Factorisation par fractions continues

Margot Funk, Antoine Hugounet

Février 2021

Table des matières

1 Explication du programme

1.1 La stucture générale du programme

Notre programme comprend deux étapes principales. La première consiste à générer, à partir du développement en fractions continues de $\sqrt{kN}$, des paires (A, Q) avec Q friable pour une base de factorisation préalablement déterminée. On associe à chaque Q ainsi produit un vecteur exposant `mpz_t exp_vect`. Ce vecteur permet de retenir les nombres premiers qui interviennent dans la factorisation de Q avec une valuation impaire. Dans le but d'augmenter le nombre de paires (A, Q) acceptées lors de cette étape, nous avons implémenté la "large prime variation". Celle-ci permet d'accepter une paire si Q se factorise grâce aux premiers de la base de factorisation et à un grand facteur premier supplémentaire. Les fonctions de cette phase de collecte sont rassemblées dans les fichiers `step_1.h` et `step_1.c`. Elles font appel, pour mettre en oeuvre la "large prime variation", aux fonctions des fichiers `lp_var.h` et `lp_var.c`.

Ces données sont traitées lors de la seconde phase dans l'espoir de trouver un facteur non trivial de N. Il s'agit de trouver des ensembles valides de paires (A, Q) par pivot de Gauss sur la matrice dont les lignes sont formées des vecteurs exposants. Chaque

ensemble valide est à l'origine d'une congruence $A^2 \equiv Q^2 \pmod{N}$ permettant potentiellement de trouver un facteur non trivial de N. Les fonctions de cette phase sont regroupées dans les fichiers `step_2.h` et `step_2.c`.

Avant d'effectuer la première étape, il convient de se doter d'une base de factorisation. Ceci est permis par une fonction de fonction **trouver nom** qui gèrent plus généralement le choix par défaut des paramètres.

explique `fact.c` `fact.h`
proposition : renommer stepA stepB stepC fact.h et changer les fonctions de fact.h et stepA.h

2 Entrées et sorties du programme

Nous avons regroupé dans une structures `Params` les paramètres d'entrée de la fonction de factorisation, à savoir :

- `N` : le nombre à factoriser, supposé produit de deux grands nombres premiers.
- `k` : le coefficient multiplicateur.
- `n_lim` : le nombre maximal de paires (A, Q) que l'on s'autorise à calculer. Ce nombre prend en compte toutes les paires produites et non uniquement les paires avec Q friable ou accepté par la "large prime variation"
- `s_fb` : la taille de la base de factorisation.
- `nb_want_AQp` : le nombre désiré de paires (A, Q) avec Q friable ou accepté par la "large prime variation"
- des booléens indiquant si la "early abort strategy" ou la "large prime variation" doivent être utilisées et des paramètres s'y rapportant.

Le programme stocke dans une structure `Results` le facteur non trivial de `N` trouvé (si tel est le cas) ainsi que des données permettant l'analyse des performances de la méthode.

Remarque 2.1. Nous avons décidé de ne pas implémenter un programme réalisant la factorisation complète de N. En effet, l'efficatité de la méthode de factorisation dépend du choix de certains paramètres : la taille de la base de factorisation et le coefficient `k`. Nous avons préféré les considérer comme paramètres d'entrée du programme plutôt que comme paramètres devant être déterminés par une sous-routine en fonction du nombre à factoriser pou avoir plus de latitude dans les tests.

Remarque 2.2. Notre programme n'est pas supposé prendre en entrée un nombre admettant un petit facteur premier (inférieur aux premiers de la base de factorisation par exemple). En effet, il ne test pas au préalable si N est divisible par des petits facteurs, et mettra donc autant de temps à trouver un petit facteur qu'un grand facteur.