|  |
| --- |
| CFPTI |
| TPI |
| Grapheur de fonction |
| **Trifunovic Ivan** |
| **Sous la direction du Professeur Christian Blanvillain** |
| **19/06/2017** |

Table des matières

[Introduction 1](#_Toc484616034)

[Fonction paramétrique 2](#_Toc484616035)

[Cahier des charges 3](#_Toc484616036)

[Etude d’opportunité 3](#_Toc484616037)

[Pourquoi ce sujet ? 3](#_Toc484616038)

[Ce que mon projet à de plus 3](#_Toc484616039)

[Description de l’existant 3](#_Toc484616040)

[Analyse fonctionnelle 4](#_Toc484616041)

[Fonctionnalités du projet 4](#_Toc484616042)

[Interface 4](#_Toc484616043)

[Analyse organique 5](#_Toc484616044)

[Architecture 5](#_Toc484616045)

[Méthodes de réalisation 5](#_Toc484616046)

[Initialisation du centre 5](#_Toc484616047)

[Calcul distance entre deux pixels 5](#_Toc484616048)

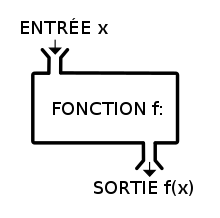
[Création axes x et y 5](#_Toc484616049)

[Annexe 6](#_Toc484616050)

[Bibliographie 7](#_Toc484616051)

# Introduction

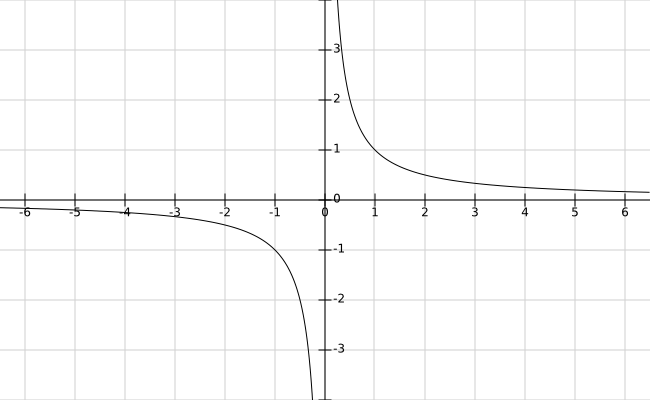
Une fonction est une relation entre deux ensembles un ensemble d’entrée et un autre de sortie.



Un exemple de fonction est la fonction

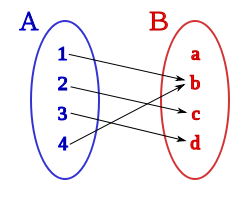
On remplace le paramètre qui par les entrées suivantes.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| x Entrée | -5 | -2 | -1 | 0 | 1 | 2 | 5 |
| f(x) Sortie | -0.2 | -0.5 | -1 | Erreur | 1 | 0.5 | 0.2 |



L’ensemble de départ A ne peut pas avoir plus d’une sortie alors que l’ensemble de sortie B peut avoir plusieurs entrées A.

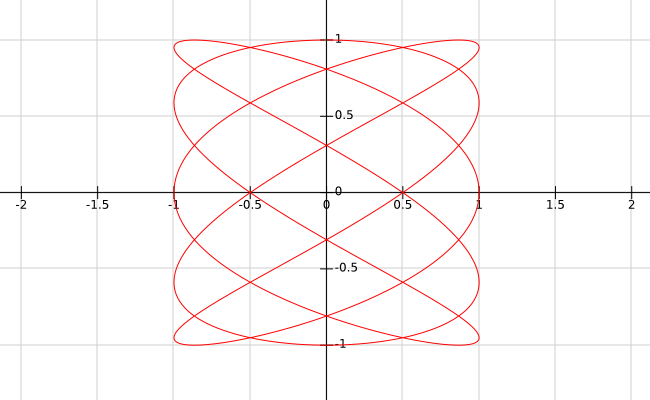
On peut voir dans l’image ci-dessous que A(1) et A(4) ont le deux la même sortie B(b) alors qu’aucune entrée ne possède deux sorties.



## Fonction paramétrique

Une fonction paramétrique est une fonction qui en entrée contient une fonction et en sortie une autre et ces deux fonctions on le même paramètre souvent noté t.

Exemple de courbe paramétrée



# Cahier des charges

L’interface présente : un menu, une zone de définition de l’équation et un graphe en 2D (x,y). Il est prévu deux types de courbes :

1. y(x) = … → Cartésiennes Affines

2. { x(t) = … ; y(t) = ... } → Cartésiennes Paramétriques

La touche entrée, valide l’équation et génère l’affichage de la courbe. Si l’équation n’est pas valide un avertissement est signalé à l’utilisateur et rien ne se passe.

Trois boutons d’action sont présents :



Le bouton Dessiner est activé uniquement lorsque l’utilisateur est en train de saisir une nouvelle équation. Il ouvre une fenêtre modale dans laquelle on peut :

● Définir l’intervalle d’affichage pour les valeurs de x ou de y, pour la zone d’affichage du graphe. L’autre dimension est toujours proportionnelle pour que le repère reste auto-normé en fonction de la taille de l’application. Par défaut centré sur zéro.

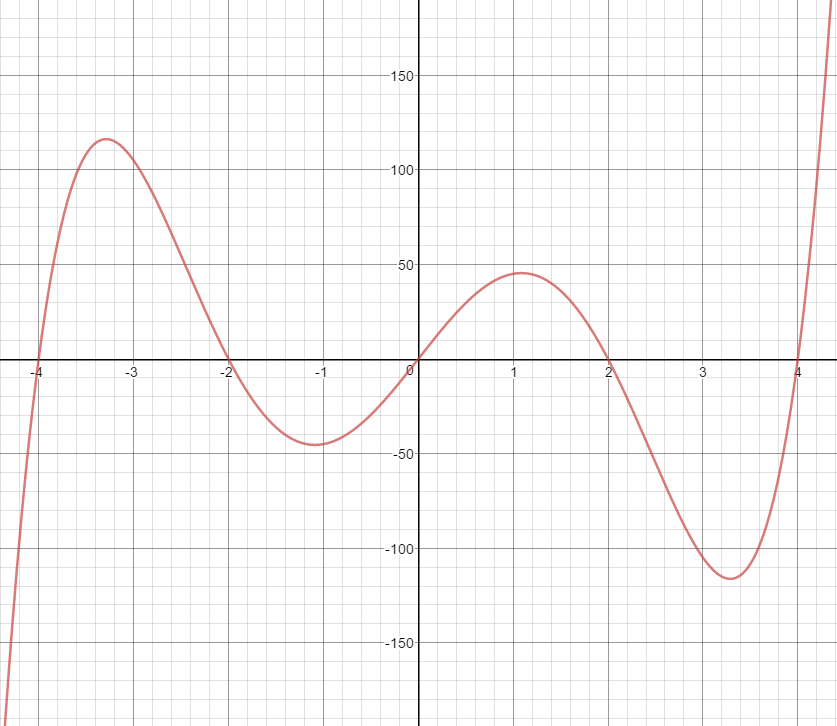
● Affiner le pas de calcul dont la valeur par défaut proposée dans la fenêtre sera calculée automatiquement en fonction du niveau de grossissement et de l’intervalle d’affichage de manière à éviter une trop forte pixellisation. La précision des calculs sera celle proposée de base par la librairie utilisée pour les calculs.

Le bouton Annuler permet de réinitialiser la saisie de l’équation d’une nouvelle courbe. Il ne se comporte donc pas comme un CTRL+Z. Il est activé uniquement lorsque l’utilisateur est en train de saisir une nouvelle équation.

Le bouton Effacer nettoie les courbes de la zone graphique d’affichage et la réinitialise, c’est à dire redéfinit l’intervalle d’affichage par défaut : [-5 ; +5] en x et y ainsi que le pas de calcul : 1 point par pixel affiché. Lorsqu’il est cliqué, un popup d’avertissement demande une confirmation avant d’effacer la zone d’affichage. Une fois la courbe effacée, il est impossible d’annuler l’opération.

Les éléments de l’interface sont en dehors du graphique.

La zone d’affichage des fonctions est similaire à la figure suivante :



L’agencement des éléments d’interface est similaire à cette capture d’écran :

# Etude d’opportunité

Dans le cadre de de mon projet TPI, j’ai voulu développer un grapheur qui consiste à reproduire le graphique d’une équation dans un plan R2 (le graphique d’une fonction). J’ai choisi de faire ce projet, car je trouve que les graphiques d’équations sont beaucoup plus parlants que les chiffres.

Dans mon projet, j’ai choisi de mettre en avant deux types de graphique. Les équations cartésiennes affines et les courbes paramétrées. Mon application possède une interface claire et rapide. Pour obtenir le résultat de l’équation et afficher la courbe il suffit juste de l’écrire de plus une option permet de régler la précision de calcul.

Dans les différents grapheur[[1]](#footnote-1) se trouvant sur le internet, beaucoup sont performants mais la plus part du temps il faut cliquer sur un bouton pour afficher le graphique. Certains ne sont pas précis et restreint à des petits intervalles avec une simple interface sans zoom. Bien sûr, il est impossible de reproduire une fonction parfaitement car on ne peut pas calculer un nombre infini de points sur un ordinateur.

# Analyse fonctionnelle

## Fonctionnalités du projet

Le programme permet de tracer le graphique d’une équation cartésienne affine et paramétrique dans un plan R2.

L’interface présente un graphique avec deux axes x et y. Deux Zone de textes sont présents en bas à gauche. L’utilisateur peut directement insérer du texte dans les champs.

En fonction de ce que l’utilisateur entre l’équation qu’il a entrée sera dessinée sur le graphique. Si l’équation est fausse le graphique de la fonction ne s’affiche pas.

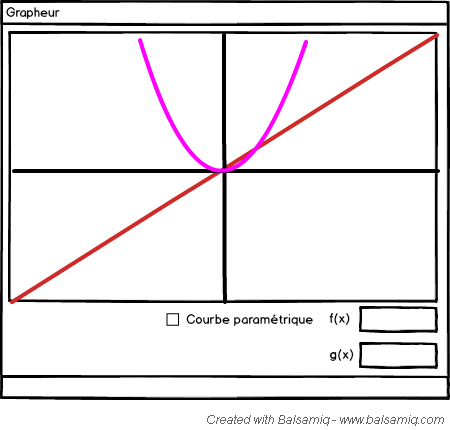
Un menu option est présent en haut à gauche permet d’afficher les différents réglages. L’utilisateur peut mettre l’intervalle du graphique qui lui correspond le plus par défaut il est à **[-5 ; +5]**.Les dimensions du graphique (intervalle) sont toujours proportionnelles entre eux.

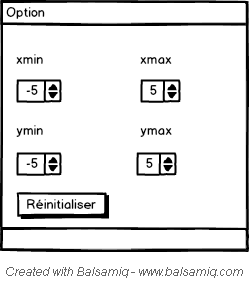
Un bouton réinitialiser est présent dans le menu option. Il permet de remettre les dimensions du graphique par défaut **[-5 ; +5]**, ainsi que la précision de calcul et nettoie toutes les courbes présentes sur le graphique.

Pour effacer une fonction l’utilisateur peut aussi enlever les équations présentes dans les zones de texte.

Un zoom est présent en scrollant la molette de la souris.

## Interface





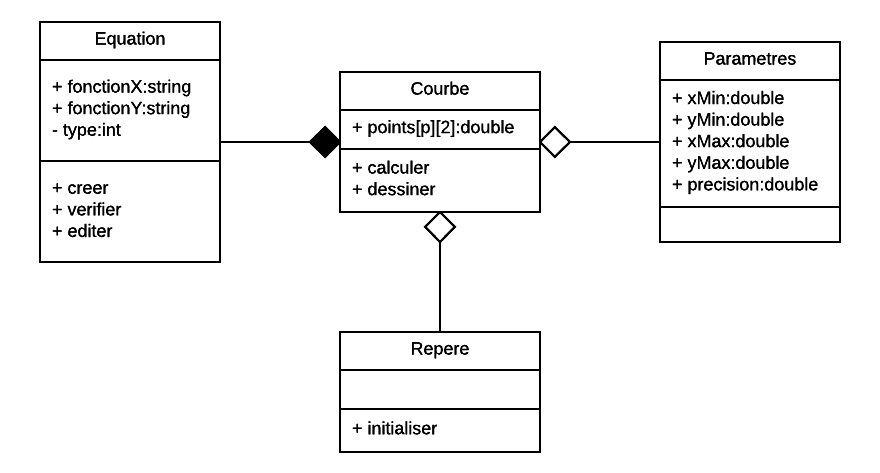
# Analyse organique

Pour élaborer mon projet, j’ai utilisé le logiciel de développement Visual Studio 2015, ayant appris le langage de programmation C# sur ce logiciel et programmé avec celui-ci pendant trois années. Durant lesquelles j’ai acquis beaucoup de connaissances sur ce programme.

Pour les sauvegardes, j’ai utilisé Git Hub et plusieurs stockages externes dont une clé USB et 4 disques durs externes. J’ai stocké mes données à plusieurs endroits pour être certain de ne pas risquer de perdre la totalité des mes données

Au niveau de l’organisation de mon code, j’ai utilisé une interface Modèle-Vue car une interface MVC modèle-vue-contrôleur aurait été obsolète pour ce type de projet.

## Architecture



Un objet *Courbe* est créé pour chaque nouvelle courbe dessinée et est capable de se dessiner dans un contexte graphique passé en paramètre. Il contient les informations saisies par l’utilisateur sous la forme de deux objets : *Equation* et *Parametres*.

L’objet *Equation* est créé et modifié par l’objet *Editeur* (non représenté ici). L’objet *Parametres* a tous les attributs nécessaires pour pouvoir calculer les *p* points en *x,y* de la courbe avant de l’afficher. Ce nombre *p* de points dépendra de la taille de la zone affichée, de la précision demandée et du type de courbe (affine ou paramétrique).

L’objet *Repere* ne fait qu’initialiser le contexte graphique dans lequel la *Courbe* se dessine.

La librairie CSharp-Eval (<http://csharp-eval.com/HowTo.php>) est utilisée pour interpréter les équations saisies par l’utilisateur et calculer les points de la courbe dessinée.

## Méthodes de réalisation

### Initialisation du centre

Initialiser le point x zéro et y zéro au centre de l’image pour que le graphique soit exactement le même que ceux utilisés habituellement en mathématiques.

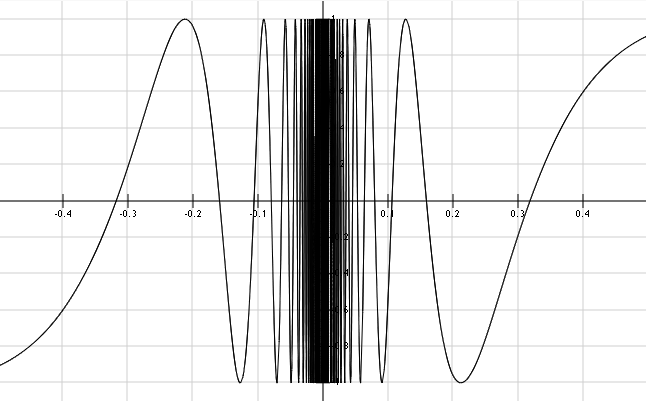
### Calcul distance entre deux pixels

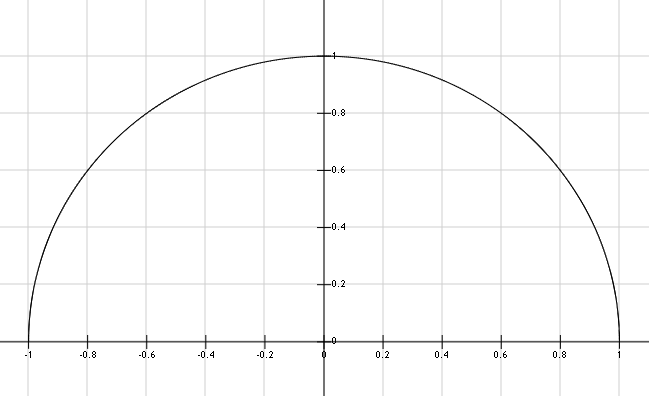
Initialisation de la taille d’un pixel selon la taille de la fenêtre en utilisant les variables de la classe Parametres.

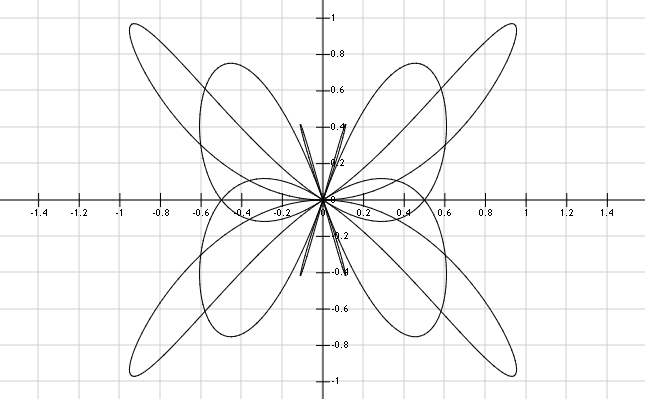
### Création axes x et y

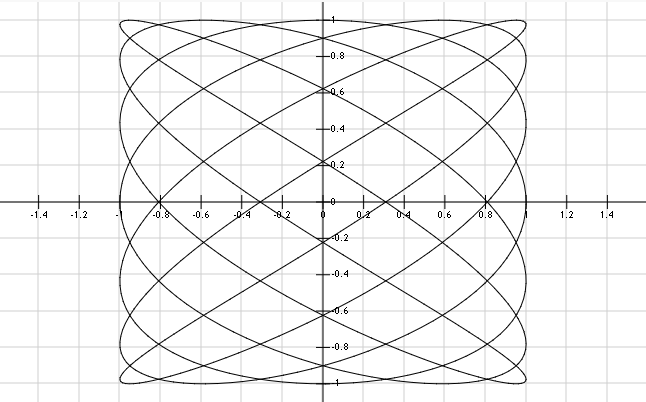
# Tests

## Jeux de test









|  |
| --- |
| Planning |
| Voici les principales étapes du projet, pour un total de 80 heures de travails (10 jours x 8h). Les tests et la documentation seront réalisés en continue, en parallèle au développement.  Jour 1  Création des forms et autres éléments d’interface : l’application, le menu, la zone du graphe. Seules les vues sont réalisées : pas les actions.  Jour 2  Création des classes définissant la courbe : *Courbe*, *Equation* et *Parametres*. Implémentation minimaliste de la méthode *Courbe*.dessiner() avec une *Equation* de test simple codée en dur (style y(x)=x).    Jour 3 et 4  Ajout du champ de saisie de l’*Equation*. Implémentation des boutons d’action. Implémentation de la méthode *Courbe*.dessiner() avec l’appel à la librairie CSharp-Eval.    Jour 5 et 6  Implémentation de la méthode *Equation*.verifier(). Ajout dans la classe *Parametre* des calculs automatiques de la précision et des intervalles de valeurs.  Jour 7 et 8  Finalisation du code et tests.  Jour 9 et 10  Finalisation de la documentation. |

# Bibliographie

Wikipédia. Fonction (mathématiques) <https://fr.wikipedia.org/wiki/Fonction_(math%C3%A9matiques)>

Mathematiquesfaciles. Les fonctions. <http://www.mathematiquesfaciles.com/les-fonctions_2_74419.htm>

Apprendre-en-ligne. Etude de courbes paramétrées. https://www.apprendre-en-ligne.net/MADIMU2/ANALY/ANALY6.PDF

1. Exemple de grapheur : <https://www.graphsketch.com/> [↑](#footnote-ref-1)