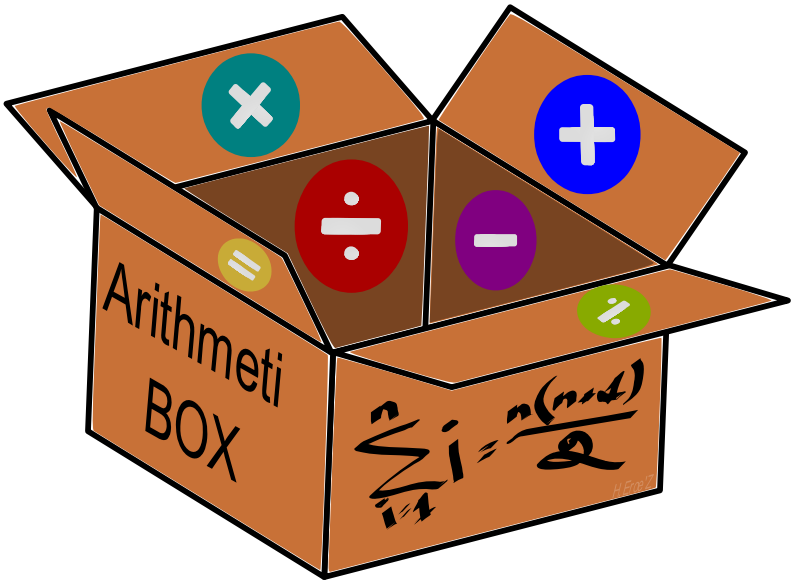
ArithmetiBox

Projet semestre 3



Team ArithmetiBox :

- RAT Quentin
- WONG Jason
- KAING Jack
- DOS SANTOS Jeremy
- MOUGAMMADOUARIBOU Fahath

Groupe: A1

SOMMAIRE

I – DESCRIPTION DU PROJET

Résumé du projet:

Le but du projet est de créer une boite à outils mathématique, sous forme de site web, qui sera par la suite intégrer à Ataraxy, le site du client. Pour le projet nous utiliserons les langages tel que le PHP, HTML, CSS et le langages de compilation mathématique LaTeX. Ce projet sera évaluer par notre enseignant et client Mr Hébert. Il s'agit de réaliser ce projet pendant le S3, pour cela nous serons cinq pour le développer, Quentin Rat sera le chef de projet charger de le rendre. La création de divers diagrammes sera demandée. Nous utiliserons essentiellement texMaker comme logiciel.

Le projet vas se dérouler en 4 grande parties:

- -La première partie consiste à:
  - -Mettre en place un espace de partage Gmail/Github pour pouvoir travailler en groupe à distance
  - -Prendre en main les logiciels et langages tel que Github et LateX
  - -Nous allons concevoir des diagrammes pour modéliser le projet
  - -Il faudra planifier les tâches avec GanttProject
- -La deuxième partie consiste à:
  - -Schématiser sur papier un aperçu graphique de la page web
  - -Créer un logo de la page (en .svg)
  - -Commencer à développer en HTML et CSS le squelette de la page web
- -La troisième partie consiste à:
  - -Commencer à coder les différentes fonctionnalités en PHP (ex: PGCD...)
  - -Intégrer les fonctionnalités au menu
  - -Générer le code Latex pour un meilleur affichage/rendu
  - (c'est trois sous parties se feront petit à petit en fonction de l'avancement du cours de cryptanalyse)
- -La quatrième partie consiste à:
  - -Corriger les erreurs
  - -Tester toute les fonctionnalités

Contraintes:

Temps: Notre principal contrainte sur ce projet sera le temps, il faudra impérativement respecter le délai.

Objectifs (classement coût, délai, qualité):

1ère: il faut respecter le délai et il faudra procéder à une présentation final, il y aussi des revues toute les semaines pour voir l'avancement du projet.
2ème: il faut que la page web soit bien codé c'est-à-dire pas de bug ou de dysfonctionnement
3ème: Le coût de la page web est nulle

II - LES SOLUTIONS MISE EN OEUVRE

La page web est un site développé dans les languages HTML, CSS, PHP et LaTeX, c'est une une boite à outils mathématique qui sera par la suite intégrer à Ataraxy, les différent outils développés seront des algorithmes d'arithmétique vus en cours de S3 (ex: PGCD, codage César...). La page web sera constitué d'un menu à gauche où il y aura plusieurs onglets répertoriés, chaque onglet correspondra à un algorithme vu en cours. La page devra aussi générer du code Latex afin de représenter les détails des résultats obtenus par les fonctionnalités.

Les parties prenantes du projet son les cinq développeurs et le professeur, aussi client, qui sera là pour noter notre travail et vérifier l'avancement du projet.

III – LES RISQUES ET PROBLÈMES RENCONTRÉS

Les risques du projet sont :
- Tombe malade
- Surcharge de travail
- Perte de données (ex : perte du code)
- Mauvaise planification du temps de travail
- Bug pendant une présentation
- Trop s'éloigner du sujet donné
- Être trop ambitieux

Les 2 risques ayant la plus grande probabilité de se réaliser sont:
- Surcharge de travail
- Mauvaise planification du temps de travail

Pour être sûr que ces situations n'arrivent pas, nous allons prendre soin de bien rédiger les tâches à accomplir et bien organiser son temps pour ne pas être surmener.
Si un des risques cité venait à se produire il faudrait tout de suite le corriger pour ne pas mettre en
péril l'avancement du projet.

Le principal problème rencontré lors du projet, qui n'était pas cité, était un problème avec la bibliothèque GMP pour le PHP qui gère les grandes valeurs, nous utilisons principalement tous MAMP sur mac et la bibliothèque n'était pas disponible et nous avons rencontré des difficultés pour l'utiliser nous avons finalement installé WAMP sur un pc avec windows qui lui bénéficie de la bibliothèque GMP déjà installée.

IV – GESTION DE PROJET

USER STORIES:

1. En tant qu'utilisateur je veux pouvoir calculer le pgcd afin de connaître le plus grand diviseur commun.

2. En tant qu'utilisateur je veux pouvoir calculer le plus grand commun diviseur (pgcd) en appliquant l'algorithme d'Euclide étendue afin de connaître le pgcd entre deux nombres et d'avoir le détail du calcul sous forme d'un tableau (a, b, r, q).

3. En tant qu'utilisateur je veux pouvoir calculer non seulement leur plus grand commun diviseur (PGCD), mais aussi un de leurs couples de coefficients de Bézout afin de connaître le pgcd et d'avoir le détail du calcul sous forme d'un tableau (a, b, r, q, u, v).

4. En tant qu'utilisateur je veux pouvoir calculer l'inverse modulaire d'un nombre afin de savoir si l'inverse de ce nombre modulo n, existe ou non.

5. En tant qu'utilisateur je veux pouvoir calculer les puissances modulaires plus communément appelées exponentiation modulaire rapide afin de connaître les puissances entières d'un nombre.

6. En tant qu'utilisateur je veux pouvoir calculer les diviseurs d'un nombre grâce à l'algorithme de factorisation afin de connaître tout les diviseurs de ce nombre et ensuite d'avoir sa décomposition en produit de nombres premiers.

7. En tant qu'utilisateur je veux pouvoir calculer l'inverse d'une matrice modulaire afin de connaître l'inverse d'une matrice cette fonction sera aussi utile pour décrypter un code crypté avec le chiffrement de Hill.

8. En tant qu'utilisateur je veux pouvoir afficher la liste des nombres premiers jusqu'à n afin de savoir si un nombre est premier ou non.

9. En tant qu'utilisateur je veux pouvoir calculer la valuation p-adique afin de connaître de connaître la valuation p-adique de ce nombre.

10. En tant qu'utilisateur je veux pouvoir calculer la congruence afin de résoudre par exemple des équation diophantiennes.

11. En tant qu'utilisateur je veux pouvoir crypter et décrypter un message chiffré avec un chiffrement de Hill afin pour pourvoir lire clairement ce message ou pour le cacher.

12. En tant qu'utilisateur je veux que le résultat de mes calculs soient affichés avec du LateX afin d'avoir un affichage plus esthétique.

13. En tant qu'utilisateur je veux pouvoir saisir des nombres très grand afin de de pouvoir faire différents calculs (pgcd, congruence...) avec ces grand nombres.

14. En tant qu'utilisateur je veux pouvoir faire une attaque par force brut ou par dictionnaire afin de décrypter les messages.

15. En tant qu'utilisateur je veux pouvoir naviguer sur la page web simplement afin de ne pas perdre de temps.

16. En tant qu'utilisateur je veux pouvoir crypter et décrypter un message chiffré en césar afin de pourvoir lire clairement ce message ou le crypter.

17. En tant qu'utilisateur je veux pouvoir crypter et décrypter un message avec la méthode affine afin de pourvoir lire clairement ce message ou le crypter.

21. En tant qu'utilisateur je veux pouvoir crypter et décrypter un message avec la méthode RSA afin de pouvoir lire clairement ce message ou le crypter.

22. En tant qu'utilisateur je veux pouvoir crypter et décrypter un message avec la méthode de substitution afin de pouvoir lire clairement ce message ou le crypter.

23. En tant qu'utilisateur je veux que quand mes saisies de données soient vérifier et retourne un message d'erreur en cas de données incorrectes afin de pouvoir limiter les erreurs de calculs ou de saisie de l'utilisateur.

24. En tant qu'utilisateur je veux pouvoir joindre un fichier avec un texte à crypter ou décrypter contenu dans ce fichier afin de ne pas avoir a copier coller le message dans la zone requise.

Stories techniques

18. En tant que développeur, je dois installer le logiciel Wamp/Mamp afin de pouvoir éxécuter du code php.

19. En tant que développeur, je dois disposer d'un gitHub afin de pouvoir partager mon code avec le reste de l'équipe.

20. En tant que développeur, je veux pouvoir disposer d'un groupe de messagerie afin de pouvoir discuter avec le reste de l'équipe projet sur par exemple un problème rencontré.

Stories

1. créer une fonction PGCD
2. créer une fonction Euclide
3. créer une fonction Euclide étendu
4. créer une fonction Inverse modulaire
5. créer une fonction Exponentiation modulaire rapide
6. créer une fonction de Décomposition de nombre
7. créer une fonction Inverse d'une matrice modulaire
8. créer une fonction Test de primalité
9. créer une fonction Valuation p-adique
10. créer une fonction Congruence
11. créer une fonction Chiffrement Hill
12. afficher le résultat de toutes les fonctions en LateX

13. inclure la bibliothèque GMP au fonction pour pouvoir manipuler des grand nombres
14. faire des attaques par force brute ou par dictionnaire
15. faire une page web ergonomique et simple avec un menu
16. créer une fonction chiffrement César
17.créer une fonction chiffrement Affine
18.Installer logiciel requis pour le projet
19.Installer git et apprendre à maitriser git/gitHub
20.Créer un espace de messagerie pour pouvoir échanger
21.créer une fonction pour RSA
22.créer une fonction pour Substitution
23.faire Hill en dimension n
24.filtrer les saisies et afficher un message d'erreur lorsque la saisie n'est pas correcte
25.inclure la possibilité de joindre un fichier texte et le décrypter au lieu de le copier/coller ou l'écrire dans l'espace prévu

Estimation du temps et de la difficultés

N° des stories	Facile	Moyen	Compliquer
18	2		
19	5		
20	1		
15		23	
1	3		
2	4		
3		5	
4		4	
5		3	
6	3		
7		6	
8		4	
9	3		
10	3		
11			17
16			15
17			19
12			20
13			26
14			10
21			16
22			18
23			8
24	4		
25	3		

Total= 225 points
Nous avons une équipe de 5 développeurs
1 développeur peut faire 17 points 5*17=85 points par mois
225/85= 2,64 mois

Liste des Sprints

Sprint 1 - Date Estimer: 3 octobre - Cout : 31
-Cas 18 : Installer logiciel requis pour le projet
-Cas 19 : Installer git et apprendre à maitriser git/gitHub
-Cas 20 : Créer un espace de messagerie pour pouvoir échanger
-Cas 15 : faire une page web ergonomique et simple avec un menu
    -Schématiser sur papier un aperçu de la page web
    -Créer un logo
    -Créer le squelette de la page

Nous avons présenter cette partie le 12 octobre, nous étions donc dans les temps

Sprint 2 - Date Estimer: 2 novembre - Cout: 109
-Cas 1 : créer une fonction PGCD
-Cas 2 : créer une fonction Euclide
-Cas 3 : créer une fonction Euclide étendu
-Cas 4 : créer une fonction Inverse modulaire
-Cas 5 : créer une fonction Exponentiation modulaire rapide
-Cas 6 : créer une fonction de Décomposition de nombre
-Cas 7 : créer une fonction Inverse d'une matrice modulaire
-Cas 8 : créer une fonction Test de primalité
-Cas 9 : créer une fonction Valuation p-adique
-Cas 10 : créer une fonction Congruence
-Cas 11 : créer une fonction Chiffrement Hill
-Cas 16 : créer une fonction chiffrement Cesar
-Cas 17 : créer une fonction chiffrement Affine
-Cas 12 : afficher le résultat de toute les fonction en LateX

Nous avons rendue une version beta du projet le 2 novembre pour ce sprint nous avons du augmenter un peu le rythme de travail pour être dans les temps

Sprint 3 - Date Estimer: 3 janvier - Cout : 85
-Cas 13 : inclure la bibliothèque GMP au fonction pour pouvoir manipuler des grand nombres
-Cas 14 : faire des attaques par force brute ou par dictionnaire
-Cas 21 : créer une fonction RSA
-Cas 22 : créer une fonction Substitution
-Cas 23 : Faire Hill avec dimension 3
-Cas 24 : Filtrer les saisies des utilisateurs
-Cas 25 : inclure la possibilité de joindre des fichiers contenant le message à crypter ou décrypter

Rendue et présentation final du projet le mardi 13 janvier

Répartition des taches :

-Fonction PGCD =>Jeremy
-Fonction Euclide =>Jeremy
-Fonction Euclide étendu =>Quentin
-Fonction Inverse modulaire =>Jack
-Fonction Exponentiation modulaire rapide =>Jack
-Fonction Décomposition d'un nombre => Jason
-Fonction Inverse d'une matrice modulaire => Quentin
-Fonction Test de primalité => Jeremy
-Fonction Valuation p-adique => Quentin
-Fonction Congruence => Jack
-Fonction Chiffrement Hill =>Fahath
-Fonction chiffrement Cesar => Jason
-Fonction chiffrement Affine => Quentin
-Fonction RSA =>Jeremy
-Fonction Substitution =>Jack

V – SYNTÈSE DES COMPTES RENDUES DES RENDEZ-VOUS

Les rendez-vous au début étais programmé chaque semaine une fois le projet bien lancer le professeur nous a donné rendez-vous toute les 2 semaines, à chaque rendez-vous un compte rendue étais écrit pour voir les point qui étais à changer au niveau du projet, le compte rendue étais alors après mis en ligne sur GitHub pour que tout les membres du groupe puisse le consulter et voir les modifications à faire pour la prochaine séance.

Premier rendez-vous le 22 janvier:
-Explication du projet

Compte rendue 29 septembre:
-Présentation d'un début de page web et de quelque fonctionnalité
A FAIRE POUR LE PROCHAIN RENDEZ-VOUS:
-Donner un vrai nom aux variables
-Présenter la gestion de projet (diagramme, CDCF…)
-Apprendre LateX
A NOTER:
-Le bouton latex sera supprimé

Compte rendue 6 octobre:
A FAIRE POUR LE PROCHAIN RENDEZ-VOUS:
-Faire l'algorithme d'Euclide étendue
-Faire Euclide
-Faire inverse modulaire (trouver a-1)
-Faire matrice modulaire, pour la saisie de la matrice c'est un champ texte(1 nombre, 1 espace...)
-Faire un algorithme de décomposition (on rentre un entier n, on retourne l'ensemble des diviseurs de l'entier)

-Faire exponentiation modulaire rapide
-Faire l'inverse d'une matrice modulaire
changement de base...
-Faire test de primalité
-Faire un onglet « à propos » où l'on présente les acteurs du projet et le but de ce projet ?
-Ajouter MaitreAtaraxy sur gitHub

Compte rendue 12 octobre:
A FAIRE POUR LE PROCHAIN RENDEZ-VOUS:
-Faire Hill et inverse modulaire et changer la saisie des matrices avec un textarea (sur plusieurs lignes)
-Faire l'algorithme de décomposition (ex: 12 = 2*2*3)
-Déterminer la valuation p-adique
-Faire l'algorithme d'exponentiation modulaire rapide
-Décryptage/Cryptage césar force brut et par dictionnaire et pareil pour affine
-en php, utiliser la librairie GMP qui permet l'utilisation de grand nombres dans toutes les fonctionnalités
-préparer un fichier zip pour le professeur pour pouvoir l'intégrer à Ataraxy

Compte rendue 2 novembre:
A FAIRE POUR LE PROCHAIN RENDEZ-VOUS:
-Le test de primalité de Miller-Rabin
-Mettre un bouton à part pour Eratosthène
-proposer plusieurs résolutions pour les nombres premier
-3 fonctionnalités ,« attaque », « crypter », « décrypter » qui propose le choix entre affine, Cesar et Hill..
-Mettre des exemples de format de saisie
-/times pour changer le signe multiplication
-rajouter module inverse modulaire
-rajouter des titres au fonction quand on est dedans
-retirer lien vers ataraxy
-changer quand le pcgd n'est pas égal à 1 on n'affiche pas u et v
-ordonner par importance dans l'ordre:
    -PGCD
    -Euclide
    -Euclide étendue
    -Congruence
    ...
-LateX pour Décomposition de nombre, on enlève les diviseurs possible (2^2*3) et non 2*2*3 utiliser valuation p-adique
-Deux partis Math/Crypto

Compte rendue 16 novembre:
A FAIRE POUR LE PROCHAIN RENDEZ-VOUS:
-Faire en sorte de filtrer la saisie et de mettre un message d'erreur lorsque la saisie n'est pas correcte
-Commencer RSA
-Commencer Substitution
-Commencer à se renseigner sur Hill en dimension n

Compte rendue 8 décembre:
FAIRE POUR LA PRESENTATION FINAL:
- Faire Hill a dimension n
- Faire RSA
- Faire Substitution
- Faire un rapport comportant un sommaire, une introduction, le description du projet, les solutions mise en oeuvre, les problèmes rencontrés et leurs résolutions, une présentation des algorithmes, une synthèse des comptes rendus des points de rendez vous avec l'enseignant et le client, une conclusion et En annexe : un peu de code, copies d'écran, etc ... Ainsi que un glossaire et les références bibliographique
- Faire un diaporama pour la présentation (mettre peu de texte, privilégier les explication à l'oral)

V1 – EXPLICATION DES ALGORITHMES

La fonction substitution en détail

```
479     function substitution(){
480       $Alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
481       $_alphabet = str_split($Alphabet);
482
483       if(isset($_FILES['messagecode']) and trim($_FILES['messagecode']['tmp_name'])!='')
484         MessageDansFichier();
485
486       if( empty($_POST['message']) or trim($_POST['message'])=='' )
487         exit();
488       else{
489         if(empty($_POST['alphabet']) or trim($_POST['alphabet'])=='' or !preg_match("#^[A-Za-z]{26}$#", $_POST['alphabet']))
490           exit();
491         else
492           $_POST['alphabet'] = strtoupper($_POST['alphabet']);
493       }
494
495       $_POST['message'] = mb_strtoupper($_POST['message'], "utf-8");
```

Après vérification des champs (lignes 483-493), le message qui doit être crypté se trouve dans $_POST['message']. On met ensuite le message non crypté en majuscule grâce à mb_strtoupper($_POST['message'], "utf-8").

```
497         $_POST['message'] = preg_replace("#É|È|Ë|Ê#", "E", $_POST['message']);
498         $_POST['message'] = preg_replace("#Î|Ï#", "I", $_POST['message']);
499         $_POST['message'] = preg_replace("#Ô#", "O", $_POST['message']);
500         $_POST['message'] = preg_replace("#À|Â#", "A", $_POST['message']);
501         $_POST['message'] = preg_replace("#Ù#", "U", $_POST['message']);
502         $_POST['message'] = preg_replace("#Ç#", "C", $_POST['message']);
```

Par la suite, on remplace les lettres accentuées par leur équivalent sans accent.

```
504         $Message = $_POST['message'];
505         $_Message = str_split($Message);
506
507         $_customAlphabet = str_split($_POST['alphabet']);
```

On convertit ensuite le message en tableau dont chaque valeur sera un caractère (lettre) du message. Ce tableau est stocké dans la variable $_Message. On fait de même pour $_POST['alphabet'] qui est l'alphabet utilisé pour crypter le message, et on le stocke dans $_customAlphabet.

```
509         foreach($_Message as $v){
510           for($i=0; $i<26; $i++){
511             if($v == $_alphabet[$i]){
512               $_MessageCrypt[] = $_customAlphabet[$i];
513             }
514           }
515         }
516
517         $MessageCrypt = implode($_MessageCrypt);
518         echo "Votre message crypté: <br>";
519         echo '<p class="message">'.$MessageCrypt.'</p>';
520       }
```

Pour chaque caractère du message non crypté, on cherche sa position $i dans $_alphabet (qui est un tableau rangé alphabétiquement dont chaque valeur est un caractère de l'alphabet). Dans le tableau $_MessageCrypt, on stocke la valeur qui se trouve à la position $i dans le tableau $_customAlphabet. Enfin, on rassemble les valeurs du tableau $_MessageCrypt en une chaîne qui sera stocké le message crypté.

La fonction érathostène en détail avec GMP

```
80    function era($n){
81              $res=array();
82              $indice=0;
83              gmp_init($indice);
84              $i=2;
85              gmp_init($i);
86              $compare=gmp_cmp($i,$n);
87              for($i; $compare<0;$i++){
88                  if((is_primary($i))==true){
89                      $res[$indice]=$i;
90                      $indice++;
91                  }
92                  $compare=gmp_cmp($i,$n);
93              }
94              return $res;
95          }
```

Le crible d'Erathostène a pour but de trouver tous les nombres premiers jusqu'à un entier N (donné par l'utilisateur).

La variable $res est un tableau qui contiendra tous les nombres premiers, $indice est la variable qui permet d'avoir un tableau ordonné au niveau des indices car on ne garde que les nombres premiers. La variable $compare est notre variable de test, tant que $i est inférieur a $n, $compare est positif, si $i égale $n alors $compare est égale à 0 sinon il est négatif.

Pour effectuer notre boucle for on utilise l'indice $i qui est initialisé 2 car 0 et 1 ne sont pas premiers. Ensuite le test effectuer est un test de primalité effectuer par une fonction que nous avons créée aussi.

```php
function is_primary($n){
    for($i=2;$i<$n;$i++){
        if(gmp_mod($n,$i)==0) return false;
    }
    return true;
}
```

Une fonction qui commence à deux et pour les mêmes raisons que précédemment et qui retourne false si un nombre $n est divisible par un autre entier que lui-même.
A la suite de ce test la si la variable est un nombre premier on la stocke dans le tableau $res à l'indice $indice que l'on incrémentera ensuite en la convertissant en entier à cause de quelque problème avec GMP.
On refait le test de comparaison pour que le test soit mis à jour avec la nouvelle valeur de $i.
Enfin on retourne le tableau contenant les nombres premiers que l'on pourra ensuite afficher en LaTeX ou réutiliser pour des fonctions comme RSA.
La complexité de la fonction est $O(n^2)$, en effet on a une boucle for qui va de 2 à n dans une boucle fort qui va aussi de 2 à n.
Nous avions une fonction plus rapide mais avec GMP c'était impossible car cette méthode créait des conflits. La fonction de base était de l'ordre de O(n).

La fonction Hill en détail

```php
if (preg_match('#^([-]?[0-9]*)(\ )([-]?[0-9]*)(\s*)([-]?[0-9]*)(\ )([-]?[0-9]*)#', $_POST['clecode'], $Accod)) {

  $melema   = $Accod[1]; //Matrice element a
   $melemb   = $Accod[3]; //Matrice element b
   $melemc   = $Accod[5]; //Matrice element c
   $melemd   = $Accod[7]; //Matrice element d
   //$alphabet = array();
  $alphabet = str_split($_POST['alphabet']);
   $modulo   = count($alphabet);
   $Gamma    = (($melema * $melemd) - ($melemc * $melemb)); //Calcul de det(A) avec Gamma
   //verifier si cle valide XO
   $mod      = $Gamma % $modulo; //Mod en fonction de lalphabet
```

Pour crypter/décrypter/attaquer avec la méthode de chiffrement de Hill nous avons besoin d'une matrice (*qui sera ici de dimension 2x2*). On récupère chaque élément de la matrice et on l'insère dans une case du tableau (*$Accod*) mais comme pour toute méthode de cryptage, nous avons besoin d'un modulo qu'on calculera en fonction de l'alphabet du message (*$alphabet*). Avec les éléments récupérés dans le tableau on peut calculer le déterminant de la matrice (*$Gamma*).

```php
    if ($mod < 0)
       $mod = $mod + $modulo;

    echo "\$\$ \\Large det(A) = (($melema \\times $melemd)-($melemc \\times $melemb)) <br>\$\$";
    echo "\$\$ \\Large det(A) = $Gamma <br>\$\$";
    echo "\$\$ \\Large det(A) \\equiv_{ $modulo } $mod <br>\$\$";
```

Une fois le calcul fait on récupère l'équivalent modulaire en fonction du modulo calculé précédemment (*$mod*), ces étapes finies on affiche ces résultats grâce au LateX à l'utilisateur pour lui permettre de comprendre les étapes et ainsi avoir une meilleure compréhension de ce qu'il se passe (*affichage de la matrice et le détail du calcul euclidien*).

```
91          if ($Gamma != 0)
92              euclid($modulo, $mod);
93          $pgcd = PGCD($modulo, $mod); // Calcul du PGCD
94
95          echo "\$\$ \\Large PGCD($modulo,$mod) = $pgcd <br>\$\$";
96
97          if ($pgcd == 1) {
98              $invmod = inverseModulaire($mod, $modulo);
99              echo "\$\$ \\Large \\text{Cle valide} <br>\$\$";
100         }
101     }
```

Nous devons maintenant vérifier que la clé est valide pour cela il faut calculer le PGCD entre le déterminant et le modulo *($pgcd*) et avoir un PGCD qui vaut 1, si le PGCD est différent de 1 on affichera un message disant que la clé rentrée n'est pas valide. Avant d'afficher la clé valide si l'utilisateur voulait décrypter le message on devrait récupérer l'inverse modulaire de la matrice grâce à la fonction créer par mon collègue ($*invmod*), tout cela possible si bien sur le PGCD vaut 1, on pourra donc commencer le décryptage une fois la matrice récupérer et le message affiché. La clé prête on peut passer au processus de conversion du message en chiffre qu'on utilisera pour le produit de matrice. En fonction du choix cryptage/décryptage on utilisera la matrice qu'il faudra et par simple produit de la matrice et de l'alphabet codé on obtient le nouveau message chiffré, que l'on convertira en lettre pour pouvoir récupérer le message crypter/décrypter.

La fonction César en détail

```
21      function cesar(){
22      if(isset($_FILES['messagecode']) and trim($_FILES['messagecode']['tmp_name'])!='')
23        MessageDansFichier();
24          if(isset($_POST['alphabet']) and trim($_POST['alphabet'])!='' and isset($_POST['paquet']) and trim($_POST['paquet'])!=''
            and preg_match('#^[0-9]*$#',$_POST['paquet']) and isset($_POST['message']) and trim($_POST['message'])!='' and isset($
            _POST['methode'])){
25            $Amess=RenvoyerMessage();
26        $dico=Dictionnaire();
27        $maxoccurence=0;
28              $alphabet=str_split($_POST['alphabet']);
29              $nbcarac=strlen($_POST['alphabet'])-1;
30              $mod = 0;
31              for($i=0 ; $i<$_POST['paquet'] ; $i++) $mod = gmp_add(gmp_mul(100,$mod),$nbcarac);
32              $mod=gmp_add($mod,1);
```

Le chiffrement de César permet de crypter ou décrypter un message.
Ligne 22-23 : On vérifie si l'utilisateur a joint un fichier à crypter ou décrypter, s'il a joint un fichier, on utilise la fonction MessageDansFichier() qui permet de lire le fichier et d'en retourner son contenu.
Ligne 24 : On vérifie les saisies de l'utilisateur.
Ligne 25 : On utilise la fonction RenvoyerMessage() pour transformer le message sous format code ou alphabet et le mettre dans un tableau avec les codes de chaque caractère de l'alphabet.
Ligne 26 : On créer le dictionnaire qui sera utilisé pour trouver la bonne clé de chiffrement en comptant les occurrences de mots.
Ligne 28-32 : On prépare le modulo en fonction du nombre de caractère dans l'alphabet et le paquet, qui permettra de connaître la dernière clé possible (de 0 à modulo).

```
33                  for($clef = 0 ; $clef<$mod ; $clef++){
34                      $test = true;
35                      $decrypt = "";
36                      foreach($Amess as $x){
37                          $y=gmp_sub($x,$clef);
38                          $y=gmp_mod($y,$mod);
39                          if($y<0) $y=gmp_add($y,$mod);
40
41                          $Y=array();
42                          for($i=0 ; $i<$_POST['paquet'] and $test==true; $i++){
43                              $Y[$i] = gmp_mod($y,100);
44                              $y=gmp_div(gmp_sub($y,$Y[$i]),100);
45
46                              if($Y[$i]>$nbcarac) {
47                                  $test=false;
48                                  break;
49                              }
50                          }
51                          if($test==false) break;
52                          $Y=array_reverse($Y);
53                          foreach($Y as $c => $v){
54                              $Y[$c]=gmp_intval($v);
55                              $decrypt = $decrypt.$_POST['alphabet'][$Y[$c]];
56                          }
57                      }
58                      if($test==false) continue;
59                      $occurence=0;
60                     // $text[$clef]=$decrypt;
```

À partir de la ligne 33, on débute le décryptage par force brut, c'est-à-dire que l'on teste chaque clé de 0 au modulo calculé précédemment. On récupère chaque valeur($x) du tableau obtenu par la fonction RenvoyerMessage() que l'on va soustraire par la clé (qui varie selon le nombre de tours de boucles). Ensuite on calcule l'équivalent modulaire de la valeur récupérée. C'est cette valeur($tmp) que l'on va utiliser pour retrouver le message, si le paquet est supérieur à 1, on doit d'abord décomposer la valeur selon le paquet. On crée une boucle selon le nombre de paquet, puis à chaque passage on prend l'équivalent modulaire modulo 100 que l'on va récupérer dans le tableau ($tab), en suite on fait un décalage de 2 chiffre vers la droite (pour cela on supprime la dizaine et l'unité puis on divise par 100).
Une fois la boucle terminée, le tableau ($tab) contient le code de chaque caractère du message que l'on va retransformer en lettre.

```
61                  foreach($dico as $v){
62                      $occurence=gmp_add($occurence,substr_count(strtolower($decrypt),$v));
63                  }
64                  if($occurence>$maxoccurence){
65                      $clefpossible= $clef;
66                      $decryptpossible=$decrypt;
67                      $maxoccurence=$occurence;
68                  }
69              }
70              echo "<p class='message'><br>Message en César le plus probable est celui de la clé : $clefpossible <br> $
                  decryptpossible</p>";
71          }
72      }
```

Ligne 61-68 : On cherche le maximum d'occurrence de mots du dictionnaire pour trouver la clé de chiffrement la plus probable.

La fonction Euclide Étendue en détail