# Manuel Utilisateur

Ce document présente les différentes spécificités du compilateur réalisé par nos soins; il présente les fonctionnalités et les limitations du projet. La gestion d'erreurs et la commande decac y sont détaillés, ainsi que les limites de la class Math implémentée.

## Fonctionnalités

Deca est un langage similaire au Java. Le compilateur permet la compilation d'un programme sans objet avec inclusion possible de fichiers deca tiers.

```
/**
 * Fibonacci programme test
 */
{
    // compteur
    int cpt = 10;
    int a = 1;
    int b = 2;
    int c;
    while(cpt >= 0){
       c = a + b;
       a = b;
       b = c;
       println(c);
       cpt = cpt - 1;
    }
}
```

La déclaration des classes est possible mais la création d'objets est pour le moment impossible.

## Limitations

- **Initialisation d'objets impossible** : Il est impossible d'instancier des objets avec le mot-clé new et par conséquent d'utiliser les attributs et méthodes de classe.
    - Le compilateur ne parvient pas à générer le code assembleur pour la gestion des classes. L'utilisateur peut créer une classe en déclarant ses méthodes et ses attributs, il peut aussi y accéder en utilisant this. Cependant, la génération en assembleur du corps d'une méthode avec des opérandes (+, -, ...), des valeurs de retour, des paramètres ne fonctionnent pas.

Le programme deca suivant résume notre gestion des classes en assembleur :

```
class A{
      int a;
      void methA(){
            this.a = 5;
      }
}
{
      println("hello");
}
```

- **Problèmes d'affichage :**
    - Une division entre flottants affiche un résultat erroné.
- **Opérande unaire de soustraction :**
    - Affecter un nombre négatif à une variable va à la place attribuer son équivalent positif.
- **While et If imbriquées :**
    - Il n'est pas possible d'imbriquer un while dans un autre while de la même façon qu'il n'est pas possible d'avoir des if imbriqués.

# Erreurs

## [Syntaxe Contextuelle]

- Affectation error, type: <type1>, expected type: <type2>
  - Erreur levée lorsqu'une valeur est affectée à une variable et que le type de cette valeur diffère du type de la variable.
  - Cette erreur est aussi être levée lorsque l'appel d'une fonction est réalisée et que la signature de l'appel diffère de la signature de la définition.
  - Enfin cette erreur peut être levée lorsque le retour d'une fonction diffère du type de celle-ci.

- The condition must be a boolean
  - Erreur levée lorsque la condition d'une structure de contrôle (if ou while) contient en condition un élément d'une nature autre que booléenne.

- Type <type> not supported by the operation
  - Erreur levée lorsqu'une opération est réalisée sur un type qui ne supporte pas cette opération.

- Operation between the type <type1> and the type <type2>
  - Erreur levée lorsqu'une opération entre deux types incompatibles est réalisée.

- Type <type2> used on a boolean operation
  - Erreur levée lorsqu'une opération booléenne est appelée sur un élément d'un autre type qu'un booléen.

- Comparison between two different types (<type1> & <type2>)
  - Erreur levée lorsqu'une comparaison est réalisée entre deux types incompatibles.

- Printing a boolean with print/printx
  - Erreur levée lorsque les fonctions {print, println, printx, printlnx} sont appelées avec un paramètre booléen.

- Trying to cast a class with a not parent class
  - Erreur levée lorsqu'un transtypage est exécuté avec une classe de <type1> sur un élément dont le type n'hérite pas de <type1>.

- Class undefined

- ➢ Erreur levée si un héritage avec une classe inexistante est réalisée.

❖ The class must inherit from an existing class
- ➢ Erreur levée lorsqu'une classe hérite d'un type qui n'est pas une classe.

❖ Trying to declare a class that already exist
- ➢ Erreur levée lors de la déclaration d'une classe déjà déclarée.

❖ Void variable
- ➢ Erreur levée lors de la déclaration d'une variable de type void.

❖ <field> isn't a parent class
- ➢ /!\ Faute d'écriture ici /!\ le message devrait être "<field> isn't a field"
- ➢ Erreur levée lorsqu'un champ est défini dont l'identificateur est déjà défini dans une classe parent sans être lui aussi un champ.

❖ Field already defined
- ➢ Erreur levée lors de la déclaration d'un champ déjà déclaré dans l'environnement local.

❖ <method> isn't a method
- ➢ Erreur levée si l'identificateur de la méthode est déjà déclaré dans une classe parente et ne correspond pas à une méthode.

❖ Trying to redefine a function with a different signature
- ➢ Erreur levée lorsqu'une méthode est redéfinie avec une signature différente.

❖ The type returned by the function <type1> must inherit from <type2>
- ➢ Erreur levée lorsque le type de retour attendu d'une fonction est différent du type retourné réel.

❖ Method already defined
- ➢ Erreur levée lors de la déclaration d'une méthode déjà déclarée dans l'environnement local.

❖ Parameter already declared
- ➢ Erreur levée lors de la déclaration d'un paramètre déjà déclaré dans l'environnement local.

❖ Void Parameter
- ➢ Erreur levée lors de la déclaration d'un paramètre de type void.

❖ Void variable
- ➢ Erreur levée lors de la déclaration d'une variable de type void.

❖ Variable already declared

- ➢ Erreur levée lors de la déclaration d'une variable déjà déclarée dans l'environnement local.

❖ Use of an undefined identifier
- ➢ Erreur levée lors de l'appel d'un identificateur non déclaré dans l'environnement local.

❖ <method> isn't a method from <class>
- ➢ Erreur levée lors de l'appel d'une méthode d'une classe dont l'environnement ne contient pas la méthode.

❖ The method <method> isn't defined
- ➢ Erreur levée lors de l'appel d'une méthode qui n'est pas déclarée dans l'environnement local.

❖ Unsupported type for % (must be an integer)
- ➢ Erreur levée lors de la réalisation d'un modulo avec des types qui ne sont pas des entiers.

❖ Type not supported by the operation
- ➢ Erreur levée lors de la réalisation d'un Not sur un type non booléen.

❖ Return void type
- ➢ Erreur levée lorsque la variable retournée par une fonction est de type void.

❖ <type> isn't a class
- ➢ Erreur levée lors de la sélection d'un champ ou d'une méthode sur un identificateur qui n'est pas une classe.

❖ <identifier> is PROTECTED
- ➢ Erreur levée lors de la sélection d'un champ d'une classe dont la visibilité est protected (appel réalisé hors de la classe).

❖ Identifier This called outside of a class
- ➢ Erreur levée lorsque le mot clé this est appelé dans le bloc Main.

❖ Unsupported type by the operation
- ➢ Erreur levée lorsque le moins unitaire est utilisé sur un élément dont le type ne supporte pas cette opération.

## [GenCode]

- ❖ Pile pleine
  - ➢ Vérifie que toutes les variables peuvent entrer dans la pile en utilisant l'instruction assembleur TSTO.
  - ➢ Lève le flag OV (Overflow) dans le cas nécessaire, et l'instruction de branchement BOV est effectué pour sauter au label d'erreur, qui affiche le message "erreur de débordement" et qui termine par l'instruction ERROR.
- ❖ Déréférencement null
  - ➢ Accéder aux attributs ou méthodes d'un objet sans l'avoir instancié avec *new*. Exemple: si *ClassA* est une classe et *instA* une instance déclarée avec *ClassA instA*. Erreur si on exécute *instA.methA()*.
- ❖ Tas plein
  - ➢ Le tas est utilisé lors de la déclaration et l'utilisation d'objets. Celui-ci a une taille mémoire limitée, et n'instancie pas un objet avec un nombre d'attributs trop élevé.
- ❖ Débordement durant une opération arithmétique
  - ➢ Durant une opération arithmétique, le résultat est plus grand que le maximum sur 32 bit (que ce soit en entier ou en flottant).
- ❖ Input ou output
  - ➢ Lorsqu'un entier ou un flottant est lu en utilisant les fonctions readInt et readFloat, l'entrée est ininterprétable. Exemple: l'utilisateur écrit un string quand le programme attend un entier.
- ❖ CircularInclude
  - ➢ Si l'inclusion d'un fichier est faite et que celle-ci conduit à une inclusion circulaire.
- ❖ IncludeFileNotFound
  - ➢ Si le fichier inclus est introuvable.

# Commande Decac

L'exécution du compilateur est effectuée par la commande decac, qui peut être utilisée avec différentes options pour obtenir des résultats spécifiques sur un ou plusieurs programmes Deca. Le mode d'utilisation de cette commande et ses options sont décrits ci-dessous.

- **decac** : Affiche sur console les options disponibles avec une brève description de chacune des options.

```
[figueroa@ensipc63 Projet_GL]$ decac

 Options pour le compilateur

  -a
    Compiler le fichier source et generer le fichier .ass
  -b
    Affiche une bannière indiquant le nom de l'équipe.
  -p
    Arrête decac après l'étape de construction de
    l'arbre, et affiche la décompilation de ce dernier.
  -v
    Arrête decac après l'étape de vérifications.
  -n
    Supprime les tests de débordement à l'exécution.
  -r
    Limite les registres banalisés disponibles.
  -d
    Active les traces de debug.
  -P
    S'il y a plusieurs fichiers sources,
    lance la compilation des fichiers en parallèle.

[figueroa@ensipc63 Projet_GL]$
```

*Image 1. Utilisation de la commande decac*

- **decac <Fichier>** : Réalise le processus complet de compilation et génère le fichier en assembleur. Le fichier assembleur .ass est stocké dans le dossier source du fichier deca.
- **decac -b** : Affiche le nom de l'équipe. Cette option ne peut pas être utilisée avec d'autres options. Si c'est le cas, un message d'erreur s'affiche.

```
[figueroa@ensipc63 Projet_GL]$ decac -b
Équipe gl48
[figueroa@ensipc63 Projet_GL]$
```

*Image 2. Utilisation de la commande decac -b*

- **decac -p <Fichiers>** : Affiche la décompilation de l'arbre construit. Cette option peut être utilisée avec un ou plusieurs fichiers deca. Cette option n'est pas compatible avec l'option -v.

```
[figueroa@ensipc63 Projet_GL]$ decac -p src/test/deca/syntax/valid/created/while.deca
{
        while (true) {
                (5 + 2);
        }
}
[figueroa@ensipc63 Projet_GL]$
```

*Image 3. Utilisation de la commande decac -p*

```
[figueroa@ensipc63 Projet_GL]$ decac -p src/test/deca/syntax/valid/created/while.deca
src/test/deca/syntax/valid/created/print.deca
{
        while (true) {
                (5 + 2);
        }
}
{
        println("Alex le Boss");
}
[figueroa@ensipc63 Projet_GL]$
```

*Image 4. Utilisation de la commande decac -p avec plusieurs fichiers*

- **decac -v <Fichiers>** : Vérifie les fichiers deca.

```
[figueroa@ensipc63 Projet_GL]$ decac -v src/test/deca/syntax/valid/created/times.deca
[figueroa@ensipc63 Projet_GL]$
```

*Image 5. Utilisation de la commande decac -v et un fichier deca valide.*

- **decac -n <Fichiers> :** Supprime les vérifications pendant la génération du code assembleur. Bien qu'il est possible d'utiliser cette option avec les options -p et -v, l'option -n ne prend effet qu'au moment de la génération de code assembleur.

- **decac -r <X>** : Prend une valeur numérique entre 4 et 16. Un message d'erreur s'affiche sur la console si une valeur autre est inséré. Soit X la valeur numérique saisie pour l'option -r, elle permet de limiter les registres banalisés disponibles. Par exemple, lorsqu'on entre l'option -r 5, les registres disponibles seront R(0) à R(X - 1).

```
[figueroa@ensipc519 created]$ decac -r 3 while.deca
X must have a value between 4 and 16
[figueroa@ensipc519 created]$ decac -r 17 while.deca
X must have a value between 4 and 16
[figueroa@ensipc519 created]$
```

*Image 6. Message d'erreur lors de la saisie d'une valeur invalide pour l'option -r.*

- **decac -d <Fichiers> :** Active les messages de debug, l'option peut être utilisée plus d'une fois pour configurer le niveau des messages de trace souhaité. Si l'option n'est pas saisie, la configuration par défaut des traces sera utilisée.


*Image 7. Utilisation de la commande decac -d.*

- **decac -P <Fichiers> :** Compile chaque fichier parallèlement afin d'accélérer le processus de compilation. Un message d'erreur est affiché si on n'entre qu'un seul fichier.


*Image 8. Utilisation de la commande decac avec l'option -P.*

## Les extensions éventuelles de la [BibliothequeStandard].

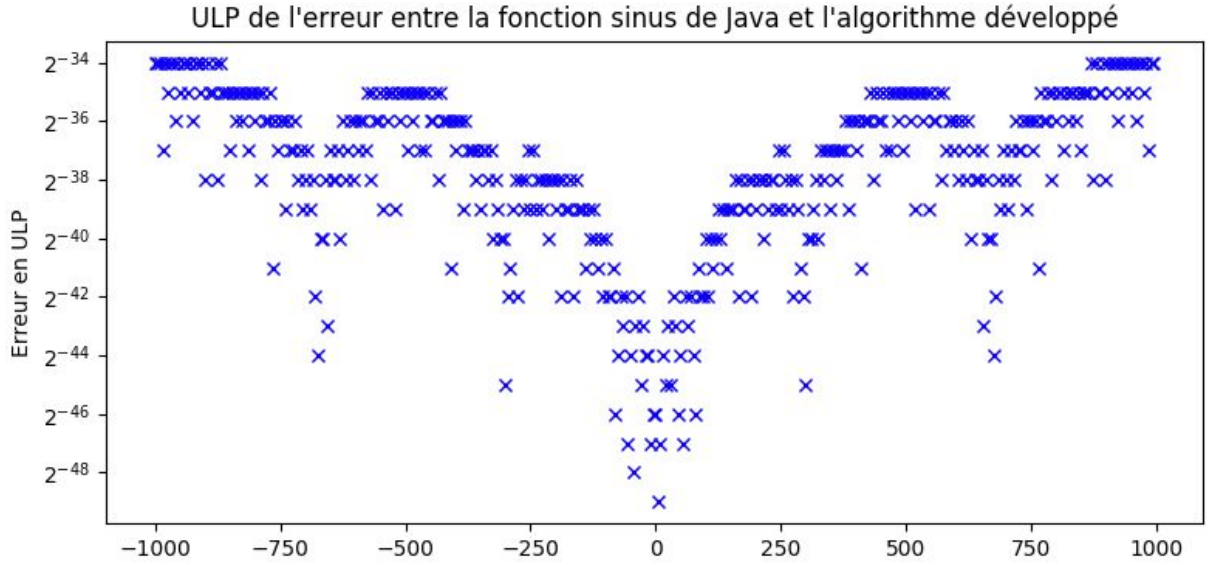
La Bibliothèque Standard a été étendue avec la classe Math car nous avons choisi de réaliser l'extension TRIGO. La classe Math contient donc les fonctions trigonométriques et le calcul d'ULP demandés, ainsi que d'autres algorithmes moins performants que ceux utilisés pour ces fonctions et des méthodes optimisants les calculs intermédiaires.

## Classe Math

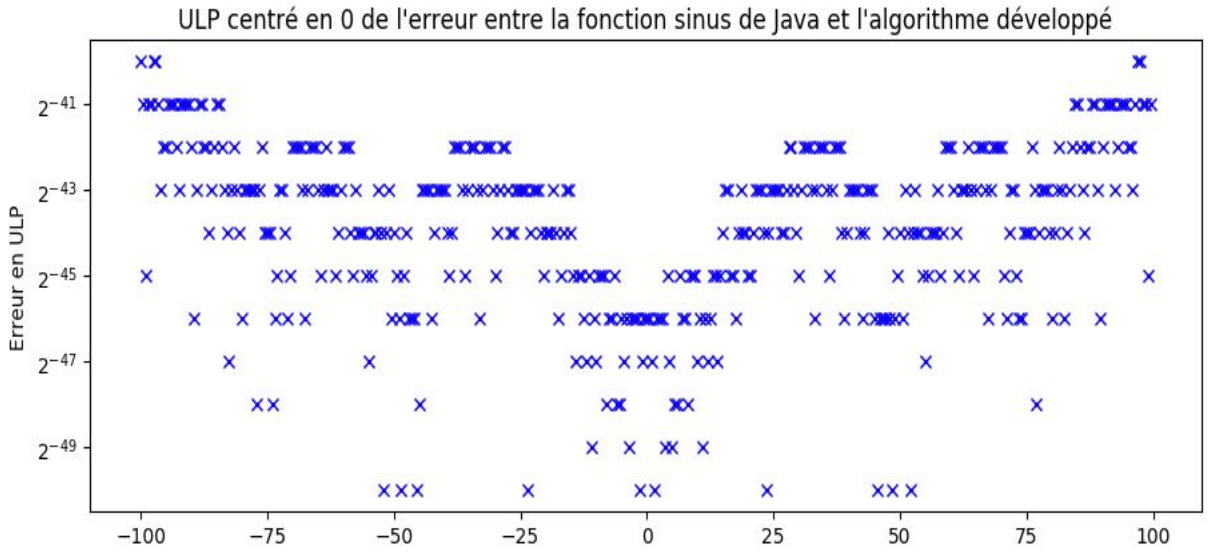
La classe Math de la Bibliothèque Standard est à inclure de la manière suivante: *#include "Math.decah"*.

La partie Objet de l'étape C n'ayant été que partiellement traitée, les résultats suivants sont obtenus en utilisant les algorithmes de la classe Math réécrit dans un fichier java. Ils profitent donc de la gestion des flottants plus aboutie de Java.

<u>Sinus :</u>

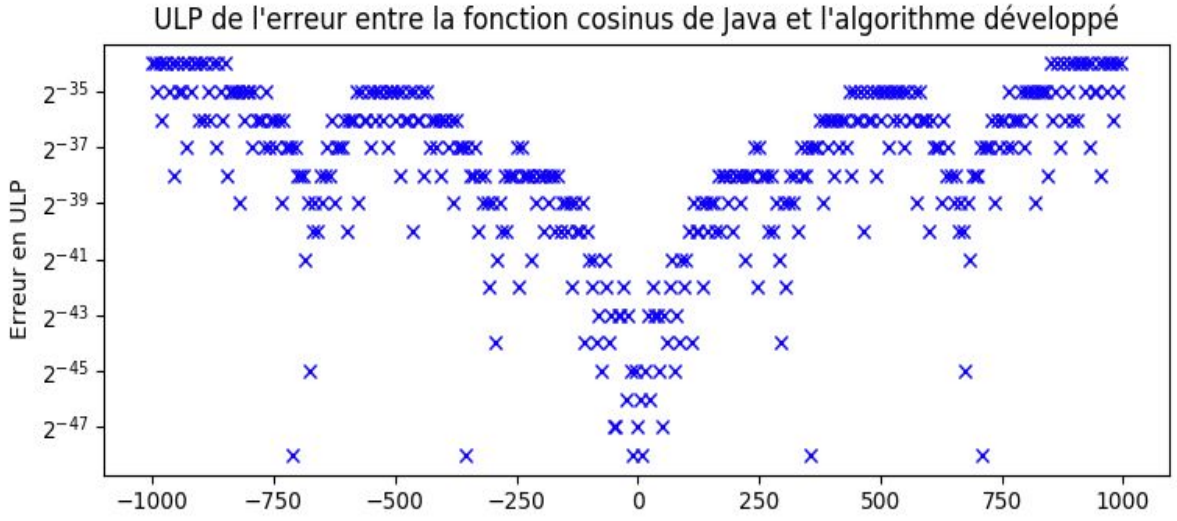



Erreur maximale : 0x1.0p-34
Erreur moyenne : 0x1.d8e81ep-37
Ecart-type : 0x1.204aecp-36

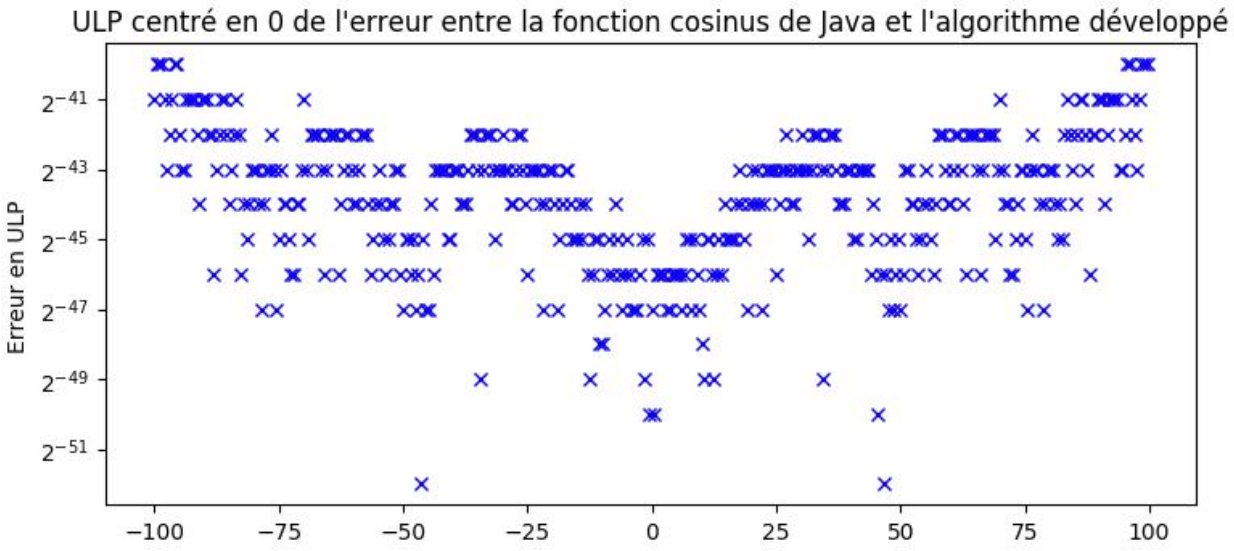



L'algorithme implémenté propose une marge d'erreur maximale de $2^{-34}$ par rapport la fonction sinus calculée par Java.

<u>Cosinus :</u>

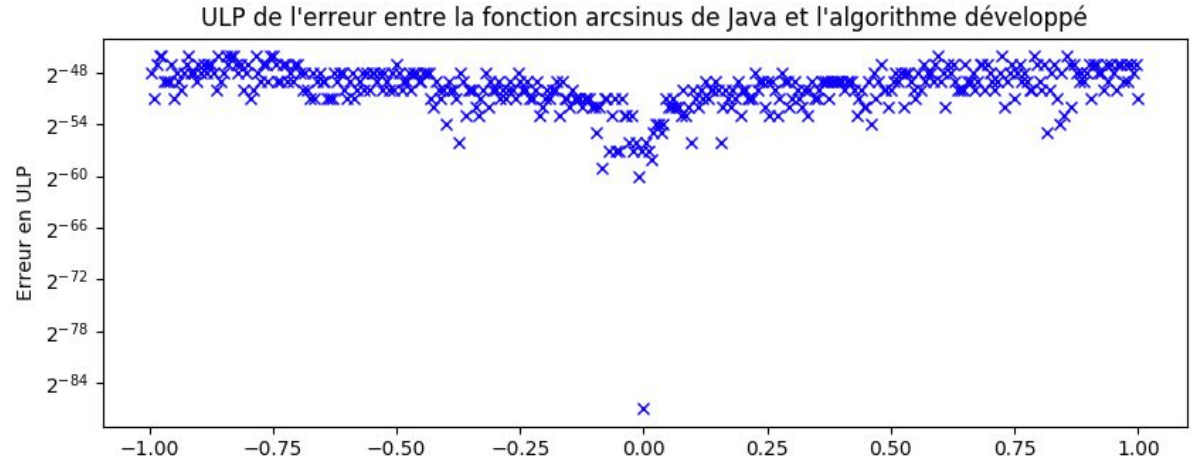



Erreur maximale : 0x1.0p-34
Erreur moyenne : 0x1.d18732p-37
Ecart-type : 0x1.1e30e6p-36




L'algorithme implémenté propose une marge d'erreur maximale de $2^{-34}$ par rapport la fonction sinus calculée par Java.

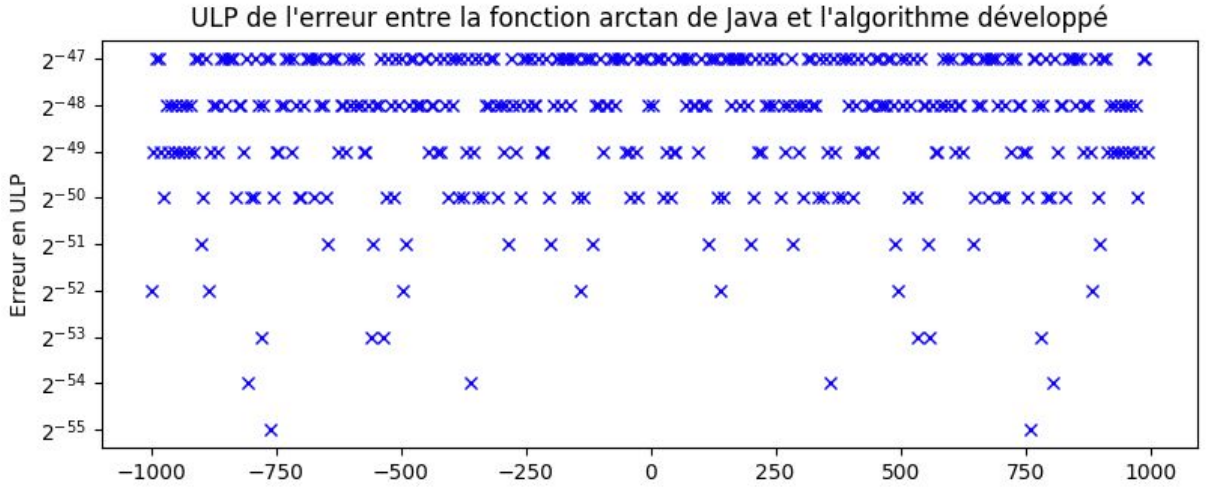
Arcsinus :




Erreur maximale : 0x1.0p-46
Erreur moyenne : 0x1.63f79ap-49
Ecart-type : 0x1.bcc39cp-49

L'algorithme implémenté propose une marge d'erreur maximale de $2^{-46}$ par rapport la fonction sinus calculée par Java.

Arctangente :




Erreur maximale : 0x1.0p-47
Erreur moyenne : 0x1.1dcdf4p-48
Ecart-type : 0x1.7369a6p-49

L'algorithme implémenté propose une marge d'erreur maximale de $2^{-47}$ par rapport la fonction sinus calculée par Java.