

Effet magnétocalorique du gadolinium : aspects thermodynamiques et optimisation pour la réfrigération magnétique

EL JAZOULY Yassine

2025-2026

- Introduction
- L'effet magnétocalorique EMC
- Mesure expérimentale de l'EMC du gadolinium
- Optimisation de l'EMC du Gadolinium avec un cycle AMRR
- Résultats : vers une réfrigération magnétique performante ?

Introduction

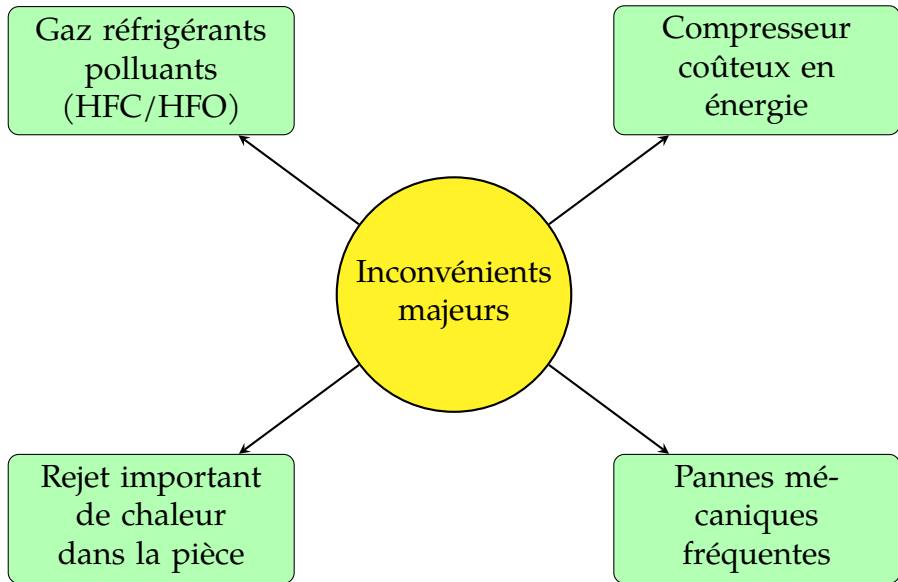


*Figure 2 : Réfrigérateur magnétique
(MAGNOTHERM)*



Figure 1 : Réfrigérateurs en supermarché

Inconvénients majeurs des réfrigérateurs classiques



Comment l'**effet magnétocalorique** du **gadolinium** peut-il être caractérisé, puis optimisé autour de sa température de Curie, notamment à l'aide de la modélisation d'un **cycle de réfrigération active (AMRR)** ?

L'effet magnétocalorique

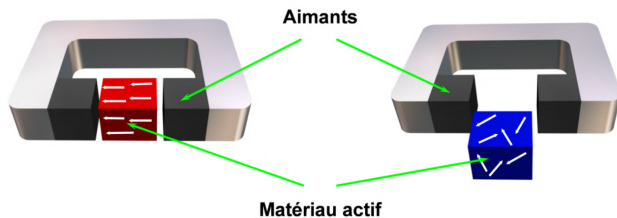


Figure 3 : Principe de l'effet magnétocalorique

L'effet magnétocalorique

On travaille dans des conditions :

- Réversible
- Adiabatique

L'entropie totale est la somme des contributions magnétique et thermique :

$$\Delta S_{\text{totale}} = 0$$

$$\implies \Delta S_{\text{magnétique}} + \Delta S_{\text{thermique}} = 0$$

$$\boxed{\Delta S_{\text{magnétique}} = -\Delta S_{\text{thermique}}}$$

Les matériaux à effet magnétocalorique

Matériaux	T _c (K)
Gd	294
La(Fe _{0.89} Si _{0.11}) ₁₃ H _{1.3}	291

Table: Propriétés magnétocaloriques de matériaux sélectionnés

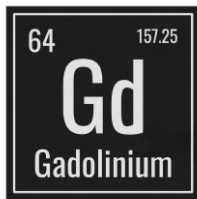


Figure 5 : Le Gadolinium



Figure 4 : Cube de Gadolinium

Traduction théorique de l'effet magnétocalorique

la relation de Maxwell : $\left(\frac{\partial S}{\partial B}\right)_T = \left(\frac{\partial M}{\partial T}\right)_B$

La capacité calorifique d'un matériau magnétique $C_B = T \left(\frac{\partial S}{\partial T}\right)_B$:

Variation de température adiabatique ΔT_{ad} :

$$\Delta T_{\text{ad}} = - \int_0^{B_1} \left(\frac{T}{C(T, B)} \right)_B \left(\frac{\partial M(T, B)}{\partial T} \right)_B dB \quad (1)$$

- $M(T, B)$: Aimantation du matériau [A · m]

Expérience



Expérience



Expérience : Mesure de l'effet magnetocalorique du Gadolinium (Montage)

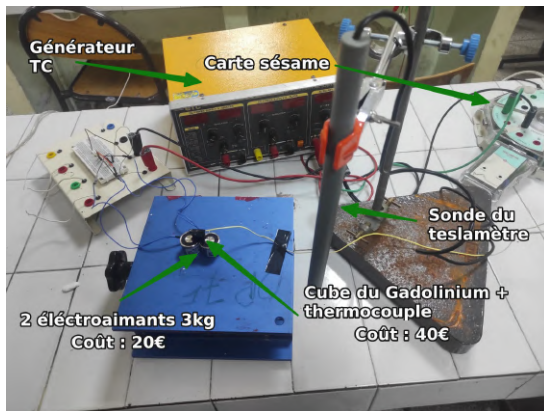


Figure 6 : Montage expérimentale

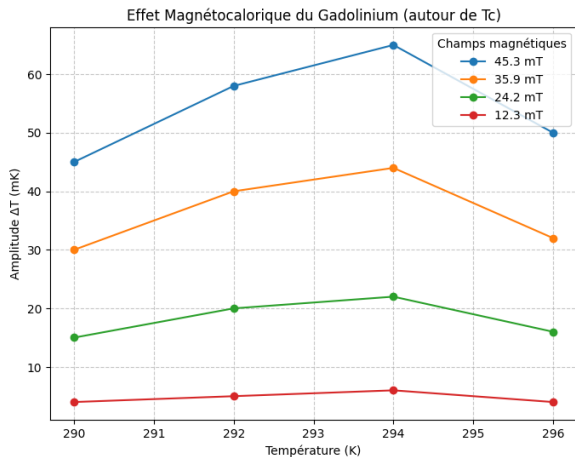


Figure 7 : Résultats expérimentaux

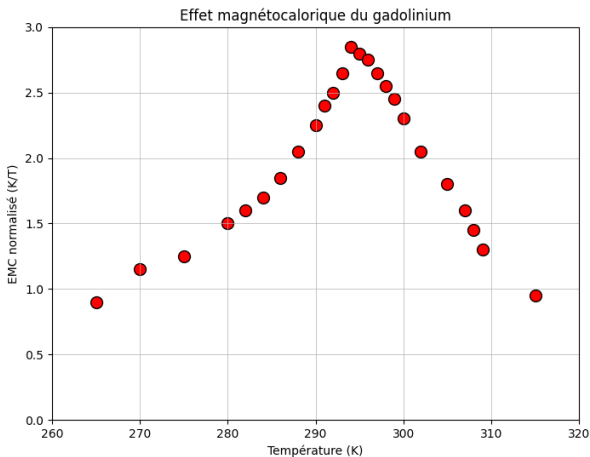
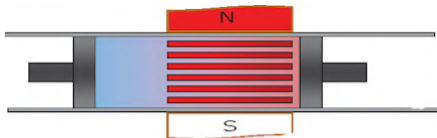


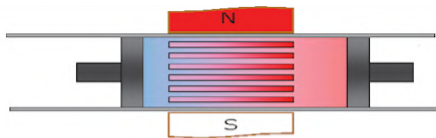
Figure 8 : Résultats expérimentaux de M.Balli

Principe du cycle AMRR



(a) Première Phase

L'aimantation du matériau entraîne une augmentation de la température



(b) Deuxième phase

Ecoulement du fluide de la source chaude vers la source froide.



(c) Troisième phase

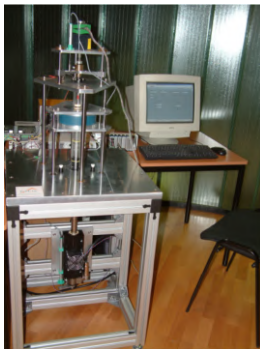
Désaimantation du matériau entraînant son refroidissement.



(d) Quatrième phase

Ecoulement du fluide de la source froide vers la source chaude.

Pourquoi la simulation plutôt que l'expérience ?



Prototype de Cédric Dupuis

- **Coût prohibitif des prototypes réels :**
 - Prototype industriel estimé entre 15 000 € et 35 000 €.
- **Réalité budgétaire (TIPE) :**
 - 2 électroaimants + gadolinium nous a coûté 60 €.

Hypothèses de simplification

- h indépendante des variations de la température.
- Conduction axiale négligée devant les autres transferts thermiques.
- EMC indépendante des variations de la température et vaut $1.75K$

Notations du modèle

- $\dot{D}(t)$: débit massique du fluide [$kg\ s^{-1}$]
- $C_f(T_f)$: capacité thermique massique du fluide [$J\ kg^{-1}K^{-1}$]
- $T_f(x, t)$: température du fluide [K]
- $T_m(x, t)$: température du matériau magnétocalorique [K]
- ρ_f : masse volumique du fluide [$kg\ m^{-3}$]
- S : section d'écoulement du canal fluide [m^2]
- L : longueur du régénérateur [m]
- h : coefficient d'échange thermique convectif [$W\ m^{-2}K^{-1}$]

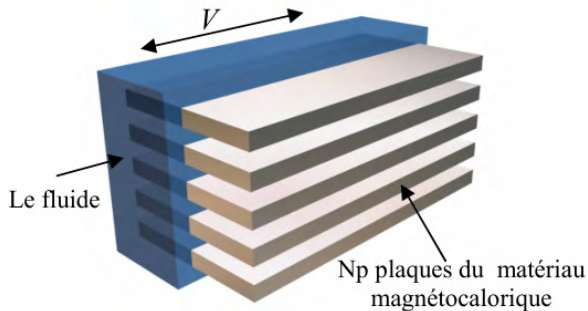
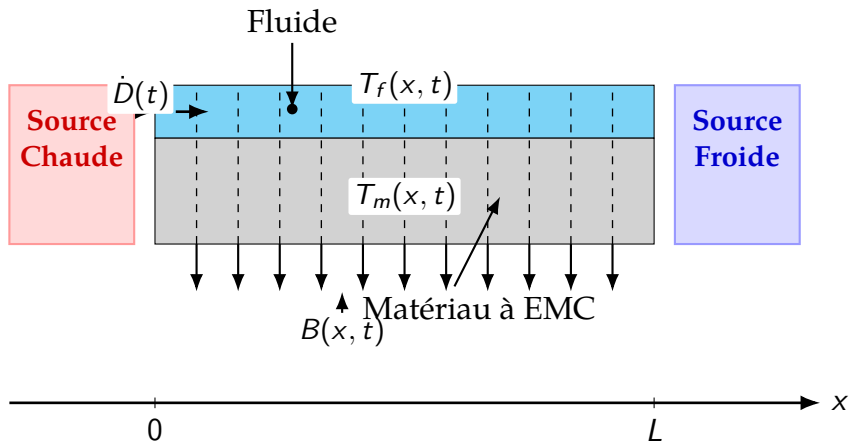
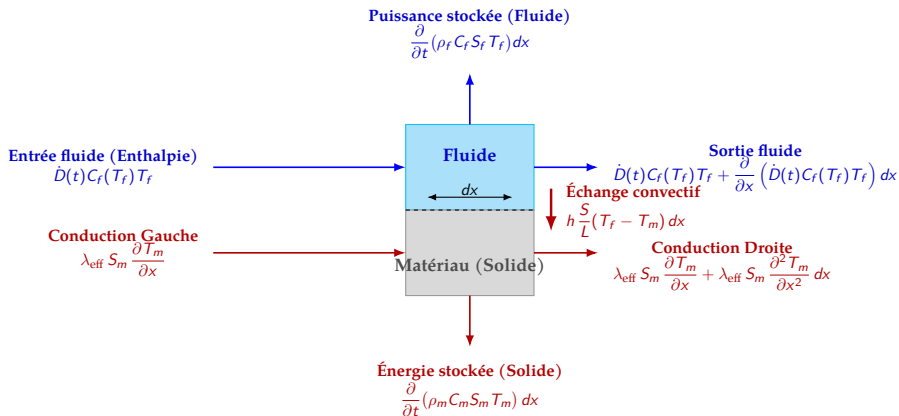


Figure 10 : Modèle d'un régénérateur magnétocalorique à plaques parallèles

Modèle numérique



Élément de référence du matériau magnétocalorique



Modèle numérique : Équations du bilan énergétique

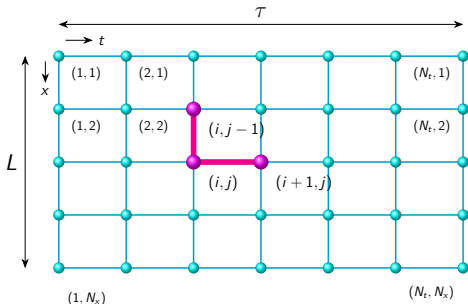
Le bilan énergétique nous donne ces deux équations :

$$m_f C_f \left(T_f \left(\frac{\partial T_f}{\partial t} + d(t) \frac{\partial T_f}{\partial x} \right) \right) = hS(T_m - T_f) \quad (2)$$

Avec le débit fictif : $d(t) = \frac{\dot{D}(t)}{\rho_f S} = d = \text{Constante}$

$$\rho_m C_m S \frac{\partial T_m}{\partial t} = h \frac{S}{L} (T_f - T_m) \quad (3)$$

Discrétisation spatio-temporelle



τ : durée totale de la simulation (période d'étude).

Maillage du domaine :

$$t_i = i \frac{\tau}{N_t}, \quad i \in \llbracket 1, N_t \rrbracket, \quad \Delta t = \frac{\tau}{N_t}$$

$$x_j = j \frac{L}{N_x}, \quad j \in \llbracket 1, N_x \rrbracket, \quad \Delta x = \frac{L}{N_x}$$

Approximation des dérivées (Euler) :

$$\frac{\partial T_f}{\partial t} \approx \frac{T_f^{i+1,j} - T_f^{i,j}}{\Delta t}$$

$$\frac{\partial T_f}{\partial x} \approx \frac{T_f^{i,j} - T_f^{i,j-1}}{\Delta x}$$

$$\frac{\partial T_m}{\partial t} \approx \frac{T_m^{i+1,j} - T_m^{i,j}}{\Delta t}$$

Modèle numérique : Système d'équations récurrentes

En remplaçant les dérivées dans le bilan, on obtient :

$$\begin{cases} T_f^{i+1,j} = A_{f1} T_f^{i,j} + A_{f2} T_f^{i,j-1} + A_{f3} T_m^{i,j} \\ T_m^{i+1,j} = A_{m1} T_m^{i,j} + A_{m2} T_f^{i,j} \end{cases}$$

Avec les coefficients constants associés :

$$A_{f1} = 1 - d \frac{\Delta t}{\Delta x} - \frac{hS}{m_f C_f} \Delta t, \quad A_{f2} = d \frac{\Delta t}{\Delta x}, \quad A_{f3} = \frac{hS}{m_f C_f} \Delta t$$

$$A_{m1} = 1 - \frac{hS}{m_m C_m} \Delta t, \quad A_{m2} = \frac{hS}{m_m C_m} \Delta t$$

Structure algorithmique complète du modèle numérique

Initialisation :

- ✓ Initialisation des paramètres géométriques, physiques.
- ✓ Les CI (Conditions Initiales).
- ✓ Calcul des paramètres intermédiaires (coefficient d'échange convectif, ...).

For $k=1:1:N_c$

Etape 1 : Aimantation ($\uparrow$ Température)

$$\begin{cases} T_{af}(i=1:j=1:N_x, k) = T_{af}(i=N_t, j=N_x: -1:1, k-1) \\ T_{af}(i=1:1:N_t, j=N_x, k) = T_{af}(i=1:1:N_t, j=N_x, k-1) \\ T_{am}(i=1:1:N_t, j=1, k) = T_{dm}(i=1:1:N_t, j=N_x, k-1) + EMC \\ T_{am}(i=1:1:N_x, k) = T_{dm}(i=N_t, j=N_x: -1:1, k-1) + EMC \end{cases}$$

Etape 3 : Désaimantation ($\downarrow$ Température)

$$\begin{cases} T_{df}(i=1, j=1:N_x, k) = T_{af}(i=N_t, j=N_x: -1:1, k) \\ T_{df}(i=1:1:N_t, j=N_x, k) = T_{af}(i=1:1:N_t, j=N_x, k) \\ T_{dm}(i=1:1:N_t, j=1, k) = T_{am}(i=1:1:N_t, j=N_x, k) - EMC \\ T_{dm}(i=j=1:1:N_x, k) = T_{am}(i=N_t, j=N_x: -1:1, k) - EMC \end{cases}$$

Etape 2 : Ecoulement Source Chaude $\rightarrow$ Source Froide

For $j=1:1:N_t$

For $i=1:1:N_x$

$$\begin{cases} T_{af}(i+1, j) = A_{r1} T_{af}(i, j) + A_{r2} T_{af}(i, j-1) + A_{r3} T_{am}(i, j) \\ T_{am}(i+1, j) = A_{m1} T_{am}(i, j) + A_{m2} T_{af}(i, j) \end{cases}$$

End

End

Etape 4 : Ecoulement Source Froide $\rightarrow$ Source Chaude

For $j=1:1:N_t - 1$

For $i=1:1:N_x$

$$\begin{cases} T_{df}(i+1, j) = A_{r1} T_{df}(i, j) + A_{r2} T_{df}(i, j-1) + A_{r3} T_{dm}(i, j) \\ T_{dm}(i+1, j) = A_{m1} T_{dm}(i, j) + A_{m2} T_{df}(i, j) \end{cases}$$

End

End

End

Exploitations

Table: Relevés des températures

Paramètre	Valeur min	Valeur max
T_{hot}	294.26 K	298.36 K
T_{cold}	288.65 K	293.03 K
ΔT	1.23 K	9.71 K
$\Delta T \text{ moyen} = 8.61 \text{ K}$		

Observations

- Écart maximal de **9.71 K** entre extrémités chaudes et froides
- Température moyenne de l'écart : **8.61 K**

Résultats : Validation du modèle par l'expérience

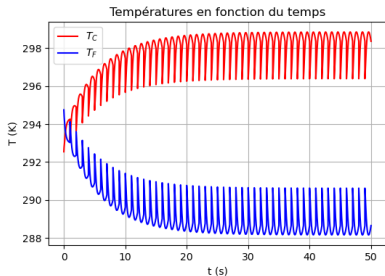


Figure 11 : Simulation

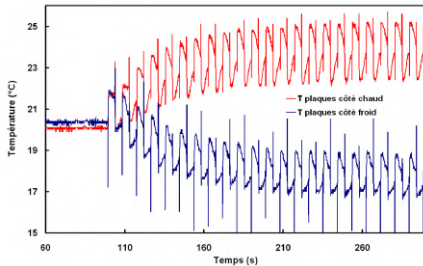


Figure 12 : Expérience (C. Dupuis)

Résultats

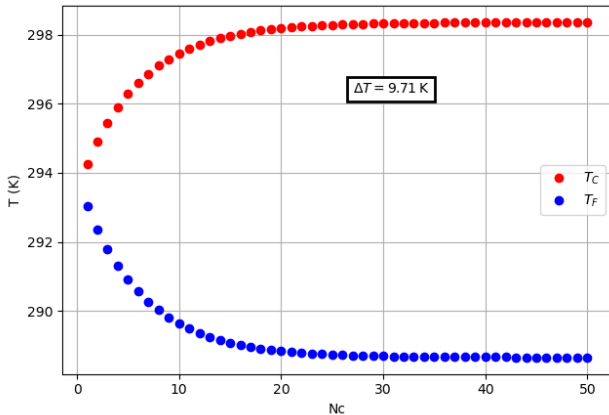


Figure 13 : Évolution de la température dans les deux côtés en fonction des cycles

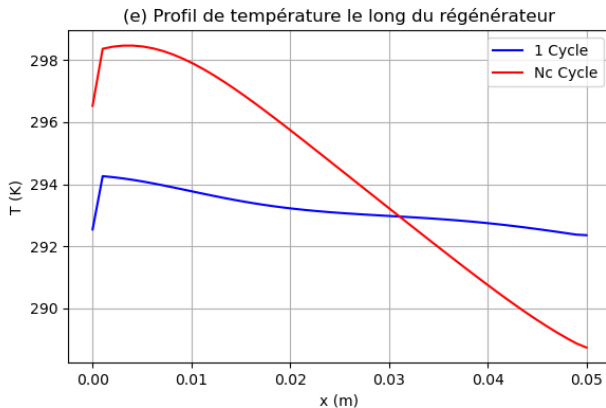


Figure 14 : Profil de température le long du régénérateur.

(d) Profils tridimensionnels de température

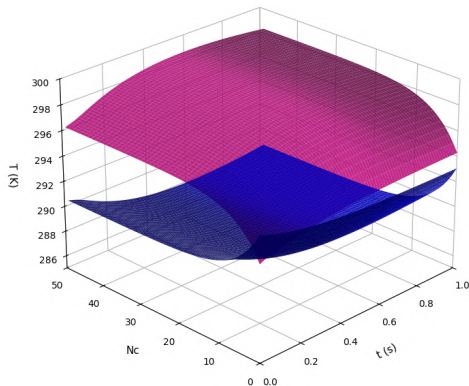


Figure 16 : Profils tridimensionnels de température.

Comparaison de performance : Effet de l'AMR

- Analyse de l'amplification du gradient thermique :

Configuration	Écart thermique (ΔT)
Effet magnétocalorique simple	1,75 K
Après régénération (AMR)	9,71 K

Conclusion

Merci pour votre attention !

Fin

Annexe : Démonstration de l'équation du fluide (2) I

- **Bilan d'énergie initial** sur un élément de référence dx :

$$\begin{aligned}\dot{D}(t)C_f(T_f)T_f &= \frac{\partial}{\partial t} \left(\rho_f C_f(T_f) S T_f \right) dx + \dot{D}(t)C_f(T_f)T_f \\ &+ \frac{\partial}{\partial x} \left(\dot{D}(t)C_f(T_f)T_f \right) dx + h(T_f - T_m) \frac{S}{L} dx\end{aligned}$$

- **Après simplification par dx et regroupement :**

$$\rho_f S \frac{\partial}{\partial t} \left(C_f(T_f) T_f \right) + \dot{D}(t) C_f(T_f) \frac{\partial T_f}{\partial x} = h \frac{S}{L} (T_m - T_f)$$

- **Hypothèse :** Terme de variation temporelle négligé ($T_f \frac{\partial C_f}{\partial t} \approx 0$).

Annexe : Démonstration de l'équation du fluide (2) II

- **Changement de variable** : Avec $d(t) = \frac{\dot{D}(t)}{\rho_f S}$, on obtient (2) :

$$m_f C_f \left(T_f \left(\frac{\partial T_f}{\partial t} + d(t) \frac{\partial T_f}{\partial x} \right) \right) = hS(T_m - T_f) \quad (2)$$

Annexe : Démonstration de l'équation du matériau (3)

- **Bilan d'énergie initial** (conduction axiale incluse) :

$$\frac{\partial}{\partial t} (\rho_m C_m S T_m) dx + \lambda_{\text{eff}} S \frac{\partial T_m}{\partial x} + \lambda_{\text{eff}} S \frac{\partial^2 T_m}{\partial x^2} dx = \lambda_{\text{eff}} S \frac{\partial T_m}{\partial x} + h(T_f - T_m) dx$$

- **Forme intermédiaire** (avec conductivité effective λ_{eff}) :

$$m_m C_m \frac{\partial T_m}{\partial t} + \lambda_{\text{eff}} \mathcal{V} \frac{\partial^2 T_m}{\partial x^2} = hS(T_f - T_m)$$

- **Hypothèse physique** : Conduction axiale négligée devant les autres transferts ($\lambda_{\text{eff}} \rightarrow 0$).
- **Résultat final** : On aboutit à la relation finale (3) :

$$\boxed{\rho_m C_m S \frac{\partial T_m}{\partial t} = h \frac{S}{L} (T_f - T_m)} \quad (3)$$

Annexe : Code Python de la Simulation AMRR (1/10)

```
1      import numpy as np
2      import matplotlib.pyplot as plt
3      from matplotlib import cm
4
5
6
7
8
9      def AMRR_coef_echange_ther_plaques(Lar, zb, Cf, landaf, rhof, v, mu_visc):
10         Dh = 4 * (Lar * zb) / (2 * (Lar + zb))
11
12         Re = (rhof * v * Dh) / mu_visc
13         Pr = (mu_visc * Cf) / landaf
14
15         if (0 < Re) and (Re < 4):
16             A, m = 0.891, 0.33
17         elif (4 <= Re) and (Re < 40):
18             A, m = 0.821, 0.385
19         elif (40 <= Re) and (Re < 4000):
20             A, m = 0.615, 0.466
```

Annexe : Code Python de la Simulation AMRR (2/10)

```
23     elif (4000 <= Re) and (Re < 40000):
24         A, m = 0.174, 0.618
25     else:
26         A, m = 0.024, 0.805
27
28     if Pr > 0.7:
29         Nu = 1.11 * A * (Re ** m) * (Pr ** 0.31)
30     else:
31         Nu = A * (Re ** m)
32
33     K = (landaf * Nu) / Dh
34     return K
35
36     # =====
37     # Exploitation
38     # =====
39     def AMRR_Exploitation(Tma, Tmd):
40         Nt, Nx, Nc = Tma.shape
41         T_hot = Tmd[Nt-1, 1, :]
42         T_cold = Tma[Nt-1, 1, :]
43         dT = T_hot - T_cold
```

Annexe : Code Python de la Simulation AMRR (3/10)

```
print("\nTEMPERATURES DES EXTREMITES :")
print(f" T_chaud (min/max): {T_hot.min():.2f} / {T_chaud.max():.2f} K")
print(f" T_froid (min/max): {T_froid.min():.2f} / {T_froid.max():.2f} K")
print(f" dT max: {dT.max():.2f} K")
print(f" dT min: {dT.min():.2f} K")
print(f" dT moy: {dT.mean():.2f} K")
return T_chaud, T_froid, dT

def run_AMRR(plot=True):
    index = 0
    q_ml_s = 5.0 # ml/s
    za, zb = 1e-3, 1.570e-4
    Lon, Lar = 50e-3, 0.573
    mu_visc, rhof, Cf, landaf = 1e-3, 1000.0, 4185.0, 0.59
    rhom, Cm = 7800.0, 240.0
    Nx, Nt, Nc = 50, 70, 50
    tp, T0, EMC = 1.0, 293.0, 1.75
    dx = Lon / (Nx - 1)
    dt = tp / (Nt - 1)
```

Annexe : Code Python de la Simulation AMRR (4/10)

```
69         v = (q_ml_s * 1e-6) / (Lar * zb)
70
71
72
73
74
75
76
77         Nc1, Eff = 50, 1.0
78         Taec = 292.0 * np.ones(Nc)
79         Taef = 294.0 * np.ones(Nc)
80         ds = Lar * dx
81         Mmat = rhom * za * Lar * dx
82         Mflui = rhof * zb * Lar * dx
83
84         K = AMRR_coef_echange_ther_plaques(Lar, zb, Cf, landaf, rhof, v, mu_visc)
85         Af1 = 1 - (K * ds / (Mflui * Cf) * dt + v * dt / dx)
86         Af2 = v * dt / dx
87         Af3 = K * ds / (Mflui * Cf) * dt
```

Annexe : Code Python de la Simulation AMRR (5/10)

```
92     Am1 = 1 - K * ds / (Mmat * Cm) * dt
93     Am2 = K * ds / (Mmat * Cm) * dt
94
95     Tma, Tfa = np.zeros((Nt, Nx, Nc)), np.zeros((Nt, Nx, Nc))
96     Tmd, Tfd = np.zeros((Nt, Nx, Nc)), np.zeros((Nt, Nx, Nc))
97     Tma[:, :, 0], Tfa[:, :, 0] = T0 + EMC, T0
98
99     for k in range(Nc):
100         if k > 0:
101             Tfa[:, 0, k] = Tfd[:, Nx-1, k-1]
102             Tma[:, 0, k] = Tmd[:, Nx-1, k-1] + EMC
103             Tma[:, 1, k] = Tmd[:, Nx-2, k-1] + EMC
104             Tfa[0, :, k] = Tfd[Nt-1, :-1, k-1]
105             Tma[0, :, k] = Tmd[Nt-1, :-1, k-1] + EMC
106
107         for i in range(Nt - 1):
108             for j in range(1, Nx):
109                 Tfa[i+1, j, k] = Af1*Tfa[i, j, k] + Af2*Tfa[i, j-1, k] + Af3*Tma[i, j, k]
```

Annexe : Code Python de la Simulation AMRR (6/10)

```
115     Tma[i+1, j, k] = Am1*Tma[i, j, k] + Am2*Tfa[i, j, k]
116
117     if k >= Nc1:
118         Tfa[:, Nx-1, k] = Tfa[:, Nx-1, k] + Eff * Qc0 / (Cf * Mflui)
119
120     Tfd[:, 0, k] = Tfa[:, Nx-1, k]
121     Tmd[:, 0, k] = Tma[:, Nx-1, k] - EMC
122     Tmd[:, 1, k] = Tma[:, Nx-2, k] - EMC
123     Tfd[0, :, k] = Tfa[Nt-1, :-1, k]
124     Tmd[0, :, k] = Tma[Nt-1, :-1, k] - EMC
125
126     for i in range(Nt - 1):
127         for j in range(1, Nx):
128             Tfd[i+1, j, k] = Af1*Tfd[i, j, k] + Af2*Tfd[i, j-1, k] + Af3*Tmd[i, j, k]
129             Tmd[i+1, j, k] = Am1*Tmd[i, j, k] + Am2*Tfd[i, j, k]
```

Annexe : Code Python de la Simulation AMRR (7/10)

```
138         if k >= Nc1:
139
140             T_chaud, T_froid, dT = AMRR_Exploitation(Tma, Tmd)
141             return Tma, Tmd, Tfa, Tfd
142
143         if __name__ == "__main__":
144             Tma, Tmd, Tfa, Tfd = run_AMRR(plot=False)
145
146         tp = 1.0
147         # ===== FIGURE (a) : T vs temps =====
148         Nt, Nx, Nc = Tma.shape
149         t_cycle = np.linspace(0, 1.0, Nt)
150         t_global, Tc_global, Tf_global = np.zeros(Nt * Nc), np.zeros(Nt * Nc), np.zeros(Nt * Nc)
151         for k in range(Nc):
152             t_global[k*Nt:(k+1)*Nt] = t_cycle + k*1.0
153             Tc_global[k*Nt:(k+1)*Nt] = Tmd[:, 1, k]
154             Tf_global[k*Nt:(k+1)*Nt] = Tma[:, 1, k]
```

Annexe : Code Python de la Simulation AMRR (8/10)

```
161 plt.figure(figsize=(6,4))
162 plt.plot(t_global, Tc_global, 'r', label=r'$T_C$')
163 plt.plot(t_global, Tf_global, 'b', label=r'$T_F$')
164 plt.xlabel('t (s)'); plt.ylabel('T (K)'); plt.legend(); plt.grid(True); plt.show()
165
166 # ===== FIGURE (b) : Tc rouge, Tf bleu =====
167 Tc, Tf = Tmd[Nt-1, 1, :], Tma[Nt-1, 1, :]
168 dT = Tc - Tf
169 cycles = np.arange(1, Nc+1)
170
171 plt.figure(figsize=(6.5,4.5))
172 plt.plot(cycles, Tc, 'o', color='red', label=r'$T_C$', linestyle='None')
173 plt.plot(cycles, Tf, 'o', color='blue', label=r'$T_F$', linestyle='None')
174 plt.xlabel('Nc'); plt.ylabel('T (K)'); plt.grid(True)
175 dT_ss = float(dT[-1])
176 plt.text(int(0.55 * Nc), 0.5*(Tc.min()+Tc.max()), rf'$\Delta T = \{dT\_ss:.2f\}\, \mathrm{{K}}$',
177         bbox=dict(facecolor='white', edgecolor='black', linewidth=2))
178 plt.legend(loc='center right'); plt.tight_layout(); plt.show()
```

Annexe : Code Python de la Simulation AMRR (9/10)

```
184
185     # ===== FIGURE (e) : profil spatial =====
186     x = np.linspace(0, 50e-3, Tma.shape[1])
187     plt.figure(figsize=(6,4))
188     plt.plot(x, Tmd[-1, :, 0], color='blue', label='1 Cycle')
```

Annexe : Code Python de la Simulation AMRR (10/10)

```
207 plt.plot(x, Tmd[-1, :, -1], color='red', label='Nc Cycle')
208 plt.xlabel('x (m)'); plt.ylabel('T (K)'); plt.legend(); plt.grid(True); plt.show()
209
210 # =====
211 # ===== FIGURE (d) : Graphique 3D des enveloppes (t, Nc) =====
212 # =====
213 t_cycle_3d = np.linspace(0, tp, Nt)
214 cycles_3d = np.arange(0, Nc)
215 T_mesh, Nc_mesh = np.meshgrid(t_cycle_3d, cycles_3d)
216 Z_hot, Z_cold = Tmd[:, 1, :].T, Tma[:, 1, :].T
217
218 fig = plt.figure(figsize=(8, 7))
219 ax = fig.add_subplot(111, projection='3d')
220 ax.xaxis.set_panel_color((1.0, 1.0, 1.0, 0.0))
221 ax.yaxis.set_panel_color((1.0, 1.0, 1.0, 0.0))
222 ax.zaxis.set_panel_color((1.0, 1.0, 1.0, 0.0))
223 surf_hot = ax.plot_surface(T_mesh, Nc_mesh, Z_hot, color='mediumvioletred', edgecolor='
    none', alpha=0.9, shade=True)
224 surf_cold = ax.plot_surface(T_mesh, Nc_mesh, Z_cold, color='mediumblue', edgecolor='none
    ', alpha=0.9, shade=True)
225 ax.set_xlabel('t (s)'); ax.set_ylabel('Nc'); ax.set_zlabel('T (K)')
226 ax.set_xlim(0, tp); ax.set_ylim(0, Nc); ax.set_zlim(285, 300); ax.view_init(elev=25,
    azim=-135)
227 plt.title('(d) Profils tridimensionnels de temperature', fontsize=12); plt.show()
```

Figure 7

```
1 import matplotlib.pyplot as plt
2 temps = [290, 292, 294, 296]
3 data = {
4     45.3: [45, 58, 65, 50],
5     35.9: [30, 40, 44, 32],
6     24.2: [15, 20, 22, 16],
7     12.3: [4, 5, 6, 4]
8 }
9
10 plt.figure(figsize=(8, 6))
11
12 for champ, valeurs in data.items():
13     plt.plot(temps, valeurs, marker='o', label=f'{champ} mT')
14
15 plt.title('Effet Magnétocalorique du Gadolinium (autour de Tc)')
16 plt.xlabel('Température (K)')
17 plt.ylabel('Amplitude  $\Delta T$  (mK)')
18 plt.legend(title="Champs magnétiques")
19 plt.grid(True, linestyle='--', alpha=0.7)
20 plt.show()
```



Figure 8

```
1 import matplotlib.pyplot as plt
2
3 T = [265, 270, 275, 280, 282, 284, 286, 288, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 302, 305, 307, 308, 309, 315]
4
5 EMC = [0.9, 1.15, 1.25, 1.5, 1.6, 1.7, 1.85, 2.05, 2.25, 2.4, 2.5, 2.65, 2.85, 2.8, 2.75, 2.65, 2.55, 2.45, 2.3, 2.05, 1.8, 1.6, 1.45, 1.3, 0.95]
6
7 plt.figure(figsize=(8, 6))
8
9
10 plt.scatter(T, EMC, color='red', edgecolors='black', s=100)
11
12 |
13 plt.xlim(260, 320)
14 plt.ylim(0, 3)
15 plt.xlabel("Température (K)")
16 plt.ylabel("EMC normalisé (K/T)")
17 plt.title("Effet magnétocalorique du gadolinium")
18 plt.grid(True, which='both', linestyle='-', linewidth=0.5)
19
20 plt.show()
```