libcasimir Documentation

Release 0.4.1

Michael Hartmann

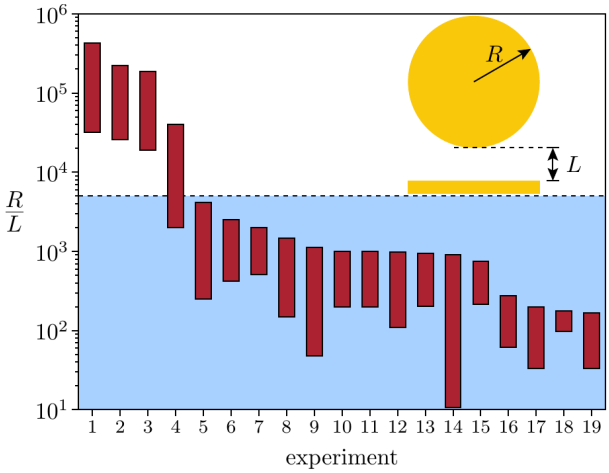
Jan 10, 2019

CONTENTS

CHAPTER

ONE

OVERVIEW AND FEATURES



[1] Masuda, Sasaki (2009)
[2] Sushkov *et al.* (2011)
[3] Lamoreaux (1997)
[4] Garcia-Sanchez *et al.* (2012)
[5] van Zwol *et al.* (2008)
[6] Zwol *et al.* (2008)
[7] Man *et al.* (2009)
[8] Decca *et al.* (2003)
[9] Chan *et al.* (2001)
[10] Munday *et al.* (2008)
[11] Munday *et al.* (2009)
[12] Mohideen, Roy (1998)
[13] Decca *et al.* (2007)
[14] Krause *et al.* (2007)
[15] Bimonte *et al.* (2016)
[16] Banishev *et al.* (2013)
[17] Jourdan *et al.* (2009)
[18] Chang *et al.* (2012)
[19] Torricelli *et al.* (2011)

Fig. 1: Experiments in the plane-sphere geometry. The blue area denotes the aspect ratios that are accessible using libcasimir. The inset depicts the plane-sphere geometry.

libcasimir is an implementation of the Casimir effect in the plane-sphere geometry. The geometry consists of a sphere of radius R separated by a distance L from an infinite plate. The main goal of the library and the associated programs is to compute the free energy $\mathcal{F}$ depending on the radius of the sphere, the separation between sphere and plate, the temperature, and the dielectric properties of the objects. The library is highly optimized and allows you - depending on parameters and your hardware - to compute the free energy for aspect ratios up to $R/L \sim 10\,000$.

1.1 Features

- Calculate the free energy for different separations and temperatures
- Calculate the free energy in the high temperature limit
- Full support for perfect reflectors, Drude metals, and generic metals described by a user-defined dielectric function
- libcasimir is fast and reliable
- ready to use programs: you don't have to modify the code
- libcasimir is free software – you may use it or even modify it

1.2 Further reading

Some of the numerical ideas used in this library are described in Hartmann, Ingold, Maia Neto, "Advancing numerics for the Casimir effect to experimentally relevant aspect ratios", Phys. Scr. 93, 114003 (2018). A more detailed description can be found in Hartmann, "Casimir effect in the plane-sphere geometry: Beyond the proximity force approximation", phd thesis (2018).

CHAPTER

TWO

INSTALLATION

In the following, we assume the operating system to be Ubuntu 18.04. The commands should also work on other Debian-like systems.

2.1 Compilation

The easiest way to get the source code is to use git. To install git, run

```
$ sudo apt install git
```

in a terminal. Once git is installed, the command

```
$ git clone https://github.com/michael-hartmann/libcasimir-dev.git
```

will get you the complete libcasimir repository and stores it in the directory `libcasimir-dev/`.

The libcasimir library and the programs are written in C and C++ using LAPACK and MPI. In order to compile the source files, you need a C and C++ compiler, the development files for LAPACK and MPI, and the build tool make. You can install all dependencies with:

```
$ sudo apt install gcc g++ libc6-dev libc++-dev make libopenmpi-dev openmpi-bin␣
↪liblapack-dev libgfortran-7-dev gfortran-7
```

In order to compile the code, run `make` in the directory `libcasimir-dev/src/`:

```
$ cd libcasimir-src/dev
$ make
```

This command compiles the HODLR library and builds the shared object `libhodlr.so` in the directory `libhodlr/`. Then, the Makefile compiles the libcasimir library `libcasimir.so` and builds the programs `casimir` and `casimir_logdetD`.

In order to run the programs, the system must be able to find the libraries `libhodlr.so` and `libcasimir.so`. As both libraries are not in the default search path, you have to add the directories to `LD_LIBRARY_PATH`

```
$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/hendrik/libcasimir-dev/src:/home/
↪hendrik/libcasimir-dev/src/libhodlr
```

where we have assumed that the libcasimir repository is in the directory `/home/hendrik`.

It is also possible to use different compilers than gcc. To compile the code with clang and clang++ run

```
$ make clean
$ CC=clang CXX=clang++ make
```

and for Intel's compiler:

```
$ make clean
$ CC=icc CXX=icpc make
```

Please make sure to run `make clean` first.

2.2 Testing

In order to check whether the compilation was successful, you can build and run the unit tests in `src/`:

```
$ make casimir_tests
```

All tests should pass. Running the tests takes up (depending on your hardware) about 7 minutes.

2.3 Adapting the Makefile

In order to improve performance, it might be neccessary to tweak some compiler options. The options are described in the file `src/Makefile`. The most interesting option is `-mmarch=native` which tells the compiler to optimize the code for the architecture the compiler is running on. This might improve performance by ~5%.

CHAPTER

THREE

PROGRAMS

3.1 casimir

The program `casimir` computes the free Casimir energy $\mathcal{F}$ for the plane-sphere geometry as a sum

$$\mathcal{F} = \frac{k_\mathrm{B}T}{2} \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} \log\det\left(1 - \mathcal{M}^m(\xi_n)\right)$$

over the Matsubara frequencies $\xi_n = 2\pi n k_\mathrm{B}T/\hbar$. For zero temperature $T = 0$, the sum over the Matsubara frequencies becomes an integration.

The program supports a wide variety of options. You can get a summary of all options using `casimir --help`. By default, the temperature is set to $T = 0$, and the sphere and plane are assumed to be perfect reflectors.

3.1.1 Mandatory options

There are two mandatory options: the separation L between sphere and plane, and the radius of the sphere R. The program expects the lengths given in units of meters. As an example, the following command computes the Casimir interaction at $T = 0$ for perfect reflectors for a sphere of radius $R = 50\mu\mathrm{m}$ and a separation $L = 2\mu\mathrm{m}$:

```
$ mpirun -n 8 ./casimir -R 50e-6 -L 2e-6
```

The command `mpirun` will set up the environment for MPI and the flag `-n` specifies how many processes the program should use. If you want to utilize the full capacity of your machine and your processor has N cores, set `-n` to N+1. The output of this command looks similar to:

```
# compiler: gcc
# compile time: Jan  2 2019 12:35:13
# compiled on: Linux jonas.physik.uni-augsburg.de 4.9.0-8-amd64 #1 SMP Debian 4.9.130-
↪2 (2018-10-27) x86_64 GNU/Linux
# git HEAD: 51b8df47618d15747e59321aee77e1f86fab32a8
# git branch: master
# pid: 23336
# start time: Thu Jan  3 12:27:38 2019
#
# LbyR = 0.04
# RbyL = 25
# L = 2e-06
# R = 5e-05
# T = 0
# cutoff = 1e-09
# epsrel = 1e-06
```

(continues on next page)

(continued from previous page)

```
# iepsrel = 1e-08
# ldim = 176
# cores = 8
# quad = adaptive Gauss-Kronrod
#
# xi*(L+R)/c=13, logdetD=-2.35165297953047, t=1.14694
# xi*(L+R)/c=3029.84719296933, logdetD=0, t=9.53674e-07
    ...
# xi*(L+R)/c=61.4928297600332, logdetD=-0.0548151501755969, t=1.37253
# xi*(L+R)/c=51.8422518708047, logdetD=-0.115164831511918, t=1.38072
#
# ier=0, integral=-6.41428305795366, neval=135, epsrel=2.91062e-07
#
# 6154 determinants computed
# stop time: Thu Jan  3 12:30:01 2019
#
# L/R, L, R, T, ldim, E*(L+R)/(hbar*c)
0.03999999999999999, 2e-06, 5e-05, 0, 176, -26.54248623166202
```

The output is in the format of a CSV file and additional comments start with a pound (#). The program first outputs some information on the compilation, i.e., time of compilation, name of compiler, machine where it was compiled and so on. Then, information about the geometry (radius, seperation, aspect ratio and inverse of aspect ratio), numerical parameters (cutoff, epsrel, iepsrel, ldim, cores) are printed. We will discuss the numerical parameters in more detail later. The value of cores is the number of MPI processes that are used for the computation. Then, the determinant of the scattering matrix for different Matsubara frequencies are printed. The comment starting with `ier` gives the result of the integration and is and `ier=0` if the integration was successful. The program ends by printing the result of the computation. The free energy is outputed in units of $(L+R)/\hbar c$, i.e., for this example the free energy is

$$\mathcal{F} \approx \frac{-26.54\hbar c}{50\mu\mathrm{m} + 2\mu\mathrm{m}} \approx -1.61 \times 10^{-20}\mathrm{J}.$$

The PFA result in this case is $\mathcal{F}_{\mathrm{PFA}} \approx -1.64 \times 10^{-20}\mathrm{J}$.

The desired relative accuracy of the integration over the Matsubara frequencies can be set using `--epsrel`. By default, `EPSREL` is 10^{-6}. Note that the integrand needs to be sufficiently smooth. In particular, for very low values of `EPSREL` you might need to decrease the value of `CUTOFF` using `--cutoff`. The value of `CUTOFF` determines when the summation over m is stopped:

$$\frac{\log\det\left(1 - \mathcal{M}^m(\xi)\right)}{\log\det\left(1 - \mathcal{M}^0(\xi)\right)} < \mathrm{CUTOFF}$$

The default value of `CUTOFF` is 10^{-9}. As a rule of thumb, in order that the integrand is sufficiently smooth for the integration routine, `CUTOFF` should be at least two orders of magnitude smaller than `EPSREL`.

By default, the integration routine uses an adaptive Gauss-Kronrod method provided by CQUADPACK. For perfect reflectors it is sometimes faster to use an adaptive exponentially convergent Fourier-Chebshev quadrature scheme (FCQS), see Boyd, "Exponentially convergent Fourier-Chebshev quadrature schemes on bounded and infinite intervals", JOSC 2, 2 (1987). You can use FCQS using the flag `--fcqs`. Since the adaptive algorithm is not well tested, this option is considered experimental.

3.1.2 Temperature

You can set the temperature using `-T`. The following program computes the free energy just like in the last example but at room temperature $T = 300\mathrm{K}$:

```
$ mpirun -n 8 ./casimir -R 50e-6 -L 2e-6 -T 300
# compiler: gcc
# compile time: Jan  2 2019 12:35:13
# compiled on: Linux jonas.physik.uni-augsburg.de 4.9.0-8-amd64 #1 SMP Debian 4.9.130-
↪2 (2018-10-27) x86_64 GNU/Linux
# git HEAD: 51b8df47618d15747e59321aee77e1f86fab32a8
# git branch: master
# pid: 24111
# start time: Thu Jan  3 12:51:15 2019
#
# LbyR = 0.04
# RbyL = 25
# L = 2e-06
# R = 5e-05
# T = 300
# using Matsubara spectrum decomposition (MSD)
# cutoff = 1e-09
# epsrel = 1e-06
# iepsrel = 1e-08
# ldim = 176
# cores = 8
# model = perfect reflectors
#
# xi*(L+R)/c=0, logdetD=-6.16165739556559, t=0.025461
# xi*(L+R)/c=42.8046294355951, logdetD=-0.230987113168236, t=1.22125
# xi*(L+R)/c=85.6092588711902, logdetD=-0.00858232692932167, t=1.186
# xi*(L+R)/c=128.413888306785, logdetD=-0.000319205031747869, t=1.10549
# xi*(L+R)/c=171.21851774238, logdetD=-1.18538336130793e-05, t=0.949226
# xi*(L+R)/c=214.023147177976, logdetD=-4.39335203111237e-07, t=0.77597
#
# 302 determinants computed
# stop time: Thu Jan  3 12:51:21 2019
#
# L/R, L, R, T, ldim, E*(L+R)/(hbar*c)
0.03999999999999999, 2e-06, 5e-05, 300, 176, -45.24539531432269
```

For finite temperature the free energy is no longer given as an integral, but as a sum over the Matsubara frequencies ξ_n. The summation is stopped once

$$\frac{\log\det\left(1-\mathcal{M}(\xi_n)\right)}{\log\det\left(1-\mathcal{M}(0)\right)} < \mathrm{EPSREL}.$$

By default, `EPSREL` is 10^{-6}. You can change the value of `EPSREL` using the option `--epsrel`.

By default, the free energy is computed summing over the Matsubara frequencies ξ_n, also called Matsubara spectrum decomposition (MSD). Another option is to compute the free energy using Padé spectrum decomposition (PSD). PSD is an optimal sum-over-poles expansion scheme, more information can be found in Hu, Xu, Yan, "Padé spectrum decompositions of quantum distribution functions and optimal hierarchical equations of motion construction for quantum open systems", J. Chem. Phys. 133, 101106 (2010). The PSD requires less terms to be computed compared to the MSD. You can tell the program to use PSD with the flag `--psd`. The order is determined automatically to achieve a relative error of the order specified by `--epsrel`. You can also manually set the order using `--psd-order`. Since the automatic determination of the order is not well tested, this option is considered experimental.

If you are only interested in the high-temperature limit, the flag `--ht` will only compute $\log\det(1-\mathcal{M}(0))$ and output the Casimir energy in the limit $T\to\infty$ in units of $k_\mathrm{B}T$.

3.1.3 Material parameters

Up to this point, we have assumed that the sphere and the plate are perfect reflectors. If you want to model the sphere and the plate using the plasma model, you can set the plasma frequency using `--omegap`. The plasma frequency is expected in units of $\mathrm{eV}/\hbar$. For example, for $R = 50\mu\mathrm{m}$, $L = 800\mathrm{nm}$, $T = 300\mathrm{K}$, and gold (plasma frequency $\omega_\mathrm{P} = 9\mathrm{eV}/\hbar$), the Casimir free energy using the plasma model is:

```
mpirun -n 8 ./casimir -R 50e-6 -L 1e-6 -T 300 --omegap 9
# compiler: gcc
# compile time: Jan  2 2019 12:35:13
# compiled on: Linux jonas.physik.uni-augsburg.de 4.9.0-8-amd64 #1 SMP Debian 4.9.130-
↪2 (2018-10-27) x86_64 GNU/Linux
# git HEAD: 51b8df47618d15747e59321aee77e1f86fab32a8
# git branch: master
# pid: 25590
# start time: Thu Jan  3 13:45:32 2019
#
# LbyR = 0.02
# RbyL = 50
# L = 1e-06
# R = 5e-05
# T = 300
# using Matsubara spectrum decomposition (MSD)
# cutoff = 1e-09
# epsrel = 1e-06
# iepsrel = 1e-08
# ldim = 351
# cores = 8
# omegap = 9
# gamma = 0
# model = plasma
#
# xi*(L+R)/c=0, logdetD=-12.9879117945873, t=0.386934
# xi*(L+R)/c=41.9814634849106, logdetD=-2.25212799211675, t=8.56138
# xi*(L+R)/c=83.9629269698212, logdetD=-0.399130249157291, t=8.79863
# xi*(L+R)/c=125.944390454732, logdetD=-0.0715772034739604, t=8.86131
# xi*(L+R)/c=167.925853939642, logdetD=-0.0128539503490175, t=8.90524
# xi*(L+R)/c=209.907317424553, logdetD=-0.00230798551800234, t=8.56306
# xi*(L+R)/c=251.888780909464, logdetD=-0.000414281644513791, t=8.18385
# xi*(L+R)/c=293.870244394374, logdetD=-7.43406394982715e-05, t=7.62512
# xi*(L+R)/c=335.851707879285, logdetD=-1.33358593258147e-05, t=6.89742
# xi*(L+R)/c=377.833171364195, logdetD=-2.39148466183249e-06, t=6.48929
#
# 685 determinants computed
# stop time: Thu Jan  3 13:46:45 2019
#
# L/R, L, R, T, ldim, E*(L+R)/(hbar*c)
0.02, 1e-06, 5e-05, 300, 351, -123.3743917511159
```

To describe the objects using the Drude model, you can additional specify the relaxation frequency γ (also in units of $\mathrm{eV}/\hbar$). For gold, $\gamma = 35\mathrm{meV}/\hbar$, so the same example as above for Drude gives:

```
mpirun -n 8 ./casimir -R 50e-6 -L 1e-6 -T 300 --omegap 9 --gamma 0.035
# compiler: gcc
# compile time: Jan  2 2019 12:35:13
# compiled on: Linux jonas.physik.uni-augsburg.de 4.9.0-8-amd64 #1 SMP Debian 4.9.130-
↪2 (2018-10-27) x86_64 GNU/Linux
```

(continues on next page)

(continued from previous page)

```
# git HEAD: 51b8df47618d15747e59321aee77e1f86fab32a8
# git branch: master
# pid: 25643
# start time: Thu Jan  3 13:48:50 2019
#
# LbyR = 0.02
# RbyL = 50
# L = 1e-06
# R = 5e-05
# T = 300
# using Matsubara spectrum decomposition (MSD)
# cutoff = 1e-09
# epsrel = 1e-06
# iepsrel = 1e-08
# ldim = 351
# cores = 8
# omegap = 9
# gamma = 0.035
# model = drude
#
# xi*(L+R)/c=0, logdetD=-7.09741176750412, t=0.000463963
# xi*(L+R)/c=41.9814634849106, logdetD=-2.23320258322643, t=8.40539
# xi*(L+R)/c=83.9629269698212, logdetD=-0.396003476948632, t=8.85682
# xi*(L+R)/c=125.944390454732, logdetD=-0.071024911146045, t=9.09324
# xi*(L+R)/c=167.925853939642, logdetD=-0.0127552647899639, t=8.74376
# xi*(L+R)/c=209.907317424553, logdetD=-0.00229031081195299, t=8.5974
# xi*(L+R)/c=251.888780909464, logdetD=-0.000411115126231591, t=8.13146
# xi*(L+R)/c=293.870244394374, logdetD=-7.3773492717613e-05, t=7.64662
# xi*(L+R)/c=335.851707879285, logdetD=-1.32343284458642e-05, t=7.20287
# xi*(L+R)/c=377.833171364195, logdetD=-2.37331945823411e-06, t=6.58019
#
# 627 determinants computed
# stop time: Thu Jan  3 13:50:03 2019
#
# L/R, L, R, T, ldim, E*(L+R)/(hbar*c)
0.02, 1e-06, 5e-05, 300, 351, -83.71300491448063
```

The Casimir energy in the high-temperature limit for the Drude and the plasma model differ by a factor of 2. This is the reason why in this example the Casimir energy using the plasma model is considerably larger than using the Drude model.

General materials can be described using `--material` which expects the path to a material file. A material file has the following format:

```
# Drude parameters for low frequencies
# omegap_low = 9.0eV
# gamma_low  = 0.03eV
#
# Drude parameters for high frequencies
# omegap_high = 54.475279eV
# gamma_high  = 211.48855eV
#
# xi in rad/s            epsilon(i*xi)
151900000000.0000        27202177.31278406
167090000000.0000        24722324.84701070
182280000000.0000        22655566.74077545
...
```

(continues on next page)

(continued from previous page)

```
1.4886200000000000E+018   1.002550948190463
1.5038100000000000E+018   1.002504731170786
1.5190000000000000E+018   1.002459773692494
```

Each line either starts with a pound (#) or contains a frequency ξ in units of $\mathrm{rad/s}$ and the corresponding value of the dielectric function $\epsilon(\mathrm{i}\xi)$ separated by spaces. The frequencies have to be in ascending order. The dielectric function for an arbitrary frequency is then computed using linear interpolation. For frequencies smaller than the smallest frequency provided in the file, the dielectric function is computed using the Drude model

$$\epsilon(\mathrm{i}\xi) = 1 + \frac{\omega_\mathrm{P}^2}{\xi(\xi+\gamma)}$$

with the plasma frequency given by `omegap_low` and the relaxation frequency given by `gamma_low`. If `omegap_low` and `gamma_low` are not given in the file, the dielectric function is assumed to be 1. The behaviour for frequencies larger than the largest provided frequency is analougous.

Here is an example that computes the Casimir energy for a sphere of $R = 50\mu\mathrm{m}$ at separation $L = 1\mu\mathrm{m}$ at room temperature $T = 300\mathrm{K}$ for real gold:

```
$ mpirun -n 8 ./casimir -R 50e-6 -L 1e-6 -T 300 --material ../materials/gold.csv
# compiler: gcc
# compile time: Jan  7 2019 10:28:24
# compiled on: Linux jonas.physik.uni-augsburg.de 4.9.0-8-amd64 #1 SMP Debian 4.9.130-
↪2 (2018-10-27) x86_64 GNU/Linux
# git HEAD: 5c3f8f083f2af60bd680646adc94997b179c350c
# git branch: master
# pid: 4840
# start time: Mon Jan  7 10:28:44 2019
#
# LbyR = 0.02
# RbyL = 50
# L = 1e-06
# R = 5e-05
# T = 300
# using Matsubara spectrum decomposition (MSD)
# cutoff = 1e-09
# epsrel = 1e-06
# iepsrel = 1e-08
# ldim = 351
# cores = 8
# filename = ../materials/gold.csv
# model = optical data (xi=0: Drude)
# plasma = -12.9879117939843 (logdetD(xi=0) for plasma model with omegap=9eV)
#
# xi*(L+R)/c=0, logdetD=-7.09741176750412, t=0.352728
# xi*(L+R)/c=41.9814634849106, logdetD=-2.23788117126162, t=8.44477
# xi*(L+R)/c=83.9629269698212, logdetD=-0.397789450092019, t=8.8778
# xi*(L+R)/c=125.944390454732, logdetD=-0.0716100613017328, t=9.09214
# xi*(L+R)/c=167.925853939642, logdetD=-0.0129251607796874, t=9.02829
# xi*(L+R)/c=209.907317424553, logdetD=-0.00233585322790776, t=8.67091
# xi*(L+R)/c=251.888780909464, logdetD=-0.000422670611527527, t=8.18489
# xi*(L+R)/c=293.870244394374, logdetD=-7.65823629633713e-05, t=7.80386
# xi*(L+R)/c=335.851707879285, logdetD=-1.38953641418705e-05, t=7.10853
# xi*(L+R)/c=377.833171364195, logdetD=-2.52460509493263e-06, t=6.50239
#
# 682 determinants computed
# stop time: Mon Jan  7 10:29:58 2019
```

(continues on next page)

(continued from previous page)

```
#
# L/R, L, R, T, ldim, E*(L+R)/(hbar*c)
0.02, 1e-06, 5e-05, 300, 351, -83.81029275263388
```

The energy printed in the last line assumes a Drude model for the zero-th Matsubara frequency. If you want to use the plasma model for the zero-th Matsubara frequency, you can use the value given by `# plasma =`. This number, i.e., -12.9879... is given in units of $k_{\mathrm{B}}T/2$ and corresponds to the additional contribution in the high-temperature limit to the energy in the plasma model. In this example, the free energy using the Drude model for zero-frequency is

$$\mathcal{F}_{\mathrm{Drude}} \approx -83.8\frac{\hbar c}{L+R} \approx -5.19 \times 10^{-20}\mathrm{J},$$

and assuming the plasma model for zero frequency

$$\mathcal{F}_{\mathrm{plasma}} \approx \mathcal{F}_{\mathrm{Drude}} + \frac{-12.98 k_{\mathrm{B}}T}{2} \approx -7.88 \times 10^{-20}\mathrm{J}.$$

3.1.4 Truncation of the vector space

The truncation of the vector space is described in more detail in Hartmann, "Casimir effect in the plane-sphere geometry: Beyond the proximity force approximation", phd thesis (2018). You can either specify the dimension of the vector space using `--ldim`, or you choose the vector space using the parameter `--eta`:

$$\ell_{\mathrm{dim}} = \max\left(20, \lceil \eta R/L \rceil\right).$$

The estimated error due to the truncation of the vector space depending on eta is given by:

numerical error	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}
eta	2.8	4	5.2	6.4	7.6	8.8	10

3.1.5 Other options

The computation of the matrix elements of the round-trip operator contain an integration. The desired relative error for this integration can be set using `--iepsrel`. The default value of 10^{-8} should be sufficient for almost all purposes. If you want to compute the Casimir energy to very high accuracy, to 10^{-7} or better, you might want to set a smaller value.

3.2 casimir_logdetD

The program `casimir_logdetD` computes

$$\log\det\left(1 - \mathcal{M}^m(\xi)\right).$$

depending on m, ξ, R, and L The options `-L`, `-R`, `--ldim`, `--material`, and `--iepsrel` are the same as for the program `casimir`.

Besides `-L` and `-R`, further mandatory options are `-m` and `--xi`. The frequency given by `--xi` is expected in units of $(L+R)/c$. In addition, you can specify the algorithm used to compute the determinant with `--detalg`. Valid options are HODLR, QR, LU, and Cholesky.

A typical output looks like

```
$ ./casimir_logdetD -R 100e-6 -L 1e-6 -m 1 --xi 1.5
# ./casimir_logdetD, -R, 100e-6, -L, 1e-6, -m, 1, --xi, 1.5
# L/R    = 0.009999999999999998
# L      = 1e-06
# R      = 0.0001
# ldim   = 501
# epsrel = 1.0e-08
# detalg = HODLR
#
# L, R, ξ*(L+R)/c, m, logdet(Id-M), ldim, time
1e-06, 0.0001, 1.5, 1, -6.417998208796558, 501, 0.492086
```

Sometimes, it is useful to output the round-trip matrix in numpy format. If the environment variable `CASIMIR_DUMP` is set and `detalg` is not HODLR, the round-trip matrix will be saved to the filename contained in `CASIMIR_DUMP`. Also note that if `detalg` is Cholesky, only the upper half of the matrix is computed. Here is an example:

```
$ CASIMIR_DUMP=M.npy ./casimir_logdetD -R 100e-6 -L 1e-6 -m 1 --xi 1.5 --detalg LU
# ./casimir_logdetD, -R, 100e-6, -L, 1e-6, -m, 1, --xi, 1.5, --detalg, LU
# L/R    = 0.009999999999999998
# L      = 1e-06
# R      = 0.0001
# ldim   = 501
# epsrel = 1.0e-08
# detalg = LU
#
# L, R, ξ*(L+R)/c, m, logdet(Id-M), ldim, time
1e-06, 0.0001, 1.5, 1, -6.417998208796549, 501, 0.833277

$ ls -lah M.npy
-rw------- 1 hartmmic g-103665 7,7M Jan  4 15:16 M.npy

$ python
Python 3.6.5 | packaged by conda-forge | (default, Apr  6 2018, 13:39:56)
[GCC 4.8.2 20140120 (Red Hat 4.8.2-15)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> M = np.load("M.npy")     # load matrix
>>> dim,_ = M.shape          # get dimension
>>> Id = np.eye(dim)         # identity matrix
>>> np.linalg.slogdet(Id-M) # compute determinant
(1.0, -6.417998208796572)
```

3.3 cylinder

The program `cylinder` computes the Casimir interaction in the cylinder-plane geometry. The radius of the cylinder is given by `-R` and the separation between cylinder and plate is given by `-d`. Both lengths are expected in meters. At the moment, only perfect reflectors at zero temperature is supported.

This example computes the Casimir free energy for a cylinder of radius $R = 100\mu\mathrm{m}$ and a separation of $d = 100\mathrm{nm}$:

```
$ ./cylinder -R 100e-6 -d 100e-9
# R/d = 1000
# d = 1e-07
# R = 0.0001
# T = 0
```

(continues on next page)

(continued from previous page)

```
# lmax = 6000
# epsrel = 1e-08
#
# d/R, d, R, T, lmax, E_PFA/(L*hbar*c), E_D/E_PFA, E_N/E_PFA, E_EM/E_PFA
0.001, 1e-07, 0.0001, 0, 6000, -72220981652413.5, 0.500089151077922, 0.
↪499432943796732, 0.999522094874654
```

Here, L denotes the length of the cylinder. `E_D` and `E_N` correspond Dirichlet and Neumann boundary conditions, `E_EM` is the energy for the electromagnetic field, `E_EM = E_D + E_N`.

CHAPTER

FOUR

API DOCUMENTATION

The documentation of the API is available at `manual/api.pdf` or can be generated running

```
$ make doc
```

in the directory `src/`. The documentation will be generated in `src/doc/`. You need doxygen installed on your computer.