TP n° 24 : Révisions

L'objectif de ce TP est de vérifier que vous ne rencontrerez pas de problèmes techniques pendant le dernier devoir en temps limité, ce samedi 13 juin de 8h à 12h. Nous en profitons au passage pour réviser quelques notions rencontrées cette année.

1 Consignes

⚠

Il est très important de bien lire cette partie.

Les questions qui comportent le symbole ✍ sont à rédiger impérativement sur une feuille, à lignes ou à carreaux, au stylo, à encre foncée. Vous aurez 30 minutes après le TP pour scanner ou prendre en photo votre copie et pour la déposer. Il est impératif que votre copie soit numérisée sous la forme d'un seul document au format PDF, de taille raisonnable (moins de 10 Mo), impérativement sous le nom `tp24-<login>.pdf`, où `<login>` est à remplacer par votre nom d'utilisateur du laboratoire informatique, a priori `prenom-nom`, en minuscules. Il existe de nombreux outils pour numériser des documents depuis un intelliphone et/ou pour regrouper plusieurs PDF en un seul.

Vous devez télécharger l'archive `tp24.zip` sur le site `https://cpge.info`. Il faut ensuite compléter le fichier `tp24.py` qui se trouve dans l'archive une fois décompressée, *sans en modifier le nom*. L'encodage du fichier devra être UTF-8 (*Fichier* > *Encodage* > utf-8 sous PYZO). Pour toutes les questions qui demandent d'écrire du code PYTHON ou SQL, c'est dans ce fichier et uniquement dans ce fichier qu'il s'agit d'implémenter vos réponses. Une fois que vous avez terminé et vérifié une toute dernière fois, ce fichier devra être renommé et déposé, sous le nom `tp24-<login>.py` où, comme toujours, `<login>` est votre nom d'utilisateur du laboratoire informatique.

Le code devra impérativement compiler : *aucune* erreur ne doit apparaître lorsque l'on exécute le script. L'exécution[1] simple de votre programme doit être pratiquement instantanée, aucun appel de fonction ne doit être effectué par défaut, rien ne doit s'imprimer à l'écran, aucun graphe ne doit s'afficher. N'oubliez pas de commenter les éventuels appels, tracés et tests !

1. Rappel : exécuter en tant que script pour pouvoir travailler dans le répertoire courant.

⚠ Un code qui ne compile pas, ou qui prend trop de temps à s'exécuter, entraînera inévitablement une note nulle pour la partie de programmation.

Vous pouvez vérifier que vos fonctions sont correctement prises en compte et passent les premiers tests (au DS il y en aura d'autres) en exécutant le fichier `verif_tp24.py` qui se trouve dans l'archive décompressée. Attention, il faut bien que votre code `tp24.py` se trouve dans cette même archive et, sous PYZO, choisir « *Démarrer le script* », que je vous conseille de lier à F5 chez vous si ce n'est pas déjà fait.

✔ On prendra bien soin de tester toutes les fonctions sur des cas triviaux, puis un peu plus complexes. La vérification ne garantit aucunement que vos fonctions sont correctes.

Le sujet vous demandera également de tracer des graphes à l'aide de MATPLOTLIB et de déposer votre production. La question précisera exactement le format et le nom du fichier, a priori `q<no>-<login>.png` où `<no>` sera un numéro de question et `<login>` est évidemment à remplacer par votre nom d'utilisateur du laboratoire informatique.

⚠ Une fois le graphe généré et sauvegardé, n'oubliez pas de commenter la partie correspondante dans votre code.

Lorsque l'on pose des hypothèses sur les arguments (par exemple qu'un argument est un entier naturel), il n'est pas nécessaire dans vos fonctions de le vérifier ni de prévoir le comportement si l'hypothèse est violée (par exemple si l'argument est strictement négatif).

✔ Il est absolument impératif de veiller au strict respect de toutes ces consignes.

1. ✍ Avez-vous bien lu et compris toutes ces consignes ?
2. ✍ Recopiez, en âme et conscience, la phrase suivante sur votre copie, puis signez-là :

   > Je m'engage sur l'honneur, samedi prochain, à ne pas tricher, à ne communiquer avec personne pendant l'épreuve, à ne consulter aucune ressource et je comprends que toute tentative de fraude pourra entraîner une note nulle pour ce semestre.

2 Jeux de triangles

La base `triangles.db` est constituée d'une seule table dont le schéma relationnel est

`triangles(idt:int, ab:int, ac:int, bc:int)`

Chaque ligne représente un triangle défini par un identifiant unique et par la taille de ses trois côtés AB, AC et BC.

3. ✍ Décrire en français, en une phrase, ce que représente la requête $\Pi_{idt} \circ \sigma_{ab+ac+bc\geq 42}(\texttt{triangles})$.
4. ✍ Écrire, dans le langage de l'algèbre relationnel, une requête qui donne la relation formée par les identifiants et la longueur d'un des côtés de tous les triangles équilatéraux.

5. Écrire une requête en SQL qui donne le nombre de triangles dans la base.
6. Écrire une requête en SQL qui donne le maximum des périmètres des triangles rectangles en A.
7. Écrire une requête en SQL qui donne, sans doublons, les longueurs AB, AC et BC correspondants au(x) triangle(s) qui possède(nt) la plus petite valeur du produit $AB \times AC \times BC$, parmi les triangles ABC de périmètre supérieur ou égal à 100.
8. Parmi les triangles de périmètre égale à 42, combien de triangles ABC ont un côté AB de même longueur que le côté $A'B'$ d'un autre triangle $A'B'C'$? Cette requête devra impérativement s'exécuter en moins de 5 s.

3 Un grand nombre

9. ✍ Quel est le plus grand nombre flottant que l'on peut représenter avec 64 bits ? On rappelle que la valeur $e = 1024$ de l'exposant e est réservée pour des valeurs spéciales.

4 Fabrication d'un diagramme HR

De nombreux programmes complexes ont été écrits pour modéliser l'évolution d'une étoile tout au long de sa vie. Ceux-ci nous donnent, après de très long calculs, de précieuses informations sur les différents stades d'évolution d'une étoile et sur le temps passé dans chacun de ces stades. Même s'ils sont très intéressés par ces informations, les astrophysiciens qui se spécialisent dans la dynamique stellaire (les mouvements d'étoiles dans des ensembles autogravitants) possèdent leurs propres programme pour observer l'évolution de la position d'une étoile (considérée comme un point matériel) au cours du temps et n'ont pas les ressources CPU suffisantes pour faire tourner un programme complet d'évolution stellaire pour chacune des millions d'étoiles qui gravitent dans leurs simulations.

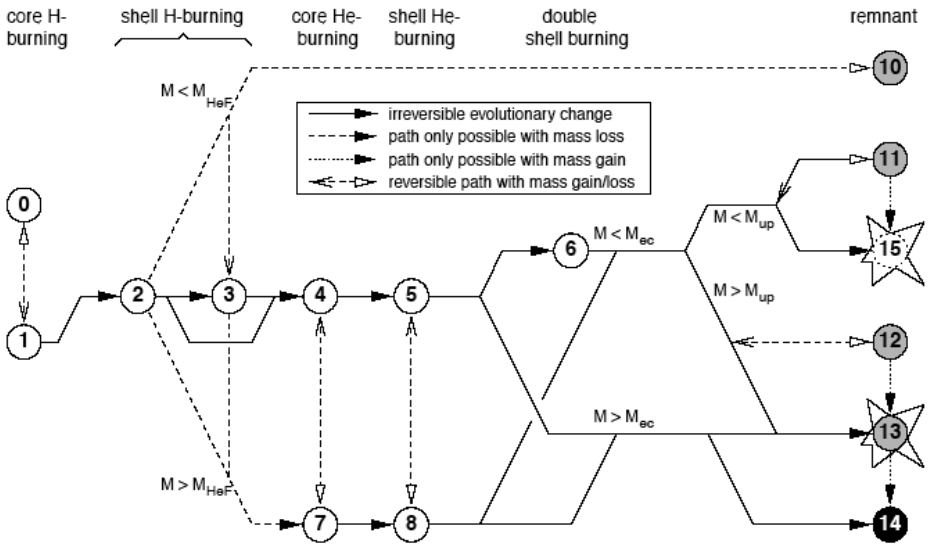


Figure 19. Possible evolution paths through the various stellar evolution phases.

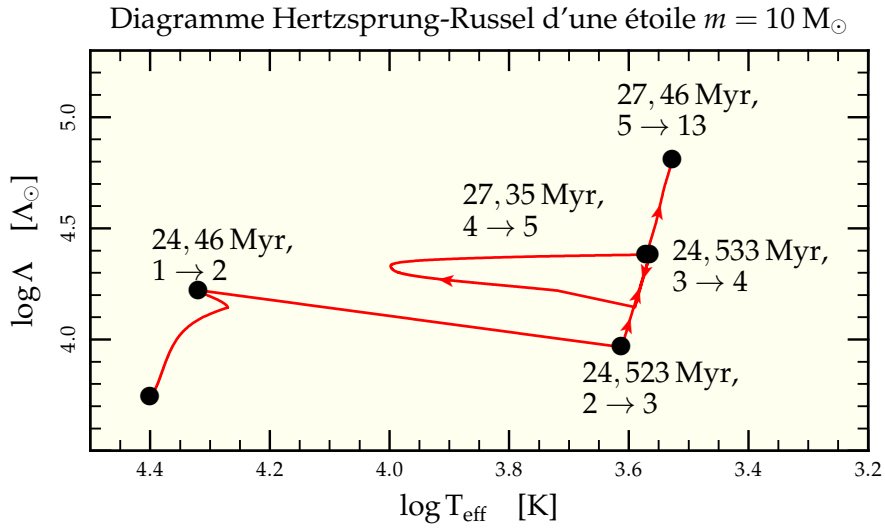
En d'autres termes, les dynamiciens stellaires aimeraient connaître des informations telle que la luminosité de chacune des étoiles de leur simulation à chaque instant, mais sans se fatiguer à faire évoluer indépendamment chaque étoile en utilisant un programme d'évolution stellaire dédié. C'est là qu'interviennent les fonctions d'interpolations mises au point par Jarod HURLEY et al. [2] qui permettent, en connaissant la masse et l'âge de l'étoile, de donner directement, entre autres choses, la luminosité, la température de surface et la phase d'évolution (voir figure ci-dessus) de l'étoile concernée.

4.1 Principe du diagramme HR et difficultés

Dans ce sujet, on se propose de comparer ces formules d'interpolation à un bon vieux code d'évolution stellaire [3]. Pour cela, on se propose de représenter l'évolution de l'étoile au cours du temps dans un diagramme dit de HERTZSPRUNG-RUSSELL qui représente le logarithme décimal de la luminosité de l'étoile en fonction de l'opposé du logarithme décimal de la température.

La difficulté principale vient du fait que les temps caractéristiques que l'étoile passe dans chacune des phases de sa vie sont très différents en ordre de grandeur. Imaginez que vous suiviez un escargot qui se déplace d'Orléans à Paris à vitesse d'escargot. Naturellement, si vous voulez suivre son périple, vous prendrez une photo satellite toutes les mois environs. Mais si, arrivé à Orly, il embarque soudainement sur un avion pour Strasbourg, il sera arrivé en moins de deux heures et il est peu probable que votre satellite puisse prendre ne serait-ce qu'une photo de son transit. Pour autant, on ne va pas prendre une photo toutes les heures durant tout le trajet car cela ne serait pas du tout adapté à la la première partie du trajet de l'escargot. Il va nous falloir ruser un peu.

Pour les étoiles, c'est exactement la même chose. Elles passent l'essentiel de leur temps sur la *séquence principale*, c'est-à-dire dans l'état 1 de l'évolution stellaire alors que les étapes suivantes se font bien plus rapidement. L'échelle de temps de base en évolution stellaire est au moins de l'ordre du million d'années (pour les étoiles de forte masse ($> 2M_\odot$)), voire du milliard d'années (pour les étoiles de faible masse ($\lesssim 2M_\odot$)). Pour fixer les idées sur les ordres de grandeur, voici ce que cela donne pour une étoile de 10 masses solaires :



La durée de vie totale de l'étoile à $7M_\odot$ que nous désirons étudier est d'environ $60Myr$.

2. HURLEY, J. R., POLS, O. R., & TOUT, C. A. 2000, MNRAS, 315, 543
3. Le plus renommé, celui de Peter Eggleton, date des débuts des années 70.

4.2 Diagramme HR à partir des données EZ-web

Le code d'Eggleton prend un soin tout particulier à inclure tout ce que l'on sait sur les mécanismes physiques à l'origine de l'évolution stellaire. Il a été revisité depuis les années 70 et on peut accéder aux résultats via l'interface EZWEB :

http://www.astro.wisc.edu/~townsend/static.php?ref=ez-web

Nous avons récupéré une simulation correspondant à l'étoile que nous allons étudier dans le fichier `EZ_summary.txt`, qui vous est donné dans l'archive et qui contient énormément d'informations (29 colonnes en tout). Pour tracer le diagramme HR correspondant, il nous faut le lire ce fichier, en extraire ce qui nous intéresse (colonne 4 pour $\log L$; colonne 6 pour $\log T$) et tracer le diagramme HR correspondant.

10. Écrire une fonction `lecture_EZ()` qui lit le fichier `EZ_summary.txt` et renvoie un triplet de listes de flottants `(t, logL, logT)` :
    - `t` est une liste contenant tous les âges présents dans la simulation (2e colonne du fichier).
    - `logL` est une liste contenant les valeurs de $\log L$ en correspondance avec les âges contenus dans `t` (4e colonne du fichier).
    - `logT` est une liste contenant les valeurs de $\log T$ en correspondance avec les âges contenus dans `t` (6e colonne du fichier).
11. Tracer le diagramme HR, c'est-à-dire $\log T_{eff}$ (en Kelvin) en fonction de $\log \Lambda$ (en luminosité solaires) pour cette étoile de 7 masses solaires. Ajouter un titre au graphique. Le titre doit impérativement et obligatoirement commencer par votre nom d'utilisateur du laboratoire informatique. Ajouter une légende aux axes. Vous êtes libre d'ajouter ou non une grille en fond. On pourra utiliser la commande `plt.gca().invert_xaxis()` pour inverser l'axe des abscisses. Enregistrer votre figure sous le nom `q11-<login>.png` où `<login>` est votre nom d'utilisateur du laboratoire informatique. N'oubliez pas de commenter la partie correspondante du code pour que le code compile sans afficher ce graphique. N'oubliez pas non plus de déposer cette figure lors du rendu.

4.3 Diagramme HR à partir des formules d'interpolation

Le module `sse`, qui est fourni dans l'archive décompressée, et plus particulièrement la fonction `sse.get_evolution(t)`, permet d'obtenir des informations sur l'état de l'étoile à partir des formules d'interpolation. Par exemple, pour obtenir à la date $t = 10Myr$, la luminosité et la température de surface, on peut utiliser :

PYTHON

```
import sse
t = 10
s, logL, logT = sse.get_evolution(t)
print("À t={} Myr, état {} avec log10(L)={} et log10(T)={}"
      "".format(t, s, logL, logT))
```

On peut obtenir de l'aide sur cette fonction avec la commande `help(sse.get_evolution)`.

Nous allons maintenant dessiner le diagramme HR de l'étoile, à partir des formules d'interpolation, en plaçant 50 points régulièrement espacés en temps sur chacune des phases (donc 300 points au final vu qu'il y aura 6 phases, les formules d'interpolations n'allant, dans notre cas, que jusqu'à l'état de géante rouge pulsante sur la branche asymptotique des géantes). On rappelle que la durée de vie totale de l'étoile à $7M_\odot$ que nous désirons étudier est d'environ $60Myr$.

12. Écrire une fonction `dates_transitions()` qui renvoie une liste à 6 valeurs où chaque valeur représente l'âge de transition de l'étape i à l'étape $i+1$ à 10^{-5} près. La dernière valeur étant l'âge limite, à 10^{-5} près, pour lequel les formules d'interpolation renvoient une valeur sensée et non `None`). On s'inspirera des techniques développées en cours concernant la recherche par dichotomie d'une valeur dans une liste ou du zéro d'une fonction.
13. Écrire une fonction `obtiens_valeurs(transitions)` qui prend en entrée la liste à 6 valeurs précédemment calculée et renvoie deux tableaux de 300 éléments, chacun contenant les valeurs en $\log L$ et $\log T$ correctement échantillonnées à raison de 50 points sur chaque phase, c'est-à-dire que les points de 0 à 49 appartiennent à la phase 1, les points de 50 à 99 à la phase 2, etc. Pour les tests, la liste de transitions donnée est telle que si $t_{i,i+1}$ correspond à la date de transition de l'état i à l'état $i+1$, alors l'état de l'étoile en $t_{i,i+1}$ est l'état $i+1$.
14. Superposer sur un même graphique, le diagramme HR obtenu avec les données EZWEB et celui utilisant les deux listes calculées à la question précédente. Ajouter un titre au graphique. Le titre doit impérativement et obligatoirement commencer par votre nom d'utilisateur du laboratoire informatique. Ajouter une légende aux axes. Vous êtes libre d'ajouter ou non une grille en fond. On pourra utiliser la commande `plt.gca().invert_xaxis()` pour inverser l'axe des abscisses. Ajouter une légende pour que l'on puisse identifier facilement les deux tracés. Enregistrer votre figure sous le nom `q14-<login>.png` où `<login>` est votre nom d'utilisateur du laboratoire informatique. N'oubliez pas de commenter la partie correspondante du code pour que le code compile sans afficher ce graphique. N'oubliez pas non plus de déposer cette figure lors du rendu.