POLYTECH SORBONNE - MAIN3

Projet d'initiation

Bibliothèque efficace d'arithmétique d'intervalle en C

BEN ALI Hala
LIU Vincent
NGUYEN Quoc Son
SAYAH Samir

Encadrant : Stef Graillat
Coordinatrice : Cecile Braunstein

2018

Table des matières

Remerciement

Nous tenons à remercier notre encadrant Monsieur **Stef Graillat** pour avoir contribué au succès de notre projet et de nous avoir fait partager son expertise à chaque séance. Grâce à lui, nous avons beaucoup appris sur la représentation des nombres en machine et l'arithmétique d'intervalles.

Nous remercions également notre coordinatrice **Cécile Braunstein** pour avoir organisé le module du projet d'initiation et avoir été présente lorsqu'il y avait nécessité.

Nous remercions également nos camarades de classe qui ont effectué le peer-review sérieusement. Cela a été très important pour nous d'avoir des critiques constructives. Nous avons alors pu améliorer notre rapport et éclaircir certaines notions ou explications qui n'étaient pas claires.

Motivation

La représentation des nombres en machine fait l'objet de nombreuses études de nos jours. La vitesse, la précision et la fiabilité des calculs sont primordiales. Elles le sont d'autant plus lorsque le domaine d'application est critique (aéronautique, finance, aérospatial). Quand on parle de précision on pense immédiatement aux nombres réels ou nombres à virgule flottante en informatique. Notre projet va porter sur la représentation de ces nombres en machine.

La norme la plus utilisée de nos jours pour représenter ces nombres est la norme IEEE 754. Bien que cette norme soit très pratique, elle présente des inconvénients et les résultats des calculs effectués ne sont pas toujours fiables. L'alternative que nous avons choisi ici est d'effectuer des calculs non pas sur des nombres, mais sur des intervalles. En mathématiques et en informatique, l'arithmétique d'intervalle est une méthode de calcul consistant à manipuler des intervalles dans le but d'obtenir des résultats rigoureux.

Cette approche permet de borner les erreurs d'arrondi et ainsi de développer des méthodes numériques **qui fournissent des résultats fiables**.
Dans l'arithmétique d'intervalle, un nombre réel x est représenté par une paire de nombres flottants (x_{inf}, x_{sup}). Dire que x est représenté par cette paire signifie que x appartient à lintervalle [x_{inf}, x_{sup}]. On parle de la notation $inf - sup$. Il existe une autre façon de représenter les intervalles. On peut définir un intervalle par son centre et son rayon. On note $< c, r >$ avec $r > 0$ l'intervalle $[c-r, c+r]$. On parle alors de notation centre-rayon.
On voit qu'il est nécessaire d'utiliser deux nombres pour coder un intervalle. Le travail de ce projet serait de définir un format "compressé" pour stocker les intervalles. Nous allons proposer un nouveau format pour représenter les intervalles. Ce qui va changer sera la représentation que l'on se fait d'un nombre binaire, un nombre ne codera plus un réel mais un **intervalle**.

Notre but sera donc d'implémenter une bibliothèques en langage C avec un format permettant de manipuler des intervalles et de créer toutes les fonctions nécessaires au calcul d'intervalles, à savoir : les opérations de base et les fonctions usuelles.

1 Représentation des nombres

1.1 La norme IEEE 754

Sur machine, il est impossible de représenter $\mathbb{R}$ tout entier. On peut seulement représenter un sous-ensemble $\mathbb{F}$ de cardinal fini de $\mathbb{R}$. Les nombres qui appartiennent à $\mathbb{F}$ sont appelés nombres à virgule flottante.
La norme IEEE 754 permet de représenter les nombres à virgule flottante en binaire avec 3 valeurs : le signe, l'exposant et la mantisse.

Soit $x \in \mathbb{F}$, alors :

$$\boxed{x = (-1)^{Signe} \times 1, Mantisse \times 2^{exposant}} \text{ où } exposant = E - decalage$$

Le bit de poids fort est le bit de signe. Le bit de signe vaut 0 si le nombre est positif, 1 si le nombre est négatif.
L'exposant du nombre peut être positif ou négatif. Pour représenter un nombre signé en binaire en général, on peut utiliser le complément à deux ou bien le décalage. Dans la norme IEEE 754, on n'utilise pas le complément à deux pour représenter l'exposant, mais le décalage.
L'exposant est décalé de telle manière à ce qu'il soit toujours positif. On va stocker l'exposant dans un entier non signé même si il est négatif ! Il est codé sur 8 bits en précision simple, et sur 11 bits en précision double. L'exposant décalé est obtenu en ajoutant le décalage à l'exposant initiale. Le décalage vaut 127 en précision simple, et 1023 en précision double.
La mantisse représente les chiffres après la virgule. Les nombres jusqu'au dernier 1 sont appelés les chiffres significatifs de x. Ce nombre est codé sous 23 bits en simple précision et 64 bits en double précision.

Précision	Signe	E	Mantisse	Décalage
simple(32 bits)	1 bit	8 bits	23 bits	127
double(64 bits)	1 bit	11 bits	52 bits	1023

La représentation d'un nombre est unique sous la norme IEEE 754.

Exemple

Comment représenter le nombre décimal 8,625 en norme IEEE 754 et en précision simple ?

1. 8,625 est positif, donc le bit de signe vaut **0**.
2. 8,625 en binaire vaut : $8,625_{10}$=$1000,101_2$ ($8,625 = 2^3 + 2^{-1} + 2^{-3}$).
3. 1000,101 en écriture scientifique vaut : $1000,101_2 = 1,\mathbf{000101}.2^{\mathbf{3}}$.
4. Les chiffres après la virgule sont les chiffres significatifs : **000101**.
   La mantisse vaut : **000101**00000000000000000 (Les 0 sont ajoutés à droite après les chiffres significatifs pour obtenir une mantisse de taille 23).
5. L'exposant vaut **3**.
   L'exposant décalé vaut : E = exposant + décalage = 3 + 127 = 130.
   (en précision simple, décalage = 127).
   E en binaire vaut : **10000010**.

La représentation de $8,625_{10}$ en norme IEEE 754 est :

Signe	E	Mantisse
0	**10000010**	**000101**00000000000000000

1.1.1 Représentation des nombres spéciaux et modes d'arrondi

Il y a deux zéros et deux infinis en norme IEEE 754 en simple précision :

Nombre	Signe	E	Mantisse
∞	0	11111111	00000000000000000000000
$+0$	0	00000000	00000000000000000000000
-0	1	00000000	00000000000000000000000
$-\infty$	1	11111111	00000000000000000000000

Une opération entre deux nombres qui appartient à l'ensemble $\mathbb{F}$ n'appartient pas forcement à $\mathbb{F}$, il faut alors arrondir le résultat.
La norme IEEE 754 dispose de quatre modes d'arrondi :

— Vers moins l'infini

— Vers plus l'infini

— Vers zéro

— Au plus près

La manipulation de ces modes d'arrondi nous sera très utile par la suite, en particulier pour borner les éventuelles erreurs d'arrondi.

1.1.2 Erreurs d'arrondi, absorption et cancellation

Lorsqu'on fait des calculs sur machine, la précision des nombres est limitée, la machine renvoie un nombre arrondi du résultat exact du calcul, car il ne peut renvoyer qu'un nombre fixe de chiffres selon le nombre de bits sur lequel le nombre est codé. L'erreur d'arrondi produite quand on remplace un réel x par son représentant $fl(x)$ dans $\mathbb{F}$ est bornée.
Le nombre $u = 2^{-53}$ est l'erreur relative maximale que l'ordinateur peut commettre en représentant un nombre réel en arithmétique finie. Ce nombre va beaucoup nous servir pour borner les erreurs d'arrondi lors de nos calculs d'intervalles.

Les erreurs d'arrondi sont en général négligeables, mais peuvent s'accumuler au cours de longs algorithmes. Une grande accumulation d'erreurs d'arrondi peut fausser un résultat et c'est ici tout l'intérêt de coder un nombre avec un intervalle dans lequel on sera sûr de retrouver le résultat de notre calcul.
En plus des erreurs dues à l'arithmétique, les deux sources principales d'erreurs d'arrondi sont : l'absorption et la cancellation.
Lorsqu'on effectue le calcul $x + y$, dès que y est beaucoup plus petit que x alors $x + y$ est arrondi à x, on dit qu'il y a eu absorption : y a été absorbé par x.
La cancellation survient lors de la soustraction de deux nombres relativement proches, lorsqu'il n'y a presque plus de bits significatifs, on parle de cancellation catastrophique.

1.1.3 Perte de l'associativité

L'ensemble $\mathbb{F}$ ne possède pas les même caractéristiques que $\mathbb{R}$: en passant du premier au second, on perd l'associativité caractéristique des nombres réels.
Il faut en fait faire attention à l'ordre lorsqu'on fait des opérations sur des nombres, surtout les nombres qui ont des ordres de grandeur très différents. Cela peut provoquer des résultats incohérents.

Exemple

Pour $2^{-40} + 2^2$, la machine va renvoyer 4. Ainsi si on fait : $(2^{-40} + 2^2) - 2^2$, la machine va renvoyer 0 au lieu de 2^{-40} en simple précision, c'est le phénomène d'absorption.
De la même façon, le calcul de $(2 \times 10^{20} + 2.5) - 2 \times 10^{20}$ renvoie 0 au lieu de 2.5, le résultat est alors complètement faux, il y a eu cancellation catastrophique.

1.2 L'arithmétique d'intervalles

Le principe de l'arithmétique d'intervalles consiste à remplacer tous les nombres par des intervalles les contenant. Ainsi comme nous l'avons mentionné auparavant, deux notations existent. La notation inf-sup classique qui consiste à représenter un intervalle sous la forme : $I = [c - r, c + r]$ c étant le centre de l'intervalle et r étant son rayon (par exemple [0,6]). Cette notation présente l'inconvénient de devoir stocker deux nombres en machine, ce n'est donc pas celle qui va nous intéresser. L'autre notation possible est la notation centre rayon qui peut s'écrire de la manière suivante : $I =< c, r >$ (avec le même exemple : <3,3>).

1.2.1 La représentation FP-Intervalle

La représentation FP-Intervalle que nous proposons consiste à coder le centre et le rayon d'un intervalle $I = [c - r, c + r] =< c, r >$ en un seul nombre. Nous allons utiliser le format IEEE 754.
À partir du centre c de l'intervalle, nous allons modifier la mantisse de sorte à ajouter une information supplémentaire sur la valeur du rayon. La nouvelle mantisse permet d'avoir un nouveau nombre qui peut être interpréter comme un FP-Intervalle ou un flottant.

Nous allons présenter deux méthodes :

— Coder un FP-Intervalle à partir d'un centre et d'un rayon.

— Extraire le centre et le rayon à partir d'un FP-Intervalle.

Comment coder un intervalle à partir du centre et du rayon ?

1. Coder le centre c sous le format IEEE 754. (C'est-à-dire connaître les nombres s, E et m)
2. Ajouter un **1** après les derniers chiffres significatifs de la mantisse m.
   La position du **1** va indiquer la valeur du rayon.
   On a : $r = 2^n$ où $n = exposant - position$.

Exemple

Comment coder l'intervalle $I =< 8,625, 2^{-4} >$?

1. Coder le centre $c = 8,625$. On a vu dans l'exemple précédent que $c = 8,625$ peut se décomposer ainsi :

Nombre	Signe	E	Mantisse m
8,625	**0**	**10000010**	**000101**00000000000000000

2. Coder le rayon $r = 2^{-4}$.
   L'exposant du centre est égale à 3. Il faut alors mettre le **1** à la position 7 ($= 3 - (-4)$) dans la mantisse.

Intervalle	Signe	E	Mantisse m'
I	**0**	**10000010**	**000101**1**0000000000000000**

   Les 0 après le **1** en rouge n'ont pas de signification particulière.
   On peut aussi donc écrire les intervalles sous cette forme :

Intervalle	Signe	E	Mantisse m'
I	**0**	**10000010**	**000101**_____

   Ou en écriture scientifique :

Intervalle	Ecriture Scientifique
I	$+1.000101\_\_\_\_\_ * 2^3$

Il est important de comprendre que le nouveau nombre caractérisé par le signe S, l'exposant décalé E et la nouvelle mantisse m' est interprété comme un FP-Intervalle, on a $I =< 8,625, 2^{-4} >$.
Attention, ce nombre peut être également interprété comme un simple flottant codé sous la norme IEEE 754 avec $I = 8,6875$.

Autre exemple

Comment coder le rayon $r = 2^{-19}$? L'exposant du centre est égale à 3. Il faut alors mettre le 1 à la position 22 ($= 3 - (-19)$) dans la mantisse.

Signe	E	Mantisse m'
0	10000010	00010100000000000000010

Pour le centre 8,625 on ne peut seulement placer le 1 entre les position 7 et position 23 dans la mantisse pour coder le rayon car, au delà de la position 7, les bits de poids forts donnent l'information sur le centre.

On ne peut représenter seulement les rayons $r = 2^n$ pour $n \in \{-20, -19, ..., -4\}$ pour $c = 8,625$ en simple précision.

Comment extraire le rayon et le centre d'un intervalle ?

1. Obtenir l'exposant du nombre.
2. Extraire le dernier 1 dans la mantisse et noter sa position.
3. La puissance de 2 du rayon de l'intervalle est caractérisé par : r=$2^{exposant-position}$.
4. Le reste du nombre binaire sans le 1 est le centre de l'intervalle.

Exemple

Comment extraire le centre et le rayon de l'intervalle $I = 4.5625$?

Intervalle	Signe	E	Mantisse m'
4,5625	0	10000001	00100100000000000000000

1. E vaut **10000001** en binaire, c'est-à-dire 129.
   L'exposant est : $exposant = E - decalage = 129 - 127 = \mathbf{2}$.
2. On extrait le dernier 1 dans la mantisse. Il se trouve à la position **6**.
3. Le rayon est alors $r = 2^{2-6} = 2^{-4}$.
4. Le reste du nombre binaire sans le 1 est le centre de l'intervalle :

Nombre	Signe	E	Mantisse m'
4,5	0	10000001	00100000000000000000000

   Donc $c = 4.5$.

Le flottant 4,5625 peut donc être interprété comme un FP-Intervalle où $I =< 4.5, 2^{-4} >$.

Restriction de la représentation

Étant donnée qu'on représente tout sur un nombre, il y a des restrictions à propos de la représentation.

1. Le rayon est une puissance de 2.
2. Tous les intervalles ne sont pas représentables.

En particulier, il n'est pas possible de représenter un intervalle dont le rayon est plus grand que le centre.

2 Opérations

Dans cette partie, nous allons décrire les opérations de bases entre les intervalles sous le format FP-Intervalle. Chaque preuve sera suivie d'un exemple de résultat après l'implémentation de la fonction. Les nombres seront codés avec précision double sur 64 bits.
Pour effectuer une quelconque opération, il convient de borner les éventuelles erreurs d'arrondi.
Pour les calculs du centre de notre intervalle, les erreurs sont bornées en ajoutant l'erreur relative maximale pouvant être commise : $u = 2^{-53}$.
En ce qui concerne les calculs du rayon, toutes les erreurs sont bornées par le mode d'arrondi que nous avons sélectionné : l'arrondi vers plus l'infini.

Opération	Centre	Rayon
Addition	$arrondi(c_1 + c_2)$	$r_1 + r_2 + u.2^{exposant}$
Soustraction	$arrondi(c_1 - c_2)$	$r_1 + r_2 + u.2^{exposant}$
Multiplication	$arrondi(c_1.c_2)$	$(\|c_1\| + r_1).r_2 + r_1.\|c_2\| + \eta + u.2^{exposant(c_3)}$
Division	voir 2.4	voir 2.4

$u = 2^{-53}$ et $\eta = 2^{-1074}$

2.1 Addition

Soit A et B deux intervalles tel que :
$A =< c_1, r_1 >= \{z_1 : c_1 - r_1 \leq z_1 \leq c_1 + r_1\}$
$B =< c_2, r_2 >= \{z_2 : c_2 - r_2 \leq z_2 \leq c_2 + r_2\}$
Alors l'intervalle addition de A et B est noté C, avec :
$C = A + B =< c_3, r_3 >= \{z_3 : c_3 - r_3 \leq z_3 \leq c_3 + r_3\}$

Déterminons le nouveau centre et rayon c_3 et r_3 :
Soit $z_1 \in A, z_2 \in B$, alors on sait que :
$|z_1 - c_1| \leq r_1$
$|z_2 - c_2| \leq r_2$
Donc, $|z_1 + z_2 - (c_1 + c_2)| \leq |z_1 - c_1| + |z_2 - c_2| \leq r_1 + r_2$ par inégalité triangulaire

Lors de l'implémentation, on ne peut pas directement considérer que $c_3 = c_1 + c_2$ et $r_3 = r_1 + r_2$. En effet, l'opération $c_1 + c_2$ en machine calcule seulement l'arrondi de $c_1 + c_2$. On commet une erreur sur le calcul du centre, cette erreur doit être borner lors du calcul du nouveau rayon r_3.
A retenir : $r_3 \neq r_1 + r_2$

Soit c3 = $arrondi(c_1 + c_2)$
$$\begin{aligned} |z_1 + z_2 - c_3| &= |z_1 + z_2 - (c_1 + c_2) + (c_1 + c_2) - c_3| \\ &\leq |z_1 - c_1| + |z_2 - c_2| + |(c_1 + c_2) - c_3| \\ &\leq r_1 + r_2 + u.2^{exposant(c_3)} \\ &\leq r_3 \end{aligned}$$
L'erreur d'arrondi $|(c_1 + c_2) - c_3|$ est bornée par $u.2^{exposant(c_3)}$ avec $u = 2^{-53}$.
Comme r_3 doit être une puissance de 2, on prend r_3 la puissance de 2 qui est supérieur ou égale à la quantité $r_1 + r_2 + u.2^{exposant(c_3)}$.

Exemple

Nombre	Ecriture Scientifique
$A =< 1, 2^{-40} >$	$+1.00000000000000000000000000000000000000\_\_\_\_\_ * 2^0$
$B =< 1, 2^{-30} >$	$+1.00000000000000000000000000000\_\_\_\_\_ * 2^0$
$A + B =< 2, 2^{-30} >$	$+1.00000000000000000000000000000\_\_\_\_\_ * 2^1$

2.2 Soustraction

Pour implémenter la soustraction, la démarche est la même.
Considérons C = A - B
Soit c3 = $arrondi(c_1 - c_2)$
$|z_1 - z_2 - c_3| = |z_1 - z_2 - (c_1 - c_2) + (c_1 - c_2) - c_3|$
$\leq |z_1 - c_1| + |z_2 - c_2| + |(c_1 - c_2) - c_3|$
$\leq r_1 + r_2 + u.2^{exposant(c_3)}$
$\leq r_3$
L'erreur d'arrondi $|(c_1 - c_2) - c_3|$ est bornée par $u.2^{exposant(c_3)}$ avec $u = 2^{-53}$.
On prend r_3 la puissance de 2 qui est supérieur à la quantité $r_1 + r_2 + u.2^{exposant(c_3)}$.

Exemple

Nombre	**Ecriture Scientifique**
$A =< 2, 2^{-40} >$	$+1.00000000000000000000000000000000000000\_\_\_\_\_ * 2^1$
$B =< 1, 2^{-30} >$	$+1.00000000000000000000000000000\_\_\_\_\_ * 2^0$
$A - B =< 1, 2^{-29} >$	$+1.00000000000000000000000000000\_\_\_\_\_ * 2^0$

2.3 Multiplication

De même, C = A . B
Soit c3 = $arrondi(c_1.c_2)$ Déterminons le nouveau rayon :

$|z_1 \times z_2 - c_3| = |z_1 \times z_2 - (c_1 \times c_2) + (c_1 \times c_2) - c_3|$
$\leq |z_1 \times z_2 - (c_1 \times c_2)| + |(c_1 \times c_2) - c_3|$
$\leq |(z_1 - c_1) \times z_2 + (z_2 - c_2) \times c_1 + |(c_1 \times c_2) - c_3|$
$\leq r_1 \times |z_2| + r_2 \times |c_1| + |(c_1 \times c_2) - c_3|$
$\leq r_1 \times (r_2 + |c_2|) + r_2 \times |c_1| + |(c_1 \times c_2) - c_3|$
$\leq r_1 \times r_2 + r_1 \times |c_2| + r_2 \times |c_1| + |(c_1 \times c_2) - c_3|$
$\leq (|c_1| + r_1) \times r_2 + r_1 \times |c_2| + \eta + u \times 2^{exposant(c_3)}$
$\leq r_3$

L'erreur d'arrondi $|(c_1.c_2) - c_3|$ est bornée par $\eta + u \times 2^{exposant(c_3)}$ avec $u = 2^{-53}$ et $\eta = 2^{-1074}$.
On prend r_3 la puissance de 2 qui est supérieur à la quantité $(|c_1|+r_1)r_2+r_1\times|c_2|+\eta+u\times2^{exposant(c_3)}$.

Exemple

Nombre	**Ecriture Scientifique**
$A =< 5, 2^{-40} >$	$+1.01000000000000000000000000000000000000\_\_\_\_\_ * 2^2$
$B =< 2, 2^{-30} >$	$+1.00000000000000000000000000000\_\_\_\_\_ * 2^1$
$A * B =< 10, 2^{-27} >$	$+1.01000000000000000000000000000\_\_\_\_\_ * 2^3$

2.4 Inversion

Une opération fondamentale à implémenter pour rendre notre norme opérationnelle est l'inversion. En effet, le calcul de la division de deux intervalles consiste à inverser le premier intervalle puis à multiplier le résultat de cette inversion par le second intervalle. Pour calculer l'inverse de notre intervalle, nous allons passer par la notation inf-sup dont nous avons parlé précédemment.

On pose : $B =< c_2, r_2 >$ $C =< c_3, r_3 >$ C = 1/B.

$\tilde{c_1} = ((-1)/(-|c_2| - r_2))$
$\tilde{c_2} = ((-1)/(-|c_2| + r_2))$

Il est évident que $\tilde{c_1}$ est la borne inf de notre intervalle et $\tilde{c_2}$ en est la borne sup. Le nouveau centre est donné par un simple calcul du centre de l'intervalle :

$c_3 = (\tilde{c_1} + 0.5 \times (\tilde{c_2} - \tilde{c_1}))$

Le rayon, lui correspond à la moitié de la taille de notre intervalle qui nous est donné par le calcul suivant :

$r_3 = (c_3 - \tilde{c_1})$ Afin de bien retrouver l'inverse de notre intervalle C, il faut lui redonner son signe initial, ce qui se traduit par la multiplication suivant sur le nouveau rayon obtenu r_3 : $r_3 = signe(c_2) \times r_3$

Exemple

Nombre	**Ecriture Scientifique**
$A =< 0.25, 2^{-30} >$	$+1.00000000000000000000000000\_\_\_\_\_ * 2^{-2}$
$1/A =< 4, 2^{-25} >$	$+1.00000000000000000000000000\_\_\_\_\_ * 2^{2}$

2.5 Division

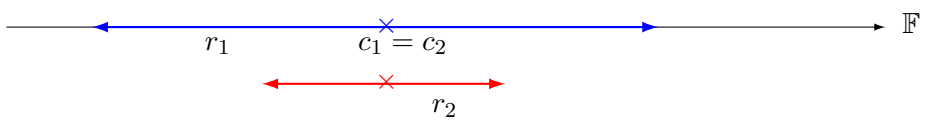
Comme nous l'avons mentionné dans le calcul de l'inverse d'un intervalle, le calcul de la division de deux intervalles consiste à inverser le premier intervalle puis à multiplier le résultat de cette inversion par le second intervalle. On a donc :
$C = A/B = A \times (1/B)$
Les fonctions utilisées sont celles de l'inversion et de la multiplication, une fois que nous disposons de celles-ci, le calcul est trivial.

2.6 Intersection

Soit C = $A \cap B$. Soit A et B deux intervalles tel que :
$A =< c_1, r_1 >$
$B =< c_2, r_2 >$

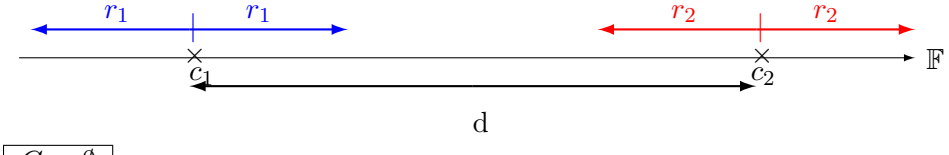
On se place dans le cas où $c_1 \leq c_2$

— **Si $\mathbf{c_1} = \mathbf{c_2} = \mathbf{c}$** : Alors : $\boxed{C =< c, min(r_1, r_2) >}$



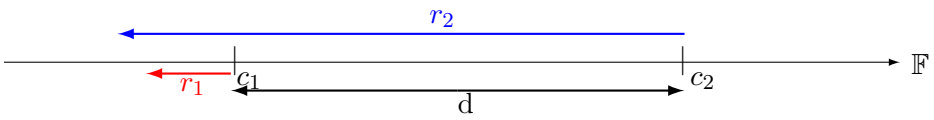
Sinon (les deux centres ne sont pas égaux) :

— Si $r_1 + r_2 < dist(c_1, c_2)$: alors les ensembles sont disjoints.

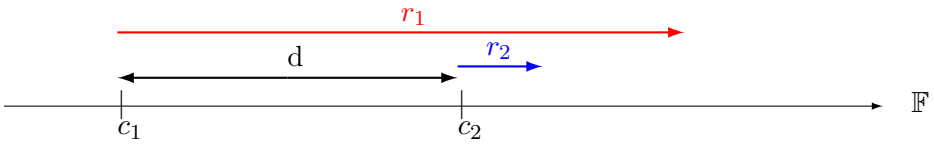


$\boxed{C = \emptyset}$

— Si $dist(c_1, c_2) + r_1 < r_2$ alors $< c_1, r_1 >$ est inclus $< c_2, r_2 >$ $\boxed{C = A =< c_1, r_1 >}$



— Si $dist(c_1, c_2) + r_2 < r_1$ alors $< c_2, r_2 >$ est inclus dans $< c_1, r_1 >$ $\boxed{C = B =< c_2, r_2 >}$



— Sinon il faut passer en notation inf-sup afin de calculer le nouveau intervalle.

$$\boxed{c_3 = \frac{c_1 + r_1 - c_2 + r_2}{2}}$$
$$\boxed{r_3 = \frac{(c_1 - c_2 + r_1 + r_2) + u \times 2^{exposant(c_3)}}{2}}$$

Exemple

Nombre	**Ecriture Scientifique**
$A =< 42, 2^{-52} >$	$+1.01010000000000000000\_\_\_\_\_ * 2^5$
$B =< 42, 2^{-40} >$	$+1.01010000000000000000000000000000000000\_\_\_\_\_ * 2^5$
$A \cap B =< 42, 2^{-52} >$	$+1.01010000000000000000000000000000000000\_\_\_\_\_ * 2^5$

3 Algorithme de Newton par intervalle

3.1 Principe de la méthode de Newton par intervalle

Nous avons testé notre bibliothèque d'intervalles constituée des opérations élémentaires afin d'approcher $\sqrt{2}$ par la méthode de Newton.
Sur $\mathbb{R}$, on aurait approcher la valeur $\sqrt{2}$ en résolvant l'équation

$$f(x) = 0$$

avec

$$f(x) = x^2 - 2$$

où f est une fonction de $\mathbb{R}$ dans $\mathbb{R}$.

La méthode de Newton s'écrit :

$$x_{k+1} = x_k - \frac{x_k^2 - 2}{2 \times x_k}$$

en prenant x_0 assez proche de $\sqrt{2}$.

On ne peut pas simplement remplacer les réels x_k par des intervalles X_k car il y a une **dépendance des données**.
En effet, l'algorithme va calculer pour $x, y, z \in X_k$:

$$x - \frac{y^2 - 2}{2 \times z}$$

au lieu de :

$$x - \frac{x^2 - 2}{2 \times x}$$

pour $x \in X_k$.
Donc on ne peut pas simplement remplacer x_k par des intervalles X_k, car l'algorithme va effectuer des itérations avec trois nombres x,y et z appartenant à X_k différents.

3.2 Calcul de $\sqrt{2}$ par intervalle

Nous allons procéder autrement pour la méthode de Newton par intervalle. Nous utiliserons le résultat suivant pour $f(x) = x^2 - 2$.
Soit f : $\mathbb{R} \to \mathbb{R}$ une fonction dérivable, $X = [x_1, x_2] \subset \mathbb{R}$ et $\tilde{x} \in X$. Supposons de plus que $0 \notin f'(X)$.
On définit alors :

$$N(\tilde{x}, X) := \tilde{x} - \frac{f(\tilde{x})}{f'(X)}$$

Si $N(\tilde{x}, X) \subset X$, alors X contient une racine de f.
Si $N(\tilde{x}, X) \cap X = \emptyset$, alors $f(x) \neq 0$ pour tout $x \in X$.

3.3 Résultat obtenu

On souhaite approcher la valeur $\sqrt{2}$, qui est codé sous le format IEEE 754 de cette manière :

Nombre	**Ecriture Scientifique**
$\sqrt{2}$	$+1.01101010000010011110011001100111111100111011111001101 * 2^0$

On part de $X_0 =< 1.4, 2^{-4} >$ qui contient bien $\sqrt{2}$.

Les itérations obtenus sont :

$+1.011\_\_\_\_\_ * 2^0$
$+1.0110100011\_\_\_\_\_ * 2^0$
$+1.01101010000010100010111111101\_\_\_\_\_ * 2^0$
$+1.01101010000010011110011001100111111110110011011\_\_\_\_\_ * 2^0$
$+1.01101010000010011110011001100111111100111011110011 0\_ * 2^0$

On voit que le résultat converge vers X_4.
Les chiffres significatifs de X_4 sont les mêmes que $\sqrt{2}$ jusqu'au dernier 1 en position 52. Donc l'intervalle itéré par l'algorithme de Newton X_4 contient $\sqrt{2}$: $\sqrt{2} \in X_4$.

Le rayon de cette intervalle X_4 est 2^{-52}, c'est-à-dire que nous avons obtenu une approximation de $\sqrt{2}$ avec une précision de 2^{-52}.

On observe que lorsque l'on passe de X_n à X_{n+1}, on double le nombre de chiffres significatifs pour le centre. Cela correspond à un résultat vu en cours de Systèmes non linéaires et Optimisation :
La méthode de Newton converge quadratiquement.

4 Fonctions usuelles

Dans cette partie, nous allons expliquer la manière de procéder afin d'implémenter des fonctions usuelles en utilisant le format FP-Intervalle. En général, lorsque la fonction considérée est strictement monotone, il suffit de passer en notation inf-sup, et de calculer les nouvelles bornes de l'intervalle.
Une subtilité réside pour les fonctions périodiques de type sinus, cosinus où il faudra gérer les cas des intervalles où la fonction est monotone, et là où la fonction ne l'est pas.

4.1 Exponentielle

On cherche à trouver $exp(I)$ avec $I =< C_I, r_I >$
$I =< C_I, r_I >= \{z_I : C_I - r_I \leq z_I \leq C_I + r_I\}$
$I =< C_I, r_I >= \{z_I : a \leq z_I \leq b\}$ avec $a = C_I - r_I$ et $b = C_I + r_I$
Alors l'intervalle image de I par la fonction exponentielle qui est **strictement croissante** est :
$K = exp(I) =< C_k, r_k >= \{z : exp(a) \leq z \leq exp(b)\}$
donc :

$$\boxed{C_k = \frac{exp(a)+exp(b)}{2}}$$
$$\boxed{r_k = \frac{exp(a)-exp(b)}{2} + u \times 2^{exposant(C_k)}}$$

Exemple

Nombre	**Ecriture Scientifique**
$A =< 0, 2^{-10} >$	$+1.\_\_\_\_\_ * 2^{-10}$
$e^A =< 1, 2^{-21} >$	$+1.00000000000000000000\_\_\_\_\_ * 2^0$

4.2 Logarithme

On cherche à trouver $log(I)$ avec $I =< C_I, r_I >$
$I =< C_I, r_I >= \{z_I : C_I - r_I \leq z_I \leq C_I + r_I\}$

$I =< C_I, r_I >= \{z_I : a \leq z_I \leq b\}$ avec $a = C_I - r_I$ et $b = C_I + r_I$
Alors l'intervalle image de I par la fonction logarithme qui est **strictement croissante** est :
$K = log(I) =< C_k, r_k >= \{z : log(a) \leq z \leq log(b)\}$
donc :

$$\begin{cases} C_k = \frac{log(a)+log(b)}{2} \\ r_k = \frac{log(a)-log(b)}{2} \end{cases}$$

$$\boxed{C_k = log(\sqrt{a \times b})}$$
$$\boxed{r_k = log(\sqrt{\frac{a}{b}}) + u \times 2^{exposant(C_k)}}$$

Exemple

Nombre	**Ecriture Scientifique**
$A =< exp(1), 2^{-52} >$	$+1.0101101111110000101010001011000101000101011101101 00\_ * 2^1$
$log(A) =< 1, 2^{-51} >$	$+1.11111111111111111111111111111111111111111111111111\_\_\_ * 2^{-1}$

4.3 Racine

On cherche à trouver $\sqrt{I}$ avec $I =< C_I, r_I >$
$I =< C_I, r_I >= \{z_I : C_I - r_I \leq z_I \leq C_I + r_I\}$
$I =< C_I, r_I >= \{z_I : a \leq z_I \leq b\}$ avec $a = C_I - r_I$ et $b = C_I + r_I$
Alors l'intervalle image de I par la fonction racine qui est **strictement croissante** est :
$K = \sqrt{I} =< C_k, r_k >= \{z : \sqrt{a} \leq z \leq \sqrt{b}\}$
donc :

$$\boxed{C_k = \frac{\sqrt{a}+\sqrt{b}}{2}}$$
$$\boxed{r_k = \frac{\sqrt{a}-\sqrt{b}}{2} + u \times 2^{exposant(C_k)}}$$

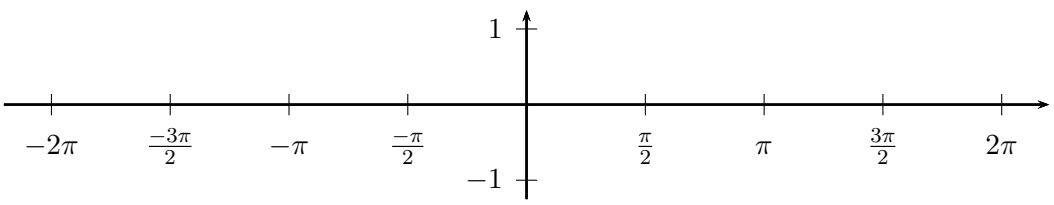
Exemple

Nombre	**Ecriture Scientifique**
$A =< 2, 2^{-10} >$	$+1.0000000000\_\_\_\_\_ * 2^1$
$\sqrt{A} =< \sqrt{2}, 2^{-52} >$	$++1.0110101000001001111001100110011111110011101111001 10\_ * 2^0$

4.4 Sinus / Cosinus / Tangente

4.4.1 Sinus

on cherche à trouver $sin(I)$ avec $I =< C_I, r_I >= [a, b]$ avec $a = C_I - r_I$ et $b = C_I + r_I$



la fonction $f : \mathbb{R} \to [-1, 1]$
$x \to sin(x)$ est periodique de periode 2π, donc pour simplifier le problème on reduit notre intervalle de telle sorte qu'on se ramène sur [0,2π].
soit $K = sin(I) =< C_k, r_k >= [min, max]$ tel que min est la borne inférieure de K et max est la

borne supérieure de K.
on a quatre cas :
a) $\mathbf{I \subset [0,\frac{\pi}{2}]}$ **ou** $\mathbf{I \subset [\frac{\pi}{2},\frac{3\pi}{2}]}$ **ou** $\mathbf{I \subset [\frac{3\pi}{2},2\pi]}$:
la fonction est monotone donc :

$$max = max(sin(a), sin(b))$$
$$min = min(sin(a), sin(b))$$

b) $\frac{\pi}{2}$ **et** $\frac{3\pi}{2} \in \mathbf{I}$:
la fonction atteint ses bornes supérieure est inférieure en $\frac{\pi}{2}$ et $\frac{3\pi}{2}$ donc :

$$max = 1$$
$$min = -1$$

c) **seulement** $\frac{\pi}{2} \in \mathbf{I}$:
la fonction atteint sa borne supérieure en $\frac{\pi}{2}$ donc :

$$max = 1$$
$$min = min(sin(a), sin(b))$$

d) **seulement** $\frac{3\pi}{2} \in \mathbf{I}$:
la fonction atteint sa borne inférieure en $\frac{3\pi}{2}$ donc :

$$max = max(sin(a), sin(b))$$
$$min = -1$$

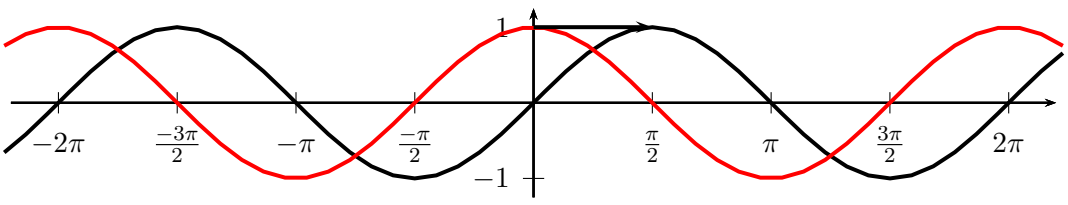
Conclusion :

$$\boxed{C_k = \frac{max+min}{2}}$$
$$\boxed{r_k = \frac{max-min}{2} + u \times 2^{exposant(C_k)}}$$

4.4.2 Cosinus

il suffit de remarquer que :

$$\boxed{cos(x) = sin(x + \pi/2)}$$



4.4.3 Tangente

il suffit juste d'utiliser les deux resultats obtenues par les deux fonctions précédentes puisque :

$$\boxed{tan(x) = \frac{sin(x)}{cos(x)}}$$

Exemple

Nombre	**Ecriture Scientifique**
$A =< 3, 2^{-10} >$	$+1.10000000000000000000000000000000000000000000000000\_\_ * 2^{1}$
$cos(A) =< -0.989992, 2^{-52} >$	$-1.11111010111000000100101111101000010111100101110100\_\_ * 2^{-1}$

5 Matrices

Dans cette partie, nous avons implémenté des matrices d'intervalles et le produit matriciel pour notre norme.

5.1 Produit Matriciel

$$\begin{bmatrix} < C_1, r_1 > & < C_2, r_2 > & < C_3, r_3 > \\ < C_4, r_4 > & < C_5, r_5 > & < C_6, r_6 > \\ < C_7, r_7 > & < C_8, r_8 > & < C_9, r_9 > \end{bmatrix} \begin{bmatrix} < C_a, r_a > & < C_b, r_b > & < C_c, r_c > \\ < C_d, r_d > & < C_e, r_e > & < C_f, r_f > \\ < C_i, r_i > & < C_j, r_j > & < C_k, r_k > \end{bmatrix} = \begin{bmatrix} l & m & n \\ o & p & q \\ r & s & t \end{bmatrix}$$

Comme dans la multiplication matricielle de deux matrices de nombres réels, on effectue la multiplication et l'addition terme à terme avec les lignes et les colonnes.
En utilisant les résultats trouvés précédemment lors de l'addition et la multiplications de deux FP-INTERV :
l=$< C_l, r_l >$
avec

$$\begin{cases} C_l = C_1C_a + C_2C_d + C_3C_i \\ r_l = (|c_1| + r_1).r_a + r_1.|c_a| + \eta + u.2^{exposant(c_1C_a)} + (|c_2| + r_2).r_d + r_2.|c_d| + \eta + u.2^{exposant(c_2C_d)} \\ +(|c_3| + r_3).r_i + r_3.|c_i| + \eta + u.2^{exposant(c_3C_i)} \end{cases}$$

La méthode est analogue pour les autres coefficients m, n, o, p, q, r, s, t.

Conclusion

En conclusion de ce projet, nous avons pu proposer en langage C Une bibliothèques de fonctions complète et efficace. Elle nous permet de manipuler des intervalles codés avec notre norme, le format FP-INTERV qui est un format compressé basé sur la norme IEEE 754. Les avantages que ce format nous apporte sont nombreux, il nous permet d'avoir des résultats précis et fiables.

Les fonctions que nous avons implémenté couvrent des opérations très classiques telles que l'addition et la multiplication d'intervalles, mais aussi des fonctions plus sophistiquées telles que l'exponentielle et le logarithme d'intervalles, et enfin des matrices d'intervalles.
Un résultat important de notre projet était la possibilité d'obtenir une approximation de $\sqrt{2}$ avec une précision de 2^{-52} à partir de l'algorithme de Newton adapté pour les intervalles.
C'est donc avec une bonne dynamique de groupe que nous avons pu mettre en place notre bibliothèque d'arithmétique d'intervalle, chaque membre a su faire avancer le projet de manière conséquente pour en faire une bibliothèque aboutie et opérationnelle.

Une perspective d'amélioration de notre projet aurait été de manipuler les bibliothèques CBLAS afin de rendre notre produit matriciel plus performant car même s'il est opérationnel, celui-ci reste relativement lent ($O(n^3)$).

Notre projet est disponible sous le lien :

https ://gitlab.com/vinceliu/bibliIntervalleC

D'un point de vue personnel, ce projet nous a permis de comprendre parfaitement comment l'ordinateur stocke les nombres à virgule flottante. Nous avons pu utiliser de nouvelles bibliothèques en langage C telle que la bibliothèque $fenv$ ou $iEEE754$. Cela fut un très bon entraînement afin de mieux maîtriser ce langage de programmation.
Nous avons également appris à mieux travailler en équipe afin de surmonter les difficultés, nous avons pu échanger nos idées, les confronter et tirer le meilleur de ce que chacun avait à proposer, cela a contribué à la réussite de ce projet.

6 Bibliographie

[1] David Defour. FP-ANR : A representation format to handle floating-point cancellation at run-time, June 2017. working paper or preprint.
[2] John L. Gustafson. The end of error. Chapman & Hall/CRC Computational Science Series. CRC Press, Boca Raton, FL, 2015.
[3] Nicholas J. Higham. Accuracy and stability of numerical algorithms. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, second edition, 2002.
[4] Ramon E. Moore, R. Baker Kearfott, and Michael J. Cloud. Introduction to interval analysis. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2009.

7 Annexe

Addition entre FP-Intervalle

```
// Functions to perform addition over FP_INTER format
FP_INTERV FpInterAdd(FP_INTERV A, FP_INTERV B)
  {

  int exprA, exprB, exprC;
  double rayonC, centreA, centreB, centreC;

  if (A == 0)
  {
    return B;
  }
  if (B ==0)
  {
    return A;
  }

  // On recupere le rayon et le centre des 2 intervalles
  centreA = FPInterv2CenterRadius(A,&exprA);
  centreB = FPInterv2CenterRadius(B,&exprB);

  // nouveau centre: Arrondi de C1 + C2
  centreC = centreA + centreB;

  int e;
  // On recupere la puissance de 2 du centre
  frexp(centreC,&e);
  e = e - 1;
    // On change le mode d'arrondi
  fesetround(FE_UPWARD);
  // On calcule le nouveau rayon
  rayonC = pow(2,exprA) + pow(2,exprB) + pow(2,-53+e);
  // On change le mode d'arrondi
  fesetround(FE_TONEAREST);
  //frexp(rayonC,&e);
  // On calcule la puissance de 2 du rayon
  exprC = (int) ceil(log(rayonC)/log(2)) ;

  return CenterRadius2FPInter(centreC,exprC);

  }
```

Voici un exemple d'implémentation en langage C de l'opération élémentaire qui, à partir de deux intervalles de type "FP_Interv" retourne l'intervalle additionnée. Le code source se trouve dans le fichier "*operation.c*".
A partir de deux intervalles A et B, on extrait les centres c_A et c_B et les exposants des rayons $exprA$, et $exprB$ avec une fonction de conversion "*FPInterv2CenterRadius*" que nous avons écrite. On calcule l'arrondi $c_C = c_A + c_B$, puis on prend la puissance de 2 de c_C avec la fonction $frexp$ qui se trouve dans la bibliothèque *math*.
L'erreur d'arrondi lors du calcule de c_C est bornée par $u.2^{exp(c_C)}$, donc par le terme $pow(2, -53+e)$. Lors du calcul du rayon r_c, on change le mode d'arrondi, pour borner l'erreur d'arrondi de l'opération $r_A + r_B$. On va arrondir vers $+\infty$. On utilise pour cela, la fonction $fsetround$ qui se trouve dans la bibliothèque $fenv$.
Enfin après avoir obtenu la puissance de 2 du nouveau rayon, on renvoie la fonction "*CenterRadius2FPInter*". Cette fonction, à partir du centre et du rayon retourne l'intervalle correspondant.

Exponentielle de FP-intervalle

```
// Function to perform Exp function over FP-INTER format
FP_INTERV FpInterExp(FP_INTERV I)
{
  double center,centerExp,RadiusExp;
  int exprExp,expr,e;
  double a,b;

  center=FPInterv2CenterRadius(I,&expr);
  // On passe en notation inf-sup
  fesetround(FE_DOWNWARD);
  a = center-pow(2,expr);
  fesetround(FE_UPWARD);
  b = center+pow(2,expr);

  // Centre en inf-sup
 fesetround(FE_TONEAREST);
  centerExp = (exp(b)+exp(a))/2;
  // On récupère la puissance de 2 du centre
  frexp(centerExp,&e);
  e--;
  // Calcul du nouveau rayon
 fesetround(FE_UPWARD);
  RadiusExp = (exp(b)-exp(a))/2 + pow(2,-53+e) + pow(2,-1074);
  // Puissance de 2 du nouveau rayon
  exprExp = (int) ceil(log(RadiusExp)/log(2));

  return CenterRadius2FPInter(centerExp, exprExp);
}
```

$w(e_{5,2}) = \frac{|N_5 \bigcap N_2|}{|N5 \bigcup N_2|} = \frac{|\{1\}|}{|\{1,3,4\}|} = \frac{1}{3}$

Voici un second exemple d'implémentation en langage C d'une de nos fonction : la fonction exponentielle. La fonction exponentielle appliquée à un intervalle I de type "FP_Interv" prend en entrée l'intervalle I et retourne l'intervalle image par la fonction exponentielle. Le code source de la fonction se trouve dans le fichier "*function.c*".
A partir de l'intervalle I, on extrait comme précédemment le centre ainsi que la puissance de 2 du rayon de l'intervalle.
Il faut ensuite calculer la borne inférieur a et la borne supérieur b de notre intervalle I. Pour cela, il faut gérer les modes d'arrondis avec la fonction $fsetround$.
A partir des deux nombres réels a et b, les bornes de l'intervalle $exp(I)$ sont simplement e^a et e^b. On en déduit le nouveau centre de $exp(I)$ par $\frac{e^a+e^b}{2}$ sans oublier le mode d'arrondi correspondant. Ensuite, la méthode pour calculer le nouveau rayon se fait de manière analogue que la première fonction.
Enfin, pour calculer le nouveau intervalle, il faut faire simplement appel à la fonction de conversion qui prend en paramètre le nouveau centre et la puissance de 2 du rayon.