Q Quadratic: GP guide

(version 0.1)

A PARI/GP package for integral binary quadratic forms (and coming soon: quaternion algebras) over $\mathbb{Q}$, with an emphasis on indefinite quadratic forms and indefinite quaternion algebras.

James Rickards
Department of Mathematics and Statistics,
McGill University, Montreal
Personal homepage
Github repository

© James Rickards 2020

Contents

1 Introduction

The roots for this library came from my thesis project, which involved studying intersection numbers of geodesics on modular and Shimura curves. To be able to do explicit computations, I wrote many GP scripts to deal with indefinite binary quadratic forms, and indefinite quaternion algebras. This library is a revised version of those scripts, rewritten in PARI ([The20]) for optimal efficiency.

While there already exist some PARI/GP methods to compute with quadratic forms and quaternion algebras (either installed or available online), I believe that this is the most comprehensive set of methods yet.

The package has been designed to be easily usable with GP, with more specific and powerful methods available to PARI users. More specifically, the GP functions are all given wrappers so as to not break, and the PARI methods often allow passing in of precomputed data like the discriminant, the reduced orbit of an indefinite quadratic form, etc.

Note that the current version (0.1) only includes algorithms for quadratic forms; the quaternionic algorithms will be in the next update, which will hopefully be ready by October 2020.

1.1 Overview of the main available methods

For integral binary quadratic forms, there are methods available to:

- Generate lists of (fundamental, coprime to a given integer n) discriminants;
- Compute the basic properties, e.g. the automorph, discriminant, reduction, and equivalence of forms;
- For indefinite forms, compute all reduced forms, the Conway river, left and right neighbours of river/reduced forms;
- Compute the narrow class group and a set of generators, as well as a reduced form for each equvalence class in the group;
- Output all integral solutions (x, y) to $Ax^2 + Bxy + Cy^2 + Dx + Ey + F = n$ for any integers A, B, C, D, E, F, n;
- Solve the simultaneous equations $Ax^2+By^2+Cz^2+Dxy+Exz+Fyz = n_1$ and $Ux+Vy+Wz = n_2$ for any integers $A, B, C, D, E, F, U, V, W, n_1, n_2$.

1.2 Upcoming methods

The next project is to implement methods relating to quaternion algebras over $\mathbb{Q}$. Planned methods include:

- Initialize the algebra given the ramification, and initialize maximal/Eichler orders (with specific care given to algebras ramified at <= 2 finite places);
- Compute the fundamental domain of unit groups of Eichler orders in indefinite algebras (Shimura curves);
- Solve the principal ideal problem in indefinite quaternion algebras;
- Compute all optimal embeddings of a quadratic order into a quaternion algebra, and arrange them with respect to the class group action and their orientation.

I will also be adding methods to compute intersections of indefinite binary quadratic forms and closed geodesics on Shimura curves, as this was one of the goals of my thesis project.

1.3 How to use the library

As a first word of warning, this library is only guaranteed to work on Linux. The essential files (.so) were created with GP2C, and they are not usable with Windows (I don't think it works on Mac, but I don't know). However, the workaround for Windows is to install the Linux Subsystem for Windows, and install PARI/GP there (in fact, this is my current setup, and it works well).

The files required are **libqquadratic.so**, and **qquadratic.gp**. Move them to the same folder, open up GP, and type "\r qquadratic" to install the methods!

I would love to be able to make this work cross-platform, but at the moment I don't know how to do that and it's not a prioity. If you do know how to do this, please let me know!

1.4 Validation of methods

I have made an effort to systematically check that the methods in this libary have been programmed correctly. This involved testing the methods with random data, and checking that basic properties are obeyed/the methods are consistent with other library methods/a less efficient but simpler algorithm produces the same results. Of course this isn't "proof" that I have no errors lurking in obscure parts of the algorithms, but it does provide good support. If you do happen to find a bug, then please let me know!

1.5 How to use this manual

Sections 2-3 contain detailed descriptions of every function: the input, output, and what the function does. The sections are labeled by source files, and are divided into subsections of "similar" methods. If you are seeking a function for a certain task, have a look through here.

Section 4 contains simply the method declarations, and is useful as a quick reference. Clicking the name of a method in this section will take you to its full description in Sections 2-3, and clicking on the name there will take you back to Section 4.

In each method, optional arguments are given inside curly braces, and the default value is given (for example, `{flag=1}` means `flag` is optional and is defaulted to `1`).

2 c_base

This is a collection of "basic" functions and structures, which are useful in various places. The main interesting method here is "sqmod", which allows you to compute square roots modulo any integer n, and not just primes (which is already built into PARI/GP).

2.1 Euclidean geometry

These methods will likely be moved the geometry package, when I write that (the geometry package will support finding the fundamental domain for a discrete subgroup of $\mathrm{PSL}(2,\mathbb{R})$).

Name:	`crossratio`
Input:	`a, b, c, d`
Input format:	`a`, `b`, `c`, `d` complex numbers or infinity, with at most one being infinity
Output format:	Complex number or $\pm$`oo`
Description:	Returns the crossratio `[a,b;c,d]`.

Name:	mat_eval
Input:	M, x
Input format:	M a 2x2 matrix and x a complex number or infinity
Output format:	Complex number or ±oo
Description:	Returns M acting on x via Mobius transformation.

2.2 Infinity

In dealing with the completed complex upper half plane, the projective line over $\mathbb{Q}$, etc., we would like to work with ∞, but currently PARI/GP does not support adding/dividing infinities by finite numbers. The functions here are wrappers around addition and division to allow for this.

Name:	addoo
Input:	a, b
Input format:	a, b complex numbers or infinity
Output format:	Complex number or ±oo
Description:	Returns a+b, where the output is a if a is infinite, b if b is infinite, and a+b otherwise.

Name:	divoo
Input:	a, b
Input format:	a, b complex numbers or infinity
Output format:	Complex number or ±oo
Description:	Returns a/b, where a/0 will return $\pm\infty$ (depending on the sign of a), and $\pm\infty$/b will return $\pm\infty$ (depending on the sign of b). Note that both 0/0 and ∞/∞ return ∞.

2.3 Linear equations and matrices

lin_intsolve is essentially just gcdext, but it outputs to a format that is useful to me.

Name:	lin_intsolve
Input:	A, B, n
Input format:	Integers A, B, C
Output format:	0 or $[[m_x, m_y], [x_0, y_0]]$.
Description:	Solves $Ax+By = n$ using gbezout, where the general solution is $x = x_0+m_xt$ and $y = y_0 + m_xt$ for $t \in \mathbb{Z}$. If there are no solutions or A=B=0, returns 0.

Name:	mat3_complete
Input:	A, B, C
Input format:	Integers A, B, C with $\gcd(A, B, C) = 1$
Output format:	Matrix
Description:	Returns a 3x3 integer matrix with determinant 1 and first row A, B, C.

2.4 Square roots modulo n

In PARI/GP you can take square roots modulo p^e very easily, but there is not support for a general modulus n, and if the number you are square rooting is not a square, an error will occur. `sqmod` is designed to solve this problem, and uses the built in methods of `Zp_sqrt` and `chinese` to build the general solution.

Name:	`sqmod`
Input:	`x, n`
Input format:	`x` a rational number with denominator coprime to `n`, a positive integer
Output format:	0 or `v=[S, m]`.
Description:	Returns the full solution set to $y^2 \equiv x \pmod{n}$, where the solution set is described as $y \equiv s_i \pmod{m}$ for any $s_i \in S$.

2.5 Time

Name:	`printtime`
Input:	-
Input format:	-
Output format:	-
Description:	Prints the current time.

3 c_bqf

These methods primarily deal with primitive integral homogeneous positive definite/indefinite binary quadratic forms. Such a form $AX^2 + BXY + CY^2$ is represented by the vector $[A, B, C]$. Some of the basic methods support non-primitive, negative definite, or square discriminant forms (like `bqf_disc` or `bqf_trans`), but more complex ones (like `bqf_isequiv`) may not.

On the other hand, the method `bqf_reps` allows non-primitive forms, as well as negative definite and square discriminant forms. Going further, `bqf_bigreps` allows non-homogeneous binary quadratic forms (but the integral requirement is never dropped).

In this and subsequent sections, a **BQF** is an integral binary quadratic form, an **IBQF** is an indefinite BQF, a **DBQF** is a positive definite BQF, a **PIBQF/PDBQF** is a primitive indefinite/positive definite BQF respectively, and a **PBQF** is either a PIBQF or a PDBQF.

3.1 Discriminant methods

These methods deal with discriminant operations that do not involve quadratic forms.

Name:	`disclist`
Input:	`D1, D2, {fund=0}, {cop=0}`
Input format:	Integers `D1`, `D2`, `fund=0, 1`, `cop` an integer
Output format:	Vector
Description:	Returns the set of discriminants (non-square integers equivalent to `0, 1` modulo 4) between `D1` and `D2` inclusive. If `fund=1`, only returns fundamental discriminants, and if `cop`$\neq$`0`, only returns discriminants coprime to `cop`.

Name:	discprimeindex
Input:	D
Input format:	Discriminant D
Output format:	Vector
Description:	Returns the set of primes p for which D/p^2 is a discriminant.

Name:	fdisc
Input:	D
Input format:	Discriminant D
Output format:	Integer
Description:	Returns the fundamental discriminant associated to D.

Name:	isdisc
Input:	D
Input format:	-
Output format:	0 or 1
Description:	Returns 1 if D is a discriminant and 0 else.

Name:	pell
Input:	D
Input format:	Positive discriminant D
Output format:	[T, U]
Description:	Returns the smallest solution in the positive integers to Pell's equation $T^2 - DU^2 = 4$.

Name:	posreg
Input:	D
Input format:	Positive discriminant D
Output format:	Real number
Description:	Returns the positive regulator of $\mathcal{O}_D$, i.e. the logarithm of the fundamental unit of norm 1 in the unique order of discriminant D.

Name:	quadroot
Input:	D
Input format:	Discriminant D
Output format:	t_QUAD
Description:	Outputs the t_QUAD w for which $w^2 = D$.

3.2 Basic methods for binary quadratic forms

Recall that the BQF $AX^2 + BXY + CY^2$ is represented as the vector $[A, B, C]$.

Name:	`bqf_automorph`
Input:	`q`
Input format:	PBQF q
Output format:	Matrix
Description:	Returns the invariant automorph `M` of `q`, i.e. the $\mathrm{PSL}(2,\mathbb{Z})$ matrix with positive trace that generates the stabilizer of `q` (a cyclic group of order 1, 2, 3, or ∞).

Name:	`bqf_disc`
Input:	`q`
Input format:	BQF `q`
Output format:	Integer
Description:	Returns the discriminant of `q`, i.e. $B^2 - 4AC$ where `q=[A, B, C]`.

Name:	`bqf_isequiv`
Input:	`q1, q2, {tmat=0}`
Input format:	`q1` a PBQF, `q2` a PBQF or a set of PBQFs, `tmat=0, 1`
Output format:	Integer or matrix or `[i, M]`
Description:	Tests if `q` is equivalent to `q2` or a BQF in `q2` (when `q2` is a set). If `q2` is a BQF, returns 1 if equivalent and 0 if not, unless `tmat=1` where we return a transition matrix taking `q1` to `q2`. If `q2` is a set of BQFs, if `tmat=0` returns an index `i` for which `q1` is equivalent to `q2[i]`, and 0 if no such index exists. If `tmat=1`, instead returns `[i, M]` where `M` is the transition matrix taking `q1` to `q2[i]`.

Name:	`bqf_isreduced`
Input:	`q`
Input format:	`q` a PBQF
Output format:	`0, 1`
Description:	Returns `1` if `q` is reduced, and `0` is `q` is not reduced. We use the standard reduced definition when $D < 0$, and the conditions $AC < 0$ and $B > \lvert A+C\rvert$ when $D > 0$.

Name:	`bqf_random`
Input:	`maxc, {type=0}, {primitive=1}`
Input format:	`maxc` a positive integer, `type, primitive=0, 1`
Output format:	BQF
Description:	Returns a random BQF of non-square discriminant with coefficient size at most `maxc`. If `type=-1` it will be positive definite, `type=1` indefinite, and `type=0` either type. If `primitive=1` the form will be primitive, otherwise it need not be.

Name:	bqf_random_D
Input:	maxc, D
Input format:	maxc a positive integer, D a discriminant
Output format:	BQF
Description:	Returns a random primitive BQF of discriminant D (positive definite if $D < 0$).

Name:	bqf_red
Input:	q, {tmat=0}
Input format:	q a PBQF, tmat=0,1
Output format:	BQF or [q', M]
Description:	Returns the reduction of q. If tmat=0 this is a BQF, otherwise this is [q', M] where the reduction is q' and the transition matrix is M.

Name:	bqf_roots
Input:	q
Input format:	BQF q
Output format:	[r1, r2]
Description:	Returns the roots of q(x,1)=0, with the first root coming first. If D is not a square, these are of type t_QUAD, and otherwise they will be rational or infinite. If D=0, the roots are equal.

Name:	bqf_trans
Input:	q, M
Input format:	BQF q, $M \in \mathrm{SL}(2, \mathbb{Z})$
Output format:	BQF
Description:	Returns $M \circ q$

Name:	bqf_trans_coprime
Input:	q, n
Input format:	BQF q, non-zero integer n
Output format:	BQF
Description:	Returns a BQF equivalent to q whose first coefficient is coprime to n.

Name:	ideal_tobqf
Input:	numf, ideal
Input format:	numf a quadratic number field, ideal an ideal in numf
Output format:	BQF
Description:	Converts the ideal to a BQF and returns it.

3.3 Basic methods for indefinite quadratic forms

Methods in this section are specific to indefinite forms. The "river" is the river of the Conway topograph; it is a periodic ordering of the forms $[A, B, C] \sim q$ with $AC < 0$. Reduced forms with $A > 0$ occur between branches pointing down and up (as we flow along the river), and reduced forms with $A < 0$ occur between branches pointing up and down.

Name:	`ibqf_isrecip`
Input:	`q`
Input format:	IBQF `q`
Output format:	`0, 1`
Description:	Returns `1` if `q` is reciprocal (`q` is similar to `-q`), and `0` else.

Name:	`ibqf_leftnbr`
Input:	`q, {tmat=0}`
Input format:	IBQF `q=[A, B, C]` with $AC < 0$, `tmat=0, 1`
Output format:	IBQF or `[q', M]`
Description:	Returns the left neighbour of `q`, i.e. the nearest reduced form on the river to the left of `q`. If `tmat=0` only returns the IBQF, and if `tmat=1` returns the form and transition matrix.

Name:	`ibqf_redorbit`
Input:	`q, {tmat=0}, {posonly=0}`
Input format:	IBQF `q`, `tmat, posonly=0, 1`
Output format:	Vector
Description:	Returns the reduced orbit of `q`. If `tmat=1` each entry is the pair `[q', M]` of form and transition matrix, otherwise each entry is just the form. If `posonly=1`, we only take the reduced forms with positive first coefficient (half of the total), otherwise we take all reduced forms.

Name:	`ibqf_rightnbr`
Input:	`q, {tmat=0}`
Input format:	IBQF `q=[A, B, C]` with $AC < 0$, `tmat=0, 1`
Output format:	IBQF or `[q', M]`
Description:	Returns the right neighbour of `q`, i.e. the nearest reduced form on the river to the right of `q`. If `tmat=0` only returns the IBQF, and if `tmat=1` returns the form and transition matrix.

Name:	`ibqf_river`
Input:	`q`
Input format:	IBQF `q`
Output format:	Vector
Description:	Returns the river sequence associated to `q`. The entry 1 indicates going right, and 0 indicates going left along the river.

Name:	`ibqf_riverforms`
Input:	`q`
Input format:	IBQF `q`
Output format:	Vector
Description:	Returns the forms on the river of `q` in the order they appear, where we only take the forms with first coefficient positive.

Name:	`ibqf_symmetricarc`
Input:	`q`
Input format:	IBQF `q`
Output format:	$[z, \gamma_q(z)]$
Description:	If γ_q is the invariant automorph of q, this computes the complex number `z`, where `z` is on the root geodesic of `q` and $z, \gamma_q(z)$ are symmetric (they have the same imaginary part). This gives a "nice" upper half plane realization of the image of the root geodesic of `q` on $\mathrm{PSL}(2,\mathbb{Z})\backslash\mathbb{H}$ (a closed geodesic). However, if the automorph of `q` is somewhat large, z and $\gamma_q(z)$ will be very close to the $x-$axis, and this method isn't very useful.

Name:	`mat_toibqf`
Input:	`M`
Input format:	$M \in \mathrm{SL}(2,\mathbb{Z})$
Output format:	PBQF
Description:	Returns the PBQF corresponding to the equation `M(x)=x`. Typically used when `M` has determinant `1` and is hyperbolic, so that the output is a PIBQF (this method is inverse to `bqf_automorph` in this case).

3.4 Class group and composition of forms

This section deals with class group related computations. To compute the class group we take the built-in PARI methods, which cover the cases when D is fundamental and when the narrow and full class group coincide. For the remaining cases, we "boost up" the full class group to the narrow class group with `bqf_ncgp_nonfundnarrow`.

Name:	`bqf_comp`
Input:	`q1, q2, {tored=1}`
Input format:	PBQFs `q1, q2` of the same discriminant, `tored=0, 1`
Output format:	PBQF
Description:	Returns the composition of `q1` and `q2`, where we reduce it if `tored=1`.

Name:	`bqf_ncgp`
Input:	`D`
Input format:	Discriminant `D`
Output format:	`[n, orders, forms]`
Description:	Computes and returns the narrow class group associated to `D`. `n` is the order of the group, `orders=[d1, d2, ..., dk]` where $d_1 \mid d_2 \mid \cdots \mid d_k$ and the group is isomorphic to $\prod_{i=1}^{k} \frac{\mathbb{Z}}{d_i\mathbb{Z}}$, and `forms` is the length `k` vector of PBQFs corresponding to the decomposition (so `forms[i]` has order `di`).

Name:	`bqf_ncgp_lexic`
Input:	`D`
Input format:	Discriminant `D`
Output format:	`[n, orders, forms]`
Description:	Computes and returns the narrow class group associated to `D`. The output is the same as `bqf_ncgp`, except the third output is now a lexicographical listing of representatives of all equivalence classes of forms of discriminant `D`: starting with the identity element, and the component with the highest order moves first.

Name:	`bqf_pow`
Input:	`q, n, {tored=1}`
Input format:	PBQF `q`, integer `n`, `tored=0, 1`
Output format:	PBQF
Description:	Returns a form equivalent to q^n, reduced if `tored=1`.

Name:	`bqf_square`
Input:	`q, {tored=1}`
Input format:	PBQF `q`, `tored=0, 1`
Output format:	PBQF
Description:	Returns q^2, reduced if `tored=1`.

3.5 Representation of integers by forms - description tables

This section deals with questions of representing integers by quadratic forms. The three main problems we solve are

- Find all integral solutions (X, Y) to $AX^2 + BXY + CY^2 = n$ (`bqf_reps`);
- Find all integral solutions (X, Y) to $AX^2 + BXY + CY^2 + DX + EY = n$ (`bqf_bigreps`);
- Find all integral solutions (X, Y, Z) to $AX^2 + BY^2 + CZ^2 + DXY + EXZ + FYZ = n_1$ and $UX + VY + WZ = n_2$ (`bqf_linearsolve`).

The general solution descriptions have a lot of cases, so we put the descriptions in Tables 1-3, and refer to the tables in the method descriptions.

For `bqf_reps`, let $q = [A, B, C]$ and let $d = B^2 - 4AC$. If there are no solutions the method will return 0, and otherwise it will return a vector `v`, where

$$v = [[\text{type}, v_{extra}], v_1, v_2, \dots, v_k].$$

The types are are

-1=all, 0=finite, 1=positive, 2=linear.

Each (family of) solution(s) is given by a v_i, possibly with reference to the extra data. In this table we will only describe **half** of all solutions: we are only taking one of (X, Y) and $(-X, -Y)$. If you want all solutions without this restriction, you just have to add in these negatives.

Table 1: General solution for `bqf_reps`

Type	Conditions to appear	v_{extra}	v_i format	General solution
-1	$q = 0$, $n = 0$	-	-	X, Y are any integers
0	$d < 0$	-	$[x_i, y_i]$	$X = x_i$ and $Y = y_i$
	$d = \square > 0$,[a] $n \neq 0$			
	$d = \boxtimes$,[a] $n = 0$			
1	$d = \boxtimes > 0$, $n \neq 0$	M [b]	$[x_i, y_i]$	$\binom{X}{Y} = M^j \binom{x_i}{y_i}$ for $j \in \mathbb{Z}$
2	$d = 0$, $n \neq 0$	-	$[[s_i, t_i], [x_i, y_i]]$	$X = x_i + s_iU$, $Y = y_i + t_iU$ for $U \in \mathbb{Z}$
	$d = \square > 0$, $n = 0$			

[a] $\square$ means square, and $\boxtimes$ means non-square.
[b] $M \in \mathrm{SL}(2, \mathbb{Z})$

For `bqf_bigreps`, let $q = [A, B, C, D, E]$ and let $d = B^2 - 4AC$. If there are no solutions the method will return 0, and otherwise it will return a vector `v`, where

$$v = [[\text{type}, v_{extra}], v_1, v_2, \dots, v_k].$$

The types are are

-2=quadratic, -1=all, 0=finite, 1=positive, 2=linear.

Each (family of) solution(s) is given by a v_i, possibly with reference to the extra data.

Table 2: General solution for `bqf_bigreps`

Type	Conditions to appear	v_{extra}	v_i format	General solution
-2	$d = 0$ and condition [a]	-	$[[a_i, b_i, c_i],$ $[e_i, f_i, g_i]]$	$X = a_iU^2 + b_iU + c_i$ and $Y = e_iU^2 + f_i + g_i$ for $U \in \mathbb{Z}$
-1	$q = 0$, $n = 0$	-	-	X, Y are any integers
0	$d < 0$	-	$[x_i, y_i]$	$X = x_i$ and $Y = y_i$
	$d = \square > 0$,[b] some cases[c]			
1	$d = \boxtimes > 0$, $n \neq 0$	$M, [s_1, s_2]$ [d]	$[x_i, y_i]$ [d]	$\left(\begin{smallmatrix}X\\Y\end{smallmatrix}\right) = M^j \left(\begin{smallmatrix}x_i\\y_i\end{smallmatrix}\right) + \left(\begin{smallmatrix}s_1\\s_2\end{smallmatrix}\right)$ for $j \in \mathbb{Z}$
2	$d = \square > 0$,[b] some cases[c]	-	$[[s_i, t_i], [x_i, y_i]]$	$x = x_i + s_iU$, $y = y_i + t_iU$ for $U \in \mathbb{Z}$
	$d = 0$, and condition [e]			

[a] At least one of $A, B, C \neq 0$ and at least one of $D, E \neq 0$.
[b] $\square$ means square, and $\boxtimes$ means non-square.
[c] "Some cases" refers to if the translated equation has $n = 0$ or not.
[d] $M \in \mathrm{SL}(2, \mathbb{Z})$ and s_1, s_2 are *rational*; they need not be integral. Same for x_i, y_i.
[e] $A = B = C = 0$ or $D = E = 0$. In this case, $s_i = s_j$ and $t_i = t_j$ for all i, j in fact.

For `bqf_linearsolve`, let $q = [A, B, C, D, E, F]$, and let $\mathrm{lin} = [U, V, W]$. If there are no solutions the method will return 0, and otherwise it will return a vector `v`, where

$$v = [[\text{type}, v_{extra}], v_1, v_2, \ldots, v_k].$$

The types are are

-2=quadratic, -1=plane, 0=finite, 1=positive, 2=linear.

Each (family of) solution(s) is given by a v_i, possibly with reference to the extra data.

Table 3: General solution for `bqf_linearsolve`

Type	v_{extra}	v_i format	General solution
-2	-	$[[x_1, x_2, x_3], [y_1, y_2, y_3], [z_1, z_2, z_3]]$	$X = x_1U^2 + x_2U + x_3$, $Y = y_1U^2 + y_2U + y_3$, $Z = z_1U^2 + z_2U + z_3$, for $U \in \mathbb{Z}$
-1	-	$[[a_1, a_2, a_3], [b_1, b_2, b_3], [c_1, c_2, c_3]]$ [a]	$X = a_1U + b_1V + c_1$ $Y = a_2U + b_2V + c_2$, $Z = a_3U + b_3V + c_3$, for $U, V \in \mathbb{Z}$
0	-	$[a_i, b_i, c_i]$	$X = a_i$, $Y = b_i$, and $Z = c_i$
1	$M, [s_1, s_2, s_3]$ [b]	$[a_i, b_i, c_i]$ [b]	$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = M^j \begin{pmatrix} a_i \\ b_i \\ c_i \end{pmatrix} + \begin{pmatrix} s_1 \\ s_2 \\ s_3 \end{pmatrix}$ for $j \in \mathbb{Z}$
2	-	$[[a_1, a_2, a_3], [b_1, b_2, b_3]]$	$X = a_1U + b_1$, $Y = a_2U + b_2$, $Z = a_3U + b_3$, for $U \in \mathbb{Z}$

[a] In fact, $i = 1$ necessarily (there is one plane only).
[b] $M \in \mathrm{SL}(3, \mathbb{Z})$ and s_1, s_2, s_3 are *rational*; they need not be integral. Same for a_i, b_i, c_i.

3.6 Representation of integers by forms - methods

Name:	`bqf_bigreps`
Input:	`q, n`
Input format:	`q=[A, B, C, D, E]` integral vector, `n` integer
Output format:	0 or `v=[[type, data], sol1, ...]`
Description:	Solves $AX^2 + BXY + CY^2 + DX + EY = n$, and returns ALL solutions. If no solutions returns 0; otherwise `v[1][1]` gives the format of the general solution in Table 2.

Name:	`bqf_linearsolve`
Input:	`q, n1, lin, n2`
Input format:	`q=[A, B, C, D, E, F]` integer vector, `n1` an integer, `lin=[U, V, W]` integer vector, `n2` an integer
Output format:	0 or `v=[[type, data], sol1, ...]`
Description:	Solves $AX^2+BY^2+CZ^2+DXY+EXY+FYZ=n1$ and $UX+VY+WZ=n2$, and returns ALL solutions. If no solutions returns 0; otherwise `v[1][1]` gives the format of the general solution in Table 3.

Name:	`bqf_reps`
Input:	`q, n, {proper=0}, {half=1}`
Input format:	`q=[A, B, C]` integer vector, `n` integer, `proper=0, 1`, `half=0, 1`
Output format:	0 or `v=[[type, data], sol1, ...]`
Description:	Solves $AX^2+BXY+CY^2=n$, and returns ALL solutions. If no solutions returns 0; otherwise `v[1][1]` gives the format of the general solution in Table 1. If `proper=1` and the form is indefinite/definite, we only output solutions with $\gcd(x,y)=1$ (otherwise, no restriction). If `half=1`, only outputs one of (the families corresponding to) (x,y) and $(-x,-y)$, and if `half=0` outputs both.

4 Method declarations

Methods in this section are divided into subsections by the files, and into subsubsections by their general function. They will appear approximately alphabetically in each subsubsection. Clicking on a method name will bring you to its full description in the previous sections.

4.1 c_base

4.1.1 Complex geometry

`crossratio`	`a, b, c, d`
`mat_eval`	`M, x`

4.1.2 Infinity

`addoo`	`a, b`
`divoo`	`a, b`

4.1.3 Linear equations and matrices

`lin_intsolve`	`A, B, n`
`mat3_complete`	`A, B, C`

4.1.4 Solving equations modulo n

`sqmod`	`x, n`

4.1.5 Time

printtime

4.2 c_bqf

4.2.1 Discriminant methods

disclist	D1, D2, {fund=0}, {cop=0}
discprimeindex	D
fdisc	D
isdisc	D
pell	D
posreg	D
quadroot	D

4.2.2 Basic methods for binary quadratic forms

bqf_automorph	q
bqf_disc	q
bqf_isequiv	q1, q2, {tmat=0}
bqf_isreduced	q
bqf_random	maxc, {type=0}, {primitive=1}
bqf_random_D	maxc, D
bqf_red	q, {tmat=0}
bqf_roots	q
bqf_trans	q, M
bqf_trans_coprime	q, n
ideal_tobqf	numf, ideal

4.2.3 Basic methods for indefinite quadratic forms

ibqf_isrecip	q
ibqf_leftnbr	q, {tmat=0}
ibqf_redorbit	q, {tmat=0}, {posonly=0}
ibqf_rightnbr	q, {tmat=0}
ibqf_river	q
ibqf_riverforms	q
ibqf_symmetricarc	q
mat_toibqf	M

4.2.4 Class group and composition of forms

bqf_comp	q1, q2, {tored=1}
bqf_ncgp	D
bqf_ncgp_lexic	D
bqf_pow	q, n, {tored=1}
bqf_square	q, {tored=1}

4.2.5 Representation of integers by forms

`bqf_bigreps`	`q, n`
`bqf_linearsolve`	`q, n1, lin, n2`
`bqf_reps`	`q, n, {proper=0}, {half=1}`

References

[The20] The PARI Group, Univ. Bordeaux. *PARI/GP version `2.11.3`*, 2020. available from `http://pari.math.u-bordeaux.fr/`.