Quelques Calculs numériques

Christophe Labourdette

Septembre 2016

1 Premières expériences Numériques

- Ecrivez un programme affichant les tailles des différentes représentations des variables entières (on pourra utiliser la fonction sizeof),
- Ecrivez un programme montrant ce qui se passe lorsque l'on utilise des valeurs entières plus grandes que la capacité des variables, on démarera par exemple juste avant le plus grand entier puis on incrémentera de 1 la valeur de l'entier trouvé pendant 4 ou 5 itérations,
- Ecrivez un programme affichant la représentation binaire d'un entier positif, on pourra utiliser le fait que les caractères représentant les chiffres sont consécutifs et commencent par '0', (En $C++11$, il existe un type bitset)
- Ecrivez un programme affichant les tailles des différentes représentations des variables flottantes,
- Utilisez les fonctions frexp et ilogb pour afficher l'exposant d'un flottant ou sa décomposion en fraction normalisée.
  (on pourra regarder http ://pwet.fr/man/linux/fonctions_bibliotheques/frexp et http ://pwet.fr/man/linux/fonctions_bibliotheques/ilogb)

2 On n'a pas toujours le résultat escompté

Pour apprendre à se méfier écrivez un petit programme qui calcule et affiche les valeurs de la suite, pour chaque itérations, (jusqu'a u_{90} par exemple), programmez-le avec des $float$ puis avec des *double* :

$$\begin{cases} u_0 & = & 2 \\ u_1 & = & -4 \\ u_{n+1} & = & 111 - \frac{1130}{u_n} + \frac{3000}{u_n u_{n-1}} \end{cases}$$

vous afficherez également à chaque étape la "vraie" valeur de la suite :

$$u_n = \frac{4 \times 5^{n+1} - 3 \times 6^{n+1}}{4 \times 5^n - 3 \times 6^n}$$

3 Les sommes et les arrondis

3.1 une somme toute simple

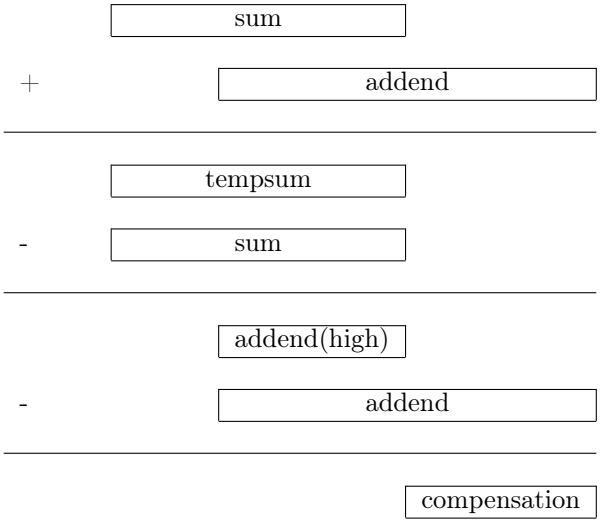
(dans cet exercice on utilisera des variables de type $float$ pour les réels) Nous allons nous intéresser à un simple problème de sommation. En considérant la

somme des N premiers entiers, $S_N = \frac{N(N+1)}{2}$, nous allons calculer pour des $n = 10, 100, 1000, \cdots, 100000000$ la somme

$$\sum_{j=1}^{j=n} \frac{j}{S_n} = 1$$

.

- Ecrivez un programme qui calcule la somme j/S_N pour les différentes valeurs de $n = 10, 100, 1000, \cdots, 100000000$ et affiche les valeurs, du dénominateur, de 1/dénominateur, de n/denominateur, de la somme et du pourcentage d'erreur ;
- Au vu de ce qui se passe on ne travaillera dans la suite qu'avec $n = 100000000$, écrivez un programme qui calcule pour $n = 100000000$ la somme par petit paquets (20 paquets de taille égale) en affichant à la fin de chaque groupe de calcul, la fraction courante et la valeur courante de la somme. Avant d'ajouter chaque fraction à la somme, on calculera la différence, entre l'exposant binaire de la fraction et l'exposant binaire de la somme courante. A la fin de chaque groupe on affichera alors la plus petite différence trouvée dans le groupe et le pourcentage des différences entre exposants, qui ont dépassé 24 ; (on pourra utiliser la fonction ilogbf)
- Ecrivez un programme identique au précédent mais qui effectue la somme dans l'ordre décroissant ;
- Ecrivez un programme qui calcule un sous-total par groupe avant de l'ajouter à la somme ;
- (Pour les courageux) Ecrivez un programme utilisant l'algorithme de sommation due à Kahan. Il consiste à ajouter une compensation provenant de l'addition précédente. autrement dit on va calculer une compensation comme suit :



3.2 Le calcul d'une exponentielle

Nous allons nous attaquer au calcul de $e^{-19.5}$ en utilisant la série de Taylor

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!} = \frac{x^0}{0!} + \frac{x^1}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

— Ecrivez un programme qui calcule la somme (tant que la somme change) et affiche, le numerateur courant, le dénominateur courant, la fraction courante et la valeur de la somme ainsi que le pourcentage d'erreur par rapport à l'appel de la fonction exponentielle.
— en utilisant $e^x = \frac{1}{e^{-x}}$ ecrivez un programme calculant la série de taylor de de $|x|$ et prenant l'inverse de la somme finale si x est négatif.
— Pour accélérer la convergence en réduisant la taille des numérateurs, utilisez l'astuce suivante : on peut diviser un exposant x en une partie entière w et une fraction f avec $0 \leq f < 1$

$$e^x = e^{w+f}$$

Si w est non nul on peut réduire la taille de l'exposant en factorisant

$$e^x = e^{w+f} = e^{(1+\frac{f}{w})w} = \left[e^{(1+\frac{f}{w})}\right]^w$$

comme $1 \leq (1+\frac{f}{w}) < 2$ la série de Taylor de $e^{(1+\frac{f}{w})}$ converge plutôt rapidement, puis on obtient le résultat en la plaçant à la puissance (entière) de w.

3.2.1 Une somme avec des signes alternés

Nous allons travailler sur la série due à Ramanujan :

$$0 = \sum_{k=0}^{\infty} (-1)^k \frac{(2k+1)^2 + (2k+1)^3}{k!} = \frac{1^2+1^3}{0!} - \frac{3^2+3^3}{1!} + \frac{5^2+5^3}{2!} - \frac{7^2+7^3}{3!} + \cdots$$

Dans un premier temps écrivez le programme qui calcule la série au fur et à mesure en affichant les valeurs à chaque étape de la valeur de k, du numérateur, de la factorielle, de la fraction et de la somme. Il suffit de s'arrêter lorsque la somme ne bouge plus.
Dans un deuxième temps écrivez le programme qui somme séparément les signes positifs et les signes négatifs. En affichant à chaque étape : la valeur de k, la fraction, la somme positive et la somme négative. On s'arrêtera lorsque l'une des deux sommes ne bouge plus et on affichera alors la somme finale.