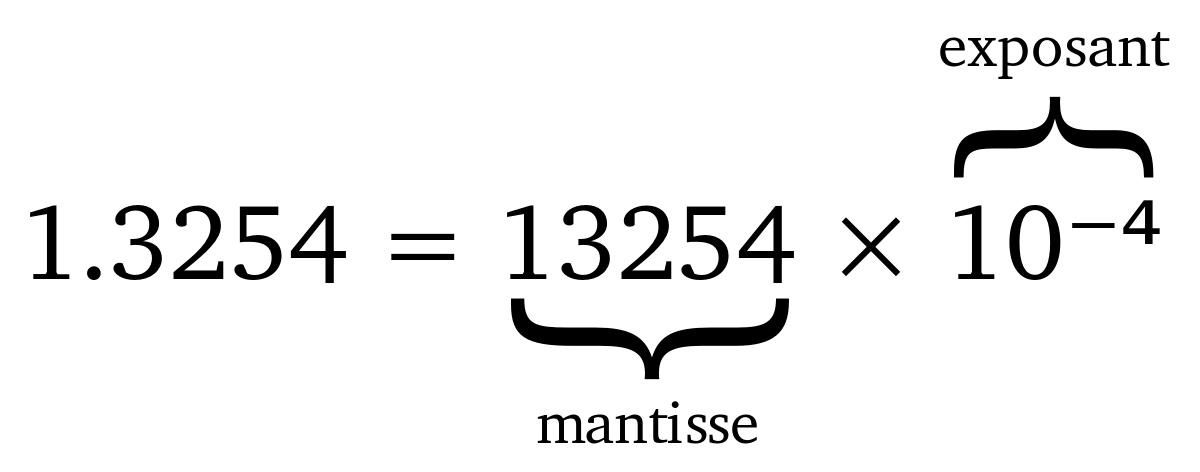
Rapport de laboratoire  
Réels sur n bits



# Table des matières

[**Table des matières**](#_3zl9gtrsf6ve) **2**

[**1. Contextualisation**](#_xn60z6c5ujxw) **3**

[**2. Méthodologie de développement**](#_epv5p9sws5ps) **3**

[2.1 Calcul de la taille de l’exposant](#_gdylror4e2h9) 4

[2.2 Calcul du biais](#_l55k7oesyuuo) 5

[2.3 Récupération du signe](#_cwrrbk1m5ske) 5

[2.4 Calcul de la mantisse et l'exposant](#_rckt8wfjxgb1) 5

[2.5 Conversions de la mantisse et de l’exposant](#_c5kgd3919293) 5

[2.5.1 La mantisse](#_8bpdrfaczzg) 5

[2.5.2 L’exposant](#_if2vcouup6es) 6

[2.6 Construire le nombre flottant](#_5g99kvro5oqb) 6

[2.7 Additions binaires & flottantes](#_hc6v1luuj5w8) 6

[2.7.1 Addition binaire](#_9qo31mx8tpu2) 6

[2.7.2 Addition flottante](#_7bhey2kz9dvy) 6

[2.7.2.1 Bit excédentaire](#_zazfkmhhnuzb) 7

[2.8 NaN et infini](#_1rjwfdj8z0zi) 7

[**3. Conclusion et perspective**](#_gv16xws2poc) **8**

[**4. Références**](#_y8iwdcbjdovn) **8**

# 

# 1. Contextualisation

Ce laboratoire a été réalisé durant le cours d’Algorithmes numériques donné par Julien Senn. Le projet est étudié par un groupe de quatre personnes.

Le but de ce laboratoire est de pouvoir additionner deux nombres binaires flottants et d’afficher le résultat de cette somme. Ce projet permet à un utilisateur d’utiliser un site web avec une simple page HTML, afin de convertir et additionner deux nombres réels en nombres flottants à virgules. Pour ce faire, il va devoir renseigner le nombre de bits sur lesquels les nombres flottants vont être codés ainsi que les deux nombres.

L’utilisation de Javascript est demandée pour concevoir ce laboratoire. Pour pouvoir additionner deux nombre flottants, il faut passer par la conversion de chaque nombre.

# 2. Méthodologie de développement

Pour résoudre le problème de l’addition flottante, il a d’abord fallu comprendre comment la conversion et l’addition fonctionnaient. En effet, la conversion d’un nombre entier en binaire et différente de celle d’un nombre réel, il faut passer par plusieurs autres étapes pour convertir la partie décimale du nombre. De plus, l’addition s’en trouve changée et certaines règles sont donc nécessaires à suivre.

Premièrement, plusieurs recherches ont été faites sur comment convertir un nombre réel en binaire ainsi que l’addition de ceux-ci. Pour ce faire, l’aide de l’enseignant ainsi que plusieurs sources ont été consultés afin d’améliorer la compréhension du problème. Finalement, des règles et concepts ont été approuvés et respectés par le groupe pour développer une solution.

Un nombre binaire à virgule flottante est séparé en différentes parties : le signe, l’exposant et la mantisse. Ci-dessous sont rassemblées les méthodes utilisées par le groupe afin de calculer et utiliser lesdites parties.

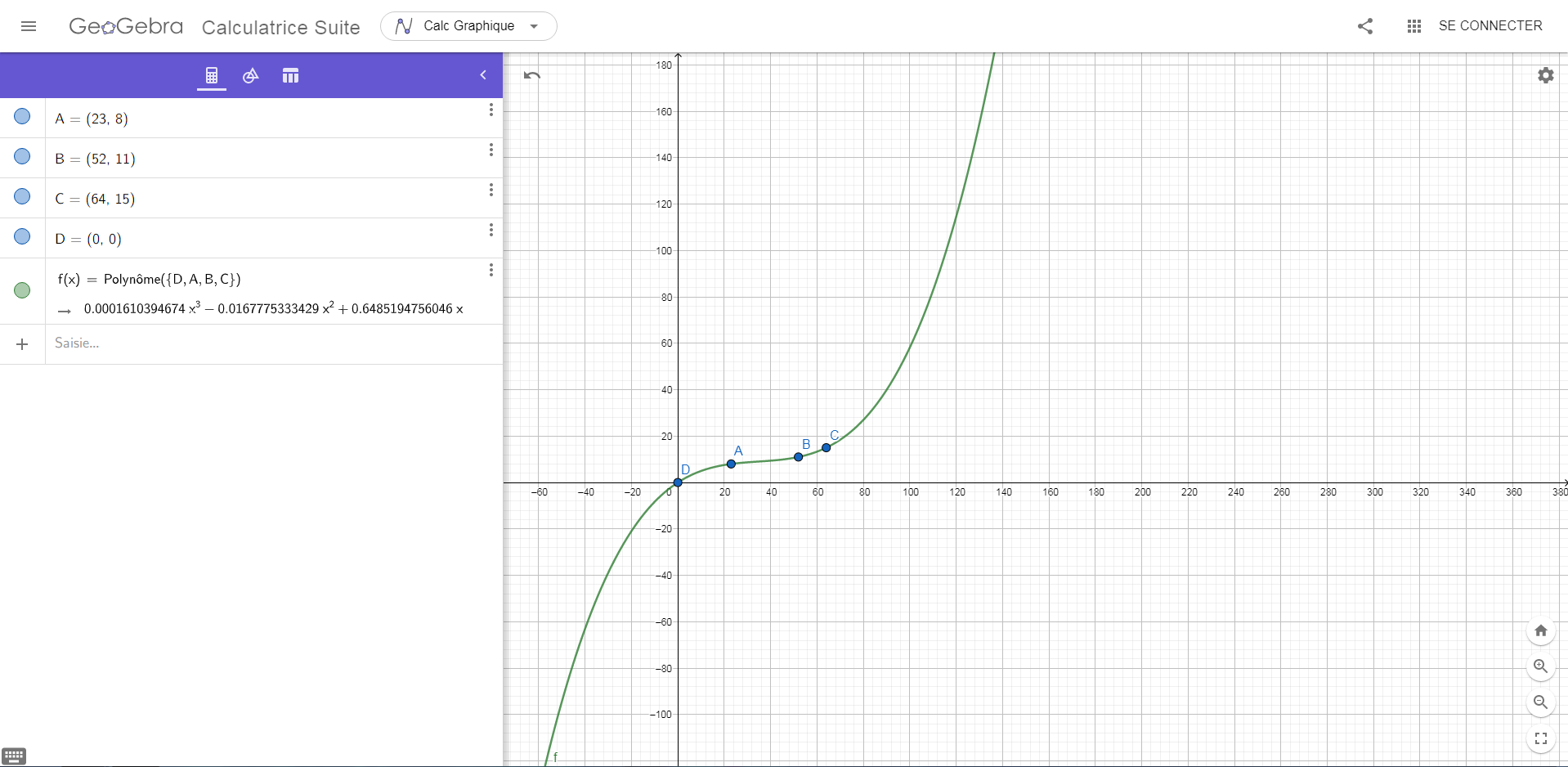
## 2.1 Calcul de la taille de l’exposant

La taille de l’exposant est normalement arbitraire, cependant, dans le cadre du laboratoire, la taille de la mantisse n’est pas fixe mais entrée par l’utilisateur. C’est pourquoi l’exposant doit également s’en trouver changé. Le groupe a récupéré les différentes valeurs des normes IEEE 754 pour former une fonction à suivre par rapport à la taille de la mantisse.

Les valeurs de la norme étaient :

|  |  |
| --- | --- |
| **Taille de la mantisse** | **Taille de l’exposant** |
| 24 bits dont 1 bit implicite | 8 bits |
| 53 bits dont 1 bit implicite | 11 bits |
| 64 bits sans bit implicite | 15 bits |
| 0 bits sans bits implicite | 0 bits |

Cette fonction en est née, c’est avec celle-ci que l’exposant est trouvé selon la taille donnée par l’utilisateur



## 2.2 Calcul du biais

Le calcule de l’exposant biaisé, ou le biais, se fait à l’aide de la taille de l’exposant en binaire :

## 2.3 Récupération du signe

La récupération du signe est simple, il suffit de récupérer le premier caractère du nombre entré. Si c’est le signe “-”, le bit sera 1 ; si c’est un chiffre, le bit sera 0.

## 2.4 Calcul de la mantisse et l'exposant

La mantisse se calcule très simplement en divisant ou en multipliant un nombre, que l’on souhaite convertir, par 2 jusqu’à ce que celui-ci atteigne une valeur entre 1 et 2. Ce nombre sera à convertir ensuite en binaire pour représenter la mantisse flottante. Le nombre d'opérations effectuées représente la puissance de 2 qui donnera l’exposant : lors d’une division, ce nombre augmentera de 1 et lors d’une multiplication, il descendra de 1.

La mantisse sera du coup une valeur décimale se trouvant entre 1 et 2.

L’exposant sera alors donné par où “n” est le nombre d’opérations expliqué ci-dessus.

## 2.5 Conversions de la mantisse et de l’exposant

### 2.5.1 La mantisse

La mantisse est convertie en binaire de manière spécifique. Une fois la partie “mantisse” décimale calculée, il suffit de récupérer cette partie décimale de la multiplier par 2 et si la partie entière vaut 1 ou 0, la partie binaire récupère cette valeur. Il faut alors répéter ce processus pour remplir la plage de bits donnée par l’utilisateur.

Voici un exemple pour mieux comprendre :

|  |  |
| --- | --- |
| Conversion en binaire de 1.657 | |
| **Décimale** | **Binaire** |
| 0.657 \* 2 = 1.314 | “1” |
| 0.314 \* 2 = 0.628 | “10” |
| 0.628 \* 2 = 1.256 | “101” |
| ... | ... |

### 2.5.2 L’exposant

Pour l’exposant, c’est une simple conversion signée car l’exposant est un nombre entier. Il suffit alors de convertir où “n” est un nombre entier, en binaire. Nous avons utilisé un algorithme de conversion binaire très connu qui utilise les restes de divisions par 2 du nombre que l’on souhaite convertir.

## 2.6 Construire le nombre flottant

Toutes les parties sont créées et calculées, il suffit à présent de les assembler à la suite ainsi :

Signe | Exposant | Mantisse

Il ne faut pas oublier que la mantisse possède un bit à 1 implicite caché juste après l’exposant. Il est très important pour les opérations mathématiques.

## 2.7 Additions binaires & flottantes

### 2.7.1 Addition binaire

L’addition binaire effectue une addition en base de 2. Comme dans les additions en base de 10, il ne faut pas oublier les retenus. Par exemple si on additionne 1+1 le résultat sera de 1 avec une retenu de 1. Ensuite si le prochain calcul est 0+0 il faudra ajouter la retenu tel que 0+0+1.

### 2.7.2 Addition flottante

Pour additionner deux nombre flottants, il faut impérativement que les mantisses des deux nombres aient la même taille ainsi que le bit implicite pris en compte. Finalement, il est nécessaire pour les exposants d’être égaux.

Si les exposants sont différents, il faut procéder à un “rajustement” en additionnant 1 suffisamment de fois à l’exposant le plus petit pour qu’il rattrape l’autre. Pour chaque ajout de 1, la mantisse doit être décalée vers la droite de 1 bit.

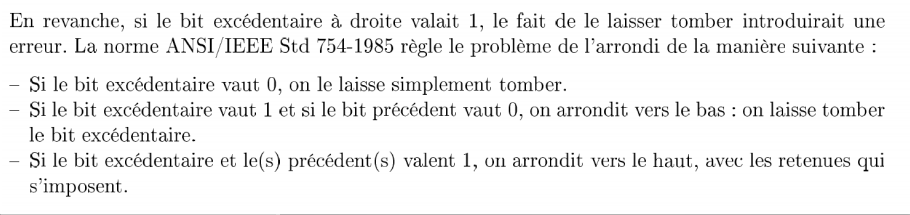
Une fois ce “rajustement” fait, il suffit d’additionner les mantisses entre elles tout en tenant compte des retenues.

Ci-dessous se trouve un exemple d’addition.

#### 2.7.2.1 Bit excédentaire

Lors de l’addition des mantisses, les retenues peuvent agrandir la taille de la mantisse : on appelle ceci, le bit excédentaire.

Il y a alors 3 possibilités de règles à respecter :

En plus de ces règles, pour chaque retenue modifiant la taille de la mantisse, il faut additionner 1 à l’exposant.

## 2.8 NaN et infini

Il est possible que la représentation des nombres flottants soit des valeurs spéciales qui peuvent être NaN ou infini. Ces deux valeurs sont facilement détectables. Pour ce faire, il faut avoir la valeur de l’exposant biaisé et la mantisse. Le tableau ci-dessous, montre les valeurs nécessaires pour être une valeur spéciale.

|  |  |  |
| --- | --- | --- |
| Type | Exposant biaisé | Mantisse |
| NaN | 2e - 1 | 0 |
| Infinis | 2e - 1 | différente de 0 |

# 3. Conclusion et perspective

L’application permet actuellement d’indiquer deux nombres ainsi que le nombre de bits pour coder la mantisse. S’ils ne sont pas remplis, une alerte indique à l’utilisateur de remplir tous les points.

Comme le voulait le cahier des charges, une fois les nombres indiqués; le signe, l’exposant, le biais ainsi que la mantisse des deux vont être calculés puis convertis en binaire. Ces nombres flottants seront premièrement affichés et additionnés pour ensuite en donner le résultat sous forme binaire.

L’application ne supporte actuellement que la conversion binaire et l’addition.

Les autres types d’opérations et fonctionnalités sont des améliorations à envisager pour le futur.

# 4. Références

Liste des auteurs: [http://fr.wikipedia.org/w/index.php?title=IEEE\_754&action=history](https://fr.wikipedia.org/w/index.php?title=IEEE_754&action=history), 2021. IEEE 754 - Wikipédia. In : Wikipedia.org. Mise à jour le 02 février 2021. [Consulté le 20 mars 2021]. Disponible à l’adresse : <https://fr.wikipedia.org/wiki/IEEE_754>.

AspenCore, Inc, 2021. Binary to Decimal and How to convert Binary to Decimal. In : electronics-tutorials.ws. Mise à jour le 14 Décembre 2020. [Consulté le 03 Mars 2021]. Disponible à l’adresse : <https://www.electronics-tutorials.ws/binary/bin_2.html>

NetInfluence, 2020. Comprendre les nombres à virgule flottante - netinfluence's blog. In : netinfluence.ch. Mise à jour le 30 Novembre 2020. [Consulté le 04 Mars 2021]. Disponible à l’addresse : <https://blog.netinfluence.ch/2009/09/24/comprendre-les-nombres-a-virgule-flottante/>

How to Learn, 2018. How to Find the Mantissa and Exponent in floating-point and 32-bit Binary. In Youtube.com. Mise à jour le 13 Mars 2019. [Consulté le 07 Mars 2021].

Disponible à l’adresse : <https://www.youtube.com/watch?v=s_8cOnuVskQ>

Abishalini Sivaraman, 2016. Decimal to IEEE 754 Floating Point Representation. In Youtube.com. Mise à jour le 10 Février 2021. [Consulté le 07 Mars 2021]. Disponible à l’adresse : <https://www.youtube.com/watch?v=8afbTaA-gOQ&ab_channel=AbishaliniSivaraman>

Wandida EPFL, 2014. Conversion du décimal vers le flottant IEEE 754 | Ronan Boulic. In : Youtube.com. Mise à jour le 29 Mai 2020. [Consulté le 12 Mars 2021]. Disponible à l’adresse : <https://www.youtube.com/watch?v=6fvHvHIq37w>