these are vectorized to apply to the entire Series efficiently.

s.str.count(pattern)
Returns a series with the integer counts in each element.

s.str.get(index)
Returns a series with the data at the given index for each element.

s.str.join(sep)
Returns a series where each element has been concatenated.

s.str.title()
Converts the first character of each word to be a capital.

s.str.len()
Returns a series with the lengths of each element.

s.str.cat()
Concatenate elements into a single string

s.str.partition(sep)
Splits the string on the first instance of the separator

s.str.slice(start, stop, step)
Slices each string

s.str.replace(pat, rep)
Use regex to replace patterns in each string.

s.str.isalnum()
Checks whether each element is alpha-numeric

Input/Output

Common file types for data input include CSV, JSON, HTML which are human-readable, while the common output types are usually more optimized for performance and scalability such as feather, parquet and HDF.

df = pd.read_csv(filepath)
Read data from csv file

df = pd.read_html(filepath)
Read data from html file

df = pd.read_excel(filepath)
Read data from xls (and related) files

df = pd.read_sql(filepath)
Read data from sql file

pd.read_clipboard()
Read text from clipboard

df.to_parquet(filepath)
Write data to parquet file

df.to_feather(filepath)
Write data to feather file

df.to_hdf(filepath)
Write data to HDF file

df.to_clipboard()
Copy object to the system clipboard

Frequently Used Options

Pandas offers some 'options' to globally control how Pandas behaves, display etc. Options can be queried and set via: **pd.options.*option_name*** (where *option_name* is the name of an option). For example:

pd.options.display.max_rows = 20
Set the **display.max_rows** option to 20.

Functions

get_option(option)
Fetch the value of the given option.

set_option(option)
Set the value of the given option.

reset_option(options)
Reset the values of all given options to default settings.

describe_option(options)
Print descriptions of given options.

option_context(options)
Execute code with temporary option settings that revert to prior settings after execution.

Display options

display.max_rows
The maximum number of rows displayed in pretty-print.

display.max_columns
The maximum number of columns displayed in pretty-print.

display.expand_frame_repr
Controls whether the DataFrame representation stretches across pages.

display.large_repr
Controls whether a DataFrame that exceeds maximum rows/columns is truncated or summarized

display.precision
The output display precision in decimal places.

display.max_colwidth
The maximum width of columns, longer cells will be truncated.

display.max_info_columns
The maximum number of columns displayed after calling **info()**.

display.chop_threshold
Sets the rounding threshold to zero when displaying a Series/DataFrame.

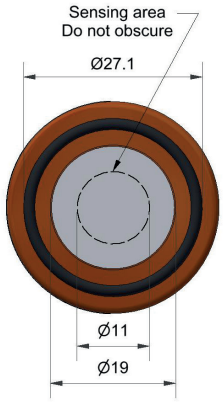
display.colheader_justify
Controls how column headers are justified.

A.4 Fiche technique du capteur *Alphasense* NO2-B43F

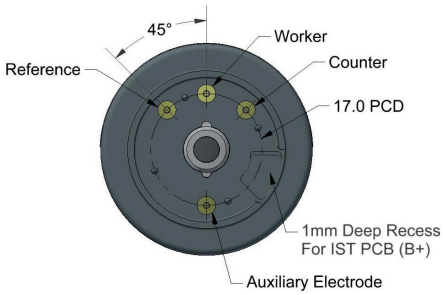
NO2-B43F/NO2-B43F+ Nitrogen Dioxide Sensor

The NO2-B43F sensor is a PPB sensor that is designed for environmental air quality applications with best-in-class baseline stability. This product is available in our standard format (NO2-B43F) and with our patented Integrated Smart Technology (NO2-B43F+) that has an IST board with a memory chip and temperature sensor integrated in the sensor. The + sensors store specific calibration, specification, and identification data on every sensor allowing plug and play operation. The on-board temperature sensor improves the accuracy and simplicity of temperature compensation algorithms.

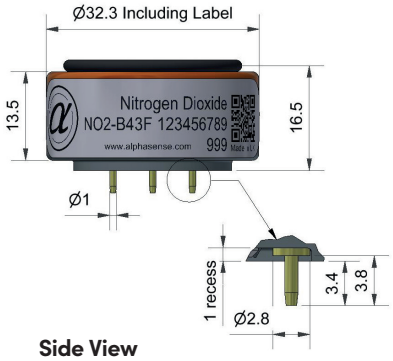
NO2-B43F Nitrogen Dioxide Sensor – 4-Electrode



Top View

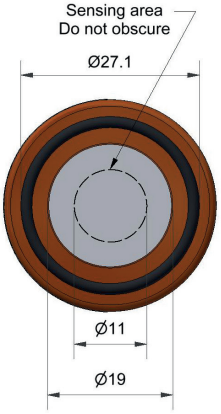


Bottom View

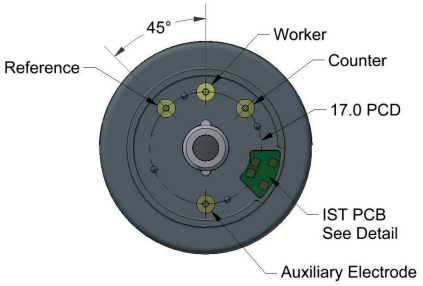


Side View

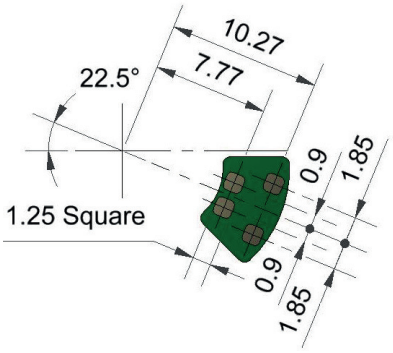
NO2-B43F+ Nitrogen Dioxide Sensor – 4-Electrode (with Integrated Smart Technology)



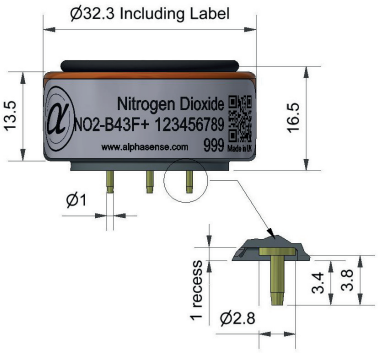
Top View



Bottom View



IST Board



Side View

B+ SERIES

INTEGRATED SMART TECHNOLOGY

Dimensions are in millimetres (± 0.15 mm).

T: +44 (0)1376 556700 E: sensors.alphasense@ametek.com W: www.alphasense.com

For further information on the performance of this sensor, on other sensors in the range or any other subject, please contact Alphasense Ltd. or visit our website at "www.alphasense.com".

Tournez la page S.V.P.

Sensor Data

Performance	Sensitivity	nA/ppm at 2ppm NO_2	-200 to -650
	Response time	t90 (s) from zero to 2ppm NO_2	< 80
	Zero current	nA in zero air at 20°C	-80 to +80
	Noise*	±2 standard deviations (ppb equivalent)	15
	Range	ppm NO_2 limit of performance warranty	20
	Linearity	ppb error at full scale, linear at zero and 20ppm	< ± 0.5
	Overgas limit	NO_2 maximum ppm for stable response to gas pulse	50
	***Tested with Alphasense ISB low noise circuit**		
Lifetime	Zero drift	ppb equivalent change/year in lab air	0 to 20
	Sensitivity drift	% change/year in lab air, monthly test	-20 to -40
	Operating life	months until 50% original signal (24-month warranted)	> 24
Environmental	Sensitivity @ -20°C	% (output @ -20°C/output @ 20°C) @ 2ppm NO_2	60 to 80
	Sensitivity @ 40°C	% (output @ 50°C/output @ 20°C) @ 2ppm NO_2	95 to 115
	Zero @ -20°C	nA	0 to 25
	Zero @ 40°C	nA	-10 to 50

Cross-sensitivity	O_3	filter capacity (ppm hrs)	@ 0.5ppm	O_3	< 500
	H_2S	sensitivity	% measured gas @ 5ppm	H_2S	< -80
	NO	sensitivity	% measured gas @ 5ppm	NO	< 5
	Cl_2	sensitivity	% measured gas @ 5ppm	Cl_2	< 100
	SO_2	sensitivity	% measured gas @ 5ppm	SO_2	< -3
	CO	sensitivity	% measured gas @ 5ppm	CO	< -3
	H_2	sensitivity	% measured gas @ 100ppm	H_2	< 0.1
	C_2H_4	sensitivity	% measured gas @ 100ppm	C_2H_4	< 0.1
	NH_3	sensitivity	% measured gas @ 20ppm	NH_3	< 0.1
	CO_2	sensitivity	% measured gas @ 5% volume	CO_2	< 0.1
	Halothane	sensitivity	% measured gas @ 100ppm	Halothane	nd

Key Specifications	Temperature range	°C	-30 to 40
	Pressure range	kPa	80 to 120
	Humidity range	% rh continuous	15 to 85
	Storage period	months @ 3 to 20°C (stored in sealed pot)	6
	Load resistor	Ω (ISB circuit is recommended)	33 to 100
	Weight	g	< 13

Figure 1 Sensitivity Temperature Dependence

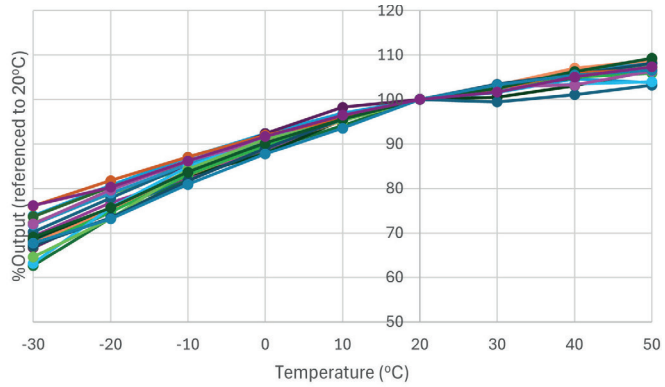


Figure 1 shows the temperature dependence of sensitivity at 2ppm NO_2.

This data is taken from a typical batch of sensors.

Figure 2 Zero Temperature Dependence

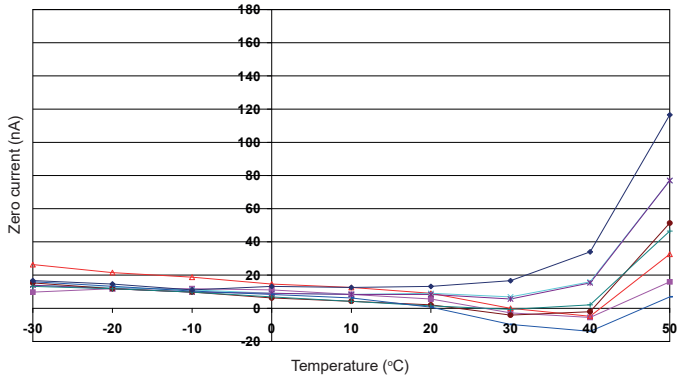
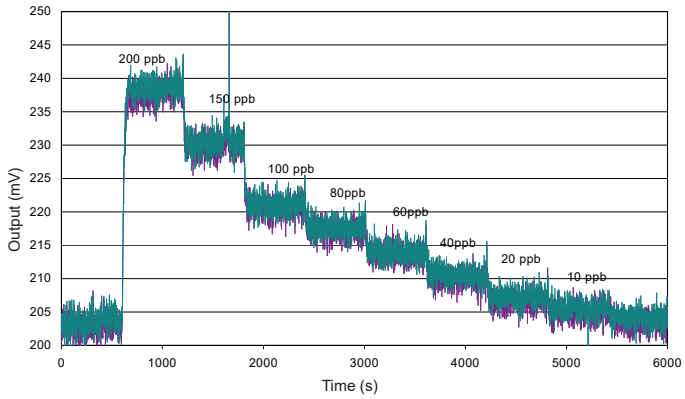


Figure 2 shows the variation in zero output of the working electrode caused by changes in temperature, expressed as nA.

This data is taken from a typical batch of sensors.

Contact Alphasense for further information on zero current correction.

Figure 3 Response from 200 ppb NO_2



With a 33 Ω load resistor, the NO2-B43F shows excellent resolution, even at the ppb level: ideal for outdoor air environmental testing.

Use of Alphasense ISB circuit reduces noise to 15ppb, with the opportunity of digital smoothing to reduce noise even further.

Offset voltage is due to intentional ISB circuit electronic offset.

IST Board Data

Interface	Communication Bus	Compatible with the 400 kHz I²C protocol
	Max. Bus Speed	Up to 1 MHz
	Input Logic Levels	High (Recessive) < 2.3 V | Low (Dominant) < 0.2 V
	Absolute Max. Input Signal	3.6 V
Electrical	Supply Voltage Range	1.7 V to 3.6 V
	Supply current – Stand-By	< 5 µA
	Supply current – Operating	< 0.15 mA (temperature reading only) < 2.15 mA (temperature reading + memory reading/writing)
	Power Supply Conditioning	Built-In 100 nF decoupling capacitor
	ESD Protection	4 kV (human body model) - Enhanced ESD / Latch-Up protection
	Bus Pins Input Capacitance	15 pF max.
Performance	Operational Temperature	–40 °C to +85 °C
	Temperature Sensor Accuracy	±1°C (-0°C to +70°C)
	Memory Data Retention	> 200 years
	Memory Write Cycles	> 4,000,000
Data & Communication	Memory IC & I2C Address	M24128X-FCU | Device Address: R - 0xA0 / W – 0xA1
	Temperature IC & I2C Address	MAX31875R0TZS+T | Device Address: R - 0x90 / W – 0x91
	Product Data Start Address	0x0900
	Calibration Data Start Address	0x0B00
	User Data Area	0x0D00 – 0x18FF (3,072 Bytes)
	CRC Polynomial	0x 01 04C1 1DB7
	Digital Signature Algorithm	SHA-256

Factory-populated data

Product Data
- Data Format Version
- Customer (OEM) ID
- Product ID
- Type of Sensor / Target Gas
- Sensor Serial Number
- End of Storage Period Date
- Sensor Replacement Date
- Product Data Checksum
- Alphasense Digital Signature
- Customer Digital Signature

Calibration
- Calibration Data Units
- Zero (clean dry air) Output
- Calibration Span
- Calibration Output
- Sensitivity
- Calibration Date
- Calibration Data Checksum
- Calibration Data Signature

Sensor Specification
- Over-gas limit
- Concentration Range
- Temperature Range Low
- Temperature Range High
- Humidity Range Low
- Humidity Range High
- Pressure Range Low
- Pressure Range High
- Specification Checksum

15,000+ locations

Customer Specific
- Custom Parameters
- Re-Calibration Due Date
- Operational Limits:
- Low | High | STEL | TWA
- Next Bump Test Due Date
- User Data Area

T: +44 (0)1376 556700 E: sensors.alphasense@ametek.com W: www.alphasense.com

For further information on the performance of this sensor, on other sensors in the range or any other subject, please contact Alphasense Ltd. or visit our website at "www.alphasense.com".

A.5 Programme *Arduino* de mesure des capteurs

```
#include <Wire.h>
#include <Adafruit_ADS1115.h>
#include <DHT.h>

// --- Definitions des broches ---
#define DHTPIN 2      // Broche du capteur DHT11
#define DHTTYPE DHT11 // Type de capteur DHT

// --- Initialisation des objets ---
Adafruit_ADS1115 ads1115;
DHT dht(DHTPIN, DHTTYPE);

// --- Structure pour les parametres du modele multi-lineaire ---
typedef struct {
  float a; // Coefficient pour WE
  float b; // Coefficient pour Aux
  float c; // Coefficient pour la temperature
  float d; // Coefficient pour l'humidite
  float e; // Terme constant
} ModelParameters;

// --- Variables globales pour le capteur NO2 ---
unsigned int uiSensNr = 6;
const char cSensId = 'H';
const float fVadcMultiplier = 0.125;
const unsigned long ulSampleInterv = 1000; // Intervalle d'echantillonnage (ms)
const unsigned long ulGasSendInterv = 48;   // Intervalle d'envoi des donnees (
    ms)

// --- Variables pour les mesures ---
unsigned int uiNrInAvg = 0;
unsigned long ulSensMeasSumWE = 0;
unsigned long ulSensMeasSumAux = 0;
unsigned long ulActMilli = 0;
unsigned long ulSampleMilli = 0;
float fSensMeas[22]; // Tableau pour le calcul de l'ecart-type
float fTemperature = 0.0;
float fHumidity = 0.0;

// --- Prototypes de fonctions ---
void measure_weather(float *temperature, float *humidity);
float calculate_NO2_concentration(float we, float aux, float temperature, float
     humidity, ModelParameters model);
char GetCheckSum(String pIn);

// --- Parametres du modele (a adapter) ---
ModelParameters model = {0.1, 0.2, 0.05, 0.01, 10.0}; // Exemple de valeurs
    arbitraires

// --- Configuration initiale ---
void setup() {
  Serial.begin(9600);
  Serial.println("Initialisation du capteur NO2 et DHT11...");

  // Initialisation de l'ADC
  ads1115.setGain(GAIN_ONE);
  ads1115.begin();

  // Initialisation du DHT11
```

Tournez la page S.V.P.

```
  dht.begin();

  // Initialisation des variables
  ulActMilli = millis();
  ulSampleMilli = ulActMilli + ulSampleInterv;
}

// --- Boucle principale ---
void loop() {
  int16_t i16ResultADC01, i16ResultADC23;
  float fSensMeasMeanWE, fSensMeasMeanAux, fStdevWE;
  float fDiffMeasMean, fSumRtDiffMeasMean = 0;
  int iPnt;
  String dataString = "";
  const char STX = 2;
  const char ETX = 3;
  char cCheckSum;

  // Mesure des donnees meteo (temperature et humidite)
  measure_weather(&fTemperature, &fHumidity);

  // Lecture des tensions WE et Aux
  ulActMilli = millis();
  i16ResultADC01 = ads1115.readADC_Differential_0_1(); // WE
  i16ResultADC23 = ads1115.readADC_Differential_2_3(); // Aux

  // Mise a jour des sommes pour le calcul des moyennes
  ulSensMeasSumWE += i16ResultADC01;
  ulSensMeasSumAux += i16ResultADC23;
  fSensMeas[uiNrInAvg++] = float(i16ResultADC01) * fVadcMultiplier;

  // Attente
  if (ulActMilli < ulSampleMilli) {
    if ((ulActMilli + ulGasSendInterv) > ulSampleMilli) {
      delay(abs(ulSampleMilli - ulActMilli));
    } else {
      delay(ulGasSendInterv);
    }
  } else {
    // Calcul des moyennes
    fSensMeasMeanWE = (float(ulSensMeasSumWE) / uiNrInAvg) * fVadcMultiplier;
    fSensMeasMeanAux = (float(ulSensMeasSumAux) / uiNrInAvg) * fVadcMultiplier;

    // Calcul de l'ecart-type pour WE
    for (iPnt = 0; iPnt < uiNrInAvg; iPnt++) {
      fDiffMeasMean = fSensMeas[iPnt] - fSensMeasMeanWE;
      fSumRtDiffMeasMean += fDiffMeasMean * fDiffMeasMean;
    }
    fStdevWE = sqrt(fSumRtDiffMeasMean / (uiNrInAvg - 1));

    // Calcul de la concentration de NO2 avec le modele multi-lineaire
    float no2_concentration = calculate_NO2_concentration(
      fSensMeasMeanWE, fSensMeasMeanAux, fTemperature, fHumidity, model
    );

    // Preparation de la chaine de donnees
    dataString = STX;
    dataString += "VQ";
    dataString += cSensId;
    dataString += String(uiSensNr);
```

```
    dataString += ";";
    dataString += String(uiNrInAvg);
    dataString += ";";
    dataString += String(fSensMeasMeanWE);
    dataString += ";";
    dataString += String(fSensMeasMeanAux);
    dataString += ";";
    dataString += String(no2_concentration);
    dataString += ";";
    dataString += String(fTemperature);
    dataString += ";";
    dataString += String(fHumidity);
    dataString += ";";
    dataString += String(fStdevWE);
    dataString += ETX;

    // Calcul du checksum
    cCheckSum = GetCheckSum(dataString);
    dataString += cCheckSum;

    // Envoi des donnees
    Serial.print(dataString);

    // Reinitialisation des variables
    uiNrInAvg = 0;
    ulSensMeasSumWE = 0;
    ulSensMeasSumAux = 0;
    ulSampleMilli = ulActMilli + ulSampleInterv;
  }
}

// --- Fonction pour mesurer la temperature et l'humidite ---
void measure_weather(float *temperature, float *humidity) {
  *humidity = dht.readHumidity();
  *temperature = dht.readTemperature();
  // Verification des erreurs de lecture
  if (isnan(*humidity) || isnan(*temperature)) {
    Serial.println("Erreur de lecture du capteur DHT11 !");
  }
}

// --- Fonction pour calculer le checksum ---
char GetCheckSum(String pIn) {
  int iXor = 0;
  int i = 0;
  while (pIn[i] != '\0') {
    iXor ^= pIn[i++];
  }
  if (iXor == 0) iXor = 1;
  return char(iXor);
}
```

Tournez la page S.V.P.

Ce sujet est basé sur des discussions avec des membres d'*Airparif* et de *Tera Sensor*, que les auteurs remercient pour leur disponibilité, ainsi que sur des informations disponibles dans la littérature scientifique. Néanmoins il n'engage pas ces organisations et ni représente leur travail. Dans les contraintes de confidentialité inhérentes à toute épreuve sélective, il repose sur des extrapolations libres effectuées par les auteurs à partir de leur compréhension du contexte technique et scientifique.

A.6 Bibliographie

[1] Organisation mondiale de la Santé, *Communiqué de presse du 25 mars 2014*
https://www.who.int/fr/news/item/25-03-2014-7-million-premature-deaths-annually-linked-to-air-pollution

[2] Santé publique France, *Communiqué de presse du 14 avril 2021*
https://www.santepubliquefrance.fr/presse/2021/pollution-de-l-air-ambiant-nouvelles-estimations-de-son-impact-sur-la-sante-des-francais

[3] Sénat, *Rapport fait au nom de la commission d'enquête sur le coût économique et financier de la pollution de l'air*, J.-F. Husson, L. Aïchi et alt., 8 juillet 2015
https://www.senat.fr/rap/r14-610-1/r14-610-11.pdf

[4] Organisation mondiale de la Santé, *Communiqué de presse n° 221 du 17 octobre 2013*
https://www.iarc.who.int/wp-content/uploads/2018/07/pr221_F.pdf

[5] Wikipedia, *Aerosol/Physics/Aerodynamic diameter*
https://en.wikipedia.org/wiki/Aerosol#Aerodynamic_diameter

[6] Organisation mondiale de la Santé, *Lignes directrices OMS relatives à la qualité de l'air*, 2021
https://iris.who.int/bitstream/handle/10665/346555/9789240035423-fre.pdf

[7] V. Ghersi, A. Rosso, S. Moukhtar et alt., *Origine des particules fines ($PM_{2.5}$) en Île-de-France*, 2012, Pollution atmosphérique, NS 5 | -1, 189-199. https://www.peren-revues.fr/pollutionatmospherique/7497?file=1

[8] Wikipedia, *Salle blanche*
https://fr.wikipedia.org/wiki/Salle_blanche

[9] Wikipedia, *Tapered element oscillating microbalance*
https://en.wikipedia.org/wiki/Tapered_element_oscillating_microbalance

[10] Wikipedia, *Beta attenuation monitoring*
https://en.wikipedia.org/wiki/Beta_attenuation_monitoring

[11] Tera Sensor, *NextPM datasheet*, version 4, 2024
https://tera-sensor.com/wp-content/uploads/2024/09/2024_NextPM-Datasheet.pdf

[12] Wikipedia, *Mie scattering*
https://en.wikipedia.org/wiki/Mie_scattering

[13] Wikipedia, *Fraunhofer distance*
https://en.wikipedia.org/wiki/Fraunhofer_distance

[14] C. Bohren, D. Huffman, *Absorption and scattering of light by small particles*, 1998, Wiley professional paperback edition.
https://archive.org/details/bohren_huffman_scattering_light_by_small

[15] Wikipedia, *Masse volumique apparente*
https://fr.wikipedia.org/wiki/Masse_volumique_apparente

[16] T. Wu, B. Boor, *Urban aerosol size distributions : a global perspective*, Atmospheric Chemestry & Physics, 21, 8883–8914, 2021.
https://acp.copernicus.org/articles/21/8883/2021/acp-21-8883-2021.pdf

[17] M. Kupper, L. Schubert et alt., *Measurement and Analysis of Brake and Tyre Particle Emissions from Automotive Series Components for High-Load Driving Tests on a Wheel and Suspension Test Bed*, Atmosphere 2024, 15, 430.
https://www.mdpi.com/2073-4433/15/4/430

[18] K. Park, D. Kittelson et alt., *Measurement of inherent material density of nanoparticle agglomerates*, Journal of Nanoparticle Research 6 : 267–272, 2004.
https://www.researchgate.net/publication/226292616_Measurement_of_Inherent_Material_Density_of_Nanoparticle_Agglomerates