LOCAL SQUARE CLASSES
USER MANUAL

PRZEMYSŁAW KOPROWSKI

Contents

1. Package installation and loading 1
1.1. For Magma 1
1.2. For SageMath 2
2. Package usage 2
2.1. Magmausage 2
2.2. Sage usage 4
3. Testing and benchmarking 5
3.1. Testing 5
3.2. Benchmarking 6
References 7
MIT License 7

This manual describes a package for computing local groups of square-classes at a non-Archimedean completion (basically dyadic) of a number field. The package comes in two versions. One for Magma [1] and the other for SageMath. Both versions offer the same functionality. The software can be downloaded for free from the author's webpage at `http://www.pkoprowski.eu/local_square_classes`. The package is released under standard MIT license. The full statement of the license is given at the end of this manual.

The library was developed on versions 2.27–2.28 of Magma and 9.5–10.4 of SageMath. The author has not tested it on other versions (yet), but will appropriate any reports on how it works in different setups. Questions and comments should be sent via email to `przemyslaw.koprowski@us.edu.pl`.

1. Package installation and loading

1.1. **For Magma.** The package does not require any installation. Download the package from the author's web page

`http://www.pkoprowski.eu/local_square_classes`

It comes in form of a zip archive. Extract the content of the subdirectory `magma` of the archive to a place where Magma will be able to find it. To test the installation go to the package directory and write

```
magma example.m
```

If the output looks something like

```
Loading file "example.m"
Use the built-in method...
Time: 2.260
```

Date: July 11, 2025.

```
Use the new method...
Time: 0.040
Verify the correctness... done
```

it means that the package works correctly. Notice that the times displayed will vary depending on the actual hardware on which the package is being executed.

The package is loaded into MAGMA by attaching its file:

```
> Attach("path_to_package/local_square_classes.m");
```

One can also add this command to MAGMA startup file (specified by the environment variable `MAGMA_USER_SPEC`) to auto-load the package when MAGMA starts. For details about MAGMA startup environment consult [2, Sections 2.3.9 and 4.3].

Warning: MAGMA versions prior to V2.28-6 contained a bug that caused our function to crash on some fields. It is strongly recommended to upgrade MAGMA to the newest version.

1.2. **For SageMath.** The package does not require any installation. Download the package from the author's web page

`http://www.pkoprowski.eu/local_square_classes`

It comes in form of a zip archive. Extract the content of the subdirectory `sage` of the archive to a place where SageMath will be able to find it. To test the installation go to the package directory and write

```
sage example.sage
```

If the output looks something like

```
Computing the local group of square-classes
Time: CPU 0.36 s, Wall: 0.36 s
Dimension: 32
```

it means that the package works correctly. Notice that the times displayed will vary depending on the actual hardware on which the package is being executed.

The package is loaded into SageMath by the following command:

```
> load("path_to_package/local_square_classes.sage")
```

2. PACKAGE USAGE

Let K be a number field, $\mathfrak{p}$ its finite prime, and $K_\mathfrak{p}$ the completion of K at $\mathfrak{p}$. The group of local square-classes $K_\mathfrak{p}^\times / K_\mathfrak{p}^{\times 2}$ can be treated as a finitely-dimensional vector space over $\mathbb{F}_2$, where vector addition is given by the multiplication of square-classes. This package provides a function that constructs this vector space.

2.1. MAGMA usage.

`LocalGroupOfSquaresClasses( p )`

The input and output specification of this function is identical to the one of the built-in method `LocalMultiplicativeGroupModSquares` (see [2, § 39.7.9]). Hence, the new routine can be used as a replacement for the built-in function.

If $\mathfrak{p}$ is prime ideal in a ring of algebraic integers $\mathbb{Z}_K$ of K, this function outputs two values: V and φ. Here, V is a vector space over $\mathbb{F}_2$ isomorphic to $K_\mathfrak{p}^\times / K_\mathfrak{p}^{\times 2}$ and φ is a map from V to K, which induces the isomorphism and can compute inverses.

Moreover, if $\mathfrak{p}$ lies above 2, $e = e(\mathfrak{p}|2) = \mathrm{ord}_{\mathfrak{p}}\, 2$ denotes the absolute ramification index of $\mathfrak{p}$ and $f = f(\mathfrak{p}|2 = \big(K(\mathfrak{p}) : \mathbb{F}_2\big)$ its inertia degree, then the basis of V corresponds to a basis of $K_{\mathfrak{p}}^{\times}/K_{\mathfrak{p}}^{\times 2}$ of the form:

$$\mathscr{B} = \{\beta_1 \cdot K_{\mathfrak{p}}^{\times 2}, \ldots, \beta_{ef} \cdot K_{\mathfrak{p}}^{\times 2}\} \cup \{\Delta \cdot K_{\mathfrak{p}}^{\times 2}, \pi \cdot K_{\mathfrak{p}}^{\times 2}\}$$

Here, π stands for a uniformizer of $\mathfrak{p}$ and is the only element of the basis with odd valuation. In particular the first $ef + 1$ basis vectors of V generate the subspace corresponding to the group $U_{\mathfrak{p}}(K_{\mathfrak{p}})^{\times}/U_{\mathfrak{p}}(K_{\mathfrak{p}})^{\times 2}$. Further, the valuations of quadratic defects of β_j's form a non-decreasing sequence and $\Delta \cdot K_{\mathfrak{p}}^{\times 2}$ is the unique square-class of $K_{\mathfrak{p}}$ with the quadratic defect $\mathfrak{p}^2$.

Example. First load the package:

```
> Attach("local_square_classes.m");
```

Now, take a number field of degree 30:

```
> Qx<x> := PolynomialRing(Rationals());
// LMFDB label:
// 30.0.5315465726642625764279559041372274540978056405738705639597 6704.1
> K<t> := NumberField(x^30 + 98*x^28 + 3977*x^26 + 88668*x^24
>   + 1211568*x^22 + 10707886*x^20 + 62933870*x^18 + 248692167*x^16
>   + 659045418*x^14 + 1153231620*x^12 + 1294105343*x^10
>   + 889400042*x^8 + 347294458*x^6 + 66308517*x^4 + 4169007*x^2
>   + 44521);
```

This field has a unique prime over 2. Denote it $\mathfrak{p}$:

```
> p := PrimeFactorization(K, 2)[1][1];
```

We shall now construct the group $K_{\mathfrak{p}}^{\times}/K_{\mathfrak{p}}^{\times 2}$. To this end we write

```
> time G, phi := LocalGroupOfSquaresClasses(p);
Time: 0.030
```

Next, extract the basis $\mathscr{B}$ applying the map φ to the canonical basis of V.

```
> B := [ g @ phi : g in Basis(G) ];
> #B;
32
```

As explained above, the last element of $\mathscr{B}$ is the only one of odd valuation at $\mathfrak{p}$:

```
> [ Valuation(b, p) : b in B ];
[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1 ]
```

The valuations of quadratics defects of the remaining elements form a non-decreasing sequence (recall that `QuadraticDefect` already outputs the valuation):

```
> [ QuadraticDefect(b, p) : b in B ];
[ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 1 ]
```

For comparison construct the same group using the built-in method

```
> time H, psi := LocalMultiplicativeGroupModSquares(p);
Time: 2.260
> C := [ h @ psi : h in Basis(H) ];
> #C;
32
```

The difference in the execution times is evident. Let us verify that both functions outputted the same group. To this end we will construct two matrices:

```
> M := Matrix([ b @@ psi : b in B ]);
> N := Matrix([ c @@ phi : c in C ]);
```

The matrix M contains the coordinates with respect to $\mathscr{C}$ of the vectors from $\mathscr{B}$. The matrix N contains the coordinates with respect to $\mathscr{B}$ of the vectors from $\mathscr{C}$. The two matrices are mutual inverses and this proves that both functions outputted the same group.

```
> IsOne(M*N);
true
```

2.2. **Sage usage.**

`quadratic_defect(K, a, p, check = true)`

Notice that, as of version 10.2, SageMath contains a bug in the build-in method `K.quadratic_defect(a, p, check = true)` in the class of a number field (see `https://github.com/sagemath/sage/issues/37656`). Therefore, we provide a corrected implementation of this function.

Given a number field K, its element a and a prime ideal $\mathfrak{p}$, this function output the valuation $\operatorname{ord}_{\mathfrak{p}} \mathfrak{d}(a)$ of the quadratic defect of a. Here, the *quadratic defect* $\mathfrak{d}(a)$ is defined by the formula (see [4, §63A]):

$$\mathfrak{d}(a) := \bigcap_{b \in K_{\mathfrak{p}}} (a - b^2)\mathcal{O}_{\mathfrak{p}}(K_{\mathfrak{p}})$$

Warning: Once the the bug in the SageMath implementation of `quadratic_defect` is corrected, this function will be removed from the package.

`local_group_of_square_classes( p )`

If $\mathfrak{p}$ is prime ideal in a ring of algebraic integers $\mathbb{Z}_K$ of K, this function outputs three values: V, φ, ψ. Here, V is a vector space over $\mathbb{F}_2$ isomorphic to $K_{\mathfrak{p}}^{\times}/K_{\mathfrak{p}}^{\times 2}$, φ is a map from V to K, which induces the isomorphism, and ψ is a map from $K^{\times}$ to V such that $\psi \circ \varphi = \mathrm{id}_V$.

Moreover, if $\mathfrak{p}$ lies above 2, $e = e(\mathfrak{p}|2) = \operatorname{ord}_{\mathfrak{p}} 2$ denotes the absolute ramification index of $\mathfrak{p}$ and $f = f(\mathfrak{p}|2 = \big(K(\mathfrak{p}) : \mathbb{F}_2\big)$ its inertia degree, then the basis of V corresponds to a basis of $K_{\mathfrak{p}}^{\times}/K_{\mathfrak{p}}^{\times 2}$ of the form:

$$\mathscr{B} = \big\{\beta_1 \cdot K_{\mathfrak{p}}^{\times 2}, \ldots, \beta_{ef} \cdot K_{\mathfrak{p}}^{\times 2}\big\} \cup \big\{\Delta \cdot K_{\mathfrak{p}}^{\times 2}, \pi \cdot K_{\mathfrak{p}}^{\times 2}\big\}$$

Here, π stands for a uniformizer of $\mathfrak{p}$ and is the only element of the basis with odd valuation. In particular the first $ef + 1$ basis vectors of V generate the subspace corresponding to the group $U_{\mathfrak{p}}(K_{\mathfrak{p}})^{\times}/U_{\mathfrak{p}}(K_{\mathfrak{p}})^{\times 2}$. Further, the valuations of quadratic defects of β_j's form a non-decreasing sequence and $\Delta \cdot K_{\mathfrak{p}}^{\times 2}$ is the unique square-class of $K_{\mathfrak{p}}$ with the quadratic defect $\mathfrak{p}^2$.

Example. First load the package:

```
sage: load("local_square_classes.sage")
```

Now, take a number field of degree 30:

```
sage: Qx.<x> = QQ[]
sage: # LMFDB label:
sage: # 30.0.531546572664262576427955904137227454097805640573870563959767O4.1
sage: K.<t> = NumberField(x^30 + 98*x^28 + 3977*x^26 + 88668*x^24
....:   + 1211568*x^22 + 10707886*x^20 + 62933870*x^18 + 248692167*x^16
```

```
....:   + 659045418*x^14 + 1153231620*x^12 + 1294105343*x^10
....:   + 889400042*x^8 + 347294458*x^6 + 66308517*x^4 + 4169007*x^2
....:   + 44521)
```

This field has a unique prime over 2. Denote it `p`:

```
sage: p = K.primes_above(2)[0]
```

We shall now construct the group $K_{\mathfrak{p}}^{\times}/K_{\mathfrak{p}}^{\times 2}$. To this end we write

```
sage: time G, phi, psi = local_group_of_square_classes(p)
CPU times: user 369 ms, sys: 0 ns, total: 369 ms
Wall time: 375 ms
```

Next, extract the basis $\mathscr{B}$ applying the map φ to the canonical basis of V.

```
sage: B = [ phi(g) for g in G.basis() ]
sage: len(B)
32
```

As explained above, the last element of $\mathscr{B}$ is the only one of odd valuation at `p`:

```
sage: print([ b.valuation(p) for b in B ])
[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1 ]
```

The valuations of quadratics defects of the remaining elements form a non-decreasing sequence. Here, we use our implementation of `quadratic_defect` introduced above:

```
sage: print([ quadratic_defect(K, b, p) for b in B ])
[ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 4, 1 ]
```

3. Testing and benchmarking

The package provides two sets of scripts. One for testing the correctness of the produced results, the other to evaluate the performance of our implementation. They are contained in the folder `tests` and `benchmarks`, respectively.

> **Important:** Before running any of the following scripts, ensure that the variable `PATH_TO_LSC` in the file `path.[m|sage]` correctly points to the package `local_square_classes.[m|sage]`.

3.1. Testing.

The tests are available for both Magma and SageMath environments. The `tests` folder contains the following scripts:

- `tests.m`

  This is the basic test script for Magma. It evaluates 12 specifically chosen fields to ensure all execution branches of the code are exercised. For each field, the tests constructs the local square class group (viewed as an $\mathbb{F}_2$-vector space). The test then verifies whether the composition of the maps from the vector space to the field and form the field to the vector space is the identity on the vector space. (This also ensures that these maps are group homomorphisms.) For small dimensions, the test is exhaustive; for larger dimensions, it runs on randomly selected elements.

  Successively, the test checks whether elements in the same square class receive identical coordinates in the produced vector space.

  The runtime of this test is approximately 100 seconds on Intel® Core™ i5 running Magma 2.28-6.

- `tests.sage`

Sage version of the `tests.m` suite with one addition field (of degree 64).

The runtime is around 95 minutes on Intel® Core™ i5 running Sage 9.8, but over 90 minutes of that is spent on the last field of degree 64.

• `tests_massive.m`

An extended version of `tests.m`, this script uses fields obtained from the LMFDB [3] instead of the 12 hand-picked ones. For each pair of ramification index e and inertia degree f, the script tests up to `NUM_OF_FIELDS` (default: 5) dyadic fields. If fewer than `NUM_OF_FIELDS` exist in the LMFDB for a given pair (e, f), all available ones are used.

The runtime of this test is about 25 minutes on Intel® Core™ i5 running Magma 2.28-6.

• `tests_massive.sage`

Sage version of the `tests_massive.m` suite with one addition field (of degree 64). The runtime is around 150 minutes on Intel® Core™ i5 running Sage 9.8, but over 90 minutes of that is spent on the last field of degree 64.

• `crosstests.m`

This test uses the same 12 fields from `tests.m`. For each field, it constructs the groups of local square classes in two ways:

- Using the function `LocalGroupOfSquaresClasses` from the package (denote the result by H),
- using Magma's built-in command `LocalMultiplicativeGroupModSquares` (denote the result by G).

The script constructs the change-of-basis matrices from G to H et vice versa. Then it checks that the two matrices are mutual inverses.

The runtime is around 4 seconds on Intel® Core™ i5 running Magma 2.28-6.

• `crosstests_massive.m`

Analogous to `crosstests.m`, but uses random fields from the LMFDB [3] instead of a fixed set. For each pair (e, f) pair, up to `NUM_OF_FIELDS` (default: 20) fields are selected. The runtime of this test is about 100 seconds on Intel® Core™ i5 running Magma 2.28-6.

3.2. **Benchmarking.** The `benchmmarks` folder contains a set of benchmark scripts for timing computations of local dyadic square classes using Magma, SageMath, and Pari/GP. Benchmark scripts are named `benchmark_XYZ.[m|sage|gp]`, where `XYZ` is one of: `totally_ramified`, `totally_ramified_big`, `unramified`.

For Magma and SageMath (except `benchmark_totally_ramified_big.m`), the following operations are timed:

(1) Construction of a basis for the group of local dyadic square classes, treated as a vector space over the binary field.
(2) Generation of random representatives of uniformly distributed local dyadic square classes.
(3) Coordinate computation of a random nonzero algebraic number with respect to the basis constructed in test 1.

Pari/GP scripts and the Magma script `benchmark_totally_ramified_big.m` perform only the first of the above tests.

To ensure reliable measurements the number of random samples in tests 2 and 3, and the number of repetitions of test 1, are dynamically adjusted. Moreover, since results of test 1 are prone to caching effects (and Magma is does the caching in a really smart way), an additional consistency check is performed. The computation

is repeated 8 times and the results are compared. We allow for a deviation of no more than 30%.

The following benchmark scripts are provided:

• `benchmark_unramified.[m|sage|gp]`

Tests are run for the unique unramified extensions of $\mathbb{Q}_2$ for degrees 2 through 50. The main limitation is the time required by MAGMA to construct the unique dyadic place in the base number field. (Which, by the way, must be the principal ideal generated by 2.)

• `benchmark_totally_ramified.[m|sage|gp]`

Tests use fields defined by generating polynomials of the form $(x-1)^d+2$, where d ranges from 2 to 100. Such fields are known to have unique totally ramified dyadic places.

• `benchmark_totally_ramified_big.[m|gp]`

This test uses the same family of polynomials with degrees $d = 2^k$, for $k \in \{7, \ldots, 11\}$.

Output format. Benchmarks for MAGMA and SageMath (i.e., with `.m` and `.sage` extensions, respectively) write their output to files named

`XYZ_[magma|sage]_`⟨*current_date*⟩`.dat`.

Pari/GP scripts print timings directly to the console.

The `.dat` files outputted by the MAGMA scripts `benchmark_totally_ramified.m` and `benchmark_unramified.m` consists of the following columns:

deg: Degree of the field extension.
t1: Average time for constructing the group using MAGMA's built-in method `LocalMultiplicativeGroupModSquares`.
N1: One-eighth of the number of repetitions used to compute `t1`.
u1: Average time using the function `LocalGroupOfSquaresClasses` provided by the package.
M1: One-eighth of the repetitions used to compute `u1`.
t2, u2: Average times to construct a representative of a random local square class using MAGMA's built-in method (`t2`) and our method (`u2`).
N2: Number of random classes used to compute `t2` and `u2`.
t3, u3: Average times to compute coordinates in the local dyadic square class group using MAGMA's built-in method (`t3`) and our method (`u3`).
N3: Number of random elements used to compute `t3` and u3.

The `.dat` file produced by `benchmark_totally_ramified_big.m` contains only the columns: `deg`, `t1`, `N1`, `u1`, and `M1`.

SageMath does not offer a built-in function to construct groups of local square-classes, Hence, the `.dat` files produced by Sage scripts contains only the columns: `deg`, `u1`, `M1`, `u2`, `N2`, `u3`, and `N3`.

REFERENCES

[1] Wieb Bosma, John Cannon, and Catherine Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997. Computational algebra and number theory (London, 1993).
[2] John Cannon, Wieb Bosma, Claus Fieker, and Allan Steel (eds.). *Handbook of Magma Functions*, 2.27-7 edition, 2022.
[3] The LMFDB Collaboration, 2024. The L-functions and modular forms database. `https://www.lmfdb.org`, accessed on 18 January 2024.
[4] O. Timothy O'Meara. *Introduction to quadratic forms*. Classics in Mathematics. Springer-Verlag, Berlin, 2000. Reprint of the 1973 edition.

MIT License

Copyright (c) 2025 Przemysław Koprowski

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

> The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.