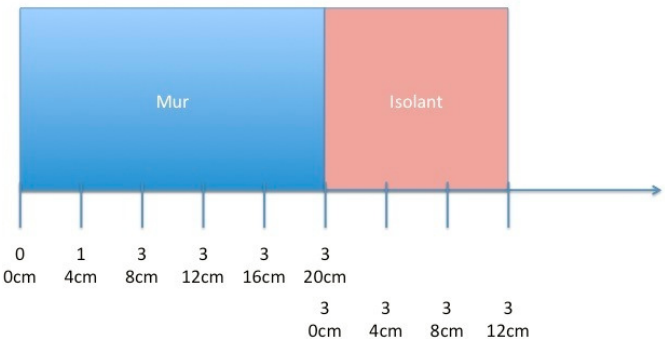
Programmation Concurrente

Durée indicative : 12 heures dont la moitié uniquement seront faites lors des séances de TD.
Travail à faire seul.

Objectifs

Dans ce projet, vous allez programmer un simulateur qui utilise un modèle très simple du phénomène de transfert de chaleur par conduction et calculer l'évolution de la température sur la face interne d'une cloison en fonction de l'évolution de la température de sa face externe . Bien évidemment, nous prendrons quelques hypothèses simplificatrices :

- le mur est composé de béton sur une épaisseur de 20 cm et de laine de verre sur une épaisseur de 12 cm, soit une épaisseur de 32 cm
- à t = 0s, l'ensemble est à l'équilibre à une température de T_0
- la face externe du mur, soumise à l'action du soleil voit sa température évoluer selon la fonction : $T_{(0,t)} = T_1 + B \sin \omega t, t > 0$ ou $T_{(0,t)} = 2 \times T_1, t > 0$
- les flux de convection sur les deux faces du mur seront ignorés
- les constantes auront les valeurs suivantes :
  - T_0 = 20°C
  - T_1 = 55°C
  - B = 40
  - $\omega = \frac{86400}{2\Pi}$



De la formule magique à la méthode de calcul

La conduction thermique est un transfert thermique spontané d'une région de température élevée vers une région de température plus basse est décrite par la loi dite de Fourier établie mathématiquement par Jean-Baptiste Biot en 1804 puis expérimentalement par Fourier en 18224 : la densité de flux de chaleur est proportionnelle au gradient de température i.e. plus l'écart de température est important, plus l'échange est important. Par conséquent, lorsque deux objets en contact ont des températures différentes, un transfert de chaleur sera produit de l'objet avec la température la plus haute vers l'objet avec la température la plus basse : un objet sera « refroidi » et l'autre sera « réchauffé ». Un modèle simplifié du phénomène de transfert de chaleur par conduction consiste donc à supposer qu'une partie de la température d'un corps sera émise aux corps voisins et perdu par ce premier. Ce modèle fait donc abstraction du phénomène de rayonnement d'énergie d'un corps et de la résistance thermique des matériels. Il n'y a aucune perte d'énergie.

Un bilan d'énergie et l'expression de la loi de Fourier dans le cas ou l'énergie produite au sein même du matériau est nulle conduit à l'équation générale de conduction de la chaleur dans un corps homogène par

$$\rho c \frac{\delta T}{\delta t} = \lambda \left(\frac{\delta^2 T}{\delta x^2} + \frac{\delta^2 T}{\delta y^2} + \frac{\delta^2 T}{\delta z^2} \right)$$ avec :

- λ est la conductivité thermique du matériau en W.m-1.K-1 (laine de verre : 0,04, brique : 0,84, granite : 2,2).
- ρ est la masse volumique du matériau en kg.m-3 (laine de verre : 30, brique : 1.400, granite : 2.700),
- c est la chaleur spécifique massique du matériau en J.kg-1.K-1 (laine de verre : 900, brique : 840, granite : 790).

Malheureusement, il n'est pas facile à l'aide d'une telle formule de calculer la température du mur en tout point et en tout temps même si dans notre cas le problème étant unidimentionnel, il est possible de simplifier la formule en $\rho c \frac{\delta T}{\delta t} = \lambda \left(\frac{\delta^2 T}{\delta x^2} \right)$. La solution analytique de cette équation reste inaccessible et il est donc nécessaire de chercher une solution approchée par une méthode numérique, par exemple en utilisant la méthode des éléments finis. La méthode des différences finies est une méthode très utilisée en analyse numérique car la discrétisation des opérateurs de dérivations sont assez triviaux et que le schéma numérique converge bien, en revanche c'est un modèle assez lent car le pas de temps doit être petit.

Petit rappel, soit une fonction f de la variable réelle x. On pose δ une quantité supposée « petite ». Alors les dérivées premières et secondes discrète de f en un point x est la fonction :

- Dérivée première : $\frac{\delta f}{\delta x}(x) = \left(\frac{f(x+\delta) - f(x)}{\delta} \right)$
- Dérivée seconde : $\frac{\delta^2 f}{\delta x^2}(x) = \left(\frac{f(x+\delta) + f(x-\delta) - 2f(x)}{\delta^2} \right)$

Lorsque $\delta \rightarrow 0$, on retrouve les dérivée usuelle, définie par la valeur limite du rapport ci-dessus.

On peut donc en déduire l'évolution de la température en fonction du temps et de la position :

$$T_{(x,t+\varepsilon)} = T_{(x,t)} + \frac{\lambda \varepsilon}{\rho c} \left(\frac{f(x+\delta) + f(x-\delta) - 2f(x)}{\delta^2} \right)$$

Comme nous pouvons voir dans l'équation précédente, la température d'un corps est influencée par les rayonnements des corps en x-δ et x+δ. Si la température du corps en x à l'instant T=t est égal à T(x,t), alors la température de ce même corps une unité de temps plus tard sera T(x,t+1) = T(x,t) + C*(T(x+δ,t) + T(x-δ,t) - 2T(x,t)), ou C représente la fraction de degrés perdu par rayonnement.

Bien évidemment, deux corps ont un comportement différents :

- celui qui représente la face externe soumise à l'action du soleil : sa température est stable,
- celui qui représente la face interne : sa temparature bouge mais elle ne dépend pas de son environnement (ce qui est faux dans la vrai vie : il faudrait prendre en compte le flux de rayonnement de la chaleur).

Pour que l'étude soit complète, il faut aussi déterminer le "pas" (i.e. la valeur de δ puisque le pas de temps est déjà fixé à 1 seconde ou minutes) permettant de garantir que l'algorithme converge : plus le pas est petit, plus les calculs sont justes (on approxime mieux les dérivées) mais plus les pas sont petits, plus le nombre de calcul et donc le temps d'exécution augmente. Pour ce TD, je vous propose de prendre un pas dans l'espace de 2 cm. La convergence est assurée puisque la température de la face au soleil ne bouge pas et que l'ensemble va s'équilibrer à terme à cette température.

Ce modèle reste donc très simple et idéal car il suppose, entre autres, une répartition homogène de la température sur la surface des corps en contact, et une isolation parfaite de ces 2 corps (aucun échange thermique n'est pris en compte entre les 2 corps en contact et l'environnement). Je vous laisse néanmoins le soin de déterminer la formule pour un corps non homogène comme le mur composée de deux parties : la première en brique ou granit et la seconde d'un isolant.

Ce que vous devez faire
Dans ce projet, vous devez créer plusieurs versions du simulateur de transfert de chaleur par conduction. Chaque version aura sa propre date de rendu :

1. Le premier simulateur est une programmation séquentielle en Java du simulateur qui doit vous permettre de préciser votre modèle, calculer la constante C, valider votre implémentation.
2. La seconde simulateur est une programmation parallèle en Java de votre simulateur dans lequel à chaque élément x de l'objet est associé un Thread. La communication entre Thread sera réalisée par des variables partagées et la synchronisation entre celles-ci, par rendez-vous.
3. Le troisième simulateur est une programmation parallèle en Java composé de N Threads (1 par coeur du processeur de votre machine). La synchronisation entre les Threads sera réalisée par une barrière.
4. Le quatrième simulateur est une programmation parallèle en C composé de N Threads Posix (1 par coeur du processeur de votre machine). La communication et synchronisation entre Thread sera réalisée un modèle "producteur-consommateur" de taille 1.

Les sorties de vos programmes

Vos Threads calculant en parallèle, il faut bien évidemment mettre un peu d'ordre dans les "sorties".

Je vous propose dans un premier temps d'utiliser un serveur de visualisation disponible à l'adresse http://erebe-vm15.i3s.unice.fr/visulabprogconc

- Le simulateur lui envoie les valeurs à afficher à l'aide d'une websocket et on récupère la visualisation dans une fenêtre d'un navigateur
- L'ordre d'écriture dans la websocket n'a pas d'importance, le serveur de visualisation remet les données dans l'ordre
- La manière de procéder est la suivante :
    - Ecriture du code
        - Créer dans votre programme une websocket et l'initialiser : `JavaWebSocketServer.getInstance();// Init the server`
        - Ajouter un sleep pour avoir le temps de faire le copier/coller demander à l'exécution : `Thread.sleep(10000);`
        - Ecrire dans la websocket les valeurs à visualiser selon le format : `<elt><time> + time + </time><X> + x + </X><value> + value + </value></elt>`
    - Ne pas faire plus de 200 itérations avec la visualisation
    - Compilation du programme
        - Compiler votre programme en le liant au deux jar suivant : ProgConcWebSocketServer.jar et webbit-0.4.15-full.jar
    - Exécution
        - Ouvrir votre navigateur web à l'URL : http://erebe-vm15.i3s.unice.fr/visulabprogconc/
        - Lancer votre programme. Copier l'URL donnée (chez moi l'URL est ws://pc21.home:9900/progconc)
        - Coller cette URL dans la page de votre naviteur et clicker sur le bouton "Connect to ProgConc server"

Pour mesurer le temps d'exécution de 100.000 cycles de simulation, il faut bien évidemment supprimer les différentes sorties qui perturbent inutilement le fonctionnement du simulateur.

Ce que vous devez rendre :

Dans le cadre de ce projet, vous devez rendre un fichier PDF de nom progi-<login>.pdf (i est la version du programme réalisé) par simulateur en détaillant :

1. La continuation de la propagation de chaleur de t=0 à t=10. Si les calculs se font en double, vous ne donnerez dans le rendu que les valeurs entière. Pour la face du mur exposé au soleil, on prendra la fonction simple i.e :
    - pour t=0, les valeurs à afficher sont : 110,20,20,20,20,20-20,20,20,20
    - la partie avant le "-" représente les températures du mur et la partie après le "-" représente les températures de l'isolant.
2. Une description en pseudo code de chacun des algorithmes implémentés
3. Le temps d'exécution pour 100.000 cycles de simulation afin de mesurer le temps d'exécution selon les solutions. Il est évident que pour mesurer le temps d'excution toute trace d'affichage doit être retirée. Vous devez expliquez dans votre rendu comment vous avez mesurer le temps d'exécution.
4. Le numéro de l'itération qui voit la face interne du mur (celle qui ne reçoit pas le soleil) changer de température (calcul du temps de propagation de la chaleur).
5. Le code de chacun des algorithmes implémentés sous la forme d'une archive tar gzip de nom progi-<login>.tgz se décompressant dans un répertoire de format : progi-<login>
6. De manière spécifique :
    - pour la 1ère version, votre calcul des différentes constantes. Au fait, combien faut-il de pas d'itération pour simuler une année complète ?
    - pour la 2ème version, une modélisation en FSP du modèle de synchronisation
    - pour la 3ème version, faire une comparaison avec les précédentes solutions. Evidemment, vous garderez pour les 3 étapes du projet les mêmes constantes y compris si vous vous étiez initialement trompés dans le calcul de celles-ci
    - pour la 4ème version, une modélisation en FSP du modèle de synchronisation

Quelques notes sur la méthode d'évaluation
A prendre en compte pour obtenir la meilleure note possible.

- L'utilisation de code Java est interdite dans le document PDF.
- Tout projet contenant uniquement le code du programme Java aura la note 0.
- Tout rendu comportant un nom de fichier pdf ou tgz, ou ne se décompressant pas par gunzip / tar ne sera pas corrigé.
- Des erreurs de compilation donneront lieu à une invalidation de votre programme Java.
- Une amélioration du modèle donné ici est un plus. Cependant, cette amélioration devra être clairement expliquée, en partant des équations jusqu'au pseudo-code.
- Le piratage de code peut être puni sévèrement
- L'absence des explications sur le schéma de synchronisation mis en oeuvre entre les Threads entraine automatiquement une annulation du code Java.
- Dans ce projet, l'utilisation correcte du langage de programmation Java est importante. De ce fait, une fois la compilation faite, les points suivants seront jugés dans votre programme (Important : Si l'un des points suivants n'est pas satisfait par votre programme, l'évaluation de votre projet s'arrête) :
    - Compilation sans erreur.
    - Exécution sans erreur. Votre programme doit être libre d'erreurs qui arrêtent son exécution.
    - Utilisation de Threads.

- Utilisation des outils de synchronisation prévu.
- Aspect divers dans la programmation : modularité, respect de la syntaxe, commentaire.
- Résultats de la simulation