UNIVERSITÉ DE NANTES

RAPPORT DE RECHERCHE TER

RÉALISÉ PAR :

NAFTI Slimen

Sujet :

CALCULS RIGOUREUX SOUS TABLEUR

Date de soutenance :

29/05/2018

ENCADRANTS :

Christophe Jermann

Frédéric Goualard

Table des matières :

I/Introduction

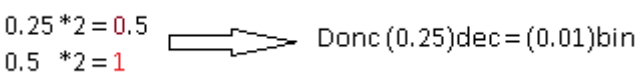
En proposant à un ordinateur un problème en arithmétique, il va s'éloigner méthodiquement, sans relâche, à la vitesse du gigahertz, jusqu'à ce qu'en fin de compte il produise la mauvaise réponse. La cause de ce problème n'est pas que le logiciel soit plein de bugs (bien que cela soit très possible aussi) ou le matériel n'est pas fiable. Le problème est simplement que les ordinateurs sont des machines discrètes et finies, et qu'ils ne peuvent pas faire face à certains des aspects continus et infinis des mathématiques. Même un nombre qui paraît simple comme 1/10 peut causer des problèmes sans fin : Dans la plupart des cas, l'ordinateur ne peut même pas le lire ou l'imprimer exactement, et encore moins effectuer des calculs précis avec un tel nombre.

Tout d'abord, voyons comment une machine affecte des valeurs binaires aux réels. Le réel « 0.25 » par exemple est représenté par le binaire 0.01 . La conversion se fait comme suit:

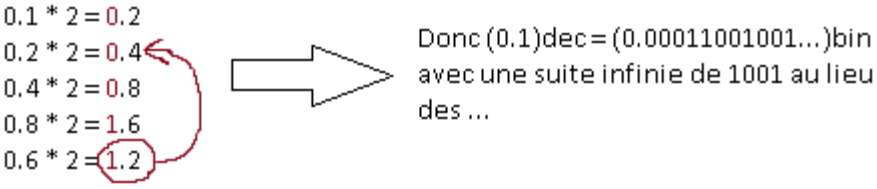
1/ on garde le « 0 » de la partie entière.
2/ On effectue des multiplications par 2 de la partie fractionnaire tout en sauvegardant la nouvelle valeur entière obtenue. Si on finit par trouver la valeur 1.0 alors on a terminé est on a ainsi trouvé une représentation exacte de notre réel.
3/ Sinon si on dépasse 1.0 on remplace la partie entière par 0 et on répète les étapes de 2.
Il est clair donc que ce process de conversion float → binaire peut être infini.

Exemples de conversion dec → bin :

le réel 0.25 :



le réel 0.1 :



Si les étapes précédentes ne se terminent pas (un nombre infini de chiffres après la virgules), la machine est obligée de faire une approximation du résultat à un nombre « n » de chiffres après la virgule qui dépends de l'architecture du langage utilisé. A titre d'exemple, sur une machine typique python utilise 53bits de précision pour représenter les floats. On aura ainsi une valeur stockée qui, malgré qu'elle est proche de la valeur réelle correcte, n'est pas 100 % exacte. Ceci peut causer des erreurs de calculs intolérables.

Dans le cas de « 0.1 » par exemple, malgré le fait que l'inexactitude de la représentation semble trop petite pour poser un problème, des opérations arithmétiques lourdes avec des valeurs qui sont pas tout a fait exactes peuvent causer des désastres dans le cas d'un calcul délicat. Ceci était malheureusement le cas dans une base militaire dans l'Arabie Saoudite. En 25 Février 1991, le système défense antimissile d'une base militaire à Dahrahn a échoué à intercepter un missile Scud. Ceci a été causé par un erreur de calcul. En effet, le système de défense de la base a suivi l'évolution de la trajectoire du missile en considérant « 0.1 » sec comme l'unité de temps et en effectuant des additions ensuite. Mathématiquement, cette procédure est tout à fait exacte. Mais sur machine, « 0.1 » était représenté en approximation binaire sur 24-bit . Malgré que cette représentation était inexacte à l'échelle de 10^{-25}, sur 4 jours elle a causé un décalage de 700 mètres sur la position du missile et 28 soldats ont malheureusement péri.

Les tableurs en particulier sont énormément affectés par les défauts de l'arithmétique flottante. Considérons par exemple une société qui utilise un tableur pour gérer ses stocks de produits et veut savoir la quantité de produits qui reste encore en stock après une année de vente. On suppose qu'elle dispose de 2 produits différents (ces produits peuvent être des vis ou des écrous par exemple) « P1 » et « P2 » et qu'au début de l'année on a fabriqué $4,9*10^{19}$ de « P1 » et $0,1*10^{19}$ de « P2 » et qu'on a vendu au total $5*10^{19}$ produits. La formule mathématique pour calculer la quantité restante est simple : « s »=$(4,9-5+0,1)*10^{19}$. Dans ce cas la valeur de « s » est clairement nulle car (4,9-5+0,1)=0. Essayons maintenant d'utiliser un tableur comme excel pour calculer la

valeur de « s ». On va alors utiliser 3 cellules pour stocker les donnée : A1=4,9;B1=-5;C1=0,1. Et on va mettre le résultat dans la cellule D1 en mettant la formule suivante : D1=SOMME(A1;B1;C1)*10^{19}. Voilà ce qu'on va obtenir :

	A	B	C	D
1	4,9	-5	0,1	3608,22483

Donc si cette société reçoit une commende de 3000 vis par exemple, elle va l'accepter en croyant que son stock courant est suffisant, alors que ce n'est pas du tout le cas.

Les tableurs ont aussi du mal à évaluer l'écart-type d'une variable dont les valeurs sont très proches de la moyenne, ce calcul finira éventuellement par donner des valeurs absurdes . Posons l'exemple d'un investisseur qui compte évaluer des actifs financiers pour savoir si c'est profitable d'investir dessus ou pas. Dans ce cas il faut prendre en compte la volatilité (dont la formule nécessite un calcul d'écart-type) de ces actifs : plus la volatilité d'un actif est élevée et plus l'investissement dans cet actif sera considéré comme risqué et par conséquent plus l'espérance de gain (ou risque de perte) sera importante. L'utilisation d'un tableur, dans ce cas, pourra ainsi aboutir à une valeur de volatilité incorrecte. Et étant donné que la décision de l'investisseur sera basée sur cette valeur. Ce dernier pourra subir des pertes financières inattendues.

On a ainsi illustré l'insuffisance de l'arithmétique flottante. Heureusement, il existe une solution potentielle (substitut pour être plus exact) qui, malgré le fait qu'elle aussi est incapable de donner la valeur exacte de «s» ou la représentation exacte de « 0.1 », pourra nous donner des encadrements avec des bornes exactes (des bornes qui sont représentées d'une manière exacte par machine comme « 0.25 ») qui sont garanties de contenir le résultat exact. C'est l'arithmétique d'intervalles.

II/L’arithmétique d’intervalles

1/ État de l’art :

En mathématiques et en informatique, l'arithmétique des intervalles est une approche de calcul qui manipule des intervalles, par opposition à des nombres (entiers ou flottants),pour aboutir à des résultats plus rigoureux. Cette approche permet de borner les erreurs d'arrondi et ainsi de développer des méthodes numériques qui fournissent des résultats fiables. Donc, l’équivalent d’un nombre flottant « a » en arithmétiques intervalles sera l’intervalle de bornes flottantes $[a^-,a^+]$ avec :
a^- : le plus grand float correctement (d’une manière finie) représentable par machine tq $a^-<a$.
a^+ : le plus petit float correctement (d’une manière finie) représentable par machine tq $a<a^+$.
Il est clair donc que si un calcul intervalle aboutit à un intervalle avec un écart important entre les bornes, alors le même calcul effectué en arithmétique flottante aura donné un résultat incorrect.Mais au moins le résultat donné par un calcul intervalles va toujours contenir le résultat correct.

L’arithmétique d’intervalles n’est pas une nouvelle idée, elle a été inventée et réinventée plusieurs fois. Et malgré ça elle n’a jamais été un outil dominant dans l'informatique numérique, et pourtant elle n'a jamais été abandonnée ou oubliée non plus. En 1931, un docteur de l’université de cambridge du nom de Rosalind Cicely Young a publié un article nommé “algebra of manyvaluedquantities” qui expliquait les règles de calculs avec intervalles et les comparait avec les calculs flottants. Bien sûr, Young et d'autres qui écrivaient à cette époque ne voyaient pas les intervalles comme une aide à l'amélioration de la fiabilité du calcul machine. En 1951, cependant, dans un manuel sur l'algèbre linéaire, Paul S. Dwyer de l'Université du Michigan décrivait l'arithmétique avec des intervalles (il les appelait «nombres de rangées») d'une manière clairement orientée vers les besoins de calcul avec les appareils numériques . Quelques années plus tard, les idées essentielles de l'arithmétique par intervalles ont été exposées indépendamment et presque simultanément par trois mathématiciens - Mieczyslaw Warmus en Pologne, Teruo Sunaga au Japon et Ramon E. Moore aux États-Unis. La version de Moore a été la plus influente, en partie parce qu'il a mis l'accent sur des solutions aux problèmes de calcul machine mais aussi parce qu'il a continué pendant plus de 40 ans à publier sur des méthodes d'intervalle et à promouvoir leur utilisation.

Aujourd'hui, la communauté des méthodes d'intervalle comprend des groupes de recherche actifs dans quelques dizaines d'universités. Un site Web de l'Université du Texas à El Paso (www.cs.utep.edu/interval-comp) fournit des liens vers ces groupes ainsi qu'une archive utile de documents historiques. La revue Reliable Computing (anciennement Interval Computations) est la principale publication du domaine; Il existe également des listes de diffusion et des conférences annuelles. Les implémentations de l'arithmétique par intervalles sont disponibles à la fois en tant que langages de programmation spécialisés et en tant que bibliothèques pouvant être liées à un programme écrit dans une langue standard.

2/ Représentation des nombres dans le cas d'un calcul intervalles:

Voyons alors comment seront représentées les valeurs de « 0.1 » et 0.25 (On va laisser la valeur de « s » pour plus tard) dans le cas d'utilisation de l'arithmétique d'intervalles (bibliothèque pyinterval de python dont un lien vers sa documentation est donné dans la partie bibleographie).

*** « 0.1 »

Dans ce cas les bornes de l'intervalle seront [a,b] tq :

. a est le nombre le plus proche de 0.1 tq a<0.1 et a est correctement représentable par la machine.

. b est le nombre le plus proche de 0.1 tq b>0.1 et b est correctement représentable par la machine.

Avec python par exemple on va finir par trouver la représentation suivante pour « 0.1 » :

0.1=[0.09999999999999999,0.1]

Ici il est clair que 0.1 > 0.09999999999999999 , mais pourquoi a-t-on gardé « 0.1 » comme borne supérieure de l'intervalle ?

Ceci est du au faite que python représente tous les floats qui n'ont pas une représentation binaire finie avec 53bits de précision, « 0.1 » en particulier se représente de la manière suivante :

$(0.1)_{déc}=(0.0001100110011001100110011001100110011001100110011001100110011010)_{bin}$

On remarque qu'à la fin de ce nombre , le 1001 est remplacé par 1010, ce qui va rendre ce nombre fini supérieur strictement à la représentation infinie de 0,1.

On aura alors $(0.1)_{déc} < (0.1)_{bin\ sur\ python}$.

***« 0.25 »

Les choses deviennent plus simple dans ce cas, « 0.25 »étant correctement représentable sur machine, sa représentation intervalle sera alors l'intervalle point [0.25, 0.25]

3/ Opérations en arithmétique d'intervalles :

Définition abstraite:

Le résultat d'une opération « # » entre **X** et **Y** est **X** # **Y**={x # y | x∈**X** , y∈**Y**}

Voici quelques exemples :

$$
\begin{array}{lcl}
[\underline{x}, \overline{x}] + [\underline{y}, \overline{y}] & = & [\underline{x} + \underline{y}, \overline{x} + \overline{y}] \\
[\underline{x}, \overline{x}] - [\underline{y}, \overline{y}] & = & [\underline{x} - \overline{y}, \overline{x} - \underline{y}] \\
[\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}] & = & [\min(\underline{x} * \underline{y}, \underline{x} * \overline{y}, \overline{x} * \underline{y}, \overline{x} * \overline{y}), \max(\mathsf{idem})] \\
[\underline{x}, \overline{x}]^2 & = & [\min(\underline{x}^2, \overline{x}^2), \max(\underline{x}^2, \overline{x}^2)] \ \text{si } 0 \notin [\underline{x}, \overline{x}] \\
 & & \text{et } [0, \max(\underline{x}^2, \overline{x}^2)] \ \text{sinon} \\
1/[\underline{y}, \overline{y}] & = & [\min(1/\underline{y}, 1/\overline{y}), \max(1/\underline{y}, 1/\overline{y})] \ \text{si } 0 \notin [\underline{y}, \overline{y}] \\
[\underline{x}, \overline{x}] / [\underline{y}, \overline{y}] & = & [\underline{x}, \overline{x}] \times (1/[\underline{y}, \overline{y}]) \ \text{si } 0 \notin [\underline{y}, \overline{y}]
\end{array}
$$

Généralement, des tableurs (Libre Office, Exel ...) sont utilisés par la quasi-totalité des entreprises pour réaliser leurs calculs. C'est pour cela que le sujet de mon TER se base sur la création d'une extension pour LibreOffice qui rend possible l'utilisation d'arithmétique d'intervalles.

III/ Plugin LibreITV pour libre office

1/ Pourquoi libre office ? :

On a voulu effectuer notre travail sur un logiciel open-source, ce qui n'était pas le cas pour Excel. On avait donc le choix entre Libre office ou le Tableur python.

On a décidé de rester sur Libre office car ça aura plus d'impact vu qu'il est plus connu et que ça base d'utilisateurs est plus grande que le tableur python. Malgré que ce dernier offre une intégration beaucoup plus souple de la bibliothèque pyinterval. On expliquera ça plus en détail dans la section « Difficultés rencontrées ».

2/Qu'est-ce qu’un plugin libre office

En informatique un *plugin* ou *plug-in*, aussi nommé *add-in* ou *add-on* en France, est un paquet qui complète un logiciel hôte pour lui apporter de nouvelles fonctionnalités. Dans le cas d’un tableur, Un "add-in" est un type d'extension permettant l'intégration complète de fonctions personnalisées dans le module Calc. Une fonction ajoutée comme add-in est ainsi disponible depuis l'assistant de fonctions et expose ses paramètres dans la barre de formule, comme une fonction native. La création du plugin libre office est possible en python, java ou C++. Nous avons choisi python car nous avions décidé d’utiliser la bibliothèque pyinterval pour définir les fonctions du plugin . Cette bibliothèque fournit une implémentation Python des fonctions mathématiques et logiques de l’arithmétique intervalle.Il y’aura une explication détaillée du process de création de add-in LibreOffice dans la section annexes.

3/Installation de LibreITV

J’ai choisi de nommer mon plugin « LibreITV ». Pour l’installer, il suffit juste d’avoir libre office version 2.0 ou supérieur, d’avoir python 3 installé et d’installer la bibleotheque pyinterval via les commandes :

***** pip install --upgrade pip

***** pip install pyinterval

Il faut noter que pour utiliser ce plugin sur Windows, il y aura du travail à faire. Car l’installation de la bibliothèque pyinterval sur Windows n’est pas évidente.

IV/Fonctionnement du plugin

1/Entrées et sorties des fonctions

Le contenu d'une cellule doit être comme suit :

- a.b ou a.b ;a.b si l'on veut mettre le réel a,b car le contenu d'une cellule doit être du texte (d'où l'utilisation du point au lieu de la virgule).
- Le nombre directement si l'on veut mettre un entier. Exemples: 5,6,-200
- a.b;c.d si l'on veut mettre l'intervalle [a,b , c,d] qui aura comme bornes les 2 réels « a,b » et « c,d».

  Exemples :

  5.3;6.99 si l'on veut mettre [5,3 , 6,99]

  -inf;5.99 si l'on veut mettre [-oo , 5,99]
- nan;nan si l'on veut mettre l'intervalle vide.

Le contenu d'une cellule sera ainsi converti en type intervalle pour rendre possible l'utilisation des fonctions de la bibpytinterval. Ce process se fait comme suit :

- Si la cellule contient une chaîne de la forme a;b, alors le constructeur de la classe intervalle sera appelé directement pour créer un intervalle équivalent à la chaîne introduite en entrée a variable a traiter dans ce cas sera égale à V=Interval([a,b]). V est viable en entrée pour toutes les fonctions de la bibliothèque pyinterval.
- Si la cellule contient un nombre flottant « x », il sera automatiquement converti en Intervalle. Le process est plus délicat dans ce cas car il faut bien choisir les bornes de l'intervalle de manière qu'il soit le plus petit possible et qu'il est sure de contenir « x ».

  Dans ce cas on va utiliser les fonctions « up » et « down » de « interval.fpu ». Ces fonctions vont arrondir « x » pour obtenir les bornes $[x^-,x^+]$ de l'intervalle de façon que ces approximations restent le plus près possible de la valeur de « x ».

  Exemples: On va reprendre l 'exemple de « 0.1 » :

  x^-=down(lambda : float(« 0,1 »))= 0.09999999999999999

  x^+=up(lambda : float(« 0,1 »))=0.1
- Une cellule vide sera ignorée durant le calcul.

Toutes les fonctions du plugin LibreITV vont donner un résultat de la forme a;b. Ce résultat sera donc utilisable en entrée pour les autres fonctions.

2/Fonctions de libreITV

Une liste qui contient toutes les fonctions de LibreITV est fournie dans la section annexes. Voici quand même quelques fonctions qui sont présentes dans LibreITV avec des exemples d'utilisation :

- SUMItv (ensemble de plage de cellules)

Elle prends en paramètre une liste de plage de cellules pour retourner le résultat de l'addition des contenus des cellules.

	C	D	E
62	5564.1;8000.8	-83.7;-10	5380.319999999998;7999.020000000002
63	-99;1.1	6.32	
64	-10.5;-2.3	3.1	

Ici E62=SUMITV(C62:C64;D62:D64)

- DIFFItv (cellule 1 ; cellule 2)

Elle prends en paramètre 2 cellules pour retourner le résultat de la soustraction des contenus des 2 cellules (cellule 1 – cellule 2).

	C	D	E
64	10	3.1	6.8999999999999995;6.9

E64=DIFFITV(C64;D64)

3/Utilité de LibreITV

Malgré que le calcul avec pyinterval pourra être imprécis, mais l'intervalle résultat qu'on aura est toujours garanti de contenir la valeur flottante correcte du calcul. Reprenons l'exemple de la société évoqué dans l'introduction. Un calcul avec LibreITV va donner le résultat suivant :

	C	D	E	F
67	4.9	-5	0.1	-5412.337245047637;3608.2248300317588
68			1,00E+19	

E67=PRODUCTITV(SUMITV(C67:E67);E68)

On remarque ici que, malgré le fait d'être parti avec des flottants en entrée, on a fini par avoir comme résultat un intervalle avec 2 bornes très écartées l'une de l'autre. Ceci indique qu'il y avait eu des erreurs de calcul liées aux approximations. Et malgré qu'on n'a pas une valeur précise du résultat (qui est normalement égale à 0), on sait qu'elle est comprise entre -5412.337245047637 et 3608.2248300317588. Et ceci est beaucoup mieux que le calcul initial (voir introduction) qui donne une valeur flottante incorrecte.

V/Difficultés rencontrées

Durant ce TER, j'ai rencontré plusieurs difficultés. Elles étaient principalement liées au fait que LibreOffice n'était pas vraiment « souple » pour modification et ajout de contenu . Au début nous avons voulu que l'intégration de la bibliothèque pyinterval se fait via la surcharge des opérateurs déjà existants de LibreOffice, et cela aurait été idéal. Si cela était possible, on aura pu garder les mêmes fonctions prédéfinies de LibreOffice tout en les modifiants pour être aussi applicables à un nouveau type « intervalle » qu'on a aimé créer. Comme ça l'utilisateur pourra facilement basculer de l'arithmétique d'intervalle vers l'arithmétique flottante. Dans ce cas, à titre d'exemple, la fonction somme (de LibreOffice) qui est applicable à une liste de plage de cellules va donner un résultat intervalle dès qu'il y a une valeur de type « intervalle » dans l'ensemble de cellules passées en paramètres. Cela n'était malheureusement pas possible à cause de plusieurs raisons. Principalement c'était dû au fait que LibreOffice n'accepte pas la surcharge des opérateurs . En effet, nous avons essayé de créer un plugin minimal qui contenait une fonction nommée somme (qui est la fonction somme de LibreOffice). Mais ceci a causé des bugs dans l'interface d'insertion des fonctions. J'ai alors cherché sur le forum officiel de LibreOffice et j'ai appris que, non seulement la surcharge des opérateurs originaux de LibreOffice est impossible, mais la création d'un nouveau type de variable l'est aussi. D'où le recours vers la représentation sous forme de String qui, malgré qu'elle ne pose pas un problème pour le fonctionnement des calculs du plugin, n'est pas très pratique pour l'introduction des données.

Pour simplifier les choses pour l'utilisateur, nous avons choisi de garder les noms des fonctions prédéfinies de LibreOffice tout en ajoutant un suffixe Itv (SUMItv,DIFFItv,etc..). Une liste avec toutes les fonctions du plugin est fournie dans la section annexes.

VI/ Conclusion

Ce travail fût une expérience vraiment enrichissante malgré toutes les limites imposées par le choix de LibreOffice qui, malgré que c'était le bon choix pour que mon travail aura plus d'impact, n'était pas souple pour l'ajout de contenu (problèmes de surcharge des opérateurs et de création de nouveau type évoqués dans la section Difficultés rencontrée).

Le fait de ne pas pouvoir créer un nouveau type dans LibreOffice m'a rendu incapable de créer un affichage personnalisé pour les valeurs de cellules. On ne peut pas donc choisir la précision de l'affichage (nombre de cellules après la virgule, etc..).

D'autre part, l'utilisation de ce plugin n'est pas limitée pour détecter les erreurs de calculs flottants. On peut aussi l'utiliser pour faire des calculs avec des valeurs incertaines. Par exemple si l'on veut estimer le profit d'une société pour l'année prochaine. En supposant que cette société vends un produit « P » dont la quantité en stock est « st ». La quantité qui sera vendue « qv » est alors comprise entre [0,st]. On pourra ainsi se baser sur cet encadrement de « qv » pour pouvoir estimer le gain/perte potentiels de cette société.

Pour résumer, posons-nous la question suivante : Serions-nous mieux si les intervalles étaient utilisés pour tous les calculs ? Peut-être, mais imaginez le sort du soldat sur le terrain: Un missile doit être tiré si et seulement si une cible entre dans une portée de 5 kilomètres, et l'ordinateur équipé d'intervalles signale que la distance est [4,6] kilomètres. C'est un peu comme les prévisions météo qui promettent une probabilité de 50% de pluie. De telles déclarations peuvent refléter avec précision notre véritable état de connaissance, mais elles ne sont pas d'une grande aide lorsque pour le pauvre soldat qui doit décider : «Est-ce que je tire ou non ? ». Ce problème pourra

ainsi devenir un problème psychologique plus qu'un problème mathématique. Peut-être la solution est de calculer avec des intervalles et les flottants en parallèle, mais à la fin, laissez la machine donner une réponse précise et ponctuelle. Le résultat du calcul intervalles ne sera pas rejeté, il va être consulté par la machine pour voir l'écart entre les bornes. Si cet écart dépasse la précision désirée par l'utilisateur, La machine doit avertir ce dernier.

VII/Annexes

1/Comment créer un plugin Libre office ? :

Un plugin Libre office est une archive au format .oxt. Elle est constituée de :

1. un fichier binaire (extension .rdb) permettant d'ajouter à la base de registre la définition des fonctions personnelles ("interface")
2. un fichier xml (extension .xcu) contenant les informations requises pour l'interface utilisateur : noms et descriptions des fonctions, noms et descriptions des paramètres, localisation, etc.
3. le fichier d'implémentation (en python,java ou c++) qui contient le corps des fonctions du plugin.
4. un fichier description.xml (facultatif) contenant les éléments d'information relatifs à l'extension .
5. un sous-répertoire META-INF contenant un fichier manifest.xml, lequel permet au programme d'identifier les éléments précités.

Après avoir créé ces fichiers, il suffit de tout compresser dans une archive .zip, et de changer ensuite l'extension de .zip à .oxt. Pour l'installation d'un plugin il suffit d'exécuter le .oxt.

2/Fonctions de LibreITV

SUMItv (liste de plage de cellules)	Elle prend en paramètre une liste de plage de cellules pour retourner le résultat de la somme des contenus des cellules.
DIFFItv (cellule 1 ; cellule 2)	Elle prend en paramètre 2 cellules pour retourner le résultat de la soustraction des contenus des 2 cellules (cellule 1 – cellule 2).
PRODUCTItv (liste de plage de cellules)	Elle prend en paramètre une liste de plage de cellules pour retourner le résultat de la multiplication des contenus des cellules.
DIVItv (cellule 1 ; cellule 2)	Elle prend en paramètre 2 cellules pour retourner le résultat de la division des contenus des 2 cellules (cellule 1 / cellule 2).
INTERItv (liste de plage de cellules)	Elle prend en paramètre une liste de plage de cellules pour retourner le résultat de l'intersection des contenus des cellules.
UNIONItv (liste de plage de cellules)	Elle prend en paramètre une liste de plage de cellules pour retourner le résultat de l'union des contenus des cellules.
INItv(cellule 1 ; cellule 2)	Elle prend en paramètre 2 cellules pour tester si la cellule 1 est incluse dans cellule 2.
POWItv(cellule 1 ; cellule 2)	Elle prend en paramètre 2 cellules pour retourner le résultat de cellule $1^{\text{cellule 2}}$.
ABSItv(cellule)	Retourne la valeur absolue du contenu d'une cellule.
TANItv (cellule)	Retourne la valeur de la tangente du contenu d'une cellule.
TANHItv (cellule)	Retourne la valeur de la tangente hyperbolique du contenu d'une cellule.
TANPIItv (cellule)	Retourne la valeur de la tangentepi du contenu d'une cellule.
ATANItv (cellule)	Retourne la valeur de l'arc tangente du contenu d'une cellule.
ATANPIItv (cellule)	Retourne la valeur de l'arc tangentepi du contenu d'une cellule.

COSItv (cellule)	Retourne la valeur du cosinus du contenu d'une cellule.
COSHItv (cellule)	Retourne la valeur du cosinus hyperbolique du contenu d'une cellule.
COSPIItv (cellule)	Retourne la valeur du cosinuspi du contenu d'une cellule.
EXPItv (cellule)	Retourne la valeur de l'exp du contenu d'une cellule.
EXPM1Itv (cellule)	Retourne la valeur de l'expM1 du contenu d'une cellule.
LOGItv (cellule)	Retourne la valeur du log du contenu d'une cellule.
LOGDIXItv (cellule)	Retourne la valeur du logbase10 du contenu d'une cellule.
LOGUNPIItv (cellule)	Retourne la valeur du logunpi du contenu d'une cellule.
LOGDEUXItv (cellule)	Retourne la valeur du logbase2 du contenu d'une cellule.
SINItv (cellule)	Retourne la valeur du sinus du contenu d'une cellule.
SINHItv (cellule)	Retourne la valeur du sinus hyperbolique du contenu d'une cellule.
SINPIItv (cellule)	Retourne la valeur du sinuspi du contenu d'une cellule.
HULLItv(liste de plage de cellules)	Retourne le plus petit intervalle [a,b] qui contient tous les intervalles de l'entrée.(1*)
COMPITv(cellule1;cellule2)	Compare l'intervalle contenu dans la cellule 1 avec celui de la cellule 2.(2*)

(1*) **E**xemple d'utilisation de HULLItv:

HULLItv([1;2];[5;6])=[1;6].

(2*) **L**a fonction COMPItv compare deux intervalles I1=[a,b] et I2=[c,d] de la manière suivante :

. COMPItv(I1;I2)=1 ssi I1 est strictement supérieur à I2. C'est à dire quelque soit x un élément de I1, on aura toujours x>y quel que soit y un élément de I2.

. COMPItv(I1;I2)=2 ssi I1 est peut-être supérieur à I2. C'est à dire quel que soit y un élément de I2, il existe x un élément de I1 tel que x>=y.

. COMPItv(I1;I2)=3 ssi I1=I2 c'est-à-dire a=c et b=d.

. COMPItv(I1;I2)=4 ssi I1 est strictement inférieur à I2. C'est à dire quelque soit x un élément de I1, on aura toujours x<y quel que soit y un élément de I2.

. COMPItv(I1;I2)=5 ssi I1 est peut-être inférieur à I2. C'est à dire quel que soit x un élément de I1, il existe y un élément de I2 tel que x<=y.

3/Bibliographie

- Documentation de pyinterval. http://pyinterval.readthedocs.io/en/latest/
- Chiriaev, Dmitri, and G. William Walster. 1998. Intervalarithmetic specification. http://www.mscs.mu.edu/~globsol/Papers/spec.ps
- Markov, Svetoslav, and Kohshi Okumura. 1999. The contri-bution of T. Sunaga to interval analysis and reliable computing. In Developments in Reliable Computing, Tibor Csendes (ed.), pp. 167–188. Dordrecht, Boston: Kluwer.
- COMPUTING SCIENCE A LUCID INTERVAL - Brian Hayes - American Scientist, Volume 91.
- Semenov, Alexander L. 1996. Solving optimization problems with help of the Unicalc solver. In Applications of Interval Computations, R. Baker Kearfott and Vladik Kreinovich (eds.), pp. 211–225; Dordrecht, Boston: Kluwer Academic.
- Skeel, Robert. 1992. Roundoff error and the Patriot missile.SIAM News 25(4):11.
- Sunaga, Teruo. 1958. Theory of interval algebra and its application to numerical analysis. In RAAG Memoirs, Ggujutsu Bunken Fukuy-kai. Tokyo, Vol. 2, pp. 29–46. Also at http://www.cs.utep.edu/interval-comp/sunaga.pdf
- van Emden, M. H. 2002. New developments in interval arithmetic and their implications for floating-point standardization. http://arXiv.org/abs/cs.NA/0210015
- Walster, G. William. 1996. Stimulating hardware and software support for interval arithmetic. In Applications of Interval Computations, R. Baker Kearfott and Vladik Kreinovich (eds.), pp. 405–416; Dordrecht, Boston: Kluwer Academic.

- Warmus, M. 1956. Calculus of approximations. Bulletin de l'Academie Polonaise des Sciences 4(5):253–257. Also at http://www.cs.utep.edu/interval-comp/warmus.pdf
- United States General Accounting Office. 1992. Patriot Missile Defense: Software Problem Led to System Failure at Dhahran, Saudi Arabia. Washington: General Accounting Office.
- Young, Rosalind Cecily. 1931. The algebra of many-valued quantities. Mathematische Annalen 104:260–290. Also at http://www.cs.utep.edu/interval-comp/young.pdf
- Powell SG, Baker KR, Lawson B (2009). “Impact of Errors in Operational Spreadsheets.” Decision Support Systems, 47, 126–132.

- Yalta AT, Yalta AY (2007). “GRETL 1.6.0 and Its Numerical Accuracy.” Journal of Applied Econometrics, 22(4), 849–854.
- Zhang WJ, Xie XF (2003). “DEPSO: Hybrid Particle Swarm with Differential Evolution Operator.” In IEEE International Conference on Systems, Man and Cybernetics, volume 4, pp. 3816–3821.
- Zoethout RWM, van Gerven JMA, Dumont GJH, Paltansing S, van Burgel ND, van der Linden M, Dahan A, Cohen AF, Schoemaker RC (2008). “A Comparative Study of Two Methods for Attaining.

Étant donné que le sujet de création de plugin pour libre office n’est pas bien documenté sur internet, j’ai crée plusieurs sujets dans le forum LibreOffice pour demander l’aide de la communauté. https://fr.libreoffice.org/get-help/nabble/