III. Principes de programmation - Corrigé

1. Switch – Case structure : Conversion d'unité d'énergie

Ecrire un petit script qui permet de convertir une quantité d'énergie parmi plusieurs unités d'énergie : joule, calorie, Ft-lb et eV. Organiser le programme afin qu'il demande d'entrer la quantité d'énergie, son unité, ainsi que la nouvelle unité. L'output sera la quantité d'énergie dans la nouvelle unité (inspirez-vous des exemples du chapitre 3).

Les facteurs de conversion entre unités sont : 1J = 0.738 ft-lb = 0.239 cal = $6.24*10^{18}$ eV.

En utilisant votre programme, déterminer a) 325J en ft-lb, b) 432 cal en Joules, c) 6.8 eV en calories.

Utiliser la commande « `disp` » pour afficher le résultat à l'écran (utiliser l'aide !)

Compléter le programme à l'aide de la structure `if - else - end` pour générer un message d'erreur si les unités sont entrées incorrectement.

```
% input la valeur d'énergie, son unité et la nouvelle unité
% désirée:
V_in = input('quantité d''énergie à convertir: ');
Ein_unit = input('entrer son unité(J, ft-lb, cal ou eV): ','s');
Eout_unit = input('entrer l''unité désirée(J, ft-lb, cal ou …
eV): ','s');

% asign la valeur 0 aux erreurs potentielles
error = 0; %l'erreur est fausse

%utilise la switch expression pour choisir entre les unités
%initiales. Pour les 4 unités initiales, convertir premièrement
%le %V_in en Joule
switch Ein_unit
    case 'J'
        E_Joule = V_in;

    case 'ft-lb'
        E_Joule = V_in/0.738;

    case 'cal'
        E_Joule = V_in/0.239;

    case 'eV'
        E_Joule = V_in/(6.24*10^18);

    otherwise
        error = 1; %l'erreur est vraie
end
```

```
% utilise la switch expression pour choisir entre les unités
% finales. Pour chaque unité finale, convertir le résultat obtenu
% précédemment (en Joules) dans l'unité finale désirée.

switch Eout_unit
    case 'J'
        E_new = E_Joule;

    case 'ft-lb'
        E_new = E_Joule*0.738;

    case 'cal'
        E_new = E_Joule*0.239;

    case 'eV'
        E_new = E_Joule*6.24e18;

    otherwise
        error = 1;
end

% Message d'erreur si les unités sont entrées incorrectement
if error
    disp('Les unités sont entrées incorrectement!')
else
    fprintf('%g %s = %g %s\n', V_in, Ein_unit, E_new, Eout_unit)
end
```

Réponse :

325 J = 239.85 ft-lb

432 cal = 1807.53 J

6.8 eV = 2.60449e-019 cal

2. Structure en boucle : la somme d'une série

- Utiliser la structure `for - end` pour calculer la somme des n termes de la série: $\sum_{k=1}^{n} \frac{(-1)^k k}{2^k}$.

Exécuter le programme pour n = 4 , n = 20, n = 10^2. En déduire pour n = +inf.

```
% input le nombre de terme n
n = input('nombre de termes n ? ')

% premièrement set la somme = 0
S = 0;

% utilise la for-end loop pour générer la somme
```

```
for k=1:n
    S = S + (-1)^k*k/2^k;
end

fprintf('la somme des %g termes de la série sont: %f \n',n, S)
```

Réponses :

la somme des 4 termes de la série sont: -0.125000

la somme des 20 termes de la série sont: -0.222216

la somme des 100 termes de la série sont: -0.222222

la somme des 1e+006 termes de la série sont: -0.222222

- La fonction cos(x) peut être écrite en une série de Taylor : cos (x) = $\sum_{k=0}^{\infty} \frac{(-1)^k x^{2k}}{(2k)!}$.

Les arguments d'entrée sont l'angle x en degré (attention : Matlab est en radian) ainsi que le nombre n de termes dans la série. Utiliser la structure for – end pour calculer cos(210°) avec n = 3 et n = 8. Comparer votre réponse avec la function cos(x) implémentée dans Matlab.

```
%input arguments: l'angle x (en degrés) et n the nombre de termes

x = input('l''angle (en degré) ')
n = input('nombre de termes de la série de Taylor ')

%transforme l'angle x en radian
xr = x*pi/180;

%set la somme = zéro
S = 0;

%utilise la boucle for-end loop pour générer la somme

for k=0:n
    S = S + (-1)^k*xr^(2*k) / factorial(2*k);
end

%display la réponse pour cos(x,n)
fprintf('la fonction cos(%g,%g)= %f\n', x, n, S)

%comparaison avec la fonction cos(x)
cosX = cos(xr);

fprintf('la fonction Matlab cos de %g = %g\n', x, cosX)
```

Réponses :

la fonction cos(210,3)= -1.564583

la fonction Matlab cos de 210 = -0.866025

la fonction cos(210,8)= -0.866023

la fonction Matlab cos de 210 = -0.866025

- La fonction f(x) = e^x peut être écrite dans une série de Taylor : $e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$.

Ecrivez un programme qui détermine e^x en ajoutant les termes de la série de Taylor jusqu'à ce que la valeur absolue du dernier terme inclus est < 10^{-4}. Pour cela, utiliser la boucle `while - end` et limiter le nombre de passage à 20. Si après 20 passage, la valeur incrémentée n'est pas inférieure à 10^{-4}, le programme s'arrête en indiquant un message (ex. 'valeur e^x imprécise'). Donner votre réponse pour e^3, e^{-4} et e^{18}.

```
%input argument:  la valeur x de la fonction exp(x)
x = input('entrer la valeur x de la fonction exp(x) ')

%définir les terms de la série de Taylor en regardant les premiers
%termes:
% exp(x) = 1 + x + x^2/2! + x^3/3!

%l'increment n, c-à-d le nombre de termes, lequel commence à 1
n = 1;

%set la somme de la fonction expX à 1, laquelle est la somme du
%premier terme

expX = 1;

%definir une variable pour le prochain terme n+1
%pour commencer la commande while, la condition est que
%n_term >=0.0001

n_term = 0.0002; % 2 ou 3 ou 100…

%utiliser la commande while pour calculer le n terme de la série
%et ajoutez-le à la somme.

%la condition est que commande while est vraie aussi longtemps que
%la valeur absolue du n terme incrémenté est >= 0.0001 et le
%nombre de passage <= 20

while abs(n_term)>=0.0001 & n<=20
    n_term = x^n/factorial(n)
    expX = expX+n_term
    n=n+1
end
```

```
%display un message d'erreur si plus de 20 termes sont nécessaires
if n>=20
    disp('valeur exp(x) imprécise!')
else
    fprintf('exp(%g) avec %g termes est de %f\n', x, n, expX)
end
```

Réponse :

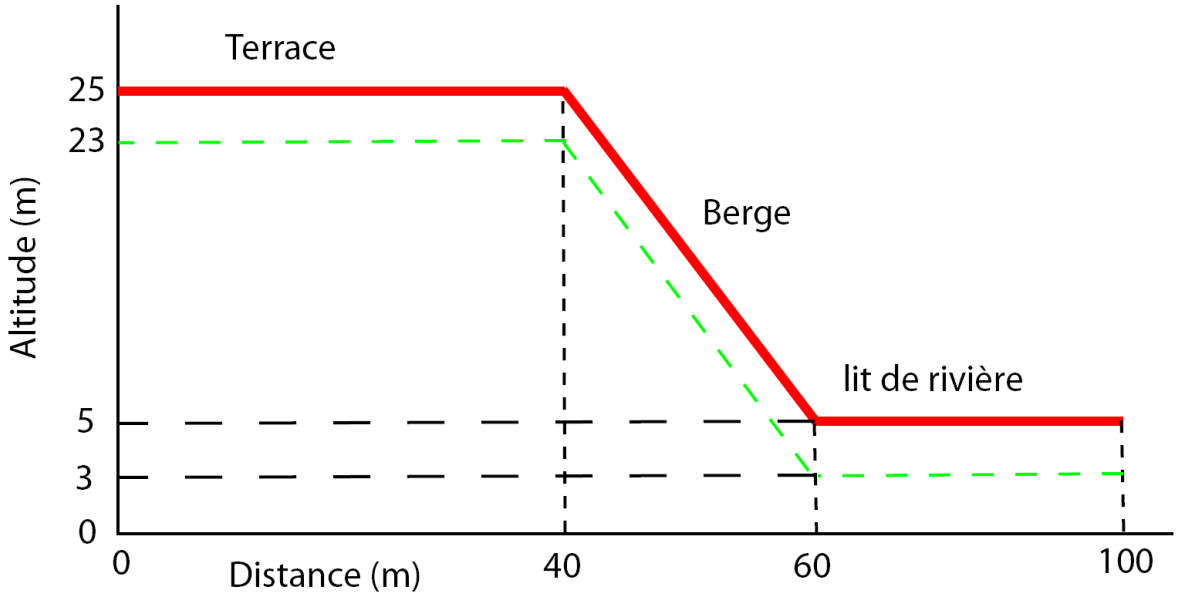
exp(3) avec 15 termes est de 20.085523

exp(-4) avec 18 termes est de 0.018307

pour exp(18) : valeur exp(x) imprécise!

3. Conditions / boucles

Créer à l'aide d'une combinaison de boucles et conditions le profile de terrain ainsi que son épaisseur de sol ci-dessous :



Les coordonnées de terrains peuvent être spécifiées en utilisant la structure « `for – end` ».

Exercice :

- créer sous forme de vecteur les coordonnées en x et z et rassembler-les sous une même matrice `xzdata` en utilisant la structure « for – end » (voir conseil dessous !).
- Representer graphiquement votre profil en utilisant la commande :

```
figure
plot(xzdata(:,1),xzdata(:,2),'-r','LineWidth',2)
axis([min(x) max(x) 0 30]);
```

- ajouter le profil de l'épaisseur du sol selon deux scénari :

  l'épaisseur du sol (normale au sol) est de 2 mètres sur toute la longueur

  l'épaisseur du sol est proportionelle à la pente, tel que : S = épaiss. sol * cos(pente)

  Pour cela, il est **conseillé** de discrétiser l'axe des x (distance) par intervalles de 1m et d'utiliser la structure « for – end ». La pente peut être calculée en utilisant :

  slope (X2-1) = atan((Z(2)-Z(1))/dx), ou dx = l'incrément (1m)

```
% x-coord. axis du profile en travers
x = [0 10 20 30 40 50 60 70 80 90 100];

% z-coord. ordinate : l'élévation z tel que z = f(x)
z = [25 25 25 25 25 15 5 5 5 5 5];

% transpose la matrice ligne x dans une matrice colonne xx
% idem pour z
xx = x';
zz = z';

% joint les deux matrices xx et zz ensemble
xzdata = [xx zz];

% séparer les matrices xzdata en deux matrices xdata et zdata
%NB: quand vous travaillez avec des boucles, il est parfois
%judicieux de pré-alloquer de la mémoire pour optimiser la vitesse
%de calcul en déclarant les variables à l'avance.

xdata = zeros(11,1);
zdata = zeros(11,1);

for i = 1:11
   xdata(i) = xzdata(i,1);
   zdata(i) = xzdata(i,2);
end

%représenter l'élévation de la surface dans un graph le long de
%l'axe des x
figure
%plot(xdata,zdata,'-r','LineWidth',2)
plot(xzdata(:,1),xzdata(:,2),'-r','LineWidth',2)
axis([min(x) max(x) 0 30]);

% ajouter le profile de l'épaisseur du sol
xmin = 0;
xmax = 100;
```

```
%x-coord. axis fait 100m de long par exemple, tel qu'on peut
%créer une donnée d'épaisseur du sol à chaque mètre par exemple
imax = 101;

%créer le pas d'augmentation le long de x-coord axis
dx = xmax/(imax - 1);

%entrer l'épaisseur du sol (normal au sol)
soilthick = 2;

%créer la x-coord
for i = 1:imax
   X(i) = (i - 1)*dx;
end

%calculer l'élévation
for i = 1:41
   Z(i) = 25;
end
for i = 42:61
   Z(i) = Z(i-1) - 1;
end
for i = 62:imax
   Z(i) = 5;
end

%calculer l'épaisseur du sol
for i = 1:imax
   S(i) = Z(i) - soilthick;
end

%représentation graphique
figure
plot(X,Z,X,S,'LineWidth',1.5)
axis([xmin xmax 0 30])

%l'épaisseur du sol de la rive est proportionnelle à la pente
%utiliser la fonction cos de la pente

%calcul de la pente du profile
slp = zeros(imax,1);
for i = 1:imax-1
   slp(i+1) =  atan((Z(i+1)-Z(i))/dx);
   %slp_deg(i) = slp(i)*180/pi;
end

%calcul de l'épaisseur du sol
for i = 1:imax
   Snew(i) = Z(i) - soilthick * cos(slp(i));
end

%représentation graphique de la nouvelle épaisseur de sol
figure
plot(X,Z,X,Snew,'LineWidth',1.5)
axis([xmin xmax 0 30])
```