TD-TP 15 INTEGRATION

Thème : Programmation de la fonction $\int_a^b$ `f(x)dx` "intégrale" par deux méthodes connues `TRAPEZE` et `TANGENTE`. Critique des deux méthodes. Algorithme de synthèse (notions vues au cours Ada numériques n°2).

Concepts : Les tests d'arrêt dans les itérations, les notions de précision. Contrôle de validité d'un algorithme. Algorithme série alternée. Tests de programme.

Introduction :

Soit `F` une fonction (réelle à valeur réelle) intégrable entre deux valeurs `INF` et `SUP`. On cherche à calculer : $\int_{INF}^{SUP}$ `F(x)dx`.

exemple : `F ≡ COS` entre `0` et `PI/2` (résultat connu : $\int_0^{PI/2}$ `cos(x)dx = 1.0`).

Méthode des trapèzes :

On découpe `[INF..SUP]` en segments de longueur `PAS = (SUP - INF)/NB` où `NB` est un nombre de segments choisi (paramétrable ?).

Le choix de `INF` et de `SUP` sera laissé à l'utilisateur (paramètres données). Le contrôle de validité doit être fait par le programme (**`exception`** sinon). `NB` est aussi un paramètre donnée de type `Positive`.

L'algorithme est simple il se réduit à la formule qui calcule la somme des aires des trapèzes approchant l'intégrale à calculer (dessinée en TD par l'enseignant) :

$$\texttt{PAS} * \sum_{i=0}^{i=NB-1} [\texttt{F(INF + i*PAS) + F(INF + (i+1)*PAS)}] / 2$$

Cette "**horrible**" formule doit être programmée **proprement** c'est à dire <u>différemment</u> avec une boucle **`for`** et surtout **sans multiplication** (sauf le produit final `PAS *`). La division par 2 **peut être évitée** <u>la plupart du temps</u>. Programmer cet algorithme `Integ_Trap` en paramétrant le type des réels, les bornes, le nombre de segments et .... la fonction. Voir les spécifications Ada page suivante.

Faire quelques tests pour des fonctions connues `COS`, `SIN`, `Ln` et aussi une fonction polynôme. Testez des valeurs négatives (pour les bornes) sauf pour `Ln` ! Le résultat aussi peut être négatif pensez-y. Sortez listing et copies d'écran exploitables (contrôlez évidemment les résultats avec vos connaissances !). Quelques exemples (voir en TP) : `COS` entre `0` et `PI/2` **puis** entre `PI/2` et `PI` **puis** entre `0` et `PI` pour les autres fonctions **imaginez** !

```
generic
      type T_Reel is digits <>;
      with function F(X : T_Reel) return T_Reel;
package P_Integr is
      function Integ_Trap(A,B : in T_Reel; Nb_Iter : in Positive)
                                    return T_Reel ;
      Exc_Erreur : exception; -- si A>= B
end P_Integr ;
```

Méthodes des tangentes :

On découpe toujours en `NB` intervalles. Mais cette fois l'intégrale est approchée par des aires obtenues avec les tangentes à la courbe prises au milieu des segments. On verra, comme pour la première méthode, un dessin (mieux qu'un long discours).
Là aussi la formule (à programmer aussi proprement et **non telle que** !) est la suivante :

$$PAS * \sum_{i=0}^{i=NB-1} [F(INF + (PAS/2) + i*PAS)]$$

Soit :

```
      function Integ_Tang(A,B : in T_Reel; Nb_Iter : in Positive)
                                    return T_REEL ;
```

Cette fonction est à ajouter aux spécifications (haut de page) du paquetage `P_Integr`.
Prendre dans le programme de test **les mêmes tests** pour les deux méthodes et comparez ! On peut voir que pour les fonctions `COS` ou `SIN` la méthode des trapèzes (entre 0 et PI/2) minore le résultat tandis que la méthode des tangentes la majore : ceci n'est pas toujours vrai (cela dépend des bornes ou de la fonction par exemple pour une fonction polynôme (à voir !)).

Il serait temps alors de **conjuguer** les deux algorithmes comme vue en cours numérique n°2 (série alternée).

Soit alors la fonction (définitive cette fois) :

```
      function Integrale (A,B : in T_Reel; Prec : in T_Reel)
                                    return T_Reel;
```

à ajouter aux spécifications du paquetage `P_Integr`. Notez que l'on travaille maintenant avec une précision et non pas avec un découpage en segments. D'où l'objectif :

```
generic
      type T_Reel is digits <>;
      with function F(X : T_Reel) return T_Reel;
package P_Integr is
      function Integ_Trap(A,B : in T_Reel; Nb_Iter : in Positive)
                                    return T_Reel ;
      function Integ_Tang(A,B : in T_Reel; Nb_Iter : in Positive)
                                    return T_REEL ;
      function Integrale (A,B : in T_Reel; Prec : in T_Reel)
                                    return T_Reel;
      Exc_Erreur : exception; -- si A>= B
end P_Integr ;
```

La fonction `Integrale` pourrait avoir l'allure suivante (à finir) :

```
function Integrale (A,B : in T_Reel; Prec : in T_Reel)
                                        return T_Reel is
      Iter : Positive := 10;
      Moy  : T_Reel;
      Tra,
      Tan  : T_Reel;
begin
      -- levées éventuelles d'exception (à faire)
      loop
            Tra := Integ_Trap(A,B,Iter);
            Tan := Integ_Tang(A,B,Iter);
            Moy := (Tra + Tan) / 2.0;
            -- des sorties possibles à écrire soyez bons!
            Iter := Iter + Iter;
      end loop;
      return Moy;
end Integrale;
```

Cette dernière fonction **méritera de jolis programmes de tests complets** avec de nombreuses fonctions (y compris des polynômes) : reprendre les données précédentes et de nombreuses variations de la précision avec un listing cohérent de sortie.
Exercice : reprendre le même problème mais avec des pointeurs sur fonction de façon à ne pas instancier le paquetage générique ! Très bel exercice.

TP 15

Il est pratiquement déjà suggéré dans le texte ci dessus :

- Créez le répertoire tp15. Copiez les fichiers proposés. Insérez votre entête.
- Réalisez le body de `P_Integr` mais seulement (et **pour commencer**) les deux fonctions `Integ_Trap` et `Integ_Tang`. Faites un corps vide pour `Integrale`.
- Testez provisoirement avec le programme `Ts_Tp15_A` (fichier `ts_tp15_a.adb`). **Critiquez vos résultats**.
- **Finissez** le body de `P_Integr` avec la réalisation de `Integrale` (qui utilise les deux premières !).
- Testez avec le programme `Ts_Tp15_B` (fichier `ts_tp15_b.adb`). Faites varier la précision de 0.1 à 0.00001 (par pas de 10 fois plus petit à chaque fois : 0.01, 0.001 etc.). **Critiquez vos résultats**.
- Testez encore avec le programme `Ts_Tp15_C` (fichier `ts_tp15_c.adb`). Ce sont des polynômes ! Vous pouvez inventez un autre polynôme (**plus compliqué**) à condition de connaître le résultat (Oracle) calculez le.
- Voyez les spécifications du fichier `p_integ_acc.ads`. Comparez avec celle de `P_Integr`. Réalisez le body (c'est très, très, très facile ! **Etonnant** !).
- Testez avec `Ts_Tp15_B_Acc` fichier `ts_tp15_b_acc.adb`.

L'erreur classique dans ce TP est de **ne pas valider ses résultats** ! Ce n'est pas par ce que le programme sort des calculs que ces derniers sont bons !