

Tuto compilation C sur Mac pour Eléments Finis

DE MOL Maxime

2 février 2013

1 Introduction

Salut à vous, tous les macoïdes qui galérez (ou allez galérer dans un futur proche) avec le problème 1 en éléments finis! Si tu es sur Windows, je vais te dire la même chose que nous a dis Olivier Bonaventure, "Tant pis pour toi"! Si tu es sur Linux, certains éléments vont être semblable, d'autre pas du tout, mais moi je m'en fout, je suis pas sur Linux. Et si enfin, tu es sur Mac, ce guide est pour toi!



Back to the future.

2 The easy way : Xcode

Comme vous l'indique le titre de la section, il y a plusieurs façon de développer (= écrire des programmes barbants dans des langages bizarres) sur Mac. Personnellement j'en connais 2. La première, et la plus simple, consiste en l'utilisation de Xcode, ce programme développé par Apple, disponible seulement sur Mac, va vous assister énormément dans l'écriture de vos petites missions! Le modus operandi est très simple.

2.1 Step 1 - Start



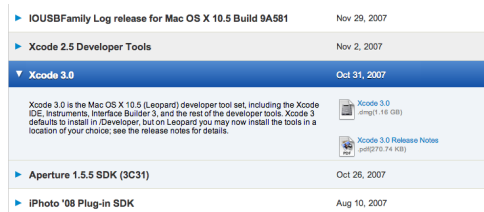
Direction le Mac App Store (si vous n'avez pas de Mac App Store sur votre Mac, c'est que probablement votre mac est trop vieux pour le dernier Xcode, donc passez directement à l'étape 2-bis).

2.2 Step 2 - Get Xcode



Simple. Cherchez pour "Xcode". Trouvez. Téléchargez. Done !

2.3 Step 2 - bis



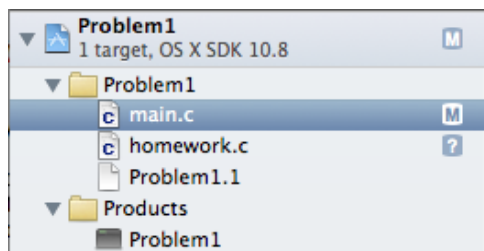
Votre Mac est vieux, vous tournez sur Leopard, ou plus ancien ? Il vous faut Xcode 3 ! Là, c'est plus compliqué que pour vos collègues plus moderne. Vous vous rendez ici : <https://developer.apple.com/downloads/index.action>. Vous devez créer un compte développeur, et une fois que vous avez accès à tous les ressources, naviguez vers la page 8, localisez Xcode 3, téléchargez, installez !

2.4 Step 3 - New Project



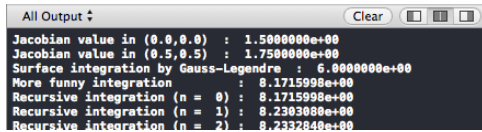
Vous devez créer un nouveau projet pour pouvoir commencer à travailler votre code ! Pour cela, dans la page d'accueil de Xcode, vous choisissez "Create a new Xcode project" » OSX » Application » Command Line Tool. Là vous donnez quelques détails, et choisissez bien dans Type : C. C'est important ! Reste plus qu'à choisir la destination de votre dossier projet, and it's done !

2.5 Step 4 - The Files



Normalement, une fois le projet créé, vous allez avoir 2 fichiers dans l'arborescence de votre projet : un main.c, et un nom_du_projet.1. Vous ne vous occupez pas du *.1, et supprimez le main.c, pour le remplacer avec le main.c fourni par Legat (simplement glisser-lâcher). Vous ajoutez aussi le homework.c dans l'arborescence (en générale, supprimer tous les *.c qu'il y a, et mettre ceux donnés par le prof). Vous pouvez coder !

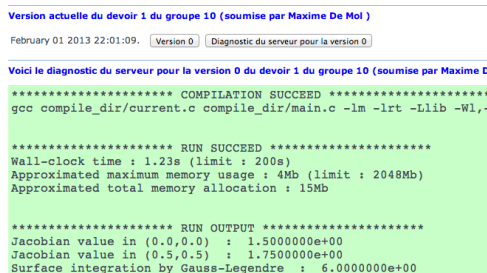
2.6 Step 5 - Execute



```
All Output
Clear
Jacobian value in (0.0,0.0) : 1.500000e+00
Jacobian value in (0.5,0.5) : 1.750000e+00
Surface integration by Gauss-Legendre : 6.000000e+00
More funny integration : 8.171599e+00
Recursive integration (n = 0) : 8.171599e+00
Recursive integration (n = 1) : 8.238388e+00
Recursive integration (n = 2) : 8.237248e+00
```

Vous pensez avoir quelque chose ? Vous voulez tester votre code ? Rien de plus simple : appuyer sur gros bouton "Play" en haut à gauche. S'il y a des erreurs, Xcode vous le fera savoir. Sinon, vous allez pouvoir voir le résultat dans le cadre en bas à droite (comme l'image ici à gauche [les couleurs peuvent changer, vous allez plutôt avoir du texte noir sur fond blanc]).

2.7 Step 6 - Submit

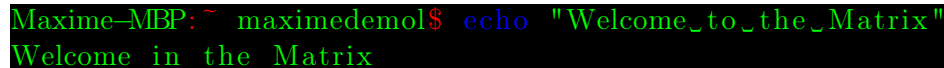


```
Version actuelle du devoir 1 du groupe 10 (soumise par Maxime De Mol)
February 01 2013 22:01:09. Version 0 Diagnostic du serveur pour la version 0
Voici le diagnostic du serveur pour la version 0 du devoir 1 du groupe 10 (soumise par Maxime D
***** COMPILATION SUCCEEDED *****
gcc compile_dir/current.c compile_dir/main.c -lm -lrt -Llib -Wl,-
***** RUN SUCCEEDED *****
Wall-clock time : 1.23s (limit : 200s)
Approximated maximum memory usage : 4Mb (limit : 2048Mb)
Approximated total memory allocation : 15Mb
***** RUN OUTPUT *****
Jacobian value in (0.0,0.0) : 1.500000e+00
Jacobian value in (0.5,0.5) : 1.750000e+00
Surface integration by Gauss-Legendre : 6.000000e+00
```

Vous avez fini ? Le programme semble correct ? Pas d'erreurs en vue ? Foncez sur le site de MECA 1120 la soumettre. Vous devez retrouver dans votre dossier projet le fichier homework.c et l'envoyer au serveur ! Le diagnostic serveur est bon ? Bueno ! Vous avez fini !

3 The hard way : Terminal

Vous trouvez que Xcode, c'est trop simple ? Vous voulez quelque chose de plus rapide et léger ? Vous voulez programmer comme des vrais bonhommes ? Alors le Terminal est pour vous !



```
Maxime-MBP:~ maximedemol$ echo "Welcome_to_the_Matrix"
Welcome in the Matrix
```

L'inconvénient du Terminal, c'est que vous êtes moins guidé que dans Xcode. Par contre l'exécution dans le terminal est plus légère, il y a moins de chipotage, est vous pouvez vous concentrer plus sur le code lui même. Vous allez par contre avoir besoin d'un bon éditeur de texte pour éditer vos codes. Personnellement j'utilise Xcode à cet effet, mais TextMate¹ est aussi très bien côté pour ça !

3.1 Step 1 - The Terminal



Vous pouvez trouver le terminal dans vos applications » utilitaires ! Suffit de lancer et bingo !

1. <https://github.com/textmate/textmate>

3.2 Step 2 - GCC

GCC c'est quoi ? C'est le compilateur C (et C++). C'est ce module du terminal qui va vous permettre, très bientôt de compiler vos programmes C en exécutable. Mais il y a des chances que celui-ci ne se trouve pas sur votre mac d'origine. Il faut l'installer ! Pour cela, foncez à l'adresse <https://developer.apple.com/downloads/index.action>, inscrivez vous si c'est pas fait, et téléchargez le dernier pack "Command Line Tools". Une fois installez, vous serez capable d'utiliser gcc !

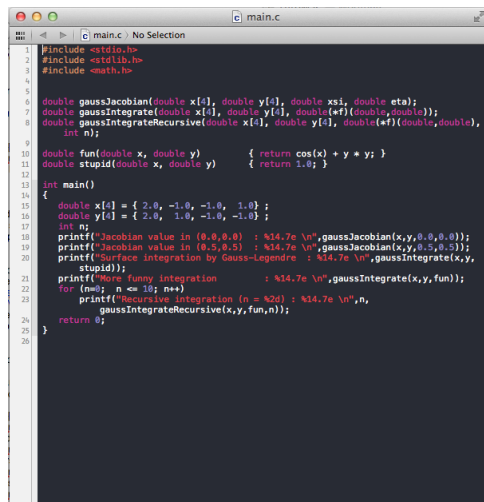
```
Maxime-MBP:Desktop maximedemol$ ls
hello.c
Maxime-MBP:Desktop maximedemol$ gcc -o hello hello.c
Maxime-MBP:Desktop maximedemol$ ls
hello  hello.c
Maxime-MBP:Desktop maximedemol$ ./hello
Hello , world
Maxime-MBP:Desktop maximedemol$
```

3.3 Step 3 - Naviguer

Maintenant, avant de pouvoir travailler, vous devez d'abord atteindre vos fichiers ! Supposons que vous avez extrait les fichiers fournis par Legat sur votre bureau. Pour y arriver, vous allez utiliser 2 commandes.

```
Maxime-MBP:~ maximedemol$ echo 'ls: voir tous les dossiers'
ls: voir tous les dossiers
Maxime-MBP:~ maximedemol$ echo 'cd: naviguer vers'
cd: naviguer vers
Maxime-MBP:~ maximedemol$ ls
Applications Desktop Documents Downloads Library Mail Movies Music
Maxime-MBP:~ maximedemol$ cd Desktop
Maxime-MBP:Desktop maximedemol$ ls
Makefile      homework.c    main.c
Maxime-MBP:Desktop maximedemol$ echo 'Nous y voila'
Nous y voila
Maxime-MBP:Desktop maximedemol$
```

3.4 Step 4 - Editer vos fichiers

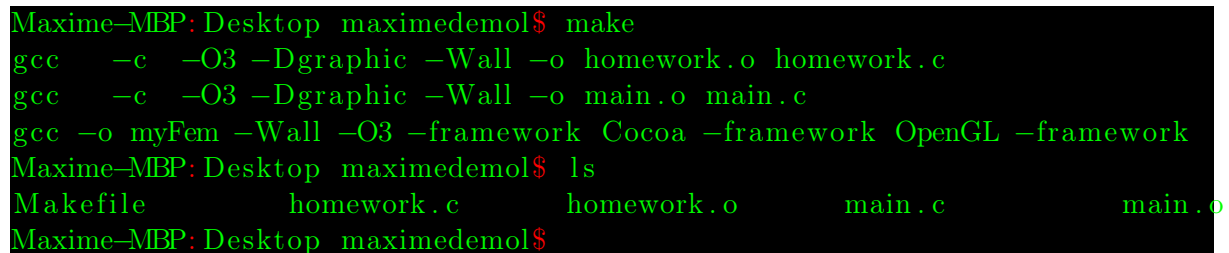


```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <math.h>
4
5 double gaussJacobian(double x[i], double y[i], double xsi, double eta);
6 double gaussIntegrateRecursive(double x[i], double y[i], double (*f)(double,double),
7     int n);
8
9 double fun(double x, double y) { return cos(x) + y * y; }
10 double stupid(double x, double y) { return 1.0; }
11
12 int main()
13 {
14     double x[i] = { 2.0, -1.0, -1.0, 1.0 };
15     double y[i] = { 2.0, 1.0, -1.0, -1.0 };
16     int n;
17     printf("Jacobian value in (0.0,0.0) : %14.7e\n",gaussJacobian(x,y,0.0,0.0));
18     printf("Jacobian value in (0.5,0.5) : %14.7e\n",gaussJacobian(x,y,0.5,0.5));
19     printf("Surface integration by Gauss-Legendre : %14.7e\n",gaussIntegrate(x,y,
20         stupid));
21     printf("More funny integration : %14.7e\n",gaussIntegrate(x,y,fun));
22     for (n=0; n <= 10; n++)
23         printf("Recursive integration (n = %2d) : %14.7e\n",n,
24             gaussIntegrateRecursive(x,y,fun,n));
25     return 0;
26 }
```

Maintenant il vous faut écrire votre programme dans le fichier homework.c. Pour cela il suffit de double cliquer sur le fichier homework.c dans votre bureau. Si vous avez installé Xcode, il va s'ouvrir avec Xcode. Si vous avez TextMate, c'est TextMate qui prend le relai. Si vous voulez l'ouvrir avec autre chose, faites "ouvrir avec".

3.5 Step 5 - Compiler

Rien de plus simple ! Legat fournit gentilement un makefile, utilisons le (commande make) !

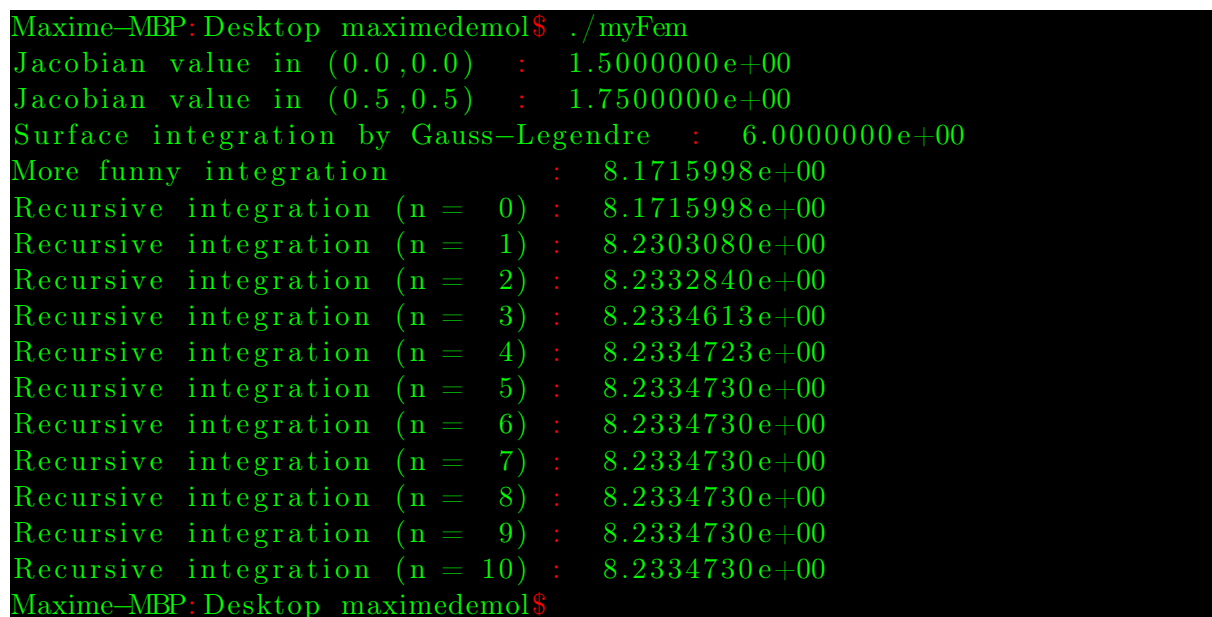


```
Maxime-MBP:Desktop maximedemol$ make
gcc -c -O3 -Dgraphic -Wall -o homework.o homework.c
gcc -c -O3 -Dgraphic -Wall -o main.o main.c
gcc -o myFem -Wall -O3 -framework Cocoa -framework OpenGL -framework
Maxime-MBP:Desktop maximedemol$ ls
Makefile      homework.c    homework.o    main.c        main.o
Maxime-MBP:Desktop maximedemol$
```

Vous voilà en possession du fichier myFem qui est l'exécutable de votre programme !

3.6 Step 6 - Executer

Encore plus simple que compiler !



```
Maxime-MBP:Desktop maximedemol$ ./myFem
Jacobian value in (0.0,0.0) : 1.5000000e+00
Jacobian value in (0.5,0.5) : 1.7500000e+00
Surface integration by Gauss-Legendre : 6.0000000e+00
More funny integration : 8.1715998e+00
Recursive integration (n = 0) : 8.1715998e+00
Recursive integration (n = 1) : 8.2303080e+00
Recursive integration (n = 2) : 8.2332840e+00
Recursive integration (n = 3) : 8.2334613e+00
Recursive integration (n = 4) : 8.2334723e+00
Recursive integration (n = 5) : 8.2334730e+00
Recursive integration (n = 6) : 8.2334730e+00
Recursive integration (n = 7) : 8.2334730e+00
Recursive integration (n = 8) : 8.2334730e+00
Recursive integration (n = 9) : 8.2334730e+00
Recursive integration (n = 10) : 8.2334730e+00
Maxime-MBP:Desktop maximedemol$
```

3.7 Step 7 - Submit

Ca marche ? Ca marche bien ? Foncez soumettre votre fichier `homework.c` qui se trouve sur le bureau ! Pour plus d'info, voir section 2.7 page 3 !

4 Conclusion

Avec ça, j'espère que tu es bien armé pour affronter le problème 1 d'éléments finis ! Bonne merde à toi et à ton binôme, et bon codage !