Le développement du système PARI/GP (1983–2013)

Karim Belabas

http://pari.math.u-bordeaux.fr/

Institut de Mathématiques de Bordeaux

PARI/GP ?

PARI/GP est un logiciel libre spécialisé en Théorie des Nombres, né en 1983 à Bordeaux. Le système est construit sur trois composantes :

- PARI, le cœur du système : bibliothèque C, permettant des calculs rapides.
- gp : interpréteur, donnant accès aux routines de PARI via le langage GP, plus simple à utiliser.
- gp2c : compilateur GP $\longrightarrow$ C ; compile et charge dans une session gp un objet externe issu de (script GP $\leftrightarrow$ code C $\leftrightarrow$ bibliothèque partagée).

21 auteurs principaux : les 4 auteurs d'origine BaBeCoOl puis The PARI Group à partir de 1997 ; actuellement 4 développeurs. La communauté des utilisateurs est importante[a], directement ou via Sage, probablement de l'ordre de 20.000 utilisateurs.

[a] Depuis le monde Unix, on récupère PARI via Git ou des distributions (Debian/Ubuntu, Fedora, SuSE, MacPorts, etc.) ; la version Windows est la seule sur laquelle nous ayons des statistiques de téléchargements : entre 10000 et 20000 par mois dont un pourcentage inconnu de robots.

Caractéristiques

Le système est essentiellement écrit en C : 93.975 lignes de code au sens de `gcov` (dont environ 15.000 lignes pour `gp2c`/ le parseur / l'interpréteur, et les entrées-sorties) ; 193.914 lignes en comptant en-têtes et commentaires (17/11/2013, 09:13).

- Libre : distribué sous licence GPL v2+ (depuis 11/2000).
- Portable : tous OS standards (Unix, Windows, MacOS, Android) et toutes architectures modernes.
- Compact : pendant 20 ans, la distribution complète a tenu sur une disquette 1,4Mo ; 3,5Mo à ce jour pour les sources compressées du système.
- Rapide : PARI est fréquemment inclus dans des benchmarks, et est rarement ridicule.
- Pas d'interface graphique native, mais possibilité d'utilisation à travers des frontaux comme le notebook de Sage, `TeXmacs` ou `PariDroid`.
- Immense majorité des utilisateurs : GP, ou à travers Sage (+ historiquement : Magma, MuPAD), ou WIMS.

Fonctionnalités

- Ce que PARI sait `bien` faire : calculs sur les entiers et flottants en précision arbitraire ; fonctions transcendantes ; corps finis ; polynômes et série formelles *univariés* ; algèbre linéaire sur $\mathbb{Z}$, $\mathbb{Q}$ ou un corps fini ; groupes abéliens ; réseaux (automorphismes, réduction) ; courbes elliptiques ; `corps de nombres` (sa principale force), théorie de Galois sur $\mathbb{Q}$.
- Ce que PARI sait faire : primalité / factorisation sur $\mathbb{Z}$, log discret sur $\mathbb{F}_q^*$; analyse numérique (calcul intégral, sommation de séries, algèbre linéaire) ; (petits) groupes de permutations ; graphismes simples.
- Ce que PARI ne sait `pas` faire : objets creux ou structurés, polynômes multivariés, systèmes polynomiaux, corps de fonctions. Groupes non abéliens. Arithmétique d'intervalle ou arrondi exact. Et bien d'autres choses encore. . .

Quelques regrets

Des choses que nous ferions différemment si on devait reprendre à 0 ou en s'autorisant à casser (lourdement) la compatibilité :

- GP est un langage très pratique mais ad hoc, inadapté pour des développements de grande envergure. Nous souhaitons le nettoyer et l'étendre pour les utilisateurs historiques. Utiliser PARI via Sage (donc Python) est une alternative de plus en plus populaire.
- Typage dynamique, trop faible, sans support abstrait pour des morphismes entre catégories d'objets. Pratique pour des objets simples (création intuitive, transtypages implicites). Rend le travail difficile pour des objets compliqués.
- Modèle d'arithmétique flottante exotique (forme faible d'arithmétique d'intervalle), difficile à contrôler pour l'utilisateur. Pas gênant pour une approche «expérimentale», mais demande croissante pour des garanties sur les résultats (preuve de théorèmes. . .). Facile à corriger localement, mais travail de Sysiphe s'il faut reprendre des centaines de fonctions.
- Nécessité de sizeof(long) == sizeof(void*).

Exemple GP (1/3) : Euclide

```
GCD(a,b) = {
   while(b, [a,b] = [b, a%b]);
   return (a);
}

/* [a,u,v] = GCDEXT(10,17); */
GCDEXT(a,b) = {
   my(u = 1, v = 0);
   while(b,
      my([q,r] = divrem(a,b));
      [a, b] = [b, r];
      [u, v] = [v, u-q*v];
   );
   return ([a,u,v]);
}
```

Exemple GP (2/3) : déterminant sur $\mathbb{Z}$ par restes Chinois

Il faut calculer les $\det M \pmod p$ pour $p < x$, tel que

$$\prod_{p<x} p > 2B, \quad \text{où } B = \texttt{Hadamard}(M) := \prod_i \|M_i\|_2.$$

Théorème (Rosser-Schoenefeld, version faible)**.**

$$\sum_{p<x} \ln p > 0.84x, \quad \textit{pour } x > 100$$

```
Hadamard(M) = sqrt( prod(i=1,#M, norml2(M[,i])) );
```

Exemple GP (2/3) : déterminant modulaire, suite

```
detZ(M) = {
   my (v, B = 2*Hadamard(M), b = max(100, log(B) / 0.84));
   v = [ matdet(M * Mod(1,p)) | p <- primes([2, b]) ];
   centerlift( chinese(v) );
}

detZ2(M) = {
   my (p, q = 1.0, B = 2*Hadamard(M), v = List());
   forprime(p = 2, /* +∞ ! */,
      listput(v, matdet(M * Mod(1,p)));
      q *= p; if (q > B, break);
   );
   centerlift( chinese(v) );
}
```

Exemple GP (3/3), partie sans facteurs carrés sur $\mathbb{F}_q[X]$

Si $T \in \mathbb{F}_q[X]$ de degré n se factorise comme $T = \prod_i T_i^{e_i}$ et

$$u = \gcd(T, T'), \quad v = T/u, \quad w = u/\gcd(u, v^n)$$

alors

$$v = \prod_{i\,:\,p \nmid e_i} T_i, \quad w = \prod_{i\,:\,p \mid e_i} T_i^{e_i} = W(X^p)$$

```
splitp(Fq, T) = {
   my(p = Fq.p, q = p^Fq.f, n = poldegree(T));
   my(u,v,w,W, X = variable(T));
   u = gcd(T,T'); v = T/u; w = u / gcd(u, lift(Mod(v,u)^n));
   W = apply(a->a^(q/p), substpol(w, X^p, X));
   return ([v, W]);
}
```

Exemple GP (3/3), partie sans facteurs carrés, suite

```
CORE(Fq, T) = {
   my(v = 1, W = T, v2);
   until (poldegree(W) == 0, [v2,W] = splitp(Fq, W); v *= v2);
   v / pollead(v);
}
Fq = ffgen(5^7, 't);
T = random(Fq * x^10) * random(Fq*x^10)^5;
CORE(Fq, T)
```

Exemple PARI, Euclide

```
GEN GCDEXT(GEN A, GEN B) {
   pari_sp av = avma;
   GEN ux = gen_1, vx = gen_0, a = A, b = B;

   if (typ(a) != t_INT) pari_err_TYPE("GCDEXT", a);
   if (typ(b) != t_INT) pari_err_TYPE("GCDEXT", b);
   if (signe(a) < 0) { a = negi(a); ux = negi(ux); }
   while (!gequal0(b)) {
      GEN r, q = dvmdii(a, b, &r), v = vx;
      vx = subii(ux, mulii(q, vx));
      ux = v; a = b; b = r;
   }
   vx = diviiexact( subii(a, mulii(A,ux)), B );
   return gerepilecopy(av, mkvec2(ux, vx));
}
```

Pourquoi PARI ?

« Mon sentiment profond est que la théorie des nombres est une véritable science de la nature au même titre que, disons, la physique. (. . .) Avec l'avènement de l'ère informatique la théorie des nombres a reçu un outil qui permet à la dialectique entre théorie et expérimentation d'être beaucoup plus efficace. Une utilisation typique peut être la suivante : 1) Obtenir suffisamment de données expérimentales sur lesquelles on puisse faire des conjectures, 2) Vérification de ces conjectures par des passages extensifs en machine, 3) Parfois aussi aide aux démonstrations elles-mêmes, soit en complétant une démonstration, soit en guidant la démonstration.

On voit que de plus en plus l'ordinateur joue en théorie des nombres un rôle analogue à celui des petites ou grosses machines utilisées par les physiciens expérimentaux. Toutefois il faut faire un constat de pauvreté : (. . .) les arithméticiens (et les mathématiciens purs en général) n'ont que de faibles moyens. Typiquement quelques micro-ordinateurs (. . .) »

Henri Cohen, « Théorie des Nombres et ordinateurs ».

(conférence à l'ENS, octobre 1983)

Projet PARI/GP, v0.1

- «Tout d'abord, il ne s'agit pas d'écrire un système de calcul formel, mais seulement la partie dont on a besoin en théorie des nombres (et qui inclut des choses que les systèmes de calcul formel classiques ne possèdent pas)
- Il y aura une grosse bibliothèque scientifique, écrite principalement en C ou/et en langage assembleur 68000. Ceci aura pour conséquence 1) Une très grande rapidité, 2) Portabilité car le langage C (ou dans une moindre mesure le 68000) est très répandu, (. . .)
- Enfin et surtout il y aura un langage spécifique, pour utiliser la puissance de la bibliothèque, et devant satisfaire à deux critères 1) Il doit être compilé, bien qu'une version interprétée soit également désirable, la communication homme-machine étant importante dans la phase expérimentale (. . .), 2) Il doit être très proche de, et même si possible contenir, un langage déjà existant pour que son apprentissage soit facile.

  Comme candidats je vois essentiellement `Pascal`, `Ada`, `C` et `Modula 2`. (. . .) Malgré tous ses défauts bien connus, j'ai choisi `Pascal`.»

Henri Cohen (conférence à l'ENS, octobre 1983)

Un peu d'histoire (1/3)

1975–1979 : projet ISABELLE (Cohen, Dress), une calculatrice multi-précision programmée sur TI980, limitée à 5894 décimales, optimisée pour 35.

```
L00010: D9=K
L00020: D0=0
L00030: D1=D2=1
L00040: D0=D0+(MOBD1)*D9/D2
L00050: GTL40*PZD9-D2=D1*D1=D1+1
L00060: K=(INVD9^.4)*K=(K=D0)-D9*6:PI*PI
L00070: GTL_1
```

Ce programme calcule

$$R(x) = \left(\sum_{a \leqslant \sqrt{x}} \mu(a) \left\lfloor x/a^2 \right\rfloor - \frac{6}{\pi^2} x \right) x^{-2/5} = O(1) \text{ sous GRH.}$$

var=K : lecture sur la console ; K=*var* affichage d'un résultat. x est contenu dans D9, la somme courante dans D0, a dans D1, a^2 dans D2. Temps pour $x = 10^7$: 85 secondes.

```
? R(x)=(sum(a=1, sqrt(x), moebius(a)*(x\a^2)) - 6*x/Pi^2) / x^0.4;
? R(10^7)
time = 4ms.
%1 = 0.031668479507398169142959440501894135146
? R(10^12)
time = 657 ms.
%2 = 0.006656129372465159570908819294924688950 1
```

Un peu d'histoire (2/3)

1983–1984. Cahier des charges du projet `PARI-68k` (Cohen–Dress).

1985–1986. Écriture d'un noyau multi-précision en assembleur 68000 (Batut, Cohen, Olivier). Programmation de types récursifs et de fonctions génériques, d'abord en Pascal, puis en C.

1986. Écriture d'un premier interpréteur `gc` (Batut)

1987. Écriture de `gcp` (Bernardi), renommé `gp`. Portage (ré-écriture) du noyau pour le 68020 (Batut, Cohen, Olivier : 6000 lignes !).

1990. Première version largement diffusée (1.31, publication via annonce sur `sci.math`).

1993. Portage sur DEC alpha (64 bit).

1990–1997. Développement des modules de théorie algébrique des nombres. Compétition amicale avec Kant/Kash ; intégration de PARI dans Magma, dans MuPAD, dans Maxima.

1997–1998. Grand nettoyage : refactorisation et disparition de la plupart des hacks historiques, des variables globales et des constantes codées en dur. Automatisation de la configuration.

1999. Système de gestion de versions public (CVS). Migrations ultérieures à subversion (2008), puis Git (2011).

Un peu d'histoire (3/3)

2000. Changement de licence (propriétaire $\longrightarrow$ GPL). Écriture du compilateur `gp2c` (Allombert)

2002. Nouveau noyau multi-précision entier, basé sur `GMP` (Allombert).

2003–2004. Un site Web pour le projet `http://pari.math.u-bordeaux.fr`. Intégration continue, Bug Tracking System. Premier Atelier PARI (IHP).

2004 Compétition amicale avec `MPFR`. Intégration systématique d'algorithmes sous-quadratiques «simples».

2007. Remplacement du parseur/évaluateur par un compilateur (à la volée) et un évaluateur de bytecode (Allombert).

2012. Les Ateliers PARI deviennent récurrents (périodicité annuelle).

Projets en cours

- Parallélisme : interface unifiée pour threads POSIX (une machine multi-cœur) + MPI (clusters). Parallélisme externe (nouvelles fonctions GP `parfor`, `parvector`, ...) et interne (`bnfinit`, ECM).
- Remplacement du noyau multi-précision flottant (précision au bit près, unification avec le code entier)
- Racines réelles d'un polynôme de $\mathbb{Q}[X]$ (Grenié).
- Groupes de classes/unités : Crible, large prime variations, représentations compactes (K.B. / Grenié).
- Algèbres de quaternions, algèbres centrales simples (Page).
- Symboles et formes modulaires pour $\Gamma_0(N)$ (K.B. / Cohen).
- Fonctions L complexes (Molin)
- Courbes elliptiques / K, corps de nombres : 2-descente, recherche de points (Simon)
- Algorithmes de Sutherland, polynômes modulaires via restes chinois (Ivey-Law).

Une page de publicité

Il est encore temps de s'inscrire pour le

4ème Atelier PARI.

Laboratoire de Mathématiques de Besançon

Université de Franche–Comté

06 – 10 Janvier 2014

Contact : christophe.delaunay@math.cnrs.fr