Documentation pour `zqadic.h`

Guillemot Alexandre, Soumier Julien

20 février 2023

Table des matières

1 Introduction

Soit p un nombre premier et $q = p^d \in \mathbb{Z}$. Ce module permet de faire des calculs sur $\mathbb{Z}_q$, en représentant l'extension comme un quotient de $\mathbb{Z}_p[X]$ par un polynôme $M \in \mathbb{F}_p[X]$ irréductible.

2 Structure de données

Un élément de $\mathbb{Z}_q$ est la classe d'équivalence d'un élément de $\mathbb{Z}_p[X]$ modulo M. On le représente donc par un élément de $\mathbb{Z}_p[X]$, que l'on réduira modulo M tout au long des

calculs. On dira qu'il est sous forme réduite s'il est réduit modulo M

Type représentant un élément de $\mathbb{Z}_q$.

```
typedef padic_poly_t zqadic_t;
```

Fonction renvoyant la précision à laquelle `x` est représenté.

```
slong zqadic_prec(zqadic_t x);
```

Fonction renvoyant la valuation de `x`.

```
slong zqadic_val(zqadic_t x);
```

3 Contexte

Un contexte d'entiers q-adiques contient les informations nécessaires aux calculs dans $\mathbb{Z}_q$, ainsi que différents éléments précalculés permettant d'accélérer certains calculs.

Différents types de polynômes pouvant représenter l'extension $\mathbb{Z}_q$ de $\mathbb{Z}_p$.

```
enum rep_type {TEICHMULLER, SPARSE};
```

Type représentant un contexte d'entiers q-adiques.

```
typedef struct _zqadic_ctx_t
{
    slong prec; // Précision maximale des calculs dans l'extension, dans le
    ↪  cas où le représentant est calculé à une précision donnée (e.g.
    ↪  module de Teichmuller)
    slong deg; // Degré de l'extension
    enum rep_type type; // Type de représentant
    fmpz_t p; // Nombre premier tel que $p^{deg} = q$
    zqadic_t* C;// Pointeur vers un tableau contenant les éléments $C_j \in
    ↪  Z_q$. Reste null si le type n'est pas TEICHMULLER
    padic_ctx_t pctx; // Contexte $p$-adique associé au sous-corps de
    ↪  l'extension
    padic_poly_t M; // Polynôme représentant de l'extension
} zqadic_ctx_t[1];
```

Procédure permettant d'initialiser un contexte `zqadic_ctx` à partir d'un polynôme `M` $\in \mathbb{Z}_p[X]$ (supposé irréductible), dans un contexte `padic_ctx`. La précision maximale de l'extension sera donnée par la précision de `M` si `type == TEICHMULLER`.

```
void zqadic_ctx_init_padic_poly(zqadic_ctx_t zqadic_ctx, padic_poly_t M,
↪  padic_ctx_t padic_ctx, enum rep_type type);
```

Procédure calculant le module de Teichmuller de `m` $\in \mathbb{F}_p[X]$, vu comme un polynôme de $\mathbb{Z}_p[X]$, à précision la précision de `M`. Le résultat est mis dans `M`. Ne marche qu'avec $p = 2$ (dans le contexte)

```
void zqadic_teichmuller_modulus(padic_poly_t M, padic_poly_t m, padic_ctx_t
↪  C);
```

Procédure permettant d'initialiser un contexte `zqadic_ctx`, avec comme représentant le module de Teichmuller de `m` $\in \mathbb{F}_p[X]$ vu comme un polynôme de $\mathbb{Z}[X]$. Les informations `min`, `max` et `mode` permettent d'initialiser le contexte p-adique dans lequel seront représentés les coefficients des polynômes représentant les éléments de $\mathbb{Z}_q$ (voir padic.h). Ne fonctionne qu'avec $p = 2$

```
void _zqadic_ctx_init_teichmuller(zqadic_ctx_t zqadic_ctx, fmpz_poly_t m,
↪  slong prec, slong min, slong max, enum padic_print_mode mode);
```

Procédure permettant d'initialiser un contexte `zqadic_ctx`, avec comme un représentant le module de Teichmuller d'un polynôme alétoire pris dans $\mathbb{F}_p[X]$. Les informations `min`, `max` et `mode` permettent d'initialiser le contexte p-adique dans lequel seront représentés les coefficients des polynômes représentant les éléments de $\mathbb{Z}_q$ (voir padic.h). Ne fonctionne qu'avec $p = 2$

```
void zqadic_ctx_init_teichmuller(zqadic_ctx_t zqadic_ctx, slong deg, slong
↪  prec, slong min, slong max, enum padic_print_mode mode);
```

Procédure permettant d'initialiser un contexte `zqadic_ctx`, avec comme représentant le relèvememnt creux de `m` $\in \mathbb{F}_p[X]$ vu comme un polynôme de $\mathbb{Z}[X]$. Les informations `min`, `max` et `mode` permettent d'initialiser le contexte p-adique dans lequel seront représentés les coefficients des polynômes représentant les éléments de $\mathbb{Z}_q$ (voir padic.h).

```
void _zqadic_ctx_init(zqadic_ctx_t zqadic_ctx, fmpz_poly_t m, fmpz_t p,
↪  slong prec, slong min, slong max, enum padic_print_mode mode);
```

Procédure permettant d'initialiser un contexte `zqadic_ctx`, avec comme un représentant le relèvement creux d'un polynôme alétoire pris dans $\mathbb{F}_p[X]$. Les informations `min`, `max` et `mode` permettent d'initialiser le contexte p-adique dans lequel seront représentés les coefficients des polynômes représentant les éléments de $\mathbb{Z}_q$ (voir padic.h).

```
void zqadic_ctx_init(zqadic_ctx_t zqadic_ctx, fmpz_t p, slong deg, slong
↪  prec, slong min, slong max, enum padic_print_mode mode);
```

Procédure permettant de récupérer le représentant d'un contexte d'entiers q-adiques `zqadic_ctx_t`. Met le résultat dans `P`.

```
void zqadic_ctx_rep(padic_poly_t P, zqadic_ctx_t ctx);
```

Procédure permettant de changer la précision maximale d'un contexte.

```
void zqadic_ctx_change_prec(zqadic_ctx_t ctx, slong prec);
```

4 Gestion de la mémoire

Permet d'initialiser la mémoire nécessaire pour un `x` $\in \mathbb{Z}_q$. La précision par défaut est donnée par la précision du contexte `zqadic_ctx`.

```
void zqadic_init(zqadic_t x, zqadic_ctx_t ctx);
```

Permet d'initialiser la mémoire nécessaire pour un `x` $\in \mathbb{Z}_q$, à précision `prec`.

```
void zqadic_init2(zqadic_t x, slong prec, zqadic_ctx_t ctx);
```

Permet de libérer la mémoire allouée pour `x`.

```
void zqadic_clear(zqadic_t x);
```

Permet de libérer la mémoire allouée pour `ctx` un contexte d'entiers q-adiques.

```
void zqadic_ctx_clear(zqadic_ctx_t ctx);
```

5 Assignement

Met la valeur de `op` dans `rop`.

```
void zqadic_set(zqadic_t rop, zqadic_t op, zqadic_ctx_t zqadic_ctx);
```

Met la valeur de `op` $\in \mathbb{Z}_p$, vu comme un polynôme constant dans $\mathbb{Z}_p[X]$, dans `rop`.

```
void zqadic_set_padic(zqadic_t rop, padic_t op, zqadic_ctx_t ctx);
```

Met dans `rop` le représentant réduit modulo le polynôme représentant $\mathbb{Z}_q$ de `op`.

```
void zqadic_set_padic_poly(zqadic_t rop, padic_poly_t op, zqadic_ctx_t ctx);
```

Met dans `rop` le représentant réduit modulo le polynôme représentant $\mathbb{Z}_q$ de l'inclusion canonique de `op` $\in \mathbb{Z}[X]$ dans $\mathbb{Z}_p[X]$.

```
void zqadic_set_fmpz_poly(zqadic_t rop, fmpz_poly_t op, zqadic_ctx_t ctx);
```

Met dans `rop` le relèvement canonique de `op` $\in \mathbb{Z}_q$, vu comme un élément de $\mathbb{Z}_p[X]$ à précision donnée, donc un élément de $(\mathbb{Z}/p^{prec}\mathbb{Z})[X]$.

```
void zqadic_get_fmpz_poly(fmpz_poly_t rop, zqadic_t op, zqadic_ctx_t ctx);
```

Met 1 dans `rop`.

```
void zqadic_one(zqadic_t rop);
```

Met 0 dans `rop`.

```
void zqadic_zero(zqadic_t rop);
```

6 Randomisation

Génère un élément de $\mathbb{Z}_q$ aléatoire. Met le résultat dans `x`.

```
void zqadic_randtest(zqadic_t x, flint_rand_t state, zqadic_ctx_t ctx);
```

7 Comparaison

Renvoie `1` si et seulement si `x` = `y`. Suppose que `x` et `y` sont réduits.

```
int zqadic_equal(zqadic_t x, zqadic_t y);
```

Renvoie `1` si et seulement si `x` = 0. Suppose que `x` et `y` sont réduits.

```
int zqadic_is_zero(zqadic_t x);
```

Renvoie `1` si et seulement si `x` = 1. Suppose que `x` et `y` sont réduits.

```
int zqadic_is_one(zqadic_t x);
```

8 Opérations arithmétiques

Réalise la division euclidienne de `A` par `B` dans $\mathbb{Z}_p[X]$. Suppose que `B` est unitaire.

```
void padic_poly_eucl_div(padic_poly_t R, padic_poly_t Q, padic_poly_t A,
↪  padic_poly_t B, padic_ctx_t C);
```

Met sous forme réduite `x` $\in \mathbb{Z}_q$.

```
void zqadic_reduce(zqadic_t x, zqadic_ctx_t C);
```

Additionne `op1` et `op2`. Met le résultat dans rop.

```
void zqadic_add(zqadic_t rop, zqadic_t op1, zqadic_t op2, zqadic_ctx_t ctx);
```

Réalise la soustration de `op1` avec `op2`. Met le résultat dans `rop`.

```
void zqadic_sub(zqadic_t rop, zqadic_t op1, zqadic_t op2, zqadic_ctx_t ctx);
```

Met l'opposé de `op` dans `rop`.

```
void zqadic_neg(zqadic_t rop, zqadic_t op, zqadic_ctx_t ctx);
```

Réalise la multiplication de `op1` avec `op2`. Met le résultat dans `rop`.

```
void zqadic_mul(zqadic_t rop, zqadic_t op1, zqadic_t op2, zqadic_ctx_t ctx);
```

Inverse `op`, en supposant qu'il est inversible. Met le résultat dans `rop`.

```
void zqadic_inv(zqadic_t rop, zqadic_t op, zqadic_ctx_t ctx);
```

Met `op` à la puissance `e` dans `rop`.

```
void zqadic_pow(zqadic_t rop, zqadic_t op, fmpz_t e, zqadic_ctx_t ctx);
```

Calcule la composition (en tant que polynômes) de `op1` avec `op2`. Met le résultat dans `rop`. Utilise l'astuce de Paterson-Stockmeyer.

```
void zqadic_padic_poly_evaluation(zqadic_t rop, padic_poly_t op1, zqadic_t
↪  op2, zqadic_ctx_t ctx);
```

9 Fonctions spéciales

Réalise la substitution du frobenius en `op`, dans l'extension spécifiée par `ctx`. Met le résultat dans `rop`.

```
void zqadic_frobenius_substitution(zqadic_t rop, zqadic_t op, zqadic_ctx_t
↪  ctx);
```

Réalise la substitution du frobenius inverse en `op`, dans l'extension spécifiée par `ctx`. Met le résulta dans `rop`.

```
void zqadic_inv_frobenius_substitution(zqadic_t rop, zqadic_t op,
↪  zqadic_ctx_t ctx);
```

Résout l'équation d'Artin-Schreier avec paramètres `alpha`, `beta` et `gamma`. Met le résultat dans `x`.

```
void zqadic_artin_schreier_root(zqadic_t x, zqadic_t alpha, zqadic_t beta,
↪  zqadic_t gamma, zqadic_ctx_t ctx);
```

10 Misc

Affiche un `x` de $\mathbb{Z}_q$, représenté comme un élément de $\mathbb{Z}_p[X]$. Les coefficients de ce polynôme (dans $\mathbb{Z}_p$) seront affichés selon le mode spécifié dans le contexte p-adique associé à `ctx` (`ctx -> ctxp`).

```
void zqadic_print(zqadic_t x, zqadic_ctx_t ctx);
```