Physique

Capacité numérique 5 – Variations de température et de pression dans l'atmosphère

Pour le mercredi 19 juin 10:00

Objectifs

- À l'aide d'un langage de programmation, étudier les variations de température et de pression dans l'atmosphère.
- Utiliser la fonction `odeint` de la bibliothèque `scipy.integrate`.

Le but de ce sujet est d'utiliser l'outil numérique afin de déterminer température et pression en fonction de l'altitude dans des modélisations plus fines que celle de l'atmosphère isotherme vue en cours.

Dans le cas isotherme, $P(z) = P_0 \exp\left(-\frac{Mg}{RT_0}z\right)$. Sur Terre, la température de surface moyenne est de $T_0 = 15\,°C$[1].

1. Tracer, dans une image « `fig1.png` », un graphique double :
— à gauche, la température en °C ;
— à droite, la pression en bar.
L'altitude sera prise entre 0 et 29 km, par pas de 10 m.
Afin de rendre ces tracés plus intuitifs, l'axe des ordonnées est *communs aux deux sous-graphiques* et correspond à l'altitude en km. Le code permettant de réaliser les deux graphiques côte à côte avec ordonnée commune est :

```
# création de la figure
figure1 = plt.figure()

# utilisation de sous-graphiques
graphT = figure1.add_subplot(1, 2, 1)
graphP = figure1.add_subplot(
    1, 2, 2,
    sharey = graphT, # partage des ordonnées
)
```

Pour tracer sur un des graphiques, utiliser par exemple `graphT.plot` au lieu de `plt.plot`.
Le tracé est réalisé en figure 1. Le code permettant de l'obtenir est ci-après.

```
# de 0 à 29 km par pas de 10 m
z_values_m = np.arange(0, 29e3, 10)

# atmosphère isotherme
T_0 = 273.15 + 15
T_K = np.ones(len(z_values_m)) * T_0

# application de la formule de l'atmosphère isotherme
P_0 = 1013.25e2 # 1013.25 hPa
M_air = 2*(.8*14+.2*16)*1e-3 # 80% de N2 et 20% de O2
g = 9.81 # g en SI
R = 8.314 # R en SI
P_isoT = P_0 * np.exp( - M_air * g / (R*T_0) * z_values_m)
```

1. Du moins, tant que le dérèglement climatique ne change pas la donne...

```
30
31 graphT.plot(
32     T_K - 273.15, # T en degC
33     z_values_m * 1e-3 # altitude en km
34 )
35 graphP.plot(
36     P_isoT / 1e5, # P en bar
37     z_values_m * 1e-3 # altitude en km
38 )
39 graphT.set_xlabel("Température (${}^\\circ$C)")
40 graphT.set_ylabel("Altitude (km)")
41 graphP.set_xlabel("Pression (bar)")
42 graphT.set_ylim(z_values_m.min()*1e-3, z_values_m.max()*1e-3)
43 graphP.set_xlim(0, 1)
44 graphT.grid() # pour afficher le quadrillage
45 graphP.grid() # pour afficher le quadrillage
46 plt.setp(graphP.get_yticklabels(), visible = False) # axe y commun, pas la peine
   ↪  d'afficher les valeurs
47 figure1.tight_layout()
48 figure1.savefig("fig1.png")
```

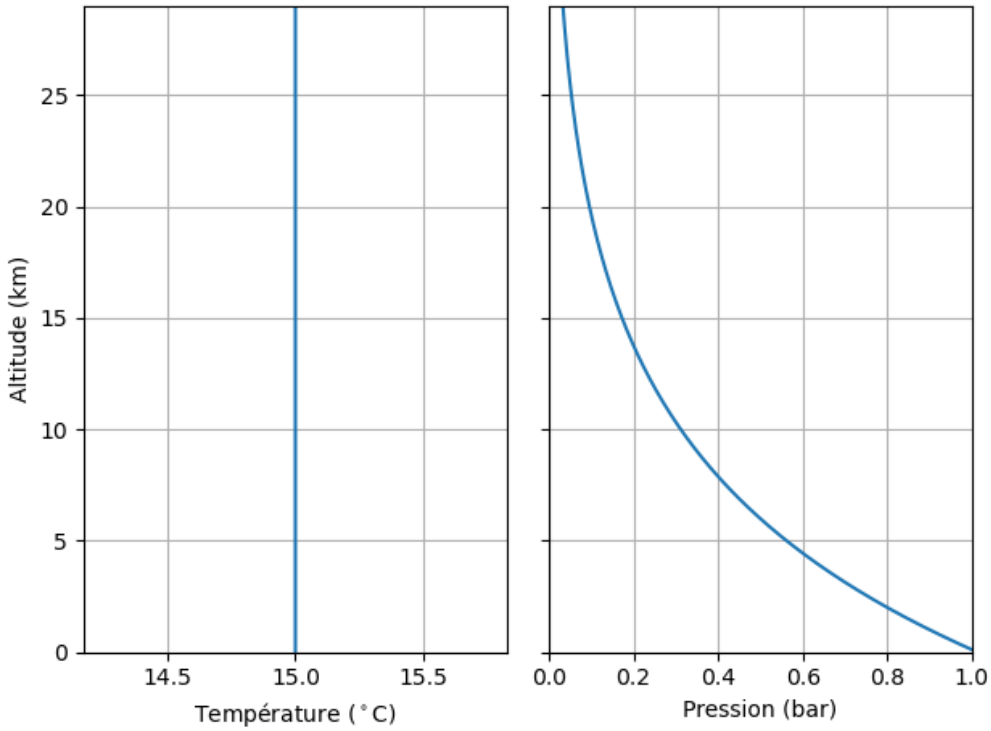

Figure 1 – *Le tracé « `fig1.png` ».*

Une autre modélisation possible est celle dite de l'atmosphère adiabatique.

2. Rappeler la loi de Laplace pour les variables P et V. En déduire l'équivalent pour les variables P et T en supposant que l'atmosphère peut être modélisée par un gaz parfait diatomique. Obtenir alors T en fonction de P, P_0 et T_0.

D'après la loi de Laplace, $PV^\gamma = \mathrm{C^{te}}$ avec $\gamma = c_P/c_V = 7/5$ pour un gaz parfait diatomique. Sachant de plus que $PV = nRT$,

$$PV^\gamma = P\left(\frac{nRT}{P}\right)^\gamma = P^{1-\gamma}T^\gamma \underbrace{(nR)^\gamma}_{\mathrm{C^{te}}} \Leftrightarrow P^{1-\gamma}T^\gamma = \mathrm{C^{te}} \Leftrightarrow \boxed{T = T_0\left(\frac{P_0}{P}\right)^{\frac{1-\gamma}{\gamma}}}.$$

3. Déterminer aussi l'équivalent pour les variables P et ρ la masse volumique de l'atmosphère. Obtenir alors $\frac{\mathrm{d}\rho}{\mathrm{d}P}$ en fonction de ρ, P et γ.

De même,

$$PV^\gamma = P\left(\frac{m}{\rho}\right)^\gamma = P\rho^{-\gamma}\underbrace{m^\gamma}_{\mathrm{C^{te}}} \Leftrightarrow \boxed{P\rho^{-\gamma} = \mathrm{C^{te}}}.$$

De cette relation, il vient en différentiant

$$\frac{P}{P_0} = \left(\frac{\rho}{\rho_0}\right)^\gamma \Leftrightarrow \ln\left(\frac{P}{P_0}\right) = \gamma\ln\left(\frac{\rho}{\rho_0}\right), \quad \frac{\mathrm{d}P}{P} = \gamma\frac{\mathrm{d}\rho}{\rho} \Leftrightarrow \boxed{\frac{\mathrm{d}\rho}{\mathrm{d}P} = \frac{1}{\gamma}\frac{\rho}{P}}.$$

4. Retrouver l'expression de ρ_0 la masse volumique de l'air à $z = 0$ et faire l'application numérique.

Comme l'atmosphère est modélisée par un gaz parfait,

$$\rho_0 = \frac{m}{V} = \frac{mP_0}{nRT_0} = \frac{MP_0}{RT_0} = 1{,}22\,\mathrm{kg{\cdot}m^{-3}}.$$

5. À partir de l'équation fondamentale de la statique des fluides, obtenir une équation différentielle sur P faisant intervenir $\rho(z)$.

D'après l'équation fondamentale de la statique des fluides, $\overrightarrow{\mathrm{grad}}(P) = \rho\vec{g}$. Alors, par projection sur (Oz) ascendant,

$$\frac{\mathrm{d}P}{\mathrm{d}z} = -\rho(z)g.$$

6. En déduire $\frac{\mathrm{d}\rho}{\mathrm{d}z}$.

En décomposant,

$$\frac{\mathrm{d}\rho}{\mathrm{d}z} = \frac{\mathrm{d}\rho}{\mathrm{d}P}\frac{\mathrm{d}P}{\mathrm{d}z} = -\frac{\rho^2}{\gamma P}g.$$

La bibliothèque `scipy.integrate` comporte une fonction `odeint` dont la documentation complète est disponible en ligne[2]. Pour un système tel que $\frac{\mathrm{d}X}{\mathrm{d}t} = F(X)$ avec X pouvant être un vecteur, son utilisation peut être simplifiée, avec les lignes nécessaires en amont, en

```
1 from scipy.integrate import odeint # import de la fonction
2 X0 = (theta_init, omega_init) # le vecteur X à l'instant initial, comme avant
3 temps = ... # la liste des instants t auxquels obtenir X(t)
4 X_liste = odeint(F, X0, temps)
```

avec `F` une fonction qui prend deux paramètres, `X` et `temps`.

7. L'idée est ici de remplacer la variable temporelle par l'altitude. Alors, `odeint` permet de résoudre les équations différentielles sur ρ et P précédemment obtenues.
Tracer ainsi, dans une image « `fig2.png` », l'équivalent de `fig1.png` mais dans le cas adiabatique en utilisant `odeint`.

Le tracé est réalisé en figure 2. Le code permettant de l'obtenir est ci-après.

```
50 # Cas adiabatique
51 from scipy.integrate import odeint
52 gamma = 1.4
53
54 def F_adiab(X, z):
55     rho = X[0]
```

2. https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.odeint.html

```
    P = X[1]
    # Application des formules
    dP_dz = - rho * g
    drho_dP = 1/gamma * rho / P
    drho_dz = drho_dP * dP_dz
    return (drho_dz, dP_dz)

rho_0 = M_air * P_0 / (R * T_0)
X0 = (rho_0, P_0) # conditions initiales

# Application de odeint
odeint_output = odeint(F_adiab, X0, z_values_m)
rhos = odeint_output[:,0]
P_adiab = odeint_output[:,1]
T_adiab = T_0 * (P_0 / P_adiab) ** ((1-gamma)/gamma)

# création de la figure
figure2 = plt.figure()

# utilisation de sous-graphiques
graphT = figure2.add_subplot(1, 2, 1)
graphP = figure2.add_subplot(
    1, 2, 2,
    sharey = graphT, # partage des ordonnées
)

graphT.plot(
    T_adiab - 273.15, # T en degC
    z_values_m * 1e-3 # altitude en km
)
graphP.plot(
    P_adiab / 1e5, # P en bar
    z_values_m * 1e-3 # altitude en km
)
graphT.set_xlabel("Température (${}^\\circ$C)")
graphT.set_ylabel("Altitude (km)")
graphP.set_xlabel("Pression (bar)")
graphT.set_ylim(z_values_m.min()*1e-3, z_values_m.max()*1e-3)
graphP.set_xlim(0, 1)
graphT.grid() # pour afficher le quadrillage
graphP.grid() # pour afficher le quadrillage
plt.setp(graphP.get_yticklabels(), visible = False) # axe y commun, pas la peine
↪  d'afficher les valeurs
figure2.tight_layout()
figure2.savefig("fig2.png")
```

8. Pourquoi ne pas aller jusqu'à 30 km d'altitude avec ce modèle ?

Dans ce modèle, la température décroit linéairement avec l'altitude. Or, il est impossible de descendre en-deçà de 0 K, qui est obtenu entre 29 et 30 km.

✎ **Sans odeint** *D'après l'équation fondamentale de la statique des fluides,* $\overrightarrow{\text{grad}}(P) = \rho\vec{g}$. *Alors, par projection sur* (Oz) *ascendant,*

$$\frac{\mathrm{d}P}{\mathrm{d}z} = -\rho g = -\rho_0 \left(\frac{P_0}{P}\right)^{-1/\gamma} g \Leftrightarrow \frac{\mathrm{d}P}{\mathrm{d}z} + \frac{1}{K_\gamma} P^{1/\gamma} = 0$$

avec

$$K_\gamma = \frac{P_0^{1/\gamma}}{\rho_0 g} = \frac{RT_0}{MP_0^{1-1/\gamma} g}\,.$$

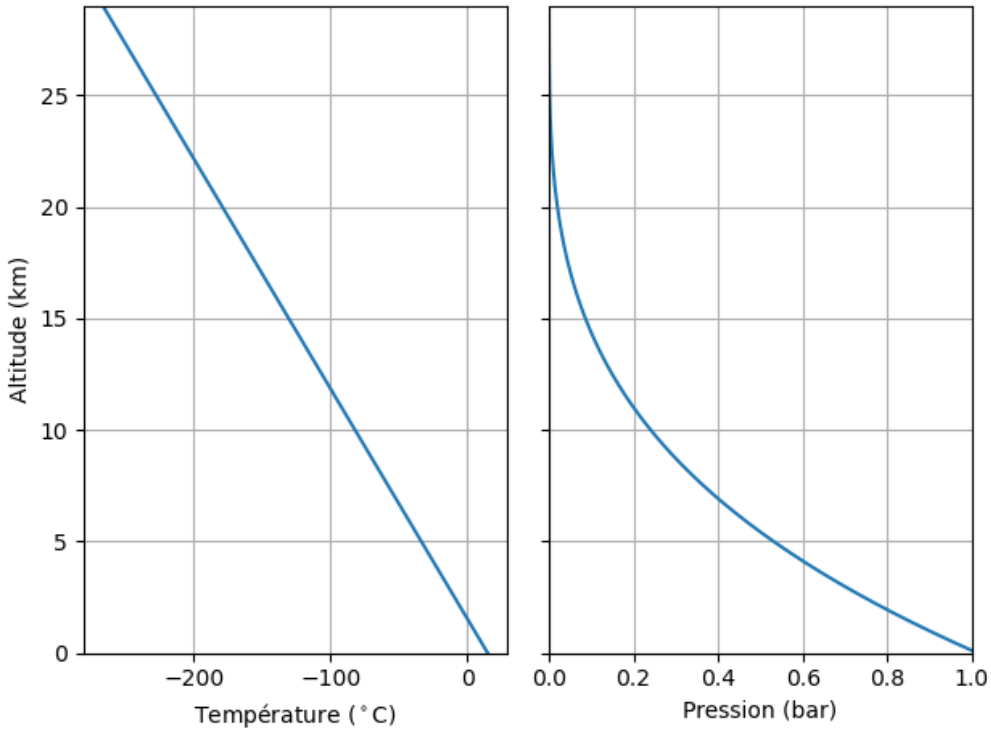

Figure 2 – *Le tracé « `fig2.png` ».*

Cette équation peut se résoudre par séparation des variables,

$$\frac{\mathrm{d}P}{\mathrm{d}z} + \frac{1}{K_\gamma} P^{1/\gamma} = 0 \Leftrightarrow \frac{\mathrm{d}P}{P^{1/\gamma}} = -\frac{\mathrm{d}z}{K_\gamma}\,.$$

Alors, par intégration entre $z = 0$ quand $P = P_0$,

$$\int_{P_0}^{P(z)} \frac{\mathrm{d}P}{P^{1/\gamma}} = -\int_0^z \frac{\mathrm{d}z'}{K_\gamma} \Leftrightarrow \frac{P^{1-1/\gamma}(z) - P_0^{1-1/\gamma}}{1-\frac{1}{\gamma}} = -\frac{z}{K_\gamma} \Leftrightarrow \boxed{P(z) = \left(P_0^{\frac{\gamma-1}{\gamma}} - \frac{z}{K_\gamma}\frac{\gamma-1}{\gamma}\right)^{\frac{\gamma}{\gamma-1}}}\,.$$

Alors,

$$T = T_0 \left(\frac{P_0}{P}\right)^{\frac{1-\gamma}{\gamma}} = T_0 \left(1 - \frac{P_0^{\frac{1-\gamma}{\gamma}}}{K_\gamma}\frac{\gamma-1}{\gamma} z\right)$$

d'où le comportement linéaire et

$$T(z_{\max}) = 0\,\mathrm{K} \Leftrightarrow z_{\max} = \frac{K_\gamma}{P_0^{\frac{1-\gamma}{\gamma}}}\frac{\gamma}{\gamma-1} = \frac{RT_0}{Mg}\frac{\gamma}{\gamma-1} = 29{,}7\,\mathrm{km}\,.$$

Les gradients de température sont en réalité connus pour l'atmosphère et consignés dans le tableau 1.

9. Écrire le code définissant la fonction `T_atm`, prenant comme paramètres la valeur `z` de l'altitude en mètres et qui rend la température à cette altitude en kelvins en se basant sur les données du tableau 1.

Le code définissant la fonction `T_atm` est

```
def T_atm(z):
    T = T_0

    # Troposphère
    grad = -6.5 # K/km
    z_max = 11e3 # base de la couche suivante en m
```

Couche	Altitude de début (km)	Gradient thermique ($\mathrm{K{\cdot}km^{-1}}$)
Troposphère	0	$-6{,}5$
Tropopause	11	0
Stratosphère	20	$+1{,}0$
Stratosphère	32	$+2{,}8$
Stratopause	47	0
Mésosphère	51	$-2{,}8$
Mésosphère	71	$-2{,}0$
Mésopause	85	0
Thermosphère	90	$+3{,}3$

Tableau 1 – *Données atmosphériques.*

```
    if z < z_max:
        T += grad*1e-3 * z
        return T
    T += grad*1e-3 * z_max

    # Tropopause
    grad = -0 # K/km
    z_min = z_max
    z_max = 20e3 # base de la couche suivante en m
    if z < z_max:
        T += grad*1e-3 * (z-z_min)
        return T
    T += grad*1e-3 * (z_max-z_min)

    # Stratosphère - 1
    grad = 1.0 # K/km
    z_min = z_max
    z_max = 32e3 # base de la couche suivante en m
    if z < z_max:
        T += grad*1e-3 * (z-z_min)
        return T
    T += grad*1e-3 * (z_max-z_min)

    # Stratosphère - 2
    grad = 2.8 # K/km
    z_min = z_max
    z_max = 47e3 # base de la couche suivante en m
    if z < z_max:
        T += grad*1e-3 * (z-z_min)
        return T
    T += grad*1e-3 * (z_max-z_min)

    # Stratopause
    grad = 0 # K/km
    z_min = z_max
    z_max = 51e3 # base de la couche suivante en m
    if z < z_max:
        T += grad*1e-3 * (z-z_min)
        return T
    T += grad*1e-3 * (z_max-z_min)

    # Mésosphère - 1
    grad = -2.8 # K/km
```

```
    z_min = z_max
    z_max = 71e3 # base de la couche suivante en m
    if z < z_max:
        T += grad*1e-3 * (z-z_min)
        return T
    T += grad*1e-3 * (z_max-z_min)

    # Mésosphère - 2
    grad = -2.0 # K/km
    z_min = z_max
    z_max = 85e3 # base de la couche suivante en m
    if z < z_max:
        T += grad*1e-3 * (z-z_min)
        return T
    T += grad*1e-3 * (z_max-z_min)

    # Mésopause
    grad = 0 # K/km
    z_min = z_max
    z_max = 90e3 # base de la couche suivante en m
    if z < z_max:
        T += grad*1e-3 * (z-z_min)
        return T
    T += grad*1e-3 * (z_max-z_min)

    # Thermosphère et au-delà
    grad = 3.3 # K/km
    z_min = z_max
    T += grad*1e-3 * (z-z_min)
    return T
```

10. En montant significativement en altitude, le champ de pesanteur g ne peut plus être considéré comme constant. Montrer qu'en réalité

$$g(z) = g_0 \frac{R_\mathrm{T}^2}{(R_\mathrm{T}+z)^2}$$

avec $g_0 = 9{,}81\,\mathrm{m{\cdot}s^{-2}}$, z l'altitude et $R_\mathrm{T} = 6371\,\mathrm{km}$ le rayon de la Terre.

En se basant sur l'expression de la force gravitationnelle qu'exerce la Terre sur un objet de masse m situé à une altitude z, correspondant à une distance $d = R_\mathrm{T} + h$ de son centre, et en particulier sa norme,

$$F_\mathrm{grav}(z) = \frac{\mathcal{G}M_\mathrm{T}m}{d^2} = \frac{\mathcal{G}M_\mathrm{T}m}{(R_\mathrm{T}+h)^2} = mg(z)\,, \quad F_\mathrm{grav}(z=0) = mg_0 = \frac{\mathcal{G}M_\mathrm{T}m}{R_\mathrm{T}^2}\,,$$

d'où

$$g(z) = \frac{\mathcal{G}M_\mathrm{T}}{(R_\mathrm{T}+h)^2}\,, \quad g_0 = \frac{\mathcal{G}M_\mathrm{T}}{R_\mathrm{T}^2}\,, \quad g(z) = g_0\frac{R_\mathrm{T}^2}{(R_\mathrm{T}+z)^2}\,.$$

11. Cet effet devra être pris en compte dans la suite de la simulation. Écrire le code définissant la fonction `g_Terre`, prenant comme paramètres la valeur `z` de l'altitude en mètres et qui rend la valeur de g à l'altitude z.

Le code code définissant la fonction `g_Terre` est

```
def g_Terre(z):
    R_T = 6371e3 # en m
    return 9.81 * (R_T/(R_T+z))**2
```

12. Exprimer ρ en fonction de P, T, M et R.

En reprenant l'équation d'état du gaz parfait,

$$\rho = \frac{m}{V} = \frac{mP}{nRT} = \frac{MP}{RT}\,.$$

13. En déduire le moyen d'obtenir le champ de pression dans l'atmosphère en utilisant `odeint`. Tracer comme précédemment T et P de 0 à 29 km par pas de 10 m dans « `fig3.png` ».

Le tracé est réalisé en figure 3. Le code permettant de l'obtenir est ci-après.

```
185 def F_gradT(X, z):
186     P = X[0]
187     # Application des formules
188     T_locale = T_atm(z)
189     rho_locale = M_air * P /(R*T_locale)
190     g_local = g_Terre(z)
191     dP_dz = - rho_locale * g_local
192     return (dP_dz)
193 
194 # Application de odeint
195 X0 = (P_0)
196 odeint_output = odeint(F_gradT, X0, z_values_m)
197 P_gradT = odeint_output[:,0]
198 T_gradT = [T_atm(z) - 273.15 for z in z_values_m] # T en degC
199 
200 # création de la figure
201 figure3 = plt.figure()
202 
203 # utilisation de sous-graphiques
204 graphT = figure3.add_subplot(1, 2, 1)
205 graphP = figure3.add_subplot(
206     1, 2, 2,
207     sharey = graphT, # partage des ordonnées
208 )
209 
210 graphT.plot(
211     T_gradT, # T en degC
212     z_values_m * 1e-3 # altitude en km
213 )
214 graphP.plot(
215     P_gradT / 1e5, # P en bar
216     z_values_m * 1e-3 # altitude en km
217 )
218 graphT.set_xlabel("Température (${}^\\circ$C)")
219 graphT.set_ylabel("Altitude (km)")
220 graphP.set_xlabel("Pression (bar)")
221 graphT.set_ylim(z_values_m.min()*1e-3, z_values_m.max()*1e-3)
222 graphP.set_xlim(0, 1)
223 graphT.grid() # pour afficher le quadrillage
224 graphP.grid() # pour afficher le quadrillage
225 plt.setp(graphP.get_yticklabels(), visible = False) # axe y commun, pas la peine
    ↪  d'afficher les valeurs
226 figure3.tight_layout()
227 figure3.savefig("fig3.png")
```

14. Dans « `fig4.png` », de 0 à 120 km par pas de 10 m, comparer les températures et pression obtenues dans les modèles isotherme (et g constant) et celui avec le gradient de température (et g variable). Afin de faciliter la lecture des valeurs de pression, les afficher en échelle logarithmique. Commenter.

Le tracé est réalisé en figure 4. Le code permettant de l'obtenir est ci-après.

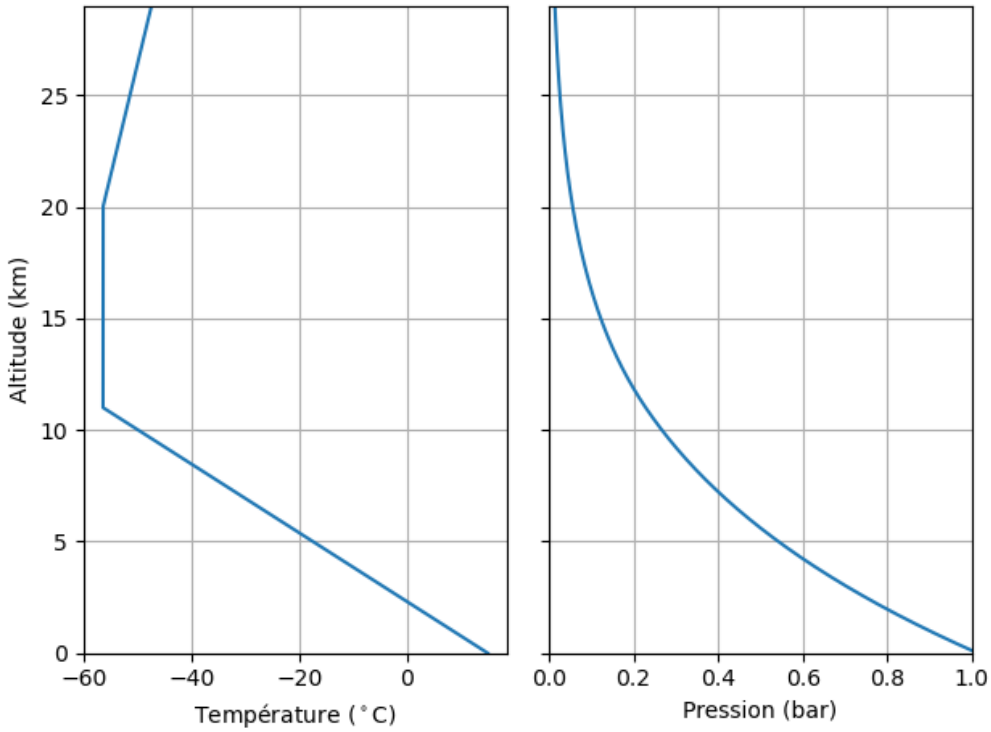

Figure 3 – *Le tracé « `fig3.png` ».*

```
229 # Application de odeint
230 z_values_m = np.arange(0, 120e3, 10)
231 X0 = (P_0)
232 odeint_output = odeint(F_gradT, X0, z_values_m)
233 P_gradT = odeint_output[:,0]
234 T_gradT = [T_atm(z) - 273.15 for z in z_values_m] # T en degC
235
236 P_isoT = P_0 * np.exp( - M_air * g / (R*T_0) * z_values_m)
237
238 # création de la figure
239 figure4 = plt.figure()
240
241 # utilisation de sous-graphiques
242 graphT = figure4.add_subplot(1, 2, 1)
243 graphP = figure4.add_subplot(
244     1, 2, 2,
245     sharey = graphT, # partage des ordonnées
246 )
247
248 graphT.plot(
249     T_gradT[0] * np.ones(len(z_values_m)), # T en degC
250     z_values_m * 1e-3, # altitude en km
251     label = "isotherme",
252 )
253 graphT.plot(
254     T_gradT, # T en degC
255     z_values_m * 1e-3, # altitude en km
256     label = "avec grad$T$",
257 )
258 graphP.plot(
259     P_isoT / 1e5, # P en bar
```

```
    z_values_m * 1e-3, # altitude en km
    label = "isotherme",
)
graphP.plot(
    P_gradT / 1e5, # P en bar
    z_values_m * 1e-3, # altitude en km
    label = "avec grad$T$",
)
graphT.set_xlabel("Température (${}^\\circ$C)")
graphT.set_ylabel("Altitude (km)")
graphP.set_xlabel("Pression (bar)")
graphT.set_ylim(z_values_m.min()*1e-3, z_values_m.max()*1e-3)
graphT.grid() # pour afficher le quadrillage
graphP.grid() # pour afficher le quadrillage
graphP.legend()
graphP.set_xscale("log") # pressions en échelle log
plt.setp(graphP.get_yticklabels(), visible = False) # axe y commun, pas la peine
    ↪ d'afficher les valeurs
figure4.tight_layout()
figure4.savefig("fig4.png")
```

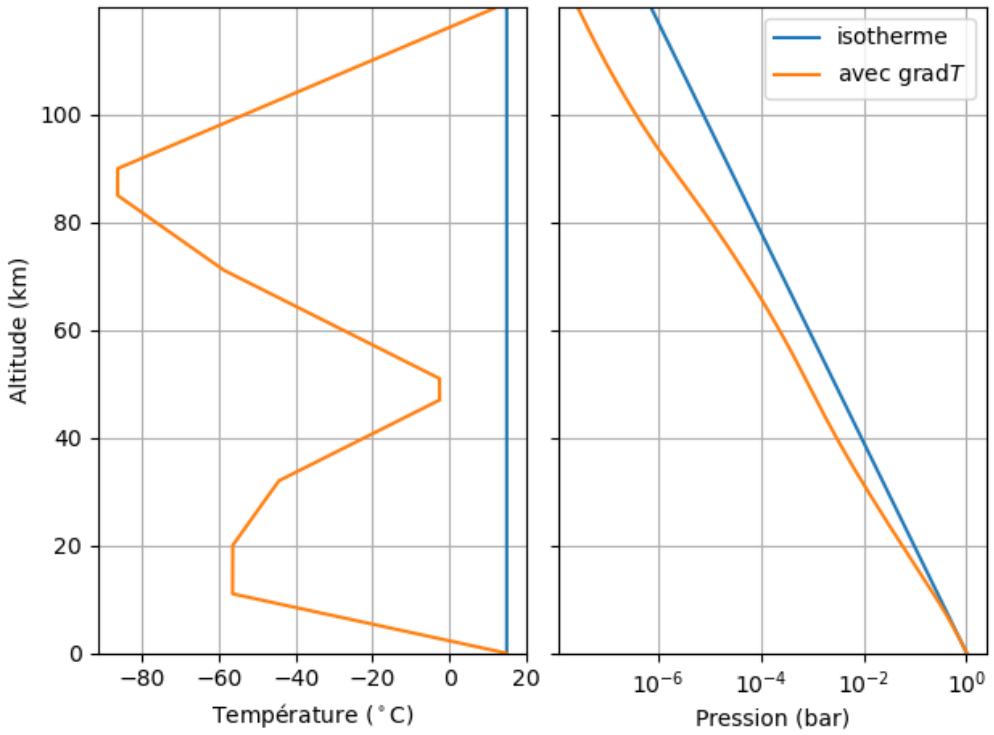

Figure 4 – *Le tracé « `fig4.png` ».*

Tracée en échelle logarithmique, la pression dans le modèle isotherme donne une droite, ce qui n'est pas étonnant vu son expression.

La pression diminue d'autant plus vite que la température est basse. Mais, le comportement reste proche de celui de l'atmosphère isotherme.