Analyse Numérique : Devoir N°3

Sasha Broux - 5920 1700

25 Novembre 2020

1 Le système

Dans le cadre de l'étude d'un puits canadien nous souhaitons modéliser l'opérateur de diffusion thermique sur une grille de différences finies, notre domaine d'étude possède trois conditions limites de Neumann et une de Dirichlet (le domaine précis est repris dans l'énoncé). Nous avons donc l'équation suivante en chaque point du domaine sauf sur les bords du puits où nous utilisons la moyenne des diffusivités thermiques D de chaque milieu.

$$\frac{T_{i,j}^t - T_{i,j}^{t-1}}{\Delta t} - D(\frac{T_{i+1,j}^t - 2T_{i,j}^t + T_{i-1,j}^t}{(\Delta x)^2} + \frac{T_{i,j+1}^t - 2T_{i,j}^t + T_{i,j-1}^t}{(\Delta y)^2}) = 0$$

$$\frac{T_{i,j}^t}{\Delta t} - D(\frac{T_{i+1,j}^t - 2T_{i,j}^t + T_{i-1,j}^t}{(\Delta x)^2} + \frac{T_{i,j+1}^t - 2T_{i,j}^t + T_{i,j-1}^t}{(\Delta y)^2}) = \frac{T_{i,j}^{t-1}}{\Delta t}$$

Chaque neoud étant sur une condition limite de Dirichlet ne sera pas compté comme inconnue mais sera remplacé par sa valeur dans les équations, les neouds (M, j) ont donc pour équation :

$$\frac{T_{i,j}^t}{\Delta t} - D(\frac{-2T_{i,j}^t + T_{i-1,j}^t}{(\Delta x)^2} + \frac{T_{i,j+1}^t - 2T_{i,j}^t + T_{i,j-1}^t}{(\Delta y)^2}) = \frac{T_{i,j}^{t-1}}{\Delta t} + D\frac{f(t)}{(\Delta x)^2}$$

Deux paramètres physiques importants sont les diffusivités thermique du sol et de l'air et du sol qui seront notés $D_s = 0.25 \times 10^{-6}$ et $D_a = 20 \times 10^{-6}$ respectivement.
Nous allons utiliser la méthode des gradients conjugués préconditionné (ou non) afin de résoudre ce système d'équations pour chaque pas de temps. Dans le cas d'une lecture par ligne du schéma, ces équations donnent lieu à une matrice A symétrique et bande de la forme suivante :

$$A = \begin{array}{c} \\ 0 \\ 1 \\ \vdots \\ \vdots \\ \vdots \\ M_L \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ 2M_L \\ \vdots \\ \vdots \end{array} \begin{array}{c} \begin{array}{ccccccccccccc} 0 & 1 & \cdots & \cdots & \cdots & M_L & \cdots & \cdots & \cdots & \cdots & 2M_L & \cdots & \cdots \end{array} \\ \begin{bmatrix} D_1 & \alpha & & & & \beta & & & & & & & \\ \alpha & D_3 & \ddots & & & & \ddots & & & & & & \\ & \ddots & \ddots & \ddots & & & & \ddots & & & & & \\ & & \ddots & D_3 & \alpha & & & & \ddots & & & & \\ & & & \alpha & D_1 & 0 & & & & \ddots & & & \\ \beta & & & & 0 & D_2 & \alpha & & & & \ddots & & \\ & \ddots & & & & \alpha & 2D_1 & \ddots & & & & \ddots & \\ & & \ddots & & & & \ddots & \ddots & \ddots & & & & \ddots \\ & & & \ddots & & & & \ddots & 2D_1 & \alpha & & & \\ & & & & \ddots & & & & \alpha & D_2 & 0 & & \\ & & & & & \ddots & & & & 0 & D_2 & \alpha & \\ & & & & & & \ddots & & & & \alpha & 2D_1 & \ddots \\ & & & & & & & \ddots & & & & \ddots & \ddots \end{bmatrix} \end{array}$$

$$\alpha = -\frac{D}{(\Delta x)^2}\ ,\ \beta = -\frac{D}{(\Delta y)^2}\ ,\ M_L = 2M + 1$$

$$D_1 = \frac{1}{\Delta t} + D(\frac{1}{(\Delta x)^2} + \frac{1}{(\Delta y)^2})\ ,\ D_2 = \frac{1}{\Delta t} + D(\frac{1}{(\Delta x)^2} + \frac{2}{(\Delta y)^2})\ ,\ D_3 = \frac{1}{\Delta t} + D(\frac{2}{(\Delta x)^2} + \frac{1}{(\Delta y)^2})$$

De manière pratique cette matrice est stockée au format `COO`, et, afin d'éviter le coût $\mathcal{O}(nnz \log(nnz))$ du tri et le potentiel $\mathcal{O}(nnz^2)$ de la suppression des doublons, j'ai décidé de la remplir dans l'ordre.

2 Théorie

2.1 Résidus de GMRES

Partons du principe de l'algorithme GMRES ; à l'itération n de l'algorithme nous approximons la solution exacte x^* par le vecteur $x_n \in \mathcal{K}_n \subseteq \mathbb{C}^m$ qui minimise les résidus r_n relatifs à x_n:

$$||r_n(x_n)|| = ||r_n|| = ||b - Ax_n|| \qquad A \in \mathbb{C}^{m\times m} \qquad r_n\ ,\ b\ ,\ x_n \in \mathbb{C}^m \tag{1}$$

Où $\mathcal{K}_n$ est le sous espace de Krylov $\langle b, Ab, A^2b, \ldots, A^{n-1}b\rangle$, nous pouvons donc utiliser la matrice de Krylov associée $K_n = (b, Ab, A^2b, \ldots, A^{n-1}b) \in \mathbb{C}^{m\times n}$ et reformuler x_n dans (1), en effet x_n est une combinaison linéaire des vecteurs formant une base de $\mathcal{K}_n$:

$$x_n \in \mathcal{K}_n \Longleftrightarrow x_n = K_n c \qquad \text{Et donc dans (1)} \qquad ||r_n|| = ||b - AK_n c||$$

L'algorithme GMRES utilise l'itération d'Arnoldi afin de construire une base orthonormée $\{q_1, q_2, \ldots, q_n\}$ de $\mathcal{K}_n$, en démarrant avec $q_1 = \frac{b}{||b||}$, la matrice associée est $Q_n = (q_1, q_2, \ldots, q_n) \in \mathbb{C}^{m\times n}$. Nous avons donc $K_n c = Q_n y$:

$$||r_n|| = ||b - AQ_n y||$$

L'itération d'Arnoldi construit la matrice Q, de la forme de Hessenberg (QHQ^*) de A, colonne par colonne en utilisant $AQ_n = Q_{n+1}\tilde{H}_n$, où $\tilde{H}_n \in \mathbb{R}^{(n+1)\times n}$ est la partie supérieure gauche de H. Nous pouvons donc écrire :

$$||r_n|| = ||b - Q_{n+1}\tilde{H}_n y||$$

Ensuite, remarquons que $b\ ,\ Q_{n+1}\tilde{H}_n y \in \langle q_1, q_2, \ldots, q_{n+1}\rangle$ (étant donné que $q_1 = \frac{b}{||b||}$). Nous pouvons donc multiplier à gauche par Q^*_{n+1}, qui est orthonormée, sans modifier la valeur de la norme :

$$||r_n|| = ||Q^*_{n+1}b - \tilde{H}_n y|| \tag{2}$$

Finalement, $Q^*_{n+1}b$ revient à $||b||e_1$ où e_1 est le vecteur $[1, 0, \ldots, 0] \in \mathbb{C}^{n+1}$. Afin de voir cela il faut se rappeler que non seulement Q_{n+1} est orthonormée mais aussi que dans le cadre de GMRES nous avons $q_1 = \frac{b}{||b||}$. En remplaçant dans (2), et en inversant le sens du vecteur dans la norme, nous obtenons l'expression des résidus recherchée :

$$Q^*_{n+1}b = \begin{bmatrix} \frac{b^*}{||b||} \\ q_2^* \\ \vdots \\ q_{n+1}^* \end{bmatrix} b = \begin{bmatrix} \frac{1}{||b||}b^*b = ||b|| \\ 0 \\ \vdots \\ 0 \end{bmatrix} \Longleftrightarrow ||r_n|| = ||\tilde{H}_n y - ||b||e_1||$$

2.2 Avantage des formats creux

Algorithm 38.1. Conjugate Gradient (CG) Iteration

$x_0 = 0,\ r_0 = b,\ p_0 = r_0$
for $n = 1, 2, 3, \ldots$

$\alpha_n = (r_{n-1}^T r_{n-1})/(p_{n-1}^T A p_{n-1})$	step length
$x_n = x_{n-1} + \alpha_n p_{n-1}$	approximate solution
$r_n = r_{n-1} - \alpha_n A p_{n-1}$	residual
$\beta_n = (r_n^T r_n)/(r_{n-1}^T r_{n-1})$	improvement this step
$p_n = r_n + \beta_n p_{n-1}$	search direction

Figure 1: Algorithme des gradients conjugués du livre de référence [1]

Un stockage creux a pour objectif de ne pas utiliser d'espace mémoire pour les entrées nulles, cela a pour effet principal de réduire drastiquement l'espace mémoire utilisé. Le deuxième effet est la possibilité de n'effectuer que

les opérations nécessaires, ce qui réduit la complexité temporelle de certains algorithmes si l'implémentation prend avantage de la structure de stockage utilisée.

Dans le cas de l'algorithme des gradients conjugués (1) nous remarquons que seul un produit matrice-vecteur est présent par itération : Ap_{n-1}. L'algorithme réalisant un produit matrice vecteur est en général assez simple et court, il prend très facilement avantage de la structure et a donc une complexité temporelle en $\Theta(nnz)$. Plus précisément, dans le cas d'un stockage au format `COO`, le pseudo code d'un produit $Av = R$ est le suivant :

```
for (i,j,k) in nnz(A) {   R[i] += k * v[j];   }
```

En suivant la notion de *flops* du livre (n'importe quelle opération en virgule flottante compte pour un *flops*), et pour une matrice $M \times M$ comportant nnz éléments non nuls, cet algorithme est en $\sim 2nnz$ *flops*. Dans le cas d'un stockage plein un produit matrice vecteur a besoin de $M^2 + M(M-1) = 2M^2 - M$ *flops*. Partons du principe que cette matrice comporte k diagonales relativement pleines et proches de la diagonale principale, alors $nnz \simeq kM$. Nous pouvons donc dire que l'économie par itération des gradients conjugués sera de l'ordre de $2M^2 - M - 2nnz = 2M^2 - M - 2kM$ *flops*. À l'itération n, nous aurons donc économisé $\sim n(2M^2 - M - 2kM) = nM(2M - 1 - 2k)$ *flops*.

Dans notre cas spécifique nous avons $A \in \mathbb{R}^{M(2M+1)\times M(2M+1)}$ et $k = 5$, si nous prenons la discrétisation la plus basse de l'énoncé, soit $M = 20$, nous économiserons environ $2 \times 820^2 - 820 - 10 \times 820 = 1\,335\,780$ *flops* par itération, soit environ 99.39% des *flops* économisés par rapport au stockage plein.

3 Pratique

3.1 Critère d'arrêt

Il nous faut trouver une condition d'arrêt de l'algorithme des gradients conjugués, utiliser la norme des résidus r_n peut être attractif, mais cela ne sera qu'une condition absolue, et dépendante du problème, ce qui ne garanti pas de précision pour la solution trouvée. Un autre critère d'arrêt plus général serait alors d'utiliser la vitesse de convergence. En cas de non convergence l'algorithme s'arrêtera après $M/3$ itérations si la matrice du système est de taille $M \times M$.

La vitesse de convergence n'est plus spécifique au problème et est :

$$\frac{||r_n||}{||r_0||} = \mu_n$$

L'algorithme s'arrête donc lorsque μ_n est plus petit qu'une précision prédéfinie.
La figure 2 montre l'évolution de cette vitesse de convergence à chaque itération, spécifiquement pour cette expérience la précision choisie est de 10^{-15} donc très proche de l'$\epsilon_{\text{machine}}$. ous n'utilisons pas la norme $||\cdot||_A$ mais bien la norme $||\cdot||_2$ pour des raisons pratiques. Ce critère d'arrêt reste valide étant donné que ce qui est petit pour la norme $||\cdot||_A$ restera petit pour la norme $||\cdot||_2$. L'une des conséquences de l'utilisation de la norme 2 est la perte de la monotonie de la convergence des résidus, le théorème 38.2 du livre [1] prédisait une convergence monotone mais seulement avec la norme A.
Le préconditionnement a l'effet attendu de converger beaucoup plus vite, cependant les deux méthode `ILU(0)` et `IC(0)` ont quasiment la même vitesse de convergence (différence indiscernable sur la fig.2). Pour notre problème il est donc plus intéressant d'utiliser le préconditionneur `IC(0)` de par son nombre de *flops* plus faible.
Pour les autres simulations les gradients conjugués (préconditionnés) s'arrêtent lorsque $\mu_n < 10^{-12}$, cette borne a été choisie en fonction du script de test reçu.

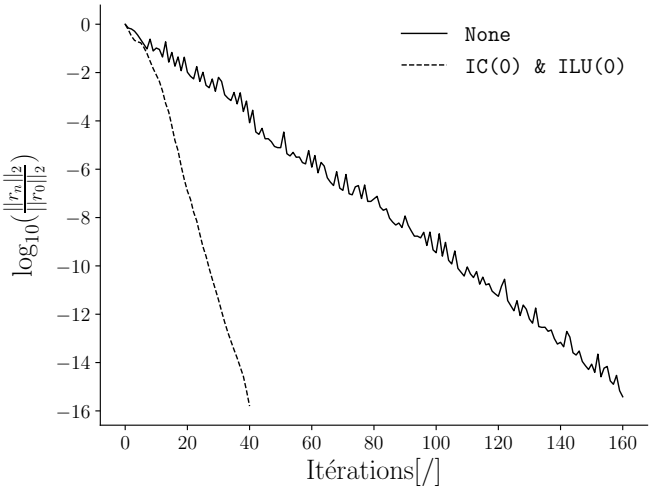

Figure 2: Vitesse de convergence $\frac{||r_n||_2}{||r_0||_2}$ à chaque itération pour $M = N = 20$, $P = 1$ an.

3.2 Régime établi

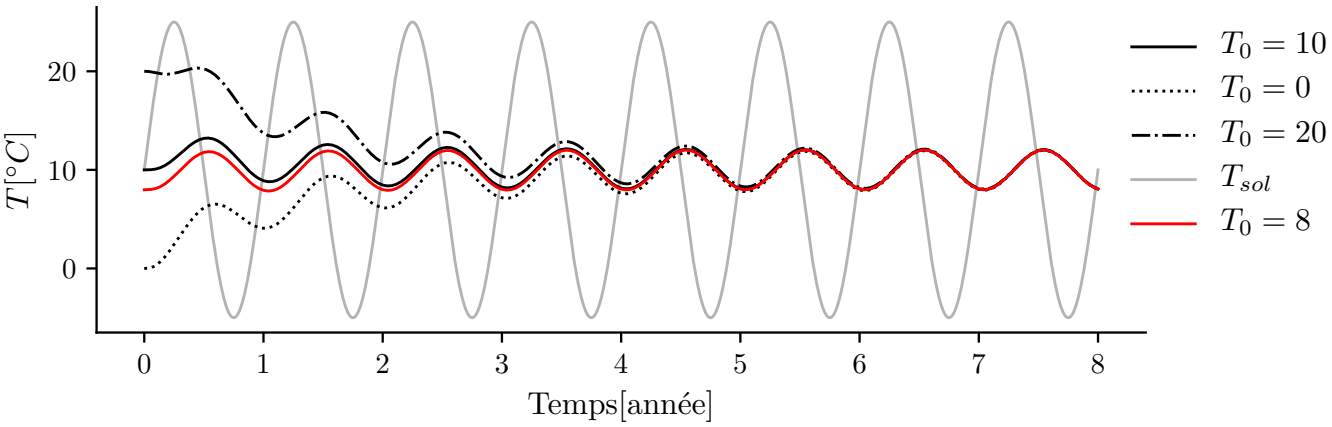

Figure 3: Évolution de la température au centre de la conduite pour $M = 20$, $N = 52$ et $P = 1$ an.

La figure 3 montre l'évolution de la température au centre de la conduite. La courbe rouge représente le régime établi, il est atteint en approximativement 4 périodes d'une année si nous partons de la solution initiale $T(x, y, t = 0) = T^0 = f(0) = 10°C$. Ce régime a pour températures maximum et minimum $\sim 12.01°C$ et $\sim 7.98°C$ respectivement. Le nombre de périodes nécessaires pour atteindre ce régime dépend de la solution initiale T^0, la figure 3 montre bien cela en comparant plusieurs solutions initiales, en utilisant donc $T^0 = 20°C$ le régime est atteint en approximativement 6 périodes. Tandis qu'en choisissant un $T^0 = 8°C$ proche de la température minimum en régime, il sera atteint en moins d'une période.

Dans le cas d'une période journalière la température va osciller mais avec une amplitude bien trop petite, nous pouvons nous y attendre en calculant la longueur $\delta = \sqrt{\frac{D_s P}{\pi}} = 8.29[cm]$. Notre conduite étant enterrée à 3 mètres de profondeur, les oscillations de température en surface s'atténueront bien trop que pour faire osciller de manière significative la température de la conduite. Cependant, il est important de noter que la température de la conduite va converger vers la température moyenne des oscillations en surface. Cela est présenté dans la figure 4, malgré des températures initiales différentes la température va converger asymptotiquement vers $14°C$, qui est la température moyenne en surface. Ces courbes peuvent être vues comme des oscillations d'amplitude infinitésimales convergeant vers un régime de manière similaire à la figure 3.

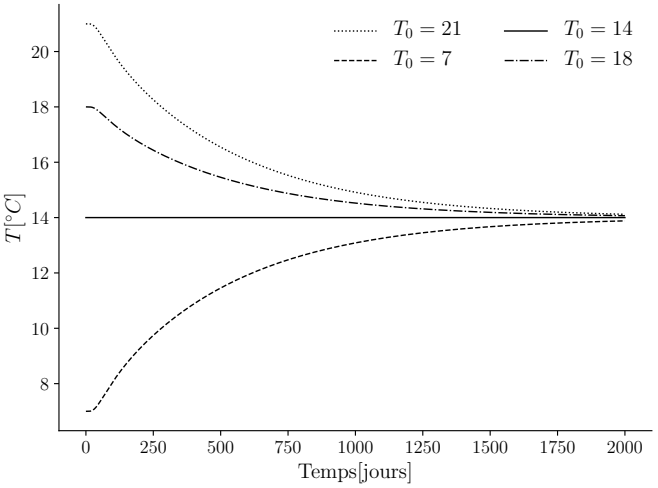

Figure 4: Évolution de la température au centre de la conduite pour $M = 20$, $N = 24$ et $P = 1$ jour.

3.3 Discrétisation

3.3.1 Variations journalières

Étudier l'efficacité η, en fonction de paramètres de discrétisation, durant l'alternance jour-nuit ne nous permettra pas de conclure. Nous avons vu à la section précédente que le régime pour l'alternance jour-nuit correspond à tendre vers une température moyenne, et donc qu'en régime nous ayons $T_{\max} \simeq T_{\min}$ dans la conduite, l'efficacité η sera donc toujours proche de zéro. De plus le temps caractéristique du problème $\tau = \frac{H_{cond}^2}{D_{sol}}$, en prenant $H_{cond} = 3[m]$, vaut un peu plus d'1 an. Étudier l'efficacité durant des variations journalières paraît donc superflu d'autant plus que nous ne prenons pas en compte la variation annuelle en compte en faisant cela.

3.3.2 Variations annuelles

Afin de rester cohérent avec l'énoncé chaque simulation utilisera $T^0 = f(0)$, ce choix implique de simuler au moins 4 ans étant donné qu'il faudra atteindre le régime. En effet, étudier l'efficacité durant une période transitoire, qui dans la réalité serait suite à l'installation du puits canadien, n'a que peu d'utilité pratique, il me paraît plus utile d'étudier l'efficacité en régime. Afin d'avoir un régime établi depuis plusieurs périodes les simulations de cette section sont sur 8 années et l'efficacité est calculée sur la dernière période (relative à P) de ces huit années, donc la dernière année.

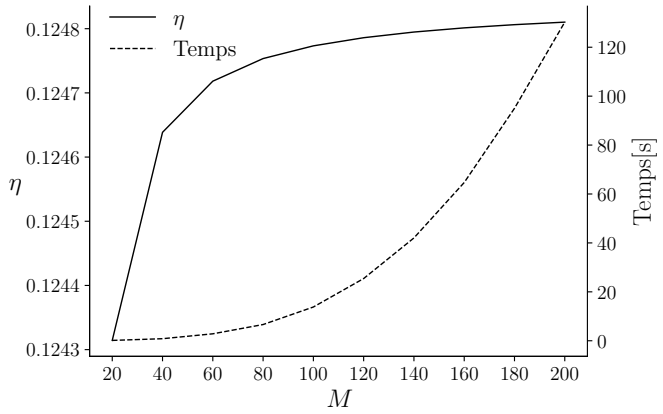

Figure 5: η en fonction de M pour $N = 20$, $P = 1$ an et $nb = 8$.

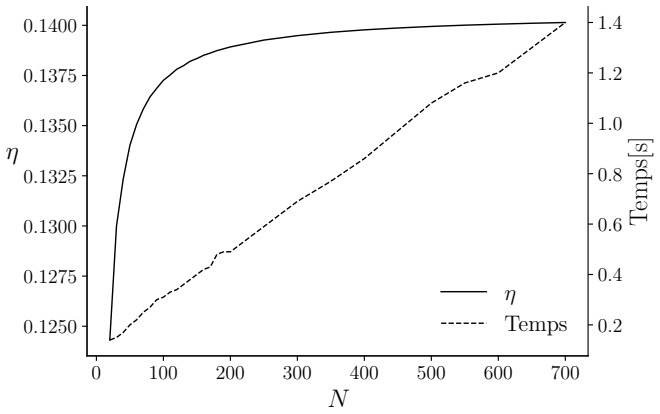

Figure 6: η en fonction de N pour $M = 20$, $P = 1$ an et $nb = 8$.

Les figures 5 et 6 présentent l'évolution de l'efficacité η en fonction des paramètres de discrétisation M et N. Les deux graphes ont été obtenus en utilisant : `./devoir3 0 8 [M] [N] 1 file.out` où `[M]` et `[N]` varient, lorsque `[M]` varie le paramètre `[N]` est fixé à 20 et inversement. La discrétisation temporelle minimum choisie est de 20, en effet le temps caractéristique $\tau = \frac{3^2}{0.25\times10^{-6}}[s]$ valant à peine plus d'un an nous souhaitons avoir au moins 20 points par longueur d'onde du phénomène étudié. Rappelons que, dans notre cas, nous sommes efficace lorsque η est proche de 0.

Dans les deux cas, il est facile de remarquer qu'augmenter la discrétisation (M ou N) augmentera la valeur η, cela est dû à la précision accrue qui donne lieu à des maxima et minima plus précis (ce qui augmente η). Les deux graphes tendent asymptotiquement vers une valeur η qui semble être la valeur la plus précise que nous pouvons obtenir.Notre but sera donc de l'approcher tout en gardant un temps d'exécution raisonnable. La figure 5 montre une croissance du temps quadratique avec M, nous pouvons estimer cela grâce aux données et un doubling ratio test : pour $M = 40$ l'algorithme prend 0.84 secondes tandis que pour $M = 80$ il prend 6.63 secondes, autrement dit lorsque l'on double M le temps est multiplié par 7.89. De plus, en comparant avec la figure 6 l'augmentation de η est faible, augmenter M afin d'améliorer la précision de η ne semble pas être une bonne idée. Cependant, si nous souhaitons observer les variations de température en surface (ou juste observer de manière plus précise le champ de température) il est alors intéressant d'utiliser une valeur de M plus élevée.

La figure 6 nous montre de meilleurs résultats, augmenter N augmente la précision de la valeur η bien plus tout en évitant une croissance quadratique du temps. Il faut cependant noter qu'augmenter M et N va améliorer la précision de manière différente, la réduction du pas de temps permet de mieux capter les maxima dans le temps tandis que diminuer le pas spatial donnera une valeur plus spécifique relative au centre de la conduite et non à une approximation de l'ensemble de la conduite.

Pour l'étude de l'efficacité de la conduite les paramètres de discrétisation suivant sont appropriés : $M = [20, 60]$ et $N = [100, 400]$. En utilisant $M = 20$, $N = 365$ et $nb = 8$ nous pouvons obtenir une efficacité $\eta = 0.1397$ en 0.8 secondes.

3.4 Simulations finales

La figure 3 présente déjà la température au centre de la conduite et la température au sol, malgré la différence entre les paramètres de discrétisation ce graphe est tout à fait valide, il sera impossible de faire la différence sur ce graphe entre les anciennes et nouvelles valeurs de régime.
Pour des variations annuelles j'ai donc choisi d'utiliser $M = 60$ et $N = 365$ afin d'estimer au mieux l'efficacité du

régime. Sur une simulation de 10 ans les températures et efficacité en régime et au centre de la conduite sont les suivantes :

$$T_{max} = 12.0994°C \qquad T_{min} = 7.9035°C \qquad \eta = 0.13986$$

Une augmentation de la diffusivité thermique du sol va faire augmenter η, les températures extérieures mettrons moins de temps à diffuser jusqu'au conduit étant donné que τ va diminuer. Le conduit se verra donc plus affecté par les variations extérieures.

Pour la simulation annuelle un GIF de l'évolution de la température durant la dernière période est disponible au lien suivant : `https://i.imgur.com/JWK62Nl.mp4` . Ci-dessous deux extraits du GIF montrant le champ de température suite à une période chaude et durant une période froide :

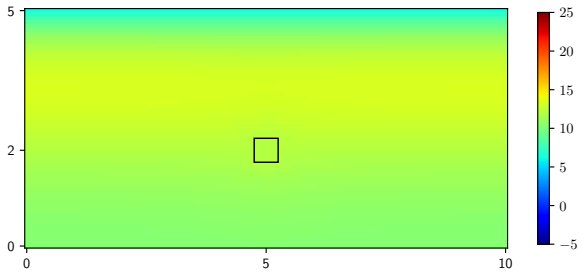

Figure 7: Champ de température suite à une période chaude

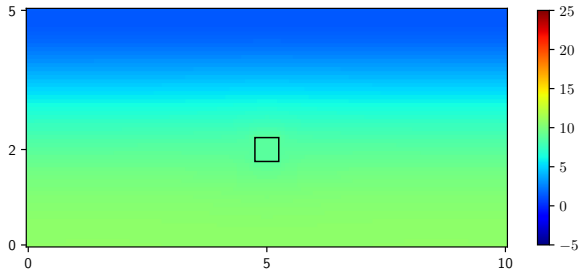

Figure 8: Champ de température durant une période froide

Sur la figure 7 nous pouvons observer l'effet de l'inertie thermique du sol : les couches en surface sont froides alors que le sous-sol est encore chaud suite à la période chaude. Augmenter la valeur D_s diminuera cet effet, ce même effet est celui qui maintient la température du puits dans un régime dont l'amplitude est bien moindre que l'amplitude des variations extérieures. Cette vague de chaleur va se diffuser dans le sol tout en s'atténuant.
La figure 8 montre une situation durant l'hiver, la conduite a alors une chaleur légèrement plus élevée que son entourage. Cela est visible par le léger halo vert autour de celui-ci, tandis que le sol à cette profondeur est de couleur déjà bleu clair.

Malheureusement je n'ai pas de GIF pour les variations jour-nuit, cependant le comportement peut être prédit en utilisant le temps caractéristique τ et la longueur $\delta == \sqrt{\frac{D_s P}{\pi}} = 8.29[cm]$, le conduit sera trop profond que pour observer des changements de température sur de courtes périodes. Sur le long terme la température du conduit convergera vers la température moyenne des oscillations extérieures.

3.5 Programme

Le programme possède quelques spécificités à mentionner, tout d'abord chaque fonction utilisée prend avantage du stockage creux, principalement les fonctions `ILU,IC,ILUSolve,ICSolve,dcooemv` avec `dcooemv` la fonction réalisant un produit matrice-vecteur. Les fonctions sont donc soit en $\mathcal{O}(nnz)$ pour `ILUSolve,ICSolve,dcooemv` soit en $\mathcal{O}(nnz * log(nnz))$ pour `ILU,IC`. Une optimisation possible mais non implémentée pour `ILU` et `IC` serait de réaliser le `bsearch` sur un tableau réduit en fonction de l'itération.
Comme mentionné au début de ce rapport, la matrice A est créée de manière séquentielle afin d'éviter le tri et la compression en $\mathcal{O}(nnz * log(nnz))$ et $\mathcal{O}(nnz^2)$ respectivement.
Le programme s'utilise et fonctionne comme requis par l'énoncé, un argument supplémentaire `-d` peut être donné afin d'obtenir des valeurs de débuggage.

References

[1] Lloyd N. Trefethen, David Bau III, *Numerical Linear Algebra*