# Introduction à Pari/GP

Bernadette Perrin-Riou

25 juin 2007

Cette introduction est destinée à prendre en main le logiciel Pari/GP dans le but de l'utiliser ensuite dans le développement d'exercices dans Wims.
Pour cela, nous allons commencer utiliser le logiciel Pari/GP directement avant de l'utiliser avec Wims. Les remarques et conseils spécifiquement destinés à l'utilisation avec Wims se trouvent en encadré.

## Table des matières

# 1 Introduction

## 1.1 Références

PARI/GP est spécialisé dans le calcul arithmétique de haut niveau et dans le calcul numérique. Il est aussi utile pour l'arithmétique polynomiale univariée et l'algèbre linéaire. La documentation est disponible en ligne : `http://pari.math.u-bordeaux.fr/doc.html`. On consultera plus particulièrement :

- La carte de référence : `http://pari.math.u-bordeaux.fr/pub/pari/manuals/refcard.pdf`
- Le manuel de l'utilisateur : `http://pari.math.u-bordeaux.fr/dochtml/html.stable`
- Le tutoriel : `http://pari.math.u-bordeaux.fr/pub/pari/manuals/2.3.1/tutorial.pdf` commence à un niveau élémentaire mais finit à un niveau mathématique substantiel ...

Le calculateur `gp` dispose aussi d'une aide en ligne (voir 2.4).

## 1.2 L'aide mémoire

Jetez un coup d'oeil à l'aide-mémoire (Reference Card). Tapez `gp` dans un terminal et essayez d'utiliser la commande `random` qui se trouve en fin de la deuxième colonne sur la première page. Il est écrit

```
random integer between 0 and N - 1\quad random({N})
```

Vous n'y arrivez pas ? Peut-être n'avez-vous pas lu la première phrase de l'aide-mémoire :

```
optional arguments are surrounded by braces { }
```

Autrement dit, vous pouvez écrire une des lignes suivantes :

```
gp > random()
%1 = 1000288896
gp > random()
%2 = 768462011
gp > random(10)
%3 = 3
gp > ?random
random({N=2^31}): random integer between 0 and N-1.
gp > random(10);
gp >
```

Les accolades signifient que l'argument qu'elles encadrent est optionnel, vous n'avez pas à écrire ces accolades. Dans l'aide obtenue à l'aide de `?random`, `random({N=2^31})` signifie que l'argument `N` est optionnel et que par défaut, c'est-à-dire si vous écrivez `random()` sans argument, il vaudra $2^{31}$, autrement dit que vous obtiendrez un nombre au hasard entre 0 et $2^{31} - 1$. Vous êtes maintenant armé pour lire avec profit la documentation en ligne ainsi que l'aide-mémoire.

# 2 Prise en main

## 2.1 Invite

La chaîne `gp >` (prompt) indique que PARI/GP attend une instruction.

## 2.2 Exécuter

```
gp > a=4
%1 = 4
gp > b=5
%2 = 5
gp > a+b
%3 = 9
gp > a=5 ; b=6 ; a+b
%4 = 11
```

Un retour chariot (`Enter`) déclenche l'exécution du groupe de commande en cours. Ainsi, on peut entrer une instruction par ligne et le résultat est affiché immédiatement. On peut aussi séparer les instructions par `;` et dans ce cas les résultats intermédiaires ne sont pas affichés. Chaque ligne de résultats a un numéro (`%1`, `%2`...) qui permet de le réutiliser ensuite.

> WIMS/PARI/GP : Dans l'interface fournie par WIMS, par exemple
>
> `htp://wims.auto.psud.fr/wims/wims.cgi?module=tool/directexec.fr`,
>
> vous pouvez rentrer une commande par ligne. Le numéro du prompt n'est pas fourni. Il est d'autre part hors de question de réutiliser un numéro de ligne de résultats !

## 2.3 Charger un fichier

On peut lire un fichier au format texte : `read(mon_fichier)` à condition qu'il se trouve dans le répertoire courant ou dans un répertoire lu par PARI/GP (voir `path` dans le manuel). Il est préférable d'écrire tout programme un peu long sous forme de fichier *texte*, puis de le charger dans la session via `read` : le comportement est essentiellement[1] le même que si les commandes avaient été rentrées au clavier.

> WIMS/PARI/GP : Cette possibilité n'est bien sûr pas utilisable si vous incorporez un programme PARI/GP dans un exercice de WIMS. Nous verrons comment faire dans le paragraphe 5.1.

## 2.4 Aide en ligne

`?`*commande* donne une documentation minimale sur *commande*.

`?` donne la liste des thèmes.

`?`*debut_d'un_mot* puis la touche `Tabulation` donne les complétions possibles de ce mot parmi les mots-clés de l'aide en ligne.

`??`*commande* affiche la partie du manuel correspondant à la fonction *commande*, dans une fenêtre `xdvi` ou dans le terminal.

`???`*mot-clé* indique les fonctions ayant un rapport avec le mot-clé ; ce dernier est une expression arbitraire, contenant éventuellement des espaces (auquel cas l'expression doit être mise entre guillemets). Par défaut, la recherche est restreinte au Chapitre 3 du manuel, celui qui décrit les fonctions GP. Si un @ est présent à la fin du mot-clé, au contraire, la recherche est étendue à l'intégralité du manuel.

[1] les résultats intermédiaires ne sont pas affichés

```
gp > ?
Help topics:
 0: list of user-defined identifiers (variable, alias, function)
  1: Standard monadic or dyadic OPERATORS
  2: CONVERSIONS and similar elementary functions
  3: TRANSCENDENTAL functions
  4: NUMBER THEORETICAL functions
  5: Functions related to ELLIPTIC CURVES
  6: Functions related to general NUMBER FIELDS
  7: POLYNOMIALS and power series
  8: Vectors, matrices, LINEAR ALGEBRA and sets
  9: SUMS, products, integrals and similar functions
 10: GRAPHIC functions
 11: PROGRAMMING under GP
 12: The PARI community

Further help (list of relevant functions): ?n (1<=n<=11).
Also:
  ? functionname (short on-line help)
  ?\             (keyboard shortcuts)
  ?.             (member functions)
Extended help looks available:
  ??             (opens the full user's manual in a dvi previewer)
  ??  tutorial   (same with the GP tutorial)
  ??  refcard    (same with the GP reference card)

  ??  keyword    (long help text about "keyword" from the user's manual)
  ??? keyword    (a propos: list of related functions).
```

```
gp > ?factor
factor(x,{lim}): factorization of x. lim is optional and can be set whenever x
is of (possibly recursive) rational type. If lim is set return partial
factorization, using primes up to lim (up to primelimit if lim=0)
```

`{lim}` signifie que la deuxième variable est optionnelle, par défaut 0.

**Exercice 1.** Regarder l'aide en ligne des commandes `divrem`, `gcd`. Calculer le quotient et le reste de la division euclidienne de 126 par 52, factoriser 3467 (calculer le pgcd de deux entiers, de 10 entiers.

## 2.5 Commentaire

\\ (le reste de la ligne est ignoré)
Pour un commentaire de plusieurs lignes `/* blablabla */`

> Wims/Pari/GP : Dans Wims, ne pas mettre de commentaires dans un programme Pari/GP. Utiliser les commentaires Wims en dehors.

## 2.6 Dernière expression évaluée

%. On peut aussi appeler l'expression évaluée à la ligne $n$ par %$n$.

WIMS/PARI/GP : Ne peut pas être utilisé dans WIMS puisque cela utilise l'interactivité du terminal.

## 2.7 Constantes et opérations élémentaires

Une suite de chiffres comportant un `.` (remplace la virgule française) est un réel flottant. Si l'expression contient `I`, c'est un complexe. (Attention à la majuscule : `I`, pas `i`.) Les calculs sont exacts si les données sont exactes (ne contiennent pas de nombre flottant) : par exemple

```
gp > 3/4
%5 = 3/4
gp > 3/4*1.
%6 = 0.7500000000000000000000000000
```

Les calculs exacts peuvent devenir très grands.

```
gp > 10^100
%3 = 1000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
```

WIMS/PARI/GP : Pour conserver un grand entier tel quel dans WIMS, il faut considérer le résultat comme `\text` et non comme `\integer` :

```
\text{a=pari(10^100)}
```

Il y a d'autres constantes classiques : `Pi` pour $\pi = 3.14...$, `Euler` pour la constante d'Euler 0.57721. La lettre `O` est réservée pour “grand O de” (développement en séries ...)
Les opérations arithmétiques élémentaires sont notées `+`, `-`, `*`, `/`, `^` (exponentiation). Les fonctions mathématiques usuelles (exp/log, trigonométrie) s'obtiennent de façon évidente. Cependant, PARI/GP n'est pas un logiciel de calcul formel : `sin(1)` est évaluée numériquement, `sin(x)` donne le développement en série de $\sin x$ en 0.

```
gp > (1+I)*(2+I)
%7 = 1 + 3*I
gp > (1+I)*(2+I)*1.
%8 = 1.000000000000000000000000000 + 3.000000000000000000000000000*I
gp > sin(1)
%3 = 0.8414709848078965066525023216
gp > sin(x)
%4 = x - 1/6*x^3 + 1/120*x^5 - 1/5040*x^7 + 1/362880*x^9 - 1/39916800*x^11 + 1/6227020800*x^1
```

**Exercice 2.** Calculer la puissance quatrième du nombre complexe $2 + 3i$. Calculer sa partie réelle et sa partie imaginaire.

```
b = 2 + 3 * I ; c = b^4
real(c)
imag(c)
```

On peut changer la précision réelle (nombre de chiffres significatifs) :

```
gp > default(realprecision, 50)
   realprecision = 57 significant digits (50 digits displayed)
```

WIMS/PARI/GP : La précision réelle (nombre de chiffres significatifs) de PARI/GP dans WIMS est par défaut de 8. Vous ne pouvez pas la changer dans un exercice OEF. Par contre cela est possible dans un module : dans le fichier `var.proc` d'un module d'exercices OEF ou dans un fichier `var.init` ou `var.proc` d'un module d'exercices, écrire :

```
pari_precision=14
```

### 2.8 Affichage

La commande `print` affiche ses arguments.

```
gp > i=4 ; print("On a i = ", i); print(i^2) ; i
On a i = 4
16
%7 = 4
```

WIMS/PARI/GP : Il vaut mieux renvoyer le résultat sous forme d'une liste ou d'une matrice, plutôt que de détourner la commande `print`.

## 3 Manipulation de matrices

On peut définir une matrice de plusieurs manières.

– On peut la définir explicitement, ligne par ligne ; les coefficients sont séparés par des virgules, les lignes par des points virgules :

```
M = [1,2,3; 4,5,6]
```

– on peut initialiser une matrice nulle, puis la modifier : on doit alors préciser sa taille (ligne, colonne) :

```
M = matrix(3, 2) ; M[1,2] = 1 ; M
```

Il est souvent commode d'utiliser une boucle (voir 4.3).

```
M = matrix(3, 3) ; for (i = 1, 3, M[i,i] = i) ; M
```

– on peut définir la matrice en donnant une expression pour ses coefficients (voir 4.2 pour le test `if`).

```
M = matrix(3,2, i,j, i+j)
M = matrix(3,2, i,j, if (i < j, i+j))
```

Les coefficients non définis sont nuls par défaut. Les opérations élémentaires sur les matrices se font avec les opérateurs usuels. Pour des manipulations plus évoluées, regarder l'aide en ligne `?8`.

**Exercice 3.**
1. Créer dans PARI/GP la matrice de taille $20 \times 20$ dont le $(i, j)$-coefficient est $|i - j|$.
2. Modifier cette matrice pour que les coefficients de la diagonale soient égaux à 1 (utilise un test ou une boucle).
3. Créer la matrice de taille $5 \times 5$ dont le $(i, j)$-coefficient est le polynôme $x^2 - ix + j$. Calculer sa puissance quatrième.

**Solution.**

```
A=matrix(20,20,i,j,abs(i-j))
B=matrix(20,20,i,j,if(i==j,1,abs(i-j)))
C=matrix(20,20,i,j,x^2-i*x+j)
primes(20)
vector(20,i,prime(i))
vector(20,i,factor(prime(i)-1))
```

**Exercice 4.** Calculer le carré des matrices précédentes, le produit des deux premières, le produit de la première par l'inverse de la deuxième.

**Solution.**

```
A^2
B^2
```

## 3.1 Listes (*vector*)

de la forme `[` *sequence* `]` Il s'agit en fait de vecteurs

```
a=[1, 2, 3]
```

Par exemple,

```
vector(5, i, 3*2^(i-1))
```

crée les cinq premiers termes d'une suite géométrique de raison 2 et de premier terme 3.

**Exercice 5.**

1. Créer le vecteur de taille 20 dont le $i$-coefficient est $p_i$ où $p_i$ est le $i$-ième nombre premier.
2. Créer le vecteur de taille 20 dont le $i$-coefficient est la factorisation de $p_i - 1$ où $p_i$ est le $i$-ième nombre premier.
3. Créer le vecteur de taille 100 contenant les termes successifs de la suite $n^2 + n + 1$ (ou de votre suite préférée) et en extraire le 40-ième terme.

**Solution.**

```
primes(20)
vector(20,i,prime(i))
vector(20,i,factor(prime(i)-1))
V = vector(5, i, i^2 + i + 1) ; V[40]
```

Un vecteur a une taille fixée à l'avance. Pour rajouter un élément en dernière position, on utilise `concat` qui permet la concatenation de vecteurs ou de matrices.
On peut méditer les exemples suivants

```
gp > ?concat
concat(x,{y}): concatenation of x and y, which can be scalars, vectors or
matrices, or lists (in this last case, both x and y have to be lists). If y is
omitted, x has to be a list or row vector and its elements are concatenated.
gp > A = [1, 2; 3, 4; 4, 6] ; B = [7; 8; 9] ; concat(A, B)
%1 =
[1 2 7]
[3 4 8]
[4 6 9]
gp > A = [1, 2; 3, 4; 4, 6] ; B = [7, 8] ; concat(A, B)
%2 =
```

```
[1 2]
[3 4]
[4 6]
[7 8]
gp > A = [1, 2; 3, 4] ; B = [7, 8; 9, 10] ; concat(A, B)
%3 =
[1 2 7 8]
[3 4 9 10]
gp > concat([[1, 2],[3, 4]])
%4 = [1, 2, 3, 4]
gp > A = [1, 2; 3, 4] ; B = [7, 8; 9, 10] ; concat(A~, B~)~
%5 =
[1 2]
[3 4]
[7 8]
[9 10]
```

Le deuxième argument est optionnel : dans le quatrième exemple, il y a un seul argument qui est une liste de listes.
Pour concatener deux matrices carrées en lignes (l'une au dessus de l'autre), le dernier exemple utilise la transposée. Une autre possibilité, ici plus compliquée, mais qui peut être utile ailleurs :

```
Mat( concat(Col(A), Col(B)) )
```

qui concatène les deux listes de lignes composant les deux matrices, puis transforme le résultat en matrice.

## 3.2 Listes (*List*)

On peut créer une liste en donnant sa taille maximale et en la remplissant ensuite :

```
L=listcreate(50) ; listput(L, 1); listinsert(L,2,3)
```

Les commandes utiles sont

```
listcreate listinsert listkill listput listsort
```

On peut convertir une liste en vecteur avec la commande `Vec` : `Vec(L)`

**Exercice 6.** 1. Créer une liste.

2. Lui rajouter deux ou trois éléments.
3. La transformer en vecteur.
4. Récupérer un des ses éléments.

Si vous savez déjà faire une boucle, faites la même chose en rajoutant tous les entiers inférieurs à 100 ayant exactement deux facteurs premiers.

**Solution.**

## 3.3 Extraction

Le nombre d'éléments d'un vecteur/liste est donné par `#L`. Si $M$ est une matrice et $V$ un vecteur

`#M` donne le nombre de colonnes de la matrice $M$.

`#V` donne le nombre de coefficients du vecteur $V$.

`matsize(M)` est la taille de la matrice $M$.

`V[i]` extrait le $i$-ème élément du vecteur $V$.

`M[i, j]` extrait le coefficient sur la ligne $i$ et la colonne $j$ de la matrice $M$.

`A = M[, i]` extrait la $i$-ième colonne de la matrice $M$.

`A = M[i,]` extrait la $i$-ième ligne d'e la matrice $M$.

> Wims/Pari/GP : Attention, il s'agit bien d'une virgule, contrairement à Wims pour lequel on doit écrire un point-virgule

De manière plus générale, on peut utiliser `vecextract`.

> Wims/Pari/GP : La manipulation peut être un peu difficile dans Wims si les bornes sont aléatoires. Par exemple, si on désire extraire les colonnes entre 2 et la variable $n$ :
>
> ```
> vecextract(A, Str(2, "..", n));
> ```
>
> pour extraire les colonnes 2 et 4 :
>
> ```
> vecextract(A, [2,4]);
> ```

La plupart des commandes sur les matrices ou sur les vecteurs commencent par `vec` ou `mat` (taper par exemple `?vec` puis utiliser la touche de tabulation) :

```
gp > ?min
min(x, y): minimum of x and y.
gp > ?vecmin
vecmin(x): minimum of the elements of the vector/matrix x.
```

# 4 Programmation

## 4.1 Booléens

Les opérateurs booléens sont `||` pour **ou**, `&&` pour **et**, `!` pour **non**. L'instruction `2 == 3` renvoie la valeur 0 (faux), `3 == 3` renvoie la valeur 1 (vrai). A distinguer bien sûr de `a = 3` qui change la valeur de `a` (et renvoie la nouvelle valeur, ici 3).

```
gp > 2 == 3
%1 = 0
gp > 3 == 3
%2 = 1
gp > a = 3
%3 = 3
gp > 1 = 3
  ***   unexpected character: 1=3
```

## 4.2 Branchements

```
if ( test , instructions1 , instructions2 )
if ( test , instructions1 )
```

Les instructions `instructions1` sont exécutées si `test` est 1 (booléen true). Les instructions `instructions2` sont exécutées si `test` est 0 (booléen false). Notons que, comme toute expression PARI/GP, `if` retourne une valeur, qui est la valeur retournée par la branche finalement évaluée : ainsi

```
if ( a == 1, b = 1 , b = 0)
```

peut s'écrire plus simplement :

```
b = if ( a == 1, 1 , 0)
```

Un test renvoie la valeur 0 ou 1 (autre chose que 0). Ainsi,

```
gp > 1 == 1
%1 = 1

gp > isprime(57)
%2 = 0

a = random(20)+1 ; b = if ( isprime(a), a , 0)
```

**Exercice 7.** Construire la matrice de taille (100, 100) dont le coefficient $(i, j)$ est 1 si $i$ et $j$ sont premiers et 0 sinon

**Solution.**

```
A = matrix(100,100, i,j, isprime(i) && isprime(j))
```

## 4.3 Boucles

```
for ( i = 1, 6, instruction)
```

L'instruction `instruction` est exécutée pour `i` allant de 1 à 6. D'autres variantes de `for` sont implémentées, rappelant que PARI/GP est un logiciel d'arithmétique

```
for fordiv forell forprime forstep forsubgroup forvec
```

Par exemple,

```
gp > ?fordiv
fordiv(n, X, seq): the sequence is evaluated, X running over the divisors of n.

gp > ?forvec
forvec(x=v, seq,{flag=0}): v being a vector of two-component vectors of length n, the
sequence is evaluated with x[i] going from v[i][1] to v[i][2] for i=n,.., 1 if flag is
zero or omitted. If flag = 1 (resp. flag = 2), restrict to increasing (resp. strictly
increasing) sequences.
```

**Exemple.** Compter les nombres premiers de la forme $4m - 1$ qui sont inférieurs à 4000 :

```
s = 0 ; for ( x = 1 , 1000, if(isprime(4*x-1), s++)) ; s
```

ou encore (légèrement plus efficace)

```
s = 0 ; forstep ( x = 3 , 4000, 4, s += isprime(x)) ; s
```

### 4.4 Procédures / fonctions

```
f(n, m) = instruction_dependant_n_et_m ; resultat
```

**Exemple.** Écrire une fonction calculant le représentant de $a \in \mathbb{Z}/N\mathbb{Z}$ entre 0 et $m-1$ :

```
repres(a, m) = a % m
```

**Exemple.** Calculer le représentant de $a^i$ mod $p$ entre 0 et $p-1$.

```
puissance(a, i, p) = lift( Mod(a, p)^i )
```

## 5 Utiliser PARI/GP dans WIMS

### 5.1 Insertion

WIMS peut appeler un certain nombre de logiciels extérieurs dont PARI/GP (on peut citer MAXIMA, OCTAVE, GAP, POVRAY, moins connu `float_calc` (qui est en fait le logiciel `bc`).
En général, cet appel se fait de la manière suivante

– dans un exercice OEF,

```
\text{ a = wims(exec nom_logiciel parametre)}
```

– dans un document,

```
\def{text a = wims(exec nom_logiciel parametre)}
```

– dans un module WIMS

```
a = !exec nom_logiciel parametre
```

Dans ce cas, si `parametre` a plusieurs lignes, chaque ligne doit se terminer par le caractère \, aucun caractère ne doit suivre sur la même ligne, même pas un espace.

Par exemple, pour calculer dans des bases de numération autres que décimales :

```
\text{h = wims(exec float_calc
  obase=5; ibase=12;\N*\M)}
```

Voici un exemple d'utilisation d'OCTAVE

```
\real{ a = randint(1..10)/10 }
\text{ fonct = 2*t*x }
\text{ reponse = wims(exec octave
  function
   xdot = f(x,t)
   xdot = \fonct;
  endfunction;
  lsode("f",1,(t=[0,\a])) }
```

et un autre exemple utilisant une commande de convolution de deux suites que l'on ne trouve pas dans PARI/GP (à ce propos, la plupart des slib de statistiques utilisent les fonctions correspondantes d'OCTAVE) :

```
\def{integer n = randint(2..5)}
\def{integer N = 10 + randint(0..\n)}
\def{integer n_iteration = 10}
\def{matrix P = slib(stat/posdiscretelaw \n,\N,Q)}
\def{text Loi_normale = wims(exec octave
  nb = \n_iteration;
  pn = [\P[1;]];
```

```
qn = pn;
for i = 1:(nb-1)
  qn = conv(pn,qn);
endfor;
disp(qn))}
```

Revenons à Pari/GP car il n'est bien sûr pas question d'apprendre à utiliser tous les logiciels. Il a le droit ainsi que Maxima à un raccourci :

```
\text{ a = pari (....)} //dans un exercice \oef
\def{text a = pari (....)} //dans un document
a = !exec pari ... //dans un module wims
```

```
a = !exec pari M = matrix(5,5) ; \
  M[1,2] = 2 ; M
```

Dans un module, il est conseillé d'écrire dans le fichier `var.init` (cela est vrai pour tous les logiciels utilisés d'ailleurs) :

```
wims_multiexec=pari
```

Cela augmente la rapidité d'exécution et permet de réutiliser des résultats intermédiaires.

### 5.2 Procédures

Dans Wims, mettre la procédure entre parenthèses :

```
\text{a=pari((puissance(a, i, p) = lift( Mod(a, p)^ i )) ;
      puissance(5, 1000, 121) )}
\text{b= pari(puissance(7, 1000, 121))}
```

La procédure `puissance` de Pari/GP est gardée en mémoire par Wims, et peut donc ensuite être utilisée ultérieurement.

## 6 Utilisation concrète

A quoi peut être utile Pari/GP dans Wims ? On ne parle bien sûr pas ici des compétences spécialisées. On peut bien sûr l'utiliser dans les probèmes d'arithmétique, lorsqu'on a besoin de faire des calculs sur des vecteurs ou des matrices, dès qu'il y a un programme à faire, des boucles importantes (Wims limite le nombre de boucles dans les exercices directement).
Quelques exercices vous sont maintenant proposés. Vous pouvez selon les cas les incorporer dans un exercice OEF ou dans un document Wims.

**Exercice 8.** Ecrire un exercice demandant le reste de la division euclidienne de $a^b$ par $n$ avec $a$ et $b$ des entiers grands.

**Solution.**

```
\title{Division euclidienne}
\language{fr}
\range{-5..5}
\email{bpr@math.u-psud.fr}
\computeanswer{no}
\format{html}
\precision{1000}
\integer{a=randint(1237..3000)}
```

```
\integer{b=randint(14000..34567)}
\integer{n=random(3,5,7,11)}
\integer{reponse= pari(lift(Mod(\a,\n)^(\b)))}

\statement{Quel est le reste de la division euclidienne de \(\a^\b) par \n ?
}
\answer{Reste}{\reponse}
```

**Exercice 9.** Ecrire un exercice demandant le point d'intersection de deux droites. On écrira les équations sous la forme $a_i x + b_i y = c_i$ et on commence par tirer au hasard une matrice inversible $\begin{pmatrix} a_1 & b_1 \\ a_2 & b_2 \end{pmatrix}$.

**Solution.**

```
\title{Intersection de deux droites}
\text{A=slib(matrix/invertible 2,10)}
\integer{b1=randint(-10..10)}
\integer{b2=randint(-10..10)}
\text{V = pari(A=[\A] ; B = [\b1;\b2] ; A*[x;y])}
\text{W = pari( (A^(-1)*B)~)}

\statement{
   Les deux droites d'équation
   <center>\(\V[1;] = \b1\) et \(\V[2;] = \b2\)</center>
   se coupent en un point. Donner ses coordonnées.
}
\answer{Coordonnées du point }{\W}{type=vector}
```

**Exercice 10.** Calculer les 10 premiers termes d'une suite géométrique de premier terme 5 et de raison 8. Faire le programme dans un document de Wims en rendant aléatoire le premier terme et la raison.

**Solution.**

```
\title{Suites}
a = 10 ; r = 8 ; vector(10, i, 10*8^i)
```

```
\def{integer a = randint(2..10)}
\def{integer r = randint(2..8)}
\def{text v = pari(vector(10, i, \a*\r^i))}
\v
```

ou encore

```
\def{integer a = randint(2..10)}
\def{integer r = randint(2..8)}
\def{text v = pari( a = \a ; r = \r ; vector(10, i, a*r^i))}
\v
```

Cela aurait pu se faire aussi dans wims de la manière suivante :

```
\def{integer a = randint(2..10)}
\def{integer r = randint(2..8)}
\def{text v = wims(values \a*\r^i for i = 1 to 10)}
\v
```

**Exercice 11.** En utilisant la commande `draw` de WIMS, dessiner en rouge les points $(i, j)$ avec $i$ et $j$ premiers inférieurs à 100.

**Solution.** Plusieurs solutions pour créer la liste des coordonnées des points en rouge :

```
\title{Dessins de nombres premiers}
\def{integer N = 100}
\def{text L = pari(N = \N ; L=listcreate(2*N^2) ;
   for(i = 1, N,
     for(j = 1,N,
        if(isprime(i) && isprime(j), listput(L, i) ; listput(L, j) )
     )
   ) ;
   Vec(L)
  )
}
```

ou

```
\def{integer N = 100}
\def{text L = pari( N = \N ; L=listcreate(2*N^2) ;
       forprime(i = 1, N, forprime(j = 1, N, listput(L, i) ; listput(L, j) ) ) ;
       Vec(L) )
}
```

ou encore

```
\def{integer N = 100}
\def{text L = pari(L = primes(primepi(\N)) ;
       concat( vector(#L, i, vector(#L, L[j], [L[i],L[j]])) )
}
```

Puis le dessin se fait facilement :

```
\draw{500, 500}{
  xrange 0,\N
  yrange 0,\N
  linewidth 5
  points red, \L
}
```

**Exercice 12.** Dans un document, dessiner les points de coordonnées entières positives $(i, j)$ inférieures à 500 telles que $i^2 + j^2$ est premier. Introduisez un entier $d$ et faire de même en demandant que $i^2 + dj^2$ soit premier.

**Solution.** Dessin premiers quadratiques

```
\def{integer N = 100}
\def{integer d = randitem(1, 2, 3, 5, 7)}
\def{text L = pari(N = \N ; d = \d ;  L=listcreate(2*N^2) ;
       for(i = 1, N,
            for(j = 1, N, if(isprime(i^2 + d* j^2), listput(L, i) ; listput(L, j)))) ;
```

```
       Vec(L) )
}
\reload{d} = \d ;
\draw{500, 500}{
  xrange 0,\N
  yrange 0,\N
  linewidth 5
  points red, \L
}
```

**Exercice 13.** Dans un document, dessiner le graphe des polynômes caractéristiques des matrices

$$\begin{pmatrix} -6 & 28 & 21 \\ 4 & -15 & -12 \\ -8 & a & 25 \end{pmatrix}$$

pour les valeurs de $a$ suivantes : 31.8, 31.9, 32, 32.1, 32.2 . On utilisera des couleurs différentes pour chacune des valeurs.

**Solution.**

```
\title{Dessin polynôme caractéristiques}
\def{matrix A = -6,28,21
      4,-15,-12
     -8,a,25}
\def{text liste = 31.8,31.9,32,32.1,32.2}
\def{text P = pari(Q = charpoly( [\A], x) ; L=[\liste] ;
       vector(#L, i, subst(Q,a,L[i]))
       )
}
\def{text color=blue,purple,red,orange,yellow}
\def{text dessin = xrange -1,4
      yrange -10,10
      hline 0,0,black
      vline 0,0, black
 }
\def{integer cnt = items(\liste)}
\for{i = 1 to \cnt}{
    \def{text dessin = \dessin
        plot \color[\i], \P[\i]}
}
Voici le graphe des polynômes caractéristiques des matrices \([\A]\)
pour les valeurs suivantes de \(a\) : \liste.
    <p align="center"> \draw{300,300}{\dessin} </p>
    Que remarquez-vous ?
```

**Exercice 14.** Vous voulez utiliser le type de réponse `range`, vous manipulez de temps en temps de très grands nombres, de temps en temps de très petits. Vous devez donc gérer l'erreur relative. Dans le type `range`, une fois que la réponse a été calculée (avec 8 chiffres significatifs), on vous demande de donner la valeur minimale acceptée, la valeur maximale acceptée et la valeur qui sera affichée. Pourquoi ne pas faire un petit programme PARI/GP qui calcule tout cela ? Vous devez donc vous donner un entier *tol* tel que l'erreur relative soit $10^{-tol}$. Faites le programme dans un document.

```
\def{text P = random(10^(-10).. 3*10^(-10))}
\def{real tol=2}
\def{integer app=\tol+1}
\def{text approx = pari((f(r,n)=
   l=if(r != 0, floor(log(abs(r))/log(10)),1) ;
   s=round(10^(-l+n)*r) ; [l,s]))}
\def{text P_approx=pari(f(\P,\app))}
\def{real P_app=\P_approx[2]*10^(-\app+\P_approx[1])}
\def{text Prange=pari(\P*[(1-1/10^\tol),(1+1/10^\tol)]),\P_app}
```

**Exercice 15.** Trouver le $n$-ième terme de la suite de Syracuse définie de la manière suivante : le premier terme $u_0$ est un entier positif ; si $u_n$ est pair, $u_{n+1} = u_n/2$ ; si $u_n$ est impair, $u_{n+1} = 3u_n+1$. Vérifier sur des entiers pris au hasard que $u_n$ finit toujours par valoir 1 et qu'on a ensuite une boucle 1, 4, 2, 1, 4, 2, ... On programmera une fonction ayant comme arguments le premier terme et $n$ et on mettra les valeurs intermédiaires dans un vecteur. En utilisant la commande `draw` de Wims, tracer les points $(n, u_n)$.

**Solution.** Le programme de Syracuse peut se faire de la manière suivante :

```
\tytle{Suite de Syracuse}
syracuse(u,n) = local(v) ; v = u ; V=vector(n) ;
    for( i = 1 , n, v = if(v %2==0, v/2, 3*v+1) ; V[i] = v ); V
```

ou bien

```
syracuse(u,n) = local(v) ; v = u ; vector(n, i, v = if(v %2==0, v/2, 3*v+1))
```

Adaptons maintenant.

```
\def{text u = randint(20..30)}
\def{integer n = randint(5..10)*10}
\def{text V = pari((syracuse(u,n) = local(v) ; v = u ;
                 vector(n, i, v = if(v %2==0, v/2, 3*v+1))
                ) ;
                n = \n ; u = \u ;
                V = syracuse(u,n) ; m = vecmax(V) ; concat( vector(n,i, [i,V[i]])) )
   }
\def{text m = pari(m)}
Les \n premiers termes de la suite de Syracuse de premier terme \u (\reload{Renouveler})
<center>
\draw{600,200}{
  xrange -0.5,\n
  yrange -0.5,\m
  hline 0,0,black
  vline 0,0,black
  linewidth 3
  points blue, \V}
 </center>
```

**Exercice 16.** Trouver les coordonnées dans $\mathbb{R}^3$ du barycentre de trois points.

**Solution.** On se donne les trois points

```
\title{Barycentre}
\def{text P = randint(1..5), randint(1..5), randint(1..5)}
\def{text Q = -randint(1..5), randint(1..5), randint(1..5)}
```

```
\def{text R = randint(1..5), -randint(1..5), randint(1..5)}
\def{text coeff= randint(1..5), -randint(1..5), randint(1..5)}
\def{text bary= pari(coeff=[\coeff] ; P=[\P] ; Q=[\Q] ; R = [\R] ;
   coeff[1] * P + coeff[1] * Q + coeff[1] * R)
}
```

**Exercice 17.** En utilisant le type **javacurve**, écrire un exercice demandant de cliquer sur le symétrique d'une figure représentée par 4 ou cinq points par rapport à une droite.

$$s_D(M) = M - 2\frac{\langle \overrightarrow{AM}, \vec{v} \rangle}{\langle \vec{v}, \vec{v} \rangle} \vec{v}.$$