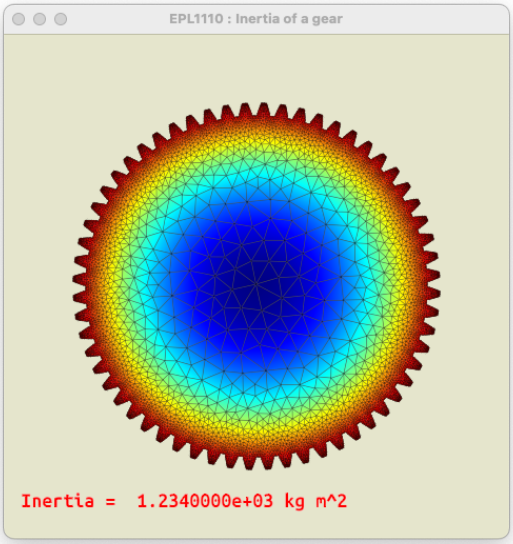
Finite elements for dummies : Quel est le moment d'inertie d'un engrenage ?



Considérons un engrenage défini par un maillage de triangles dont les coordonnées des sommets sont fournies en millimètres. L'engrenage a une épaisseur de dix centimètres et est centré à l'origine. Le moment d'inertie pour l'axe de rotation traversant l'engrenage en son centre de gravité s'obtient par :

$$I = \int\int\int \rho(x^2 + y^2)\, dx\, dy\, dz$$

où ρ est la masse volumique de l'acier.

Pour effectuer cette intégration, on utilisera une règle d'integration numérique de Hammer dont les poids et points d'intégration sont inclus dans une structure de données `femIntegration`.

```
typedef struct
    int n;
    const double *xsi;
    const double *eta;
    const double *weight;
 femIntegration;
```

Les maillages sont fournis dans des fichiers qui ont la structure suivante :

```
Number of nodes 4
     0 :  5.0000000e-01  0.0000000e+00
     1 :  1.0000000e+00  0.0000000e+00
     2 :  5.0000000e-01  5.0000000e-01
     3 :  1.0000000e+00  5.0000000e-01
Number of triangles 2
     1 :      0      3      2
     2 :      0      1      3
```

Mais dans le programme, on utilisera la structure de données `femMesh` définie comme suit :

```
typedef struct
    int *elem;        = [ 0  3  2  0  1  3 ]
    double *X;        = [ 0.5  1.0  0.5  1.0 ]
    double *Y;        = [ 0.0  0.0  0.5  0.5 ]
    int nElem;        = 2
    int nNode;        = 4
    int nLocalNode;   = 3
 femMesh;
```

Notre exemple sera donc stocké sous la forme de deux vecteurs de taille 4 et d'un vecteur de taille 6 :-) Le tableau est donc stocké sous la forme d'un vecteur !

Pour effectuer ce devoir, il faut écrire quatre fonctions. Voici ce qui est fourni par nos bons soins :

- le fichier `homework.c` est une version vide du devoir à réaliser. Les quatre fonctions sont définies mais ne font pas vraiment ce qui est prévu.
- Le fichier `fem.c` contient une série de fonctions de base. Quelques autres fonctions pourraient aussi vous aider. Ces fonctions seront disponibles lors de la correction et il est donc permis de les utiliser. Il n'est toutefois pas permis de les modifier[1]
- Les fichiers `gear8.txt`, `gear12.txt`, `gear16.txt` et `gear60.txt` donnent quelques engrenages dont on voudrait évaluer le moment d'inertie autour de son axe de rotation. On vous fournit aussi le fichier du maillage des notes de cours `stupid.txt` qui peut être bien utile pour mettre au point votre programme.
- Le fichier `glfem.c` est une petite librairie pour visualiser facilement vos résultats en temps réel, sans devoir passer par un autre programme. Il est possible d'utiliser la librairie graphique sur le serveur qui contient le même fichier `main.c` que celui que vous avez reçu.

Plus précisément, on vous demande de :

1. Tout d'abord, écrire une fonction

   ```
   double inertiaSteelRho();
   ```

   qui fournit simplement la masse volume de l'acier (en kg/m^3). Il n'y a pas vraiment grand chose à programmer : il suffit juste de trouver la masse volumique de l'acier ! Une erreur relative de 10% sera admise pour valider votre devoir ! Comme dirait mon informaticien favori, Google est votre ami...

2. Ensuite, écrire une fonction

   ```
   femMesh *inertiaMeshRead(const char *filename);
   ```

   qui lit un maillage se trouvant dans le fichier `fileName` et alloue dynamiquement la mémoire requise pour la structure de données. Comme l'équipe didactique est vraiment gentille, on a effectué une partie du travail pour vous, la fonction fournie ouvre le fichier, alloue les deux vecteurs pour les coordonnées des sommets. Ensuite, il lit les coordonnées dans le fichier et les stocke dans les deux vecteurs.

   Il vous reste donc à lire le tableau d'appartenance et à le stocker dans une vecteur que vous aurez allouer préalablement. La partie délicate à implémenter consiste à utiliser judicieusement les fonctions `malloc` et `scanf` du langage C. A nouveau, soyez astucieux et pour comprendre comment cela fonctionne, il suffit d'utiliser l'information disponible sur le web : oui, Google est vraiment ton pote :-) La première partie de la fonction peut évidemment vous inspirer pour écrire la partie manquante ! Evidemment, tel que fourni, le code va exploser tant que cette étape préliminaire n'est pas effectuée.

3. Ecrire une fonction

   ```
   double inertiaIntegrate(femMesh *theMesh, femIntegration *theRule, double rho);
   ```

   qui calcule le moment d'inertie pour un engrenage. Les arguments de cette fonction sont les structures de données qui contiennent la maillage `theMesh`, et la règle d'intégration `theRule` et le masse volumique `rho`. Il faut évidemment pas faire appel à la première fonction dans votre calcul, mais utiliser la valeur fournie dans les arguments de la fonction !

[1] Si vraiment vous voulez modifier ces fonctions, il faut en créer une nouvelle fonction avec un nom différent et inclure la nouvelle version comme une fonction distincte dans votre fichier de soumission.

Attention, nous utiliserons une autre règle d'intégration dans le correcteur automatique : il convient donc de ne pas définir directement les poids et points d'intégration dans votre fonction ! Au passage, on fera de même pour le masse volumique en utilisant celle du plastique, ou celle du cuivre ou même celle de l'or pour notre ami Trump.

4. Finalement, écrire une fonction

```
void inertiaMeshFree(femMesh *theMesh)
```

qui désalloue avec l'instruction `free` la mémoire allouée dans la fonction `inertiaMeshRead`. Oui, oui, : pour savoir ce que fait l'instruction `free`, il suffit d'être un peu astucieux et de chercher. Votre code semblera fonctionner parfaitement si vous n'écrivez rien dans cette fonction, mais vous observerez que le contrôle de la mémoire effectué avec `valgrind` s'effectue de manière incorrecte. Dans un programme C, il est essentiel de bien contrôler l'allocation dynamique de la mémoire et de bien libérer celle-ci à la fin de votre code : il faut donc lire bien attentivement les diagnostics de votre soumission pour voir si tout cela se passe parfaitement.

5. Un morceau de code `main.c` vous est fourni pour tester vos quatre fonctions.

```
#include "fem.h"

femMesh         *inertiaMeshRead(const char *filename);
double           inertiaSteelRho();
double           inertiaIntegrate(femMesh *theMesh, femIntegration *theRule, double rho);
void             inertiaMeshFree(femMesh *theMesh);

int main(void)
{
    femIntegration* theRule = femIntegrationCreate(3, FEM_TRIANGLE);
    femMesh *theMesh = inertiaMeshRead("data/gear60.txt");
    double rho = inertiaSteelRho();
    double   I = inertiaIntegrate(theMesh,theRule,rho);
    printf("Inertia = %14.7e kg m^2",I);

    femIntegrationFree(theRule);
    inertiaMeshFree(theMesh);
    exit(0);
}
```

6. Vos quatre fonctions seront incluses dans un unique fichier `homework.c`, sans y adjoindre le programme de test fourni ! Ce fichier devra être soumis via le web et la correction sera effectuée automatiquement. Il est donc indispensable de respecter strictement la signature des fonctions. Votre code devra être strictement conforme au langage `C` et il est fortement conseillé de bien vérifier que la compilation s'exécute correctement sur le serveur.

7. Il est impératif qu'aucune des fonctions n'arrête le flux du programme, quel que soit le problème rencontré. En cas d'arguments incompatibles, vous pouvez imprimer un message d'erreur et renvoyer une valeur nulle si nécessaire.

8. Attention, des avertissements (`warning`) à la compilation sur le serveur affecteront négativement votre évaluation.

9. Afin de pouvoir effectuer un test distinct de vos fonctions, des instructions de compilation ont été mis dans le fichier `homework.c`. Il est IMPERATIF de ne pas les retirer, ni de les modifier...