# Ratio

Projet de programmation objet - mathématiques pour l'informatique
imac2


**Résumé**

Ce projet se place à l'intersection de la programmation C++ et des mathématiques numériques. Il s'agit d'explorer une alternative à la représentation des réels avec des nombres à virgule flottante en représentant les réels par des rationnels, c'est à dire par le quotient de deux entiers relatifs. **Ce travail s'effectuera par binôme.**

## 1 Introduction

En `C++`, la représentation usuelle des nombres réels se fait par des nombres à virgules flottantes, codés avec un bit de signe, une mantisse et un exposant. Ce genre de représentation est très efficace sur un large panel d'applications, mais peut montrer ses limites dans certains cas. Notez par exemple la suite :

$$\begin{cases} u_0 &= \frac{1}{3} \\ u_{n+1} &= 4u_n - 1 \end{cases}$$

Cette suite est constante avec $u_n = \frac{1}{3} \quad \forall n \in \mathbb{N}$. Pourtant, en codant cette suite avec des `double` codé sur 64 bits, cette suite diverge au bout de quelques itérations. Dans ce projet, nous proposons d'explorer une alternative aux nombres à virgule flottante, en représentant les nombres réels par un nombre rationnel.

## 2 Nombres rationnels

Un nombre rationnel peut s'exprimer comme le quotient de deux nombres entiers :

$$x = \frac{a}{b} = \frac{\text{numerateur}}{\text{denominateur}}$$

L'idée de cette approche consiste à effectuer les calculs sur ces nombres rationnels et à les convertir en nombres à virgule flottante uniquement à la fin des calculs, lorsque l'on veut exploiter ou afficher le résultat. En théorie, les opérations sur les nombres rationnels n'impliquent aucune approximation numérique, et sont par conséquent très précis. En pratique, le numérateur et le dénominateur sont deux entiers codés sur un nombre fini de bits, ce qui implique des limites sur la précision des opérations effectuées.

### 2.1 Structure de données pour un rationnel

La structure de donnée la plus évidente pour représenter un nombre rationnel se compose simplement d'un numérateur et d'un dénominateur, soit deux entiers. On peut éventuellement coder le dénominateur avec un entier non signé, laissant le choix du signe au numérateur.

### 2.2 Formes standards

Dans le contexte de ce projet, nous proposons une forme standard pour les nombres rationnels :

- le dénominateur doit être positif.
- le quotient $\frac{a}{b}$ doit être une fraction irréductible, on a alors $\text{pgcd}(a, b) = 1$. On doit par exemple remplacer $\frac{18}{15}$ par $\frac{6}{5}$. Pour obtenir une fraction irréductible, il suffit de diviser le numérateur et le dénominateur par leur pgcd. En`C++`, vous pourrez utiliser la fonction `std::gcd`.
- on notera le nombre zéro : $0 = \frac{0}{1}$.
- on pourra explorer des possibilités d'exprimer l'infini comme $\infty = \frac{1}{0}$.

# 3 Partie mathématique

Cette partie est à réaliser avant la partie programmation. Il s'agit de formaliser les concepts que vous programmerez.

## 3.1 Opérations sur les rationnels

Voici quelques exemples d'opérations sur les rationnels qu'il faudra implémenter :

- somme de deux rationnels : $\frac{a}{b} + \frac{c}{d} = \frac{ad + bc}{bd}$
- produit de deux rationnels : $\frac{a}{b} \times \frac{c}{d} = \frac{ac}{bd}$
- inverse d'un rationnel : $\left(\frac{a}{b}\right)^{-1} = \frac{b}{a} \qquad \forall b \neq 0$

**Question :** Comment formaliseriez vous l'opérateur de division / ?

**Question :** Vous pourrez également explorer les opérations suivantes :

$$\sqrt{\frac{a}{b}}, \quad \cos\frac{a}{b}, \quad \left(\frac{a}{b}\right)^k, \quad e^{\frac{a}{b}}, \quad etc.$$

ou d'autres opérateurs que vous aimez bien.

## 3.2 Conversion d'un réel en rationnel

Un des points critique de ce projet consiste à convertir un réel représenté par un nombre à virgule flottante, en rationnel. Pour cela, je vous propose l'algorithme récursif 1. **Notez qu'en l'état, cet algorithme n'est valable que pour un réel positif ou nul**. Essayez de bien comprendre cet algorithme. D'après vous, à qui s'adresse la puissance $-1$ de la ligne 8 ou bien la somme de la ligne 12 ?

**Question :** Comment le modifier pour qu'il gère également les nombres réels négatifs ?

**Algorithm 1:** Conversion d'un réel en rationnel

**Function** `convert_float_to_ratio`

**Input:** $x \in \mathbb{R}^+$ : un nombre réel à convertir en rationnel
`nb_iter` $\in \mathbb{N}$ : le nombre d'appels récursifs restant

// première condition d'arrêt
**if** $x == 0$ **then** return $\frac{0}{1}$
// seconde condition d'arrêt
**if** `nb_iter` $== 0$ **then** return $\frac{0}{1}$
// appel récursif si $x < 1$
**if** $x < 1$ **then**
    return $\left( \texttt{convert\_float\_to\_ratio}\left(\frac{1}{x}, \texttt{nb\_iter}\right)\right)^{-1}$
// appel récursif si $x > 1$
**if** $x > 1$ **then**
    $q = \lfloor x \rfloor$ // partie entière
    return $\frac{q}{1} +$ `convert_float_to_ratio`$(x - q, \texttt{nb\_iter} - 1)$

### 3.3 Analyse

Une fois votre classe de rationnels implémenté, élaborez des tests pour voir dans quels cas ces nombres rationnels sont efficaces et dans quels cas ils ne le sont pas.

**Question :** D'une façon générale, on peut s'apercevoir que les grands nombres se représentent assez mal avec notre classe de rationnels. Voyez vous une explication à ça ?

**Question :** Lorsque les opérations entre rationnels s'enchaînent, le numérateur et le dénominateur peuvent prendre des valeurs très grandes, voir dépasser la limite de représentation des entiers en `C++`. Voyez vous des solutions ?

# 4 Partie programmation

Cette partie concerne ce que vous devrez implémenter en `C++`, ainsi que la façon dont vous devez l'implémenter.

## 4.1 Les rationnels

Naturellement, il vous est demandé de coder une bibliothèque de rationnels. Elle contiendra évidemment une implémentation de rationnel sous forme standard définie dans la section 2.2. Elle implémentera également des opérations sur les rationnels, comme dans la section 3.1, ainsi que des opérateurs tels que :

- une fonction pour transformer le rationnel sous forme de fraction irréductible (forme standard de la section 2.2)
- le moins unaire
- la valeur absolue
- la partie entière
- le produit d'un nombre en virgule flottante avec un rationnel et le produit d'un rationnel avec un nombre en virgule flottante.
- les opérateurs de comparaison : `==`, `!=`, `>`, `<`, `<=` et `>=`.
- une fonction d'affichage sous forme de surcharge de l'opérateur `<<`.

Enfin, vous implémenterez la conversion d'un réel en rationnel, comme dans la section 3.2. Pensez aux nombres réels négatifs.

## 4.2 Les exemples

Vous fournirez une batterie d'exemples destinés à montrer que votre bibliothèque fonctionne, ainsi que pour aider un utilisateur à commencer à utiliser vos rationnels.

## 4.3 Les tests unitaires

Tout au long de la phase de développement, vous mettrez en œuvre des tests unitaires afin de vérifier que chaque évolution de votre programme n'altère pas ce qui fonctionne déjà.

### 4.4 Pour avoir un max de points

Pour aller plus loin, essayez :

— de coder votre classe en `template`. Vous pourrez alors choisir de représenter vos nombres rationnels avec des `int`, `long int`, etc.

— de coder votre classe en `constexpr`, afin que le cas échéant, les calculs puissent se faire à la compilation.

### 4.5 Compilation

Il est largement conseillé d'utiliser une version moderne du C++ (11, 14 ou 17). La compilation de votre projet devra être gérée avec **cmake** (au moins la version 3.13). Pensez à déployer une arborescence de vos fichiers appropriée à un projet **cmake**, avec des sous projets. Pensez à agrémenter votre code d'un `readme.md` pour aiguiller l'utilisateur sur la structure de votre projet et son utilisation.

**Votre projet devra nécessairement compiler et exécuter sur les machines de l'université.**

### 4.6 Lisibilité

J'attends un code lisible et bien organisé. Bien organisé signifie que vous répartirez judicieusement votre code sur plusieurs fichiers, et que les fonctions ou classes composants chaque fichier seront elles aussi bien réparties. Par ailleurs, il est nécessaire de commenter votre code en gardant à l'esprit qu'il doit être compréhensible même sans ces commentaires : essayez de nommer vos variables et vos fonctions explicitement et n'hésitez pas découper de longs algorithmes en plus petites fonctions.

### 4.7 Documentation

La bibliothèque que vous écrirez devra être auto documenté, avec Doxygen par exemple.

### 4.8 Gestionnaire de versions

Vous devrez utiliser le gestionnaire de version Git et héberger votre dépôt sur une plateforme en ligne, telle que GitHub, GitLab ou Bitbucket. Pensez à faire des `commit` très régulièrement, ça sera pour moi une indication permettant de vérifier que vous n'avez pas tout fait au dernier moment.

### 4.9 Côté technique

Voici ce qu'il serait bon de rencontrer dans votre projet :

- classes
- classes et fonctions template
- polymorphisme
- usage d'outils de la STL
- exceptions pour traiter les erreurs
- `asserts` pour détecter et prévenir les erreurs de programmation
- espaces de nommage
- fonctions lambdas `constexpr`
- (bonus) fonctions `variadics`

Voici ce qu'il serait regrettable de rencontrer dans votre projet :

- une mauvaise organisation de vos fichiers et de vos classes
- des variables, des fonctions ou des types mal nommés
- des références non constantes alors qu'elles devraient l'être
- des passages par copies non justifiés
- des erreurs de segmentation
- des fuites de mémoires
- des bugs
- du code de debug non retiré
- du code mort (= des portions de code non utilisés par le programme)
- du code dupliqué
- du code illisible d'un point de vue sémantique (des `for` dans des `for` dans des `if` dans des `for` dans des `else` par exemple)
- du code illisible d'un point de vue esthétique (indentation irrégulière, mélange camelCase / snake_case, mélange tabs / spaces)
- des mega fonctions de plus de 30 lignes
- des mega fichiers de plus de 500 lignes
- des variables globales non `constexpr`
- des crashs lorsque l'utilisateur effectue une opération non prévue

# 5 Modalités de rendu

Ce projet sera à réaliser :

- par binôme
- livrable : xxx
- envoyez moi votre lien git dès que vous commencez.

Le jour du rendu, votre *repository* devra contenir :

- votre code (bibliothèque, exemple, tests, etc.) et un `readme.md` mentionnant vos noms.
- votre rapport au format `nom1_nom2.pdf`.

## 5.1 Rapport

Vous fournirez en version électronique un rapport de 5 pages minimum et de 10 pages maximum. **Consacrez suffisamment de temps au rapport car il représente une bonne partie de la note finale**.

Voici quelques conseils :

- Ne perdez pas de temps à réexpliquer le sujet du projet, l'enseignant le connaît déjà. De manière plus générale, ne détaillez pas des méthodes déjà expliquées dans l'énoncé à moins que vous les ayez modifiées.
- Un rapport sert surtout à montrer comment vous avez fait face aux problèmes (d'ordre algorithmique/mathématique). Certains problèmes sont connus (on en parle dans l'énoncé), d'autres sont imprévus. Montrez que vous les avez remarqués et compris. Donnez la liste des solutions à ce problème et indiquez votre choix. Justifiez votre choix (vous avez le droit de dire que c'est la méthode la plus facile à coder).
- Un rapport n'est pas un listing de votre programme où vous détaillez chaque fonction, évitez de mettre trop de code. Vous devez par contre détailler vos structures de données et mettre du pseudocode pour expliquer vos choix algorithmiques.
  Il est autorisé d'utiliser des "raccourcis" tels que "`initialiser T à 0`" plutôt que de détailler la boucle faisant la même chose.

Merci de respecter les directives décrites dans les sections suivantes.

### 5.1.1 Partie programmation : tableau bilan

Votre rapport doit commencer par une page présentant un tableau bilan pour la partie programmation. Il devra lister les fonctionnalités sur lesquelles vous avez travaillé. Vous indiquerez en particulier :

- éléments demandés et codés qui fonctionnent.
- éléments demandés et codés qui ne fonctionnent pas.
- éléments demandés mais (malheureusement) pas codés.
- éléments non demandés (options) et codés qui fonctionnent.
- éléments non demandés mais pas codés ou qui ne fonctionnent pas, mais pour lesquels vous proposez des solutions

### 5.1.2 Partie mathématiques

J'attire votre attention sur le fait que la partie mathématique sera évaluée essentiellement à travers votre rapport.

## 5.2 Evaluation

Ce projet sera évalué avec deux notes, une pour la partie programmation comptant pour la note de contrôle continu pour la matière "programmation objet" et une notre pour la partie mathématiques comptant pour la note de contrôle continu pour la matière"math-info".

## 5.3 Pour finir

N'hésietez pas à vous approprier le sujet : tout ajout non demandé sera le bienvenu. Bon courage ! !