EpsteinLib User Manual

Andreas A. Buchheit*[1], Jonathan K. Busse[1,2], and Ruben Gutendorf[1]

[1]Department of Mathematics, Saarland University, 66123 Saarbrücken, Germany
[2]Department of Mathematics, Saarland University, 66123 Saarbrücken, Germany , German Aerospace Center (DLR), 51147 Cologne, Germany

January 21, 2025

Contents

*Contact: `buchheit@num.uni-sb.de`

1 Introduction

EpsteinLib is a C library designed for the fast and efficient computation of the Epstein zeta function for arbitrary multidimensional lattices. In addition to the C library, we also offer a Python package, epsteinlib, which can be easily installed via pip. For more information on the properties of the Epstein zeta function and on the underlying algorithm, see Ref. [1].

Originally studied by Epstein [8, 9], the Epstein zeta function forms the basis for computing general multidimensional lattice sums in classical and quantum physics applications [4]. Together with its regularization, it serves as the central ingredient in the singular Euler-Maclaurin (SEM) expansion, which generalizes the 300-year-old Euler summation formula to lattice sums in higher dimensions with physically relevant power-law interactions [2, 3]. An efficiently computable representation of the Epstein zeta function is provided in [1, 5, 7]. In [1], we discuss in detail the analytical properties of the Epstein zeta function and present an algorithm for its computation, complete with error bounds.

2 Usage

For a d-dimensional lattice $\Lambda = A\mathbb{Z}^d$, with $A \in \mathbb{R}^{d\times d}$ regular, $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$, and $\nu \in \mathbb{C}$, the Epstein zeta function is defined by the Dirichlet series

$$Z_{\Lambda,\nu}\left|\begin{matrix}\boldsymbol{x}\\ \boldsymbol{y}\end{matrix}\right| = {\sum_{z\in\Lambda}}' \frac{e^{-2\pi i \boldsymbol{y}\cdot\boldsymbol{z}}}{|\boldsymbol{x}-\boldsymbol{z}|^{\nu}}, \quad \mathrm{Re}(\nu) > d,$$

which can be meromorphically continued to $\nu \in \mathbb{C}$. Here, the primed sum excludes the case $\boldsymbol{z} = \boldsymbol{x}$.

The Epstein zeta function is implemented in this library as

```c
double complex epsteinZeta(
    double nu,
    unsigned int dim,
    const double *A,
    const double *x,
    const double *y);
```

In the Python package, it is implemented as

```python
def epstein_zeta(
    nu: Union[float, int],
    A: NDArray[Union[np.integer[Any], np.floating[Any]]],
    x: NDArray[Union[np.integer[Any], np.floating[Any]]],
    y: NDArray[Union[np.integer[Any], np.floating[Any]]],
) -> complex
```

In the Mathematica package, it is implemented as

```
EpsteinZeta[\[Nu],A,x,y]
```

and evaluates to full precision over the whole parameter range up to ten dimensions.

In addition, this library includes the regularized Epstein zeta function, which is analytic around $\boldsymbol{y} = 0$, and is defined via

$$Z^{\mathrm{reg}}_{\Lambda,\nu}\begin{vmatrix}\boldsymbol{x}\\ \boldsymbol{y}\end{vmatrix} = e^{2\pi i \boldsymbol{x}\cdot\boldsymbol{y}} Z_{\Lambda,\nu}\begin{vmatrix}\boldsymbol{x}\\ \boldsymbol{y}\end{vmatrix} - \frac{\hat{s}(\boldsymbol{y})}{V_\Lambda},$$

where $V_\Lambda = |\det A|$ is the volume of the elementary lattice cell, and

$$\hat{s}_\nu(\boldsymbol{y}) = \frac{\pi^{\nu/2}}{\Gamma(\nu/2)}\Gamma\big((d-\nu)/2\big)(\pi\boldsymbol{y}^2)^{(\nu-d)/2}, \quad \nu \notin (d+2\mathbb{N}_0)$$

is the distributional Fourier transform of $|\boldsymbol{z}|^{-\nu}$, where Γ denotes the gamma function and we adopt the choice

$$\hat{s}_{d+2k}(\boldsymbol{y}) = \frac{\pi^{k+d/2}}{\Gamma(k+d/2)}\frac{(-1)^{k+1}}{k!}(\pi\boldsymbol{y}^2)^k\log(\pi\boldsymbol{y}^2), \quad k \in \mathbb{N}_0.$$

In the c library, the regularized Epstein zeta function is included as

```
double complex epsteinZetaReg(
    double nu,
    unsigned int dim,
    const double *A,
    const double *x,
    const double *y);
```

in the Python package as

```
def epstein_zeta_reg(
    nu: Union[float, int],
    A: NDArray[Union[np.integer[Any], np.floating[Any]]],
    x: NDArray[Union[np.integer[Any], np.floating[Any]]],
    y: NDArray[Union[np.integer[Any], np.floating[Any]]],
) -> complex
```

and in the Mathematica package as

```
EpsteinZetaReg[\[Nu],A,x,y]
```

3 Installation

Install our required dependencies: meson, ninja, pkg-config, python3 e.g. with

```
pacman -S meson ninja pkgconf python
```

for Archlinux and

```
brew install meson ninja pkg-config python3
```

for MacOs. Currently, we support native Windows builds only with GCC installed via `https://www.msys2.org/`. Different environments may or may not work. However, for the full out of the box development experience we encourage Windows users to use WSL2 and follow the Linux installation instructions.

3.1 Installing only the Python wrapper with pip

Create and activate a virtual environment before installing the Python wrapper, if you're not already in one.

```
cd C && python3 -m venv .venv && source .venv/bin/activate
```

EpsteinLib ist then installed by:

```
python -m pip install .
```

3.2 Installing the C library and the Python wrapper with meson

The following bash commands install the library with meson.

1. `cd C`
2. `meson setup build`
3. `meson compile -C build`

Proceed either with system-wide or local installation.

3.3 System-wide installation

Meson supports a system-wide installation of the compiled library. After that, you can use `#include <epsteinZeta.h>` and link the library with `gcc -lepsteinZeta`. This may require superuser rights.

6. To install system-wide: `meson install -C build`.
7. Try to compile the sample program in `test lattice_sum.c` with the command `gcc -o lattice_sum lattice_sum.c -lm -lepsteinZeta`. You may encounter the problem that the shared library cannot be found. In this case, you need to modify the environment variables. Please continue with the next step. Otherwise, you are done.
8. Update the environment variables to correctly locate the shared library at runtime. You can find `/path/to/library` in the output given by `meson install`.

In Linux, the environment variable is exported by

```
export $LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/path/to/library
```

and in MacOs by

```
export $DYLD_LIBRARY_PATH=$DYLD_LIBRARY_PATH:/path/to/library
```

3.4 Local installation

6. Copy the header `include/epsteinZeta.h` and move the compiled library `build/src/libepsteinZeta.so` to places of your choice. This can be done by

```
cp include/epsteinZeta.h /your/path/to/include
mv build/src/libepsteinZeta.so /your/path/to/library
```

7. To test your library, try to compile `test lattice_sum.c` with the following command:

```
gcc -o lattice_sum lattice_sum.c -lm -L/your/path/to/library -lepsteinZeta \
    -I/your/path/to/include
```

4 Documentation

You can find our API documentation at `https://epsteinlib.github.io/epsteinlib/`.

5 Usage

Minimal working examples for calculating the Madelung constant for NaCl in 3 dimensions, (see [6] for a reference value).

5.1 C example

In C, the Madelung constant is calculated as follows:

```c
// If the library is installed, compile with
// `gcc -o lattice_sum lattice_sum.c -lm -lepsteinZeta
// If the library is not installed, compile with
// `gcc -o lattice_sum lattice_sum.c -lm -L/path/to/library -lepsteinZeta \
//      -I/path/to/include`

#include <complex.h>
#include <math.h>
#include <stdio.h>
#include "epsteinZeta.h"


int main() {

    // Madelung constant found in literature
    double madelungRef = -1.747564594633182190636212035544397 4;
    unsigned int dim = 3;
    double m[] = {1, 0, 0, 0, 1,
                  0, 0, 0, 1};     // identity matrix for whole numbers
    double x[] = {0, 0, 0};        // no shift
    double y[] = {0.5, 0.5, 0.5}; // alternating sum
    double nu = 1.0;
    double madelung = creal(epsteinZeta(nu, dim, m, x, y));
    printf("Madelung sum in 3 dimensions:\t %.16lf\n", creal(madelung));
    printf("Reference value:\t\t %.16lf\n", madelungRef);
    printf("Relative error:\t\t\t +%.2e\n",
           fabs(madelungRef - madelung) / fabs(madelungRef));

    return fabs(madelung - madelungRef) > pow(10, -14);
}
```

5.2 Python example

We perform the calculation of the Madelung constant using the Python wrapper as follows.

```python
import numpy as np
from epsteinlib import epstein_zeta

madelung_ref = -1.7475645946331821906362120355443974
dim = 3
a = np.identity(dim)            # identity matrix for whole numbers
x = np.zeros(dim)               # no shift
y = np.full(dim, 0.5)           # alternating sum
nu = 1.0
madelung = np.real(epstein_zeta(nu, a, x, y))
print(f"Madelung sum in 3 dimensions:\t {madelung:.16f}")
print(f"Reference value:\t\t {madelung_ref:.16f}")
print(f"Relative error:\t\t\t +{abs(madelung_ref - madelung) / abs(madelung_ref):.2e}")
```

In the `examples/python/` folder, you can find two more Python examples:

1. `dispersion_relation_3d.py`: This script demonstrates how to use EpsteinLib to calculate quantum dispersion relations in 3D.
2. `sem_gaussian_1d.py`: This script showcases the Singular Euler-Maclaurin (SEM) expansion for a Gaussian function in 1D. It has an optional argument `--nu` that can be used to set the value of nu. For example, you can run it with `python sem_gaussian_1d.py --nu 1`. If no value is provided, it defaults to `nu` = 1.5.

These examples, along with the `lattice_sum.py` script, provide a comprehensive overview of how to use EpsteinLib in various scenarios.

5.3 Mathematica example

```mathematica
<<"EpsteinZeta.wl"

madelungRef = -1.7475645946331821906362120355443974;

dim = 3;
A = IdentityMatrix[dim];
x = ConstantArray[0, dim];
y = ConstantArray[0.5, dim];
\[Nu] = 1.0;

madelung = Re[EpsteinZeta[\[Nu], A, x, y]];

Print["Madelung sum in 3 dimensions: ", NumberForm[madelung, 16]]
Print["Reference value:              ", NumberForm[madelungRef, 16]]
```

```
Print["Relative error:                  ",
    ScientificForm[Abs[madelungRef - madelung]/Abs[madelungRef], 2]]
```

Executing this code snipped in the same folder as `EpsteinZeta.wl` and setting `SetDirectory[NotebookDirectory[]]` is the easiest way to help mathematica find the package.

6 Reproducing Results

The numerical results presented in our paper can be reproduced as follows. To regenerate all figures, evaluate `Figures.wls` located in `C/examples/mathematica`. On a standard 12-core laptop, the evaluation completes in under 2 minutes. The benchmark data can be regenerated using the following steps:

1. Navigate to the C directory: `cd C`
2. Build the project: `meson setup build && meson compile -C build`
3. Execute the benchmark programs:

```
# Regenerates benchmark values in `benchmark/csv` for epsteinZeta and epsteinZetaReg
./build/benchmark/epsteinlib_benchmark_epstein

# Regenerates benchmark values in `benchmark/csv` for the upper gamma function
./build/benchmark/epsteinlib_benchmark_gamma
```

The benchmark results are automatically saved as CSV files in `C/benchmark/csv`.

7 Development environment

We provide a nix devshell to have a reproducible development environment with the same dependencies across different operating systems. Once you have installed and configured nix, starting developing is as easy as running `nix develop`.

7.1 Nix installation instructions

1. Install nix; follow the wiki
2. Configure nix by executing

```
sudo tee -a /etc/nix/nix.conf <<CFG
max-jobs = auto
#max-jobs = 1
experimental-features = nix-command flakes auto-allocate-uids
```

```
auto-allocate-uids = true
auto-optimise-store = true
CFG
```

3. `systemctl enable --now nix-daemon.socket`
4. `usermod -a -G nix-users <your username>`
5. Reboot
6. `cd <path/to/folder>/C`
7. `bash init.sh`
8. `nix develop` or `nix run -- <your args>`

7.2 Nix-Portable based installation instructions

We offer a Nix-portable solution, which is especially helpful if you do not have root rights.

1. Install nix-portable:

```
mkdir -p ~/.local/bin
cd ~/.local/bin

curl -L https://github.com/DavHau/nix-portable/releases/latest/download/nix-portable\
    -$(uname -m) > ./nix-portable
chmod +x ./nix-portable
cat > ./nix <<NIX
#!/usr/bin/env bash
CURDIR=\$(dirname "\$(readlink -f "\$0")")
NP_RUNTIME=bwrap "\$CURDIR/nix-portable" nix \$@
NIX
chmod +x ./nix

export PATH=~/.local/bin:"$PATH"
cd ~
nix run 'nixpkgs#hello'
```

2. Configure `nix.conf` by executing

```
tee -a ~/.nix-portable/conf/nix.conf <<CFG
max-jobs = auto
#max-jobs = 1
auto-optimise-store = true
CFG
```

3. Add `.local/bin` permanently to your PATH

```
echo 'PATH=$HOME/.local/bin:"$PATH"' >> ~/.env
echo 'export $(envsubst < .env)' | tee -a .bashrc >> .zshrc
```

4. `cd <path/to/folder>/C`
5. `bash init.sh`
6. `nix develop` or `nix run -- <your args>`

8 Contributing

Pull requests are welcome. For major changes, please open an issue first to discuss what you would like to change.

References

[1] Andreas A. Buchheit, Jonathan Busse, and Ruben Gutendorf. "Computation and properties of the Epstein zeta function with high-performance implementation in EpsteinLib". In: *arXiv preprint arXiv:2412.16317* (2024).

[2] Andreas A. Buchheit and Torsten Keßler. "On the Efficient Computation of Large Scale Singular Sums with Applications to Long-Range Forces in Crystal Lattices". In: *J. Sci. Comput.* 90.1 (2022), pp. 1–20.

[3] Andreas A. Buchheit and Torsten Keßler. "Singular Euler–Maclaurin expansion on multidimensional lattices". In: *Nonlinearity* 35.7 (2022), p. 3706.

[4] Andreas A. Buchheit, Torsten Keßler, Peter K. Schuhmacher, and Benedikt Fauseweh. "Exact continuum representation of long-range interacting systems and emerging exotic phases in unconventional superconductors". In: *Phys. Rev. Res.* 5 (4 Oct. 2023), p. 043065.

[5] Andreas A. Buchheit, Torsten Keßler, and Kirill Serkh. "On the Computation of Lattice Sums without Translational Invariance". In: *Mathematics of Computation* (Oct. 2024).

[6] Antony Burrows, Shaun Cooper, and P. Schwerdtfeger. "The Madelung Constant in N Dimensions". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 478.2267 (Nov. 2022), p. 20220334.

[7] R. Crandall. "Unified algorithms for polylogarithm, L-series, and zeta variants". In: *Algorithmic Reflections: Selected Works.* PSIpress, 2012.

[8] P. Epstein. "Zur Theorie allgemeiner Zetafunctionen". In: *Math. Ann.* 56 (1903), pp. 615–644.

[9] P. Epstein. "Zur Theorie allgemeiner Zetafunktionen. II". In: *Math. Ann.* 63 (1906), pp. 205–216.