Petit guide sur l'utilisation de mes programmes

Jean Ogier du Terrail

19 août 2014

Résumé

J'ai rédigé ce petit guide pour que tu comprenne mieux ma démarche. Je pense que tu peux utiliser mon code comme base et faire beaucoup mieux que ce que j'ai fait.

1 Conversion .txt en .root (1) "gif.cpp et temp.cpp"

J'ai deux programmes : un pour les fichiers issus du gif (courants) l'autre pour les autres mesures (fichiers temp(érature)s) Les deux sont construits sur la même structure :

le premier "gif.cpp" demande à l'utilisateur un nom de fichier texte ascii à traduire (ATTENTION IL NE DOIT PAS Y AVOIR DE TITRES OU DE SYMBOLES AUTRES QUE DES FLOATS ($/, :, NaN$)). On doit écrire les noms au format xxx.txt et yyy.root mais ceci est précisé. C'est un petit programme à la syntaxe facile à comprendre est assez réduite. Dans un premier temps il lit simplement le fichier en mettant les floats dans les variables créeées à cet effet, qui formeront les branches d'un TNTuple. Dans un second temps il va créer la TBranche "dateabs" contenant les dates absolue prises à partir du 1 janvier 2013 00h00 et comptées en minutes. Il est à appliqué uniquement aux fichiers, que tu m'as donné sinon il faut modifier le code en conséquence (nombres de vaiables, ordres, etc.).

temp.cpp fonctionne exactement de la même façon pour les fichiers de températures mais pas avec le même nombre de variables ni le même nom d'ou l'utilité d'avoir deux versions du même code.

Attention il dépend donc de l'ordre et du nombre de variables du fichier texte d'entrée.

2 Conversion .txt en .root(2) "creationfusion.cpp"

Dans tous mes programmes j'ai preferé, et je crois que ce n'est pas une mauvaise idée, de construire un fichier .txt intermédiaire permettant au cas ou de recréer le fichier originel si on en a modifié une partie par erreur. Mais absolument rien n'empèche d'accoler les deux codes et de faire ca en une seule étape à utiliser avec précaution. Dans ce cas là j'ai crée un simple programme appelé creationfusion.cpp qui permet de repasser du fichier texte à un fichier root. Mais il est ultra-dépendant là encore du NOMBRE, de l'ORDRE des variables. Tous les bugs, que j'ai eu sur ce programme viennent de là. Certains fichiers contiennent par exemple le courant normalisé ou même corrigé ou alors

ils contiennent en plus de la dateabsolue les jours mois année heure minute ou pas. La charge peut aussi poser problème ! Il est possible de créér autant de fichiers qu'on a de configurations possibles mais je ne crois pas que cela soit une bonne idée. Le code étant très facilement modifiable. Je répète encore une fois tout bug de ce programme vient probablement d'une mauvaise écriture des vraiables.

3 Suppression des données non physiques "zerosuppression.cpp"

Attention ce programme est piègeux. Il prend en argument un fichier root. Comme son nom l'indique il va supprimer toutes les données ou le courant I1 est quasiment à 0 le seuil est par défaut à $10^(-9)$. Et ce n'est PAS TOUT !Il va réecrire le courant I1 en prenant l'opposé pour avoir un courant positif. Attention là aussi cela est source de problèmes. Il est facile d'écrire deux programme un zerosuppression et un valabsolue je l'ai d'ailleurs fait (ce programme n'est donc utile que pour l'utilisateur averti).

4 Normalisation "normalisation.cpp"

Comme son nom l'indique ce programme normalise un courant en utilisant la moyenne des N premières valeurs N étant un entier indiqué par l'utilisateur. Il est nécéssaire de l'appliquer avant d'utiliser les corrections. Il va créer la TBranche "I1n".

5 Correction du courant "lineac.cpp , expoc.cpp et powerc"

J'ai créé trois méthodes de fit pour corriger le courant (gain) en température et en pression. Powerc(orrection), expoc(orrection) et linea(ire)c(orrection).cpp. Power(c) est celle de Capéans c'est à dire qu'il assume une loi de type

$$I = I_0 * Pressure^f * Temperature^g \tag{1}$$

Avec f et g les exposants inconnus. Il prend un argument un fichier contenant la TBranch "I1n" le courant normalisé mais en fait on pourrait l'appliquer directement à I sachant qu'il normalise lui même. Je l'ai fait car il affiche le graphe courant normalisé et les différents courants corrigés versus le temps. Il peut créer ou non selon le vouloir de l'utilisateur une branche "Ipowerc(orrected)" contenant le courant corrigé. Pour ce faire il faut modifier la fin du fichier, qui est commentée par défaut. Expo(c) corrige de la même manière mais avec une loi exponentielle de type

$$I = I_0 * \exp(a * Temperature) * \exp(b * Pressure) \tag{2}$$

Les mêmes remarques sont valables il est possible de créer une TBranche Iexpoc. Linea(c) propose une correction polynomiale

$$I = I_0 * P(Temperature) * Q(Pressure) \tag{3}$$

Avec P et Q des polynômes de même degré. On peut la justifier comme étant le début de la série de Taylor de l'exponentielle. Ces trois programmes on la propriété d'être applicables à l'infini par les propriétés d'addition de l'exponentielle et de la puissance ou par l'extension de l'espace d'arrivée pour les polynômes (augmentation du degré). Il suffit de changer les noms des TBranch de début et de fin. Le nombre d'itérations dépend de la précision voulue par l'utilisateur. Seulement même en les appliquant un certain nombre de fois on obtient pas une precision infinie plutôt une convergence vers une courbe très différente de la fonction constante. Ils sont aussi facilement modifiables du point de vue des TGraph qu'il affiche. Une option commentée permet d'afficher la variance de la gaussienne fitée à l'histogramme du courant corrigé en fonction de f à g fixé (et inversement). On pourrait imaginer afficher le paramètre du chi2 pour avoir une idée de la qualité du caractère gaussien de l'histogramme...

6 Zooms ou cut ("cuttime.cpp et cuthisto.cpp")

On peut très bien imaginer de corriger chaque portions de la courbe d'intensité différemment. Les coefficients f et g peuvent subir de légères variations ou même carrément changer du tout au tout (application de filtres). Attention cependant la multiplication des segments de correction améliore certe le tracé mais il me semble qu'on perd de l'information. J'ai réalisé deux script l'un "cuttime.cpp" qui réalise un zoom sur le fichier en ne prenant par exemple que les valeurs entre telle dates (absolues) et telle autre date (pour les avoir regarder le graphe I=f(dateabs)). L'autre script réalise un cut directement sur les valeurs de I1. Il a été créé suite à l'observation de trois différents pics sur l'histogramme du courant. Il permet d'en isoler un et de travailler dessus. Attention cependant il revient à complétement écarter les autres pics. A utiliser avec précaution pour garder l'information. Là encore on voit l'interêt des fichiers textes intermédiaires.

7 Création de la TBranche "charge" "Addcharge.cpp"

Tout est dans le titre le programme utilise I1 comme source mais peut être modifié.

8 Moyenne "moy.cpp"

Il réalise une moyenne des mesures toutes les N mesures N étant indiqué par l'utilisateur. Il diminue la variance des mesures mais là encore on perd de l'information.

9 Chirurgies "collage.cpp" et "fusion.cpp"

Collage réalise la fusion de deux fichiers root dont les mesures couvrent différents intervalles temporels. J'en ai eu besoin par exemple avec les fichiers que tu m'a envoyé en trois parties. Ou alors pour recoller les différentes corrections effectuées par segments. Il prend en argument le nom des deux fichiers et le nom du fichier root de destination. Fusion quand à lui est beaucoup plus important

c'est lui qui trouve les coïncidence et qui effectue la suture au bon endroit entres les fichiers créés par gif et par temp. Grâce à Marco notamment ce programme est assez optimisé je ne pense vraiment pas qu'on puisse l'améliorer de beaucoup cependant il prend quand même quelques temps si toutes les mesures doivent être recopiées jour mois et années par exemple ou les erreurs sur les courants il peut être modifié en conséquences selon les envies mais là il faut plonger dans le code.

10 PROTOCOLE A SUIVRE POUR EVITER LES PROBLEMES :

L'ordre des programmes à effectuer peut être le suivant : .x gif.cpp .x temp.cpp .x fusion.cpp .x zerosuppression.cpp .x normalisation.cpp .x lineac.cpp Mais là encore c'est juste une question d'ordre et de nombre des variables à part pour lineac, qui necessite la branche I1n. Si on veut appliquer lineac, powerc ou expoc de nombreuses fois attention au nom de la branche créées et de la branche d'entrée qui ne sera plus I1n mais I1expoc1 par exemple. Attention cela est source de bug si l'on souhaite créer deux fois le même Tbranche. Si l'utilisateur peut éviter de compiler deux fois le même programme dans une même session de ROOT...cela provoque parfois des bugs. Si l'utilisateur se trompe de nom de variables ou de nombre ou d'ordre ceci est la source la plus récurrente de problèmes. Tous les fichiers et les programmes doivent être dans le même REPERTOIRE ! Je n'ai pas eu d'autres bugs.