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 le fichier `step_1.c`. Elles font appel, pour mettre en oeuvre la « large prime variation », aux fonctions du fichier `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 de la forme $A^2 \equiv Q^2 \pmod{N}$ permettant potentiellement de trouver un facteur non trivial de N. Les fonctions de cette phase sont regroupées dans le fichier `step_2.c`.

Avant d'effectuer la première étape, il convient de se doter d'une base de factorisation. Ceci est permis par une des fonctions de `init_algo.c`. Ces dernières se chargent plus généralement de l'initialisation et du choix par défaut des paramètres.

Finalement, en mettant bout à bout les deux étapes, la fonction `contfract_factor` du fichier `fact.c` recherche un facteur non trivial de N et `print_results` affiche les résultats.

1.2 Entrées et sorties du programme

Nous avons regroupé dans une structure `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` un facteur non trivial de `N` trouvé (si tel est le cas) ainsi que des données permettant l'analyse des performances de la méthode est de

Remarque 1.1. L'efficatité de la méthode de factorisation dépend du choix des paramètres ci-dessus. Pour avoir plus de latitude dans les tests, nous les considérons comme des paramètres d'entrée du programme. C'est pourquoi notre programme ne s'attèle pas à la factorisation complète d'un entier, qui aurait nécessité une sous-routine déterminant des paramètres optimaux.

Remarque 1.2. 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, comme il ne teste pas au préalable si `N` est divisible par de petits facteurs, il mettra autant de temps à trouver un petit facteur qu'un grand facteur.

1.3 Collecte des paires (A, Q) : `step_1.c` et `lp_var.c`

Décrivons tout d'abord la phase de collecte des données. Elles sont stockées au fur et à mesure de la collecte dans les tableaux `mpz_t *Ans`, `mpz_t *Qns`, `mpz_t *exp_vects` et `mpz_t *hist_vects`. A un indice correspond un paire (A, Q) donnée.

1.3.1 La fonction `create_AQ_pairs`

Sachant que seules les paires (A, Q) dont on a pu factoriser Q_n nous intéressent pour la seconde phase, nous avons décidé de ne stocker que celles-ci. Ce choix a en outre un avantage : étant donné un nombre `nb_want_AQp` représentant le nombre voulu de telles paires, il est possible d'arrêter le développement en fraction continue dès que ce nombre est atteint. Cela évite d'avoir à stocker toutes les paires (A, Q), pour ensuite sélectionner celles qui nous intéressent, en courant le risque d'en avoir trop ou pas assez.

Ce choix amène à avoir une grande fonction, en l'occurence `create_AQ_pairs`, qui au fur à mesure du développement de $\sqrt{kN}$ en fraction continue, teste si le Q_n qui vient d'être calculé est factorisable. Si c'est le cas, on crée son vecteur exposant et ajoute les données de la paire aux tableaux `Ans`, `Qns` et `exp_vects`. Pour faire cela, la fonction utilise les sous-routines `is_Qn_factorisable` et `init_exp_vect`.

1.3.2 La « early abort strategy »

La fonction `is_Qn_factorisable` teste si un Q_n est friable par divisions successives avec les premiers de la base de factorisation. Un moyen d'améliorer les performances de la méthode est de décider de ne pas poursuivre les divisions successives si après un nombre `eas_cut` de divisions la partie non factorisée de Q_n est trop grande (supérieure à une borne `eas_bound_div` proportionnelle à la borne déjà connue $\sqrt{kN}$).

1.3.3 La « large prime variation »

Etant donnée une base de factorisation $B = \{p_1, \cdots, p_m\}$, la « large prime variation » consiste à accepter lors de la collecte, non seulement des Q_n B-friables mais aussi des Q_n produits d'un entier B-friable et d'un entier lp_n inférieur à p_m^2 (et donc premier), que l'on appelera *grand premier (large prime)*. De tels Q_n deviennent intéressants s'ils partagent grand premier lp en commun.

En effet, si on on connait $Q_{n_1} = X_{n_1} lp$ et $Q_{n_2} = X_{n_2} lp$ avec X_{n_1} et X_{n_2} B-friables, alors