rijksuniversiteit groningen

Computation of the Casimir Force

Bachelor Project Thesis - Physics & Computing Science

Thomas den Hollander

Supervised by George Palasantzas & Jiri Kosinka

06-07-2018

Abstract

In this thesis, different methods for the computation of the Casimir force are discussed. The proximity force approximation, direct eigenmode summation, energy density integration and finally a stress tensor based approach. These methods are compared by their applicability to general geometries and their complexity. We discuss several ways of improving convergence and performance. The stress tensor based bounding surface integration technique is then implemented in `casimir-fdfd`, a program designed to compute the Casimir force for arbitrary geometry consisting of materials based on the Drude permittivity model. Results for the plate-plate and a sphere-plate geometries are obtained and discussed.

Contents

1 A force due to vacuum energy

In 1948 Hendrick Casimir predicted an interaction between perfectly conducting objects due to the electromagnetic zero point energy [2]. The resonating frequencies of standing waves in any geometry give rise to an energy, even in a vacuum. In his paper, Casimir considers a conducting square plate in a box. As the plate moves through the box, the possible eigenfrequencies of the electromagnetic waves change, and with them the total energy changes too. The result is a force on the object.

Let us consider the simplest one-dimensional case. We take a 'box' of size L, and add two perfectly conducting 'plates', with a distance d between them, $d \ll L$. On a perfectly conducting surface, there is no electric field: $\mathbf{E} = \mathbf{0}$. This determines the boundary conditions and limits the number of possible standing waves. By doing a Fourier transform on any wave, we can decompose it as a sum of solutions to the harmonic oscillator problem. This is called the frequency basis for the field. The energy of a harmonic oscillator is given by $U(n) = \hbar\omega(n + \frac{1}{2})$ [5]. Here ω is the frequency of the oscillation and n is the number of the oscillation. Notice how even for the ground state ($n = 0$), there is an energy, $U_0 = \frac{\hbar\omega}{2}$. The total vacuum energy is therefore given by the sum of all eigenfrequencies

$$U = \frac{\hbar}{2} \sum_{\omega} \omega \tag{1}$$

The energy for higher dimensions is found by the same formula, by summing over the eigenfrequencies in every dimension. To find the force on an object, we use the relation $\mathbf{F} = \frac{\partial U}{\partial \mathbf{r}}$, the derivative of the energy with respect to displacement of the object. Similarly, the torque is given by the derivative of the energy with respect to rotation of the object.

Although not all frequencies are solutions to the problem, there are still an infinite number of them, meaning the sum is divergent. To fix this, the contribution of a frequency is often multiplied by a function $f(\omega)$ that is close to 1 for smaller frequencies, but tends to 0 as $\omega \to \infty$ [2, 12]. Physically, it can be argued that the plates are not able to perfectly reflect waves at every frequency, so 'leaking' starts to occur as the frequency and with it the energy increases. One function that can be used is $f(\omega) = e^{-s\omega}$, where s a sufficiently small constant. The corrected energy formula becomes

$$U = \frac{\hbar}{2} \sum_{\omega} \omega e^{-s\omega} \tag{2}$$

For some special cases, like in the case of two perfectly conducting parallel plates, the eigenfrequencies can be computed analytically. Over a distance d, the possible wavenumbers are $k_n = \frac{\pi n}{d}$, where n is any positive integer. $n = 1$ corresponds to a wave without a knot, $n = 2$ has one knot, etcetera. The corresponding angular frequencies are $\omega_n = \frac{\pi c n}{d}$, where c denotes the speed of light in a vacuum. In combination with Equation 2 this gives the total energy. The force can then be found by computing the energy at two locations, and then dividing the difference in force by the difference in position, $\mathbf{F} \approx \frac{\Delta U}{\Delta \mathbf{r}}$. The force computed via this method[1] is shown in Figure 1.

In his paper, Casimir used a fully analytic approach and found a pressure (force per unit area) given by

$$P(d) = -\frac{\pi^2 \hbar c}{240 d^4} \tag{3}$$

This result is used in the proximity force approximation to calculate the Casimir force for more geometries.

2 Proximity force approximation

For a long time, the computation of the Casimir energy/force was limited to specific cases. Like the two-plates case, only simple and/or highly symmetric geometries could be considered [7]. One

[1] Source code available at `https://gist.github.com/ThomasdenH/d11a3fe376e1839c193764ce4621969e`

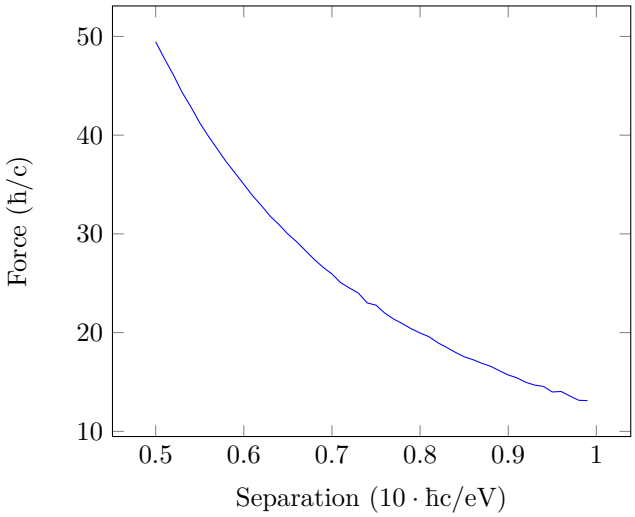

Figure 1: The Casimir force between two perfectly conducting plates plotted against their separation, calculated by taking the discrete derivative over the energy. As the distance between them decreases, the attractive force increases.

way to calculate the energy corresponding to more complex geometry was the use of the proximity force approximation (PFA). First described by B. Derjaguin in 1934 [3], it approximates two bodies as consisting of many flat parallel plates. This makes the use of Equation 3 possible. As long as the bodies are smooth, very close to each other and their surfaces are nearly parallel, the main contribution to the force between them is due to immediately opposite surfaces, and less due to the rest of the geometry. Because of this, the energy is estimated as the sum of the parallel plate energies, which can be found analytically. In particular, if the energy per unit area at a distance d, $U_{\text{density}}(d)$, is given, then the PFA gives the total energy

$$U = \int U_{\text{density}}(d)\,\mathrm{d}\sigma \tag{4}$$

Here $\mathrm{d}\sigma$ is the infinitesimal scalar area. I.e. it is a surface integral of the energy per unit area. Similarly, the force is given by an integral of the pressure over the surface of one of the objects:

$$\mathbf{F} = -\int P(d)\,\mathrm{d}\mathbf{S} \tag{5}$$

Here $\mathrm{d}\mathbf{S}$ is the surface integral normal vector, and P the pressure along this vector.

2.1 A PFA example

As an example of how the PFA can be used, we calculate the force on an example geometry shown in Figure 2. It consists of slices of two cylinders, oriented perpendicular to each other. Their radius is given by R. The origin is located on the centre of the line along which the distance between the object is the smallest. This minimal distance is called z_0. z gives the position along this line, and x and y give the position parallel to either of the cylinders. We integrate the pressure over the planar surface between them. We find that the distance between the surfaces, perpendicular to the x-y-plane, is given by

$$d(x, y) = z_0 + 2R - \sqrt{R^2 - x^2} - \sqrt{R^2 - y^2} \tag{6}$$

When we integrate over the planar surface, we should also parametrize the surface element. Instead of using $\mathrm{d}x\mathrm{d}y$, as we would on a flat surface, we have $\sqrt{\mathrm{d}x^2 + \mathrm{d}z^2}\mathrm{d}y$. Using $R^2 = x^2 + z^2$, we

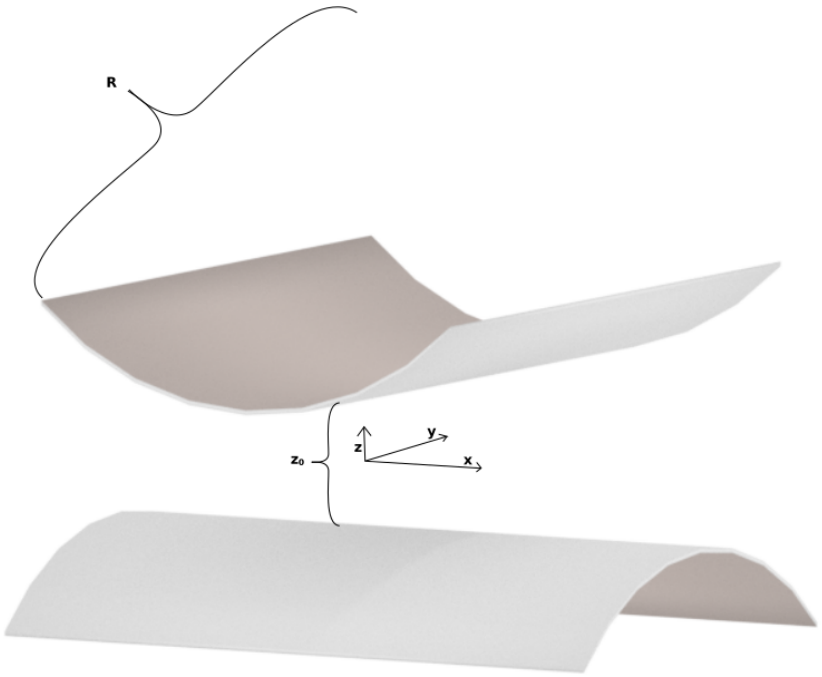

Figure 2: An example geometry for the proximity force approximation. The two objects are slices of cylinders. The slices are small compared to the radii, which means that the proximity force approximation can be used.

find that $\mathrm{d}z = \frac{x}{\sqrt{R^2-x^2}}\mathrm{d}x$. The planar surface $\mathrm{d}x\mathrm{d}y$ is therefore mapped to a cylinder surface $\mathrm{d}x\mathrm{d}y\frac{R}{\sqrt{R^2-x^2}}$. We should integrate over one surface and only use the distance to the other. This however breaks the symmetry of the problem, since we use an increasing surface element for one of the surfaces, but not for the other. The proximity force only works when the surfaces are nearly parallel, so the curvature along the integrated surface should not become too large. Here we assume that R is much larger than the dimensions of the surfaces (their size in the x and y plane is small compared to R), so $\frac{R}{\sqrt{R^2-x^2}} \approx 1$. This corresponds to the requirement that the surfaces are nearly parallel. By the same assumption, we can also do a Taylor expansion of d and discard higher order terms (third order and higher) of $\frac{x}{R}$ and $\frac{y}{R}$. Finally we use Equation 3 to obtain an expression for the force, arriving at

$$F = -\frac{\pi^2\hbar c}{240}\int_{-L}^{L}\int_{-L}^{L}\frac{xy}{z_0^4} + \frac{2x^3y}{3z_0^5R} + \frac{2xy^3}{3z_0^5R} + \frac{5x^3y^3}{9z_0^6R^2}\mathrm{d}x\mathrm{d}y \tag{7}$$

L defines the dimensions of the object; x and y can take values between $-L$ and L. We find that the force is given by

$$F(d) = -\frac{\pi^2\hbar c}{60}\left(\frac{L^2}{d^4} - \frac{4L^4}{3d^5R} + \frac{5L^6}{9d^6R^2}\right) \tag{8}$$

Here we have renamed z_0 to d since it describes the separation between the two objects.

2.2 Suitability for Casimir force computation

The assumptions that have to be made to use the PFA make it difficult to use for arbitrary geometries. Despite this, there is some none-analytical and more general use for the method. If we have two nearby surfaces, for which no direct equation is known, that can be described by heightmaps of the form $a(x, y) = \ldots$, the surface integration can very easily be done numerically, as demonstrated in Algorithm 1.

Algorithm 1 Pseudocode for a proximity force approximation of the attraction between two arbitrary surfaces given by height maps $a_1(x, y)$ and $a_2(x, y)$.

```
a ← base separation between surfaces
force ← 0
for x from 0 ... L, with dx step size do
    for y from 0 ... L, with dy step size do
        d ← a + a1(x, y) + a2(x, y)
        force ← force + dx * dy * π²ħc / 240d⁴
    end for
end for
```

This method has an optimal asymptotic efficiency. Any method that considers the height maps on N coordinates has to have complexity $\mathcal{O}(N)$ or worse. Since each iteration takes $\mathcal{O}(1)$ time, this method is $\mathcal{O}(N)$ overall. If we disregard the storage costs of the geometry itself, which could be given by an equation instead of stored in its entirety, the algorithm uses only $\mathcal{O}(1)$ memory.

When the approximations hold, the PFA has many desirable properties for a force calculation. The natural question is then, when do these approximations hold 'sufficiently'? What is the error in this computation? It is estimated that the error is in the range a/R [8], where a again is the separation and R is the effective radius of the geometry. However, there is no good theoretical basis for this estimation. The lack of an indication for the magnitude of the error is one of the disadvantages of PFA.

3 Direct computation of eigenmodes

Ideally, it would be possible to get an arbitrarily precise calculation of the force for any arbitrary geometry. In other words, the accuracy of the computation would be limited by computational resources and input parameter precision only, and not in principle by the algorithm in use. In theory, Equation 1 can be used directly for this purpose. To do this, the eigenmode frequencies of the geometry should be found.

One method to do this is called the Finite-Difference Frequency-Domain (FDFD) method [9, 15], and is used commonly for traditional electromagnetism problems. The method works in the frequency domain, and the computation quantizes derivatives to finite differences, hence its name. We use the Maxwell curl equations, given by [4]:

$$\boldsymbol{\nabla} \times \mathbf{E} + \frac{\partial \mathbf{B}}{\partial t} = \mathbf{0} \tag{9}$$

$$\boldsymbol{\nabla} \times \mathbf{B} - \mu\epsilon \frac{\partial \mathbf{E}}{\partial t} = \mu \mathbf{J} \tag{10}$$

$\mathbf{E}$ denotes the electric field, and $\mathbf{B}$ the magnetic field, and ϵ and μ are the relative electric and magnetic permittivity, respectively. Here $\mathbf{J}$ is an external current. Since we are considering standing waves, we use $\mathbf{J} = \mathbf{0}$. If we now insert a frequency basis for electromagnetic waves ($\mathbf{E} = \mathbf{E_0} e^{i(\mathbf{kr} - \omega t)}$, $\mathbf{B} = \mathbf{B_0} e^{i(\mathbf{kr} - \omega t)}$), we find

$$\boldsymbol{\nabla} \times \mathbf{E} = i\omega \mathbf{B} \tag{11}$$

$$\boldsymbol{\nabla} \times \mathbf{B} = -i\omega\mu\epsilon \mathbf{E} \tag{12}$$

For conducting materials and the frequencies where the Casimir effect is dominant, $\mu \approx 0$. In a vacuum, $\epsilon = \epsilon_0$ and $\mu = \mu_0$. For a perfect conductor, $\epsilon = \infty$. We now quantize these equations, which means that we give the components of the magnetic and electric field locations on a grid. Notice that in the Maxwell equations there is a derivative of one field on the left hand side, and another field on the right hand side. When quantizing derivatives, a difference between two neighbouring grid points is taken. Because of this, it makes sense if the electric and magnetic field points are not located on the same point, but instead alternate on a *Yee grid* [15]. One cell of a three-dimensional Yee grid is displayed in Figure 3.

The curl of a vector field can be quantized as follows:

$$(\boldsymbol{\nabla} \times \mathbf{A})_i = \left(\frac{\partial A_k}{\partial \epsilon_j} - \frac{\partial A_j}{\partial \epsilon_k} \right) \epsilon_i \tag{13}$$

$$\approx \left(\frac{A_{\mathrm{i,j,k\,+\,1}} - A_{\mathrm{i,j,k}}}{\Delta_j} - \frac{A_{\mathrm{i,j\,+\,1,k}} - A_{\mathrm{i,j,k}}}{\Delta_k} \right) \epsilon_i \tag{14}$$

In a Yee grid, the centres for both differences lie on the same new point in space. The curl of the electric field exactly lines up with the grid points of the magnetic field and vice versa. This means that the accuracy due to the Yee grid is much higher than that of a grid where all values lie on the same point in space. This discretized formula gives enough information to convert Equations 11 and 12 into the eigenvalue problem

$$A\mathbf{v} = \omega \mathbf{v} \tag{15}$$

Here the vector $\mathbf{v}$ contains all the components $E_{\mathrm{i,j,k}}$ and $B_{\mathrm{i,j,k}}$ of the EM field, and A is a matrix containing all discretized Maxwell relations. Any traditional method can be used to find the corresponding eigenvalues (eigenfrequencies).

3.1 Suitability for Casimir force computation

The direct FDFD method is a common method for traditional electromagnetic computations. However, for Casimir problems, it is much less suitable. For one, as opposed to traditional EM

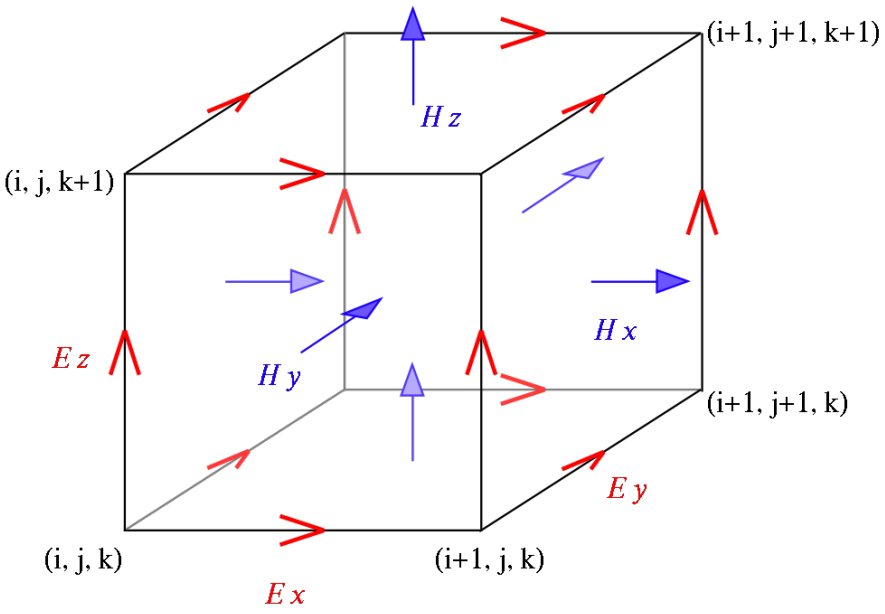

Figure 3: One unit of a Yee grid. The electric and magnetic fields originate at different points to suit the structure of the difference equations.[6]

calculations, where often only a few frequencies are wanted, the Casimir force requires calculating a very large number of frequencies. All frequencies contribute non-negligibly to the force, only to add up to a very small total. This makes numerical errors a problem in practice as well [12].

If the field is quantized into N unit cells, then the matrix A has N^2 components. The simplest approach of finding its eigenvalues has complexity $\mathcal{O}(N^3)$. However, properties of this particular matrix make more efficient methods available.

4 Fluctuation-dissipation theorem

Another way of calculating the zero point energy (and via its derivative the Casimir force) is using the photon Green functions. The fluctuation-dissipation theorem gives us a way to describe the vacuum energy as a response to an infinite number of delta function currents. The Green function is defined in the following way [12]:

$$\left[\left(\boldsymbol{\nabla}\times\frac{1}{\mu(\omega,\mathbf{r})}\boldsymbol{\nabla}\times\right)-\omega^2\epsilon(\omega,\mathbf{r})\right]\mathbf{G}_k^E(\omega,\mathbf{r},\mathbf{r}')=\delta^3(\mathbf{r}-\mathbf{r}')\hat{\mathbf{e}}_k \tag{16}$$

This equation is derived from Equations 9 and 10, where we use the external current $\mathbf{J}=\delta^3(\mathbf{r}-\mathbf{r}')\hat{\mathbf{e}}_k$, the Dirac delta function on one component of the field. $\mathbf{r}$ is the space coordinate, k gives the polarization of the incoming current. The Green function itself, $\mathbf{G}_k^E$, is therefore the electric field in response to $\mathbf{J}$. Since the Green function is already a three-component vector (the field has a vector at each location in space), the term Green tensor is used for G_{ij}^E, where j gives the polarization and i the component of the field. The magnetic field has its own Green function:

$$\left[\left(\boldsymbol{\nabla}\times\frac{1}{\epsilon(\omega,\mathbf{r})}\boldsymbol{\nabla}\times\right)-\omega^2\mu(\omega,\mathbf{r})\right]\mathbf{G}_k^B(\omega,\mathbf{r},\mathbf{r}')=\delta^3(\mathbf{r}-\mathbf{r}')\hat{\mathbf{e}}_k \tag{17}$$

These tensors are used in the electric and magnetic field correlation functions $\langle E_j(\mathbf{r})E_k(\mathbf{r}')\rangle$ and $\langle B_j(\mathbf{r})B_k(\mathbf{r}')\rangle$. These correlation functions at $\mathbf{r}=\mathbf{r}'$ in turn are used for the energy density, which

can be integrated over space and all frequencies to obtain the total energy. Computationally, this new formulation does not yet have an advantage compared to the direct summation of eigenmodes. However, we can now apply a trick. We integrate the energy density in the domain $\omega \in [0, \infty\rangle$, but due to the causality of the Green functions and as a result from complex analysis, we can instead integrate the formula over imaginary frequencies $\omega \in [0, i\infty\rangle$ [7]. For convenience, we define $\omega = i\xi$, where ξ is real and range from 0 to ∞ as before. The total energy is given by [12]

$$U = \int_0^\infty \int \frac{1}{2}\left(\frac{\mathrm{d}(\xi\epsilon)}{\mathrm{d}\xi}\langle|\mathbf{E}|^2\rangle_{i\xi} + \frac{\mathrm{d}(\xi\mu)}{\mathrm{d}\xi}\langle|\mathbf{B}|^2\rangle_{i\xi}\right)\mathrm{d}\mathbf{r}\mathrm{d}\xi \tag{18}$$

The correlation functions are

$$\langle|\mathbf{E}(\mathbf{r})|\rangle_{i\xi} = \frac{\hbar}{\pi}\xi^2 \operatorname{tr}\big(\mathbf{G}^E(i\xi, \mathbf{r}, \mathbf{r})\big) \tag{19}$$

$$\langle|\mathbf{B}(\mathbf{r})|\rangle_{i\xi} = \frac{\hbar}{\pi}\xi^2 \operatorname{tr}\big(\mathbf{G}^B(i\xi, \mathbf{r}, \mathbf{r})\big) \tag{20}$$

Here tr denotes the trace of a matrix. The switch to imaginary frequencies is called Wick-rotation. Its advantage is that the previously oscillating Green functions proportional to $e^{i\omega t}$ now suddenly decrease exponentially; as $e^{-i\xi}$. The contribution from each frequency is now not an oscillating, but a smoothly decreasing function, meaning that only a few points are needed to evaluate the integral. This reduces the time complexity by a big factor.

Wick-rotation would not be possible if the frequency-dependent quantities in Equations 16 and 17 were not defined for imaginary frequencies. We can still use $\mu = 0$, but we have to find a value for $\epsilon(i\xi)$. Using the Kramers-Kroning relation this function can be defined in terms of observables [14]. One model interpolating the permittivity is the Drude model. Using this function, every quantity is defined for imaginary frequencies as well and thus Wick-rotation is possible.

4.1 Conjugate Gradient method

To obtain the force, Equations 16 and 17 need to be inverted to obtain the Green functions. If we quantize space as before in an FDFD method, we have a linear problem in the form $A\mathbf{x} = \mathbf{b}$. Furthermore, the matrix A is sparse, which means we can do much better than the worst case $\mathcal{O}(N^3)$.

The steepest descent method is one iterative way to do this. In each iteration, the residual $\mathbf{r} = A\mathbf{x} - \mathbf{b}$ is obtained. This is a vector along which $\mathbf{x}$ should move to decrease the next residual the most. When $\mathbf{r} = \mathbf{0}$, this means that $\mathbf{x} = A^{-1}\mathbf{b}$, and we have the solution. In other words, each iteration we compute $\mathbf{r}$ and a scalar α, and set $\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{r}$. After a certain number of iterations, for example when $\|\mathbf{r}\|$ is small enough, we stop. The steepest descent method is not optimal though. Every iteration, $\mathbf{r_{i+1}}$ is orthogonal to $\mathbf{r_i}$. However, $\mathbf{r_{i+2}}$ can again move in the same direction as $\mathbf{r_i}$, creating a zig-zag pattern. There is a way to choose the direction vectors in such a way that it solves perfectly for one dimension each time [13], while keeping other directions intact. This means that after N iterations, in theory we should have reached the exact solution. The worst case complexity is again $\mathcal{O}(N^3)$, but the convergence is often much better. How much better depends on the specific matrix A. With the conjugate gradient method, we do not use the direction $\mathbf{r}$ directly, but instead compute the direction vector $\mathbf{p_{i+1}} = -\mathbf{r_{i+1}} + \beta_k\mathbf{p_k}$. β_k is a scalar factor that is computed for each $\mathbf{p_k}$. This way, each $\mathbf{p}$ is guaranteed to follow $i \neq j \Rightarrow \mathbf{p_i}^T A\mathbf{p_j} = 0$. This requirement is called A-conjugacy and it guarantees that the initial residual can be reduced one component per iteration, without successive iterations adding error for these components again.

4.2 Suitability for Casimir force computation

Pseudocode for the energy density method based on Green functions is given in Algorithm 2. To obtain the total energy, we need to integrate the energy density over space and all frequencies. Because of the Wick-rotation, the latter only takes a handfull of samples. The space integral

takes N samples, for every point in space. Each of these samples takes the number of iterations from the conjugate gradient method times the complexity of one iteration. In an iteration, we do vector and sparse matrix multiplication, making it $\mathcal{O}(N)$. The gradient descent method converges much quicker than in N iterations. In a three-dimensional cube with volume $\mathcal{O}(N)$, the sides are $\mathcal{O}(N^{\frac{1}{3}})$. The effect of a changed value propagates by one unit each iteration, so we might expect the iterations needed for convergence go as $\mathcal{O}(N^{\frac{1}{3}})$, since then a change has propagated through the entire space. This intuition turns out to be correct [7]. The total complexity of this method therefore becomes $\mathcal{O}(N) \cdot \mathcal{O}(N^{\frac{1}{3}}) \cdot \mathcal{O}(N) = \mathcal{O}(N^{2\frac{1}{3}})$. This is an improvement compared to the direct eigenmode computation, and it doesn't reflect the additional constant time reduction given by Wick-rotation.

Algorithm 2 Pseudocode for the energy density integration method of computing the Casimir energy. Note that the matrices should not be constructed explicitly, but a sparse representation should be used instead.

```
energy ← 0
for ξ ∈ [0, ∞⟩ do
    A^E ← [(∇ × (1/μ(ω,r)) ∇×) − ω²ε(ω, r)]
    A^B ← [(∇ × (1/ε(ω,r)) ∇×) − ω²μ(ω, r)]
    for r on grid do
        G^E ← conjugate_gradient(A^E, r)
        G^B ← conjugate_gradient(A^B, r)
        energy ← energy + energy_density(G^E, G^B) · Δξ · Δr
    end for
end for
```

5 Stress tensor integration

So far we have computed the force via the formula $\mathbf{F} = \frac{\partial U}{\partial \mathbf{x}}$. For a discrete derivative in three dimensions, this means computing the energy 1 + 3 times. There is a way to move the derivative earlier in the computational process, reducing the overall complexity of the problem. This alternative formulation uses the stress tensor [12]:

$$\begin{aligned}\langle T_{ij}(\mathbf{r}, i\omega)\rangle =& \mu\left[\langle B_i(\mathbf{r})B_j(\mathbf{r}))\rangle - \frac{1}{2}\delta_{ij}\,\mathrm{tr}\langle B_i(\mathbf{r})B_j(\mathbf{r}))\rangle\right] \\ &+ \epsilon\left[\langle E_i(\mathbf{r})E_j(\mathbf{r}))\rangle - \frac{1}{2}\delta_{ij}\,\mathrm{tr}\langle E_i(\mathbf{r})E_j(\mathbf{r}))\rangle\right]\end{aligned} \tag{21}$$

δ_{ij} is the Kronecker delta. The stress tensor is defined at every point in space. Integrated over a surface, it gives the force due to geometry on one side of the surface on that on the other side. When integrated over a closed surface around an object, the force of the object on itself cancels out and the total force of the object is found. Crucially, the space integral needed to compute U is now replaced by a surface integral for $\mathbf{F}$. The surface itself does not matter, the integral will always equal the total force on the material (defined by $\epsilon(\omega, \mathbf{x})$ and $\mu(\omega, \mathbf{x})$) inside it. For simple objects a bounding box suffices, for more complicated geometry a different method must be used. The formula for the force now becomes

$$\mathbf{F} = \int_0^\infty \oiint \langle \mathbf{T}(\mathbf{x}, i\omega)\rangle \mathrm{d}\mathrm{d}\xi S \tag{22}$$

Here $\mathrm{d}\mathbf{S}$ is the surface element vector, its direction is the normal vector of the surface and its length is equal to the area of the infinitesimal surface. $\oiint$ denotes an integral over a closed surface.

The final procedure is nearly identical to Algorithm 2, but the integral is replaced by a bounding surface integral instead.

The complexity of computing G_{ij}^E and G_{ij}^B is still the same, using the conjugate gradient method, $\mathcal{O}(N^{1\frac{1}{3}})$. However, we now don't have to do a volume integral with complexity $\mathcal{O}(N)$, but instead we can do a surface integral. A two-dimensional surface integral sums only over $\mathcal{O}(N^{\frac{2}{3}})$ of grid points, which reduces the total complexity to $\mathcal{O}(N^2)$. This is also the complexity of the final program, but there are still some optimizations we can do to reduce running time by constant factors.

5.1 Error forces

In theory the integral over a bounding surface should eliminate the contribution from the object itself. However, when quantizing the field and integral, these contributions will not exactly cancel and instead cause significant errors. The Yee-grid gives much more accurate results than the more obvious choice of placing all values on the same point in space, but it also creates a slight asymmetry. This, combined with the imperfect surface integral result due to its discrete variant is a major contributor to the error in the force. As the grid unit size $\Delta x \to 0$, the difference vanishes, but the running time will quickly become too long. One way to eliminate a large part of the error is to calculate the (theoretically zero) contributions to the force of objects on themselves. After computing the force for the entire geometry, it is now computed over the same surface, but now the geometry contains only one object at a time. The process is illustrated using Algorithm 3. The result of this technique is that convergence to the real solution happens much quicker for larger Δx [7]. It does impose a constant increase, since now these error forces need to be computed for every object, but this is often a more efficient increase in accuracy than lowering the grid unit size.

Algorithm 3 Pseudocode of the process of subtracting error forces of objects on themselves.

```
force ← base force with all geometry
for object in geometry do
    error_force ← force with only object as geometry
    force ← force − error_force
end for
```

5.2 Cosine basis

To integrate over the surface surrounding the object, the stress tensor should be obtained for every point on it. However, depending on the geometry, the tensor can be quite smoothly varying over space. This makes it possible to use extrapolation instead of computing the stress tensor for each point in space. But the nature of the problem allows for a more elegant way to make use of this property. The delta function excitation currents form a basis for the surface integral. However, since any orthonormal basis can be used, it is possible to purposefully distribute the force contribution unevenly between basis elements [10]. It is then necessary to compute the force only for a subset of those basis elements, because the contribution from others becomes negligible. In particular, a cosine expansion of the surface currents can be used. The one-dimensional cosine basis function is given by

$$f_n(x) = \sqrt{\frac{2}{L}} \cos\left(\frac{n\pi x}{L}\right) \tag{23}$$

Here L is the domain of f_n and x is the coordinate along this line. n can be 0, or any positive integer. The two-dimensional version of this equation is simply the product $f_{n_x}(x) \cdot f_{n_y}(y)$, where L is replaced by L_x and L_y, the lengths in either direction. Finally, we can define $f_{n_x,n_y}(\mathbf{r})$, where x and y are two perpendicular directions on the surface, and $f_{n_x,n_y}(\mathbf{r}) = f_{n_x}(x(\mathbf{r})) \cdot f_{n_y}(y(\mathbf{r}))$

when $\mathbf{r}$ is on the surface and 0 otherwise. We can now change Equation 16 as follows:

$$\left[\left(\boldsymbol{\nabla} \times \frac{1}{\mu(\omega, \mathbf{r})}\boldsymbol{\nabla}\times\right) - \omega^2\epsilon(\omega, \mathbf{r})\right]\mathbf{G}^E_{k,n_x,n_y}(\omega, \mathbf{r}, \mathbf{r}') = f_{n_x,n_y}(\mathbf{r}')\hat{\mathbf{e}}_k \tag{24}$$

The result is that we are now measuring the response to a surface current given by f_{n_x,n_y}. We can then integrate this over the domain of the basis function and use the stress tensor as before. The total force is given by the sum of all the expansion contributions.

The advantage of this step is clear; the cosine expansion converges rapidly to the real solution as n_x, n_y increases. The contributions from higher n are therefore negligible. Computing the force for each element would yield the exact answer, but after the first few terms the expansion contributes less to the total than the error due to the other program parameters (for example the grid cell size). The contribution from a term decreases as n^{-4}, although the required amount differs for each geometry. A rough and more detailed material would require more terms to be described accurately than a smooth flat plate. One possibility is to start adding contributions from $n_x = n_y = 0$ upwards, and stop when the contribution has decreases below a certain value, either absolutely or relative to the first term.

5.3 Parallel computing

The nature of the system allows for parallel processing, potentially speeding up the program significantly. When using the delta function expansion of the surface integral, the calculation for each basis element is completely independent. This means that the computation can easily be split over $\mathcal{O}(N^{\frac{2}{3}})$ cores. When the cosine expansion is used instead, the computation time can be reduced even more, but now the tasks are not as trivially divided between CPU's. Like the sequential case, the cosine expansion should start at the lowest n and stop when the contribution is small enough. A variable *min_contrib* that stores the minimal contribution thus far can be used for this purpose. When a core is looking for work, it should start the computation for the next lowest n, unless *min_contrib* is lower than a certain threshold. After the computation is done, it should update *min_contrib*. With this method, the surface integral can be distributed evenly, while still taking advantage of the cosine expansion.

The conjugate gradient algorithm is inherently sequential. This means that the complexity from $\mathcal{O}(N^{\frac{1}{3}})$ iterations is inherent to this method. However, each iteration we can do better than $\mathcal{O}(N)$. This complexity is due to two computations: Matrix multiplication ($A\mathbf{x}$) and vector dot products ($\mathbf{x}\cdot\mathbf{y}$). The electric and magnetic field operators are sparse and have a fixed number of elements in each row. This allows matrix-vector multiplication to be performed for each element in the vector in parallel. This reduces its sequential time complexity to $\mathcal{O}(1)$. Vector-vector multiplication is more difficult. We can start by computing a new vector by element-wise multiplication in parallel. Then to compute the sum of the resulting entries we can do a pairwise reduction. In parallel, we sum elements at odd and even indices together to reduce a vector with N elements to $N/2$ elements. If there is an odd number of elements, we can simply append 0 to the vector to make it even without changing the total. We repeat this process until there is only one element left, this will be the total sum. This will happen after $\log_2(N)$ iterations. The result is that each iteration of the conjugate gradient method has a time complexity of $\mathcal{O}(\log N)$, which makes the time complexity of the entire program $\mathcal{O}(N^{\frac{1}{3}} \log N)$, assuming an arbitrarily large number of cores are available.

5.4 Recursive frequency integration

To integrate the force integrand $\frac{\mathrm{d}\mathbf{F}}{\mathrm{d}\xi}$ a recursive algorithm can be used. Since the function is very smooth, a particularly simple algorithm is sufficient. Pseudocode for this method is given in Algorithm 4. For this, the computation is performed for the start and end frequencies, then for the frequency in the middle. This last value is compared to the average value of both ends, and if the difference is small enough, no recursion is employed. If the difference is too large, the process

will be repeated at both sides. For most materials the main contribution to the force will come from low frequencies, so the first value can be used as an absolute reference point for the error.

Algorithm 4 Pseudocode for recursive integration of the force integrand over a frequency domain.

```
function RECURSIVE_INTEGRATE(start_freq, end_freq, start_val, end_val)
    middle_freq ← (start_freq + end_freq)/2
    middle_val ← force_for_freq(middle_freq)
    interpolated ← (start_val + end_val)/2
    if ‖middle_val - interpolated‖ < error then
        return (1/4 start_val + 1/2 middle_val + 1/4 end_val) · (end_freq − start_freq)
    else
        return recursive_integrate(start_freq, middle_freq, start_val, middle_val) +
recursive_integrate(middle_freq, end_freq, middle_val, end_val)
    end if
end function
```

6 The program

A program called `casimir-fdfd` was created implementing most of the theory described earlier. The source code is available on GitHub[2] and the binary on crates.io[3]. The program accepts the path to a JSON file specifying the run configuration, the full explanation of which can be found in Section 10 along with usage information. It uses the stress tensor based approach with a cosine expansion basis. The surface integral is computed for each of the faces of the bounding box in parallel, with a separate thread for each face, and the error forces described in Section 5.1 are subtracted for quicker convergence. The world can contain spheres and boxes with either predefined or custom materials. Sanity checks will automatically be performed, for example making sure bounding boxes don't intersect. After this, the force integrand per frequency and finally the total force will be displayed.

Several choices have been made to promote program correctness and efficiency. First, the code was written in the Rust programming language. This means the code is strongly typed, and many invariants can be checked at compile-time. To prevent memory allocations the vector operations consume one of the vectors to recycle its memory. Rust's borrow checker makes sure the old value can't be used afterwards. The opposite is also true; a vector that is passed via an immutable reference is guaranteed to be invariant. Using two temporary vectors, an iteration of the conjugate gradient method doesn't require any additional allocations, which is essential since a vector uses $\mathcal{O}(N)$ memory, potentially causing major allocations. Another method of ensuring correctness is the use of testing and continuous integration. The basic components are mostly covered by unit tests, making sure they behave as expected. Currently, the code has 60% test case coverage[4], although measurement is somewhat imprecise.

The program is written in such a way that it can easily be extended. This includes adding new shapes (implemented via an enum) and materials (implemented as an interface with a single function). This is desirable because developments in the field could require new properties of the program or a researcher might want to explore different geometries. Additionally, since much development in computational methods for computing the Casimir force has occurred so recently and rapidly, there will likely be many improvements that can be made to improve the simulation. The code has been divided into logical modules, commented and documented to make functionality clear to future users.

[2] `https://github.com/ThomasdenH/casimir-fdfd`

[3] `https://crates.io/crates/casimir-fdfd`; if Cargo is installed, the program can be installed using `cargo install casimir-fdfd`.

[4] Viewable on `https://coveralls.io/github/ThomasdenH/casimir-fdfd`.

7 Results

First we consider the simple two-plate case[5]. In a grid of size $(50, 50, 50)^T$ two plates, oriented perpendicular to the z axis, were placed. Their dimensions are $15 \times 15 \times 1$ in the x, y and z directions, respectively. They are made of the predefined material `gold`, following the Drude model with $\omega_p = 7.79, \omega_t = 48.8$. These values are based on [14]. The force was integrated between frequencies $\omega \in [0.02, 1]$ $c/\Delta x$, where Δx denotes the grid unit size. The total force on plate 1 is $\mathbf{F_1} = (-0.64, -0.64, -15)^T$ $\hbar c/\Delta x$, while plate 2 experiences $\mathbf{F_2} = (0.84, 0.84, 16)^T$ $\hbar c/\Delta x$. The force integrand is plotted in Figures 4 and 5 for plate 1 and 2 respectively. The curve is very smooth due to the non-oscillating imaginary time Green functions. This makes sure that only 15 measurements were needed to integrate with the desired accuracy. The force integrand rapidly decreases as the frequency increases. This effect is due to two factors; first, the Wick-rotated Green functions decrease exponentially with the frequency. Secondly, the permittivity calculated using the Drude model decreases with higher frequencies. Since there are two objects in the world, the force should be equal and opposite. Therefore we can use the sum of both as an error estimate. Written explicitly, the relative error is roughly

$$2\frac{\|\mathbf{F_1} + \mathbf{F_2}\|}{\|\mathbf{F_1}\| + \|\mathbf{F_2}\|} \tag{25}$$

Using this formula, we find that the relative error is 0.08. Another measure of error is due to a different symmetry of this particular geometry. Because the world is periodic, there should be no net force over the x and y axis. This means that any deviation from 0 in these directions is due to imperfections in the simulation. We find that $\left|\frac{F_{1,x}}{F_{1,z}}\right| = \left|\frac{F_{1,y}}{F_{1,z}}\right| = 0.04$, and $\left|\frac{F_{2,x}}{F_{2,z}}\right| = \left|\frac{F_{2,y}}{F_{2,z}}\right| = 0.05$.

Another geometry we consider is the sphere-plate case.[6] In a grid of size $(40, 40, 40)^T$, a plate with dimensions $14 \times 14 \times 1$ was placed, together with a sphere of radius 7. The objects are separated by a distance 5. Once again they are made of gold. We find a force of $(7.0 \times 10^{-2}, 7.0 \times 10^{-2}, -3.3)^T$ $\hbar c/\Delta x$ on the plate, and a force of $(1.1, 1.1, 2.2)^T$ $\hbar c/\Delta x$ on the sphere. We can use symmetry in the same way as we did for the plate-plate case to find the error for this geometry. Using Equation 25, we find an error of 0.7. We also find that $\left|\frac{F_{1,x}}{F_{1,z}}\right| = \left|\frac{F_{1,y}}{F_{1,z}}\right| = 0.02$ and $\left|\frac{F_{2,x}}{F_{2,z}}\right| = \left|\frac{F_{2,y}}{F_{2,z}}\right| = 0.5$. The force integrand plot for the plate can be seen in Figure 6. The force plot on the plate is very similar to that of the plate-plate sphere, although the error forces are smaller for small values of ξ. We see a smooth curve as before. The magnitude of the force is much larger for smaller frequencies, and therefore they cause the main contribution to the force.

For the force integrand plot for the sphere in Figure 7, we notice something curious. The error forces increase rapidly for lower frequencies and are on the domain $\xi < 3.2 \times 10^{-2}$ c/Δx even higher than the true force F_z. This causes the high relative error and is likely due to the larger dimensions of the sphere. It is possible that the force error is proportional to either the integrated surface or volume, causing the larger sphere to have a greater error than the plates. Methods of improving the error in our calculations are discussed in Section 8.

[5]The two plates configuration file can be found at `https://github.com/ThomasdenH/casimir-fdfd/blob/master/worlds/plates_medium.json`. The full output is shown in Section 11.1

[6]The sphere / plate configuration file can be found at `https://github.com/ThomasdenH/casimir-fdfd/blob/master/worlds/sphere_plate_medium.json`. The full output is shown in Section 11.2.

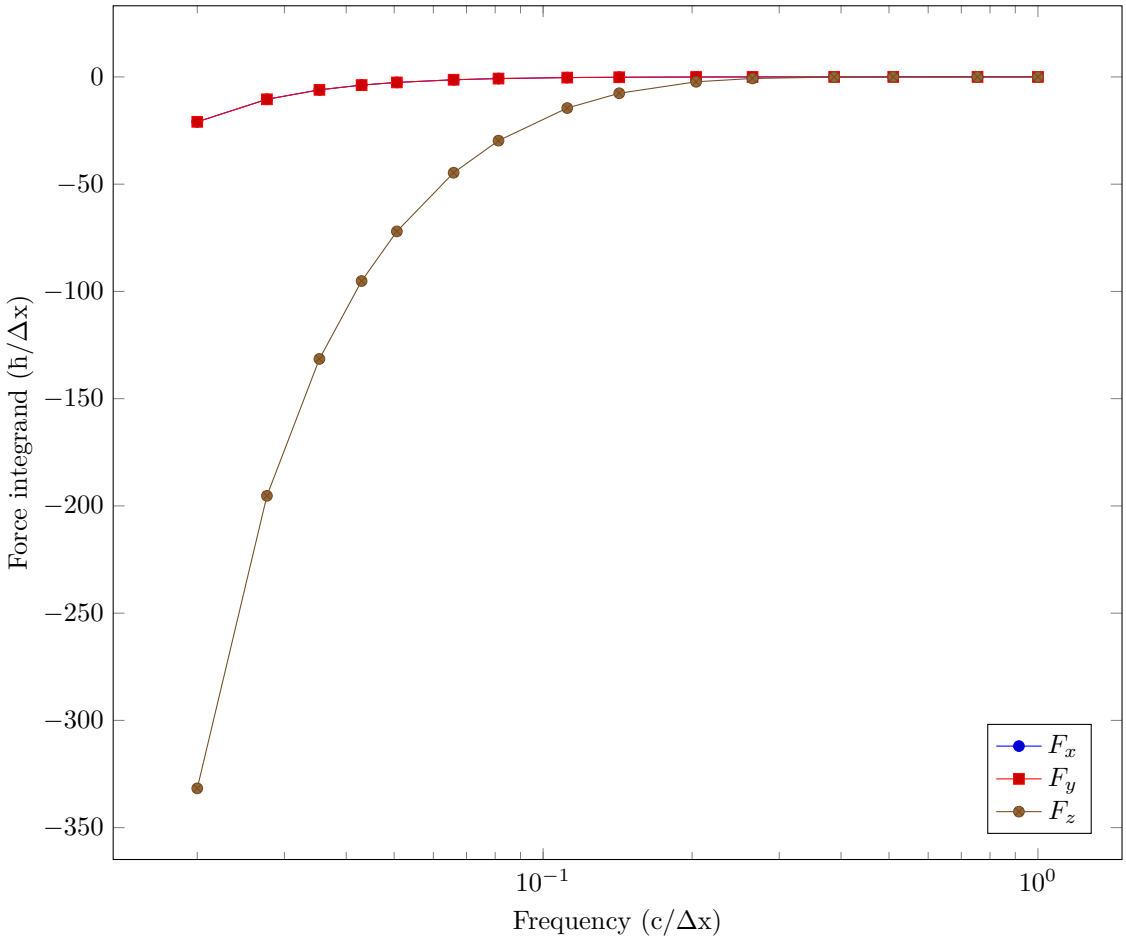

Figure 4: The computed Casimir force between two plates. The magnitude of the force rapidly decreases as the frequency increases. This is because of two reasons: the decreasing effect from the Green functions due to Wick-rotation and the decreasing permittivity of the material. Since the geometry is symmetric over the x and y axis, the force in these directions should be zero. Any deviation is an indication of the error in the calculation.

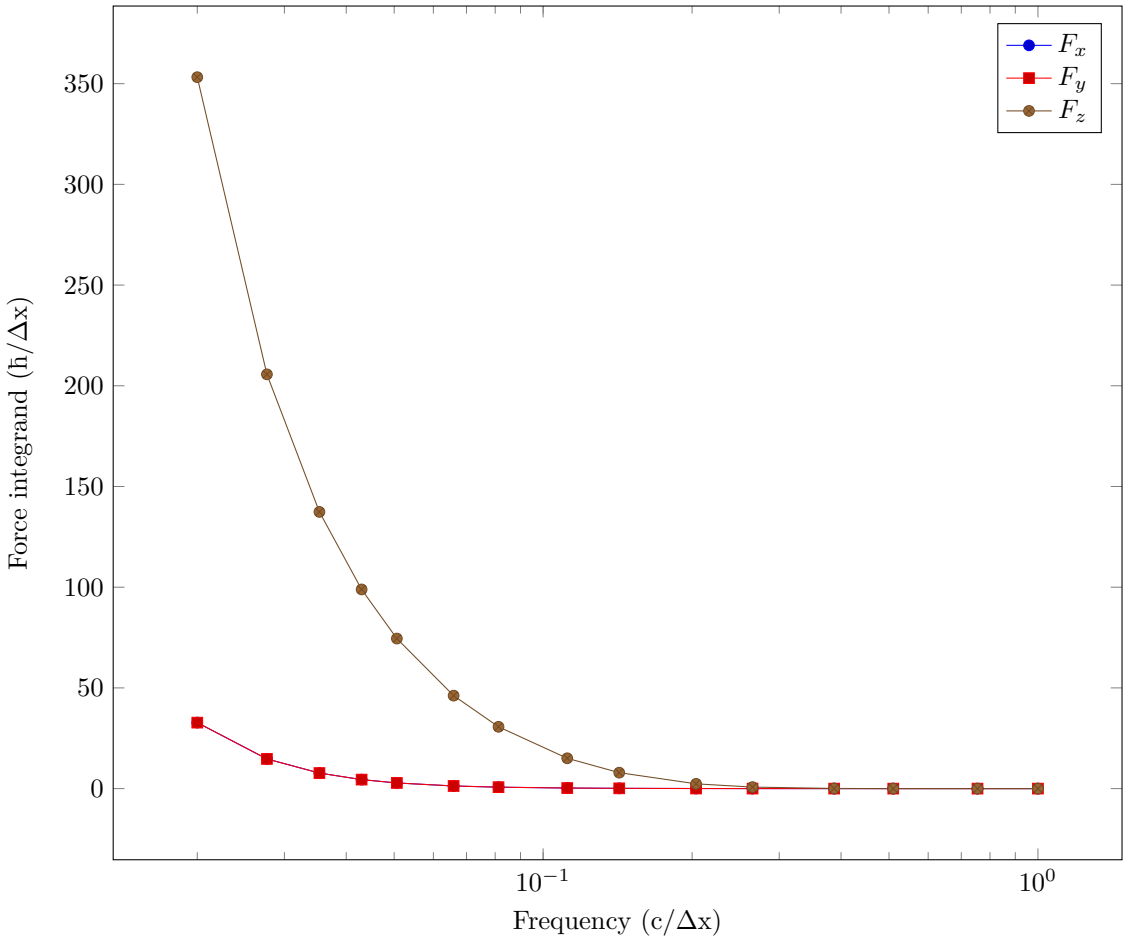

Figure 5: The computed Casimir force between two plates. This is the force on the same geometry as Figure 4, but on the other plate. The force is therefore expected to be equal and opposite.

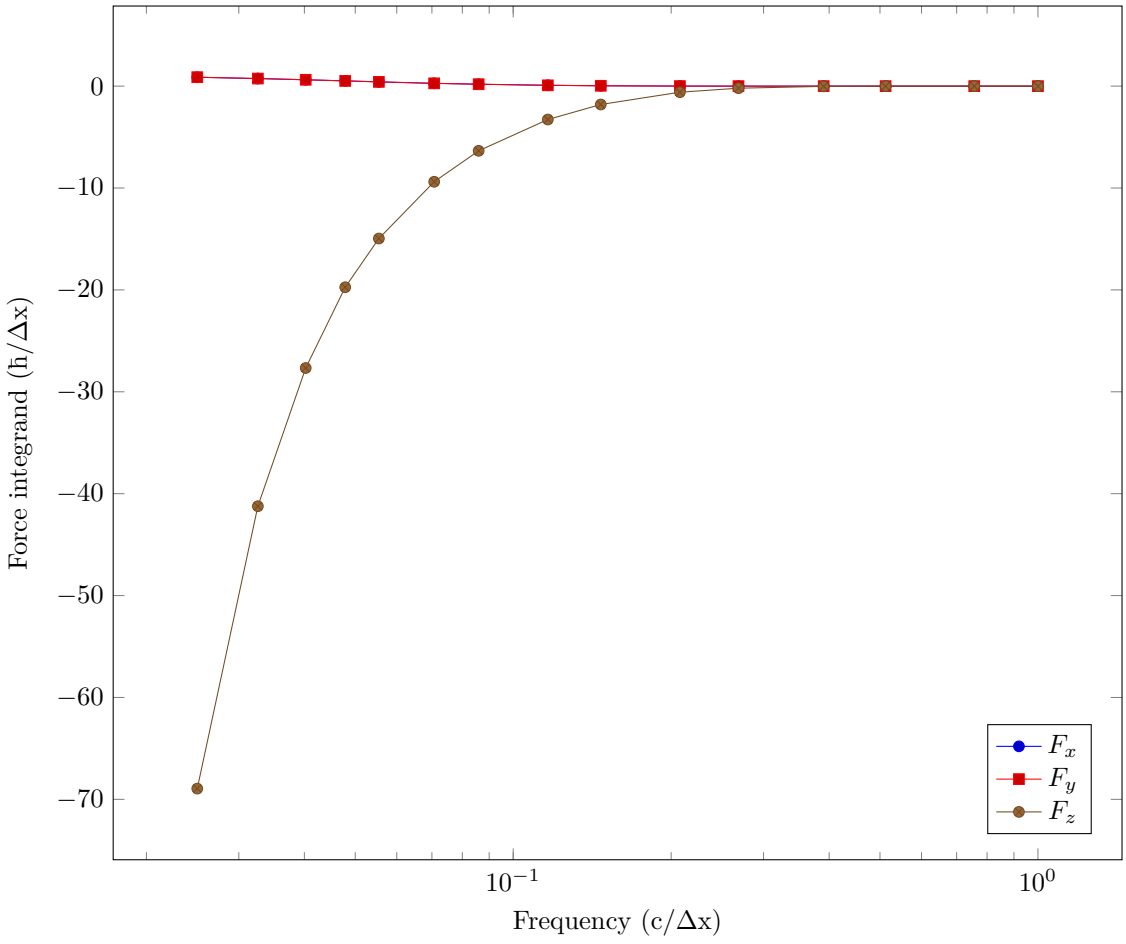

Figure 6: The computed Casimir force on the plate in the sphere-plate configuration. The force integrand F_z curves similarly to Figures 4 and 5, but the error forces are smaller for low frequencies.

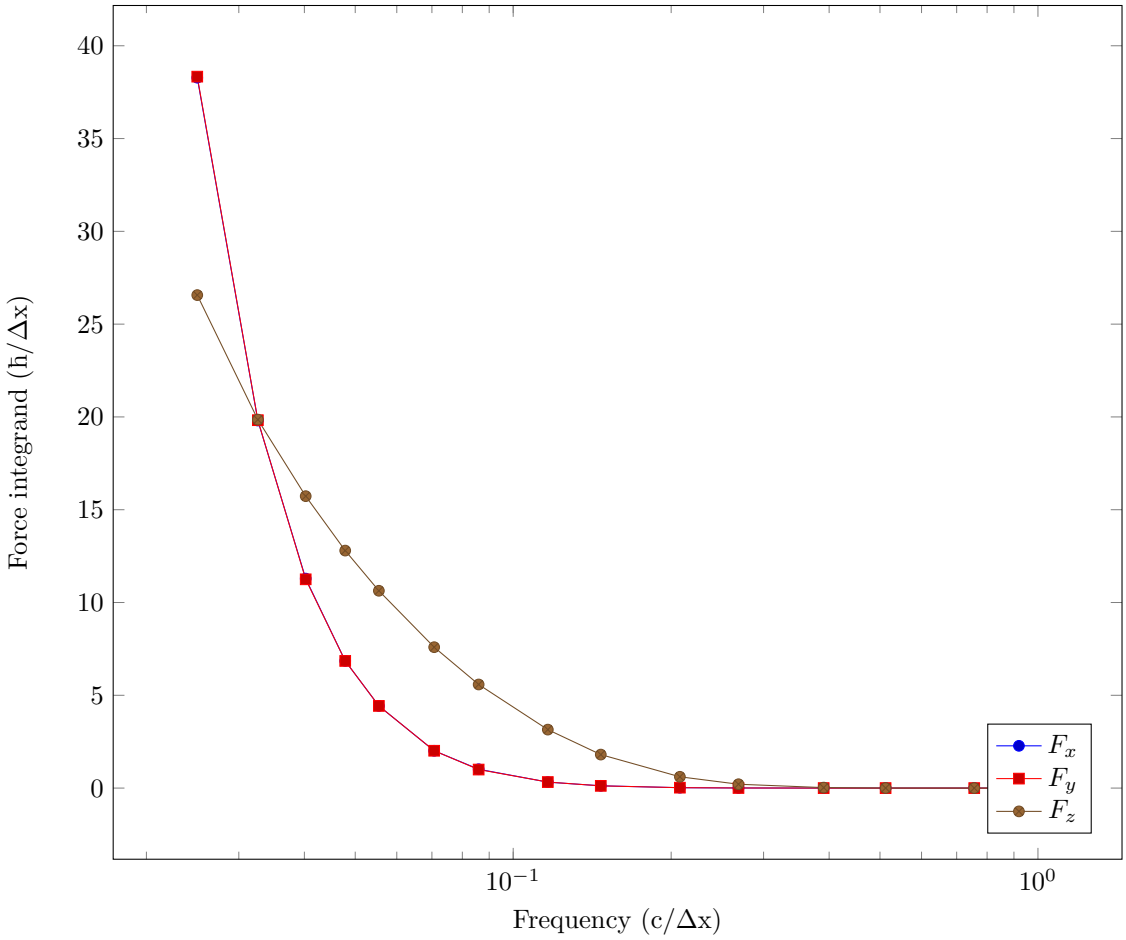

Figure 7: The computed Casimir force on the sphere in the sphere-plate configuration. The force integrand for the error forces F_x and F_y is large compared to F_z and even overtakes it at low frequencies. The force is expected to be opposite to that on the plate (Figure 6).

8 Discussion and future work

The results show that it is possible to compute the Casimir force for arbitrary geometry. While the FDFD method has advantages compared to other specialized techniques like the proximity force approximation and boundary element methods (it can deal with arbitrary permittivity fields), we find significant relative errors. To improve this error, the unit size can be decreased, but because of the time complexity this makes the running time quickly infeasible. Without further improvements, the error goes as $\mathcal{O}(\Delta x)$. This means that halving the error and thereby doubling the grid size in all directions multiplies the running time by $(2^3)^2 = 64$. A different way of reducing the error is the use of extrapolation. If the force is known for different grid unit sizes Δx, it can be extrapolated to $\Delta x \to 0$. This is an improvement that can either still be made to the program, or done manually after obtaining simulation results.

The periodic boundary conditions that are currently employed in the program are also not optimal. Instead, a perfectly matched layer can be used, a material on the boundary of the grid that absorbs incoming waves [16]. It can be implemented as simply a change in permittivity on the boundary of the domain and is equivalent to stretching the unit size towards the edge of the grid. Changing the boundary condition in this way can improve the convergence of the solution as wraparound waves are now impossible.

If the problem allows it, there are boundary element methods available that are much more efficient [11]. To use these, the objects should have discrete surface boundaries. Instead of quantizing space into a grid, it divides the surface itself into triangles. The edges of these triangles now form a basis along which surface currents run. The advantage of this technique is that the volume does not need to be quantized at all, only the surface does. Therefore, if the accuracy needs to be increased, this happens in two dimensions and not in all three. This means that the work complexity is a lot smaller than $\mathcal{O}(N^2)$. If the number of edges is called M, the total work complexity is $\mathcal{O}(M \log M)$.

The program currently supports boxes and spheres, but it is trivial to add more basic objects to the simulation. Object composition would open up interesting new possibilities as well, making it possible to compute forces for objects that are too complex to warrant their own primitive type. Adding the ability to load triangle meshes into the program would also be a solution here. Some geometry could require a different method of generating bounding surfaces. When two objects interlock this could cause local attraction but apparent global repulsion. Currently this behaviour can not be modelled in the program because the bounding boxes intersect. If the program allowed this, the force would be calculated for slices of other objects as well. Since objects are quantized in a grid, finding a discrete bounding surface is not too difficult, though features like holes could complicate things. An additional problem with these bounding surfaces is that the cosine expansion is not as straightforward as with a bounding box. This could mean that a more general stress tensor interpolation method should be used instead, like quadrature integration.

Currently only materials based on the Drude permittivity model are supported. Addition of different materials, such as a perfect electrical conductor, would be interesting. This would also allow for direct comparisons with many analytically obtained results and thereby give a way to further corroborate the correctness of the program. A similar expansion would be to move away from the $\mu = 0$ assumption, since magnetizable materials cause interesting behaviour as well, like a repulsive Casimir force [1].

9 Conclusion

Of all the methods that are explained here, the stress tensor approach has the most desirable properties. Its time complexity is lowest, it is parallelizable and there are many additional techniques to improve performance and convergence. This method was implemented in the program `casimir-fdfd`, together with most of the discussed improvements. The program is able to compute the Casimir force for boxes and spheres, but it is easily extensible to allow for other objects as well. It can load configurations from special files and is easy to install and use. To demonstrate

and test the program, we computed the force for a plate-plate and a sphere-plate geometry. We found that in the former case the accuracy is decent, while in the latter it is somewhat lacking. One way to improve this is to increase the grid size, but there are other more efficient methods as well that could improve results. Finally we discussed possible improvements and additions to the program, as well as a comparison to boundary element methods, that turn out to be superior when the force should be measured at well-defined objects.

10 Appendix 1: Configuration

If Rust is installed, the program is installable via `cargo install casimir-fdfd`. Alternatively it can be built directly from the source code. To run, pass it the path to a configuration file. The flag `--progressbar`/`-p` can be used to display a progress bar while the program is running. It is always possible to run `casimir-fdfd --help` for usage information. Example usage would be `casimir-fdfd "worlds/plates_medium.json"`.

`casimir-fdfd` accepts configuration files in the following format:

```
{
```

The option `"size"` determines the size of the grid. Beyond this, the field will be periodic and wrap around.

```
    "size": [40, 40, 40],
```

The option `"objects"` defines the geometry for the simulation. Two types of objects can be defined: boxes and spheres. Boxes are defined by two points, spheres by one centre point and a radius. Except for the radius, which can be a rational number, all values have to be integers. The bounding boxes are larger than the objects themselves, so they can't be too close to eachother or to the edge of the world. The program will warn if the object definition causes such a problem. The predefined material `"gold"` can be used, or a custom material with its Drude parameters can be given.

```
    "objects": [
        {
            "Box": {
                "p0": [2, 2, 2],
                "p1": [17, 17, 3],
                "material": "gold"
            }
        },
        {
            "Sphere": {
                "point": [9, 9, 13],
                "radius": 7
                "material": "gold"
            }
        }
    ],
```

This object contains all settings for the simulation efficiency and properties. By changing the option `frequency_threshold`, the frequency integration error can be determined. If the linear interpolation error is smaller than `frequency_threshold` multiplied by the starting force integrand value, no further subdivisions will be used. `fdfd_convergence` determines the maximum error the residue $\mathbf{r} = A\mathbf{x} - \mathbf{b}$ can have in the conjugate gradient algorithm. `cosine_cutoff` is used for the cosine expansion; further (n_x, n_y) terms will be computed until the contribution is smaller than this value multiplied by the average thus far. The value of `frequency_range` gives the range between which to integrate the force. This domain is limited because the material permittivity approximation might not hold for some frequencies. The frequencies are given in units of $\frac{c}{\Delta x}$, where c is the speed of light and Δx the grid unit size.

```
    "simulation_config": {
        "frequency_threshold": 0.01,
        "fdfd_convergence": 0.000001,
        "cosine_cutoff": 0.001,
```

```
        "frequency_range": [0.01, 1.0]
    }
}
```

11 Appendix 2: Program output

11.1 Plate-plate configuration

Below is the full output for the plate-plate configuration.

```
Geometry:
    World size: (50, 50, 50)
    0: Box: (2, 2, 2), (17, 17, 3) with material Drude material: p: 7.79,  :
        48.8, step size: 0.1, precision: 0.001
    1: Box: (2, 2, 10), (17, 17, 11) with material Drude material: p: 7.79,  :
         48.8, step size: 0.1, precision: 0.001
Simulation configuration:
    Frequency threshold: 0.005
    Frequency range: 0.02 .. 1
    FDFD convergence: 0.000001
Cosine cutoff: 0.005
Force for frequency 0.02: (-20.935608, -20.93164, -331.69)
Force for frequency 1: (0.0000007748604, 0.0000006109476, -0.0000052452087)
Force for frequency 0.51: (-0.000113487244, -0.000120162964, -0.008491516)
Force for frequency 0.265: (-0.011394024, -0.011352539, -0.7167654)
Force for frequency 0.1425: (-0.14366531, -0.14382935, -7.602659)
Force for frequency 0.08125: (-0.76039886, -0.7593994, -29.701134)
Force for frequency 0.050624996: (-2.563057, -2.5637207, -72.01967)
Force for frequency 0.035312496: (-6.017151, -6.023926, -131.47585)
Force for frequency 0.027656248: (-10.427734, -10.426758, -195.3103)
Force for frequency 0.042968746: (-3.7970886, -3.8017578, -95.158875)
Force for frequency 0.0659375: (-1.3219414, -1.3204346, -44.673107)
Force for frequency 0.111875: (-0.30808258, -0.307312, -14.511875)
Force for frequency 0.20374998: (-0.037672043, -0.03755188, -2.2695026)
Force for frequency 0.3875: (-0.0011394024, -0.0011577606, -0.076051235)
Force for frequency 0.755: (0.0000051259995, 0.0000038146973, -0.00003695488)
Calculated force for object 0: (-0.64362484, -0.64363116, -15.406083)
Geometry:
    World size: (50, 50, 50)
    0: Box: (2, 2, 2), (17, 17, 3) with material Drude material: p: 7.79,  :
        48.8, step size: 0.1, precision: 0.001
    1: Box: (2, 2, 10), (17, 17, 11) with material Drude material: p: 7.79,  :
         48.8, step size: 0.1, precision: 0.001
Simulation configuration:
    Frequency threshold: 0.005
    Frequency range: 0.02 .. 1
    FDFD convergence: 0.000001
    Cosine cutoff: 0.005
Force for frequency 0.02: (32.77942, 32.757813, 353.2198)
Force for frequency 1: (-0.0000013709068, -0.0000013262033, 0.0000026226044)
Force for frequency 0.51: (0.00020086765, 0.00018405914, 0.00877285)
Force for frequency 0.265: (0.013239384, 0.013282776, 0.7469139)
Force for frequency 0.1425: (0.13785934, 0.13848877, 7.915556)
Force for frequency 0.08125: (0.71546173, 0.71606445, 30.68718)
Force for frequency 0.050624996: (2.8070755, 2.8095703, 74.49688)
Force for frequency 0.035312496: (7.7183228, 7.7197266, 137.37428)
Force for frequency 0.027656248: (14.72998, 14.733398, 205.68079)
```

```
Force for frequency 0.042968746: (4.464737, 4.4746094, 98.88666)
Force for frequency 0.0659375: (1.3051605, 1.3061523, 46.162777)
Force for frequency 0.111875: (0.28235626, 0.28289795, 15.06148)
Force for frequency 0.20374998: (0.040741444, 0.04081726, 2.3743362)
Force for frequency 0.3875: (0.0014913082, 0.0015087128, 0.07990551)
Force for frequency 0.755: (0.000006735325, 0.0000014305115, 0.000071287155)
Calculated force for object 1: (0.8382111, 0.8384899, 16.106794)
```

11.2 Sphere-plate configuration

```
Geometry:
World size: (40, 40, 40)
0: Box: (2, 2, 2), (16, 16, 3) with material Drude material: p: 7.79,  : 48.8,
     step size: 0.1, precision: 0.001
1: Sphere: (9, 9, 15), with radius 7 and material Drude material: p: 7.79,  :
    48.8, step size: 0.1, precision: 0.001
Simulation configuration:
Frequency threshold: 0.005
Frequency range: 0.025 .. 1
FDFD convergence: 0.000001
Cosine cutoff: 0.005
Force for frequency 0.025: (0.86660767, 0.8779297, -68.94632)
Force for frequency 1: (0.00000047683716, 0.00000008940697, 0.00000059604645)
Force for frequency 0.5125: (-0.000047683716, -0.0000333786, -0.0033636093)
Force for frequency 0.26874998: (-0.00016117096, -0.00008392334, -0.20431519)
Force for frequency 0.146875: (0.033613205, 0.033813477, -1.8038292)
Force for frequency 0.0859375: (0.18613434, 0.1875, -6.3492355)
Force for frequency 0.05546875: (0.4156723, 0.4169922, -14.957832)
Force for frequency 0.040234376: (0.6233978, 0.6230469, -27.66597)
Force for frequency 0.03261719: (0.74528503, 0.7441406, -41.231995)
Force for frequency 0.047851563: (0.5150986, 0.51220703, -19.739586)
Force for frequency 0.07070313: (0.2735405, 0.27478027, -9.386662)
Force for frequency 0.11640625: (0.080286026, 0.080566406, -3.2784214)
Force for frequency 0.20781249: (0.0047011375, 0.0048065186, -0.5957489)
Force for frequency 0.390625: (-0.00028538704, -0.00025558472, -0.025082588)
Force for frequency 0.75625: (-0.0000019669533, -0.0000017881393,
    -0.00009250641)
Calculated force for object 0: (0.07005999, 0.070284605, -3.2755113)
Geometry:
World size: (40, 40, 40)
0: Box: (2, 2, 2), (16, 16, 3) with material Drude material: p: 7.79,  : 48.8,
     step size: 0.1, precision: 0.001
1: Sphere: (9, 9, 15), with radius 7 and material Drude material: p: 7.79,  :
    48.8, step size: 0.1, precision: 0.001
Simulation configuration:
Frequency threshold: 0.005
Frequency range: 0.025 .. 1
FDFD convergence: 0.000001
Cosine cutoff: 0.005
Force for frequency 0.025: (38.28055, 38.335938, 26.564636)
Force for frequency 1: (0.0000013113022, -0.0000016391277, 0.000002026558)
Force for frequency 0.5125: (0.00009393692, 0.00008392334, 0.003238678)
Force for frequency 0.26874998: (0.005750656, 0.00592041, 0.2078352)
Force for frequency 0.146875: (0.12326813, 0.12011719, 1.8049889)
Force for frequency 0.0859375: (1.0159073, 0.99853516, 5.5822296)
Force for frequency 0.05546875: (4.4272385, 4.4277344, 10.632355)
Force for frequency 0.040234376: (11.287384, 11.25, 15.729858)
```

```
Force for frequency 0.03261719: (19.805878, 19.824219, 19.854004)
Force for frequency 0.047851563: (6.8532715, 6.8515625, 12.797256)
Force for frequency 0.07070313: (2.0108948, 2.0117188, 7.59227)
Force for frequency 0.11640625: (0.32086754, 0.32495117, 3.1504593)
Force for frequency 0.20781249: (0.021015167, 0.022277832, 0.6085186)
Force for frequency 0.390625: (0.00030612946, 0.000045776367, 0.025589466)
Force for frequency 0.75625: (0.000004529953, -0.0000009536743,
    0.000074863434)
Calculated force for object 1: (1.113095, 1.1125185, 2.145407)
```

References

[1] Timothy H Boyer. "Van der Waals forces and zero-point energy for dielectric and permeable materials". In: *Physical Review A* 9.5 (1974), p. 2078.

[2] Hendrick BG Casimir. "On the attraction between two perfectly conducting plates". In: *Proc. Kon. Ned. Akad. Wet.* Vol. 51. 1948, p. 793.

[3] B V Derjaguin. "Untersuchungen über die Reibung und Adhäsion, IV". In: *Colloid & Polymer Science* 69.2 (1934), pp. 155–164.

[4] David J. Griffiths. *Introduction to Electrodynamics*. Pearson, 2013. ISBN: 978-0-321-85656-2.

[5] David J. Griffiths. *Introduction to Quantum Mechanics*. Cambridge University Press, 2017. ISBN: 978-1-107-17986-8.

[6] Steven G Johnson. URL: `https://commons.wikimedia.org/wiki/File:Yee-cube.svg`.

[7] Steven G Johnson. "Numerical methods for computing Casimir interactions". In: *Casimir physics*. Springer, 2011, pp. 175–218.

[8] DE Krause et al. "Experimental investigation of the Casimir force beyond the proximity-force approximation". In: *Physical review letters* 98.5 (2007), 050403.

[9] Man-Leung Lui and Zhizhang Chen. "A direct computation of propagation constant using compact 2-D full-wave eigen-based finite-difference frequency-domain technique". In: *Computational Electromagnetics and Its Applications, 1999. Proceedings.(ICCEA'99) 1999 International Conference on*. IEEE. 1999, pp. 78–81.

[10] Alexander P McCauley et al. "Casimir forces in the time domain: Applications". In: *Physical Review A* 81.1 (2010), 012119.

[11] MT Homer Reid, Jacob White, and Steven G Johnson. "Fluctuating surface currents: An algorithm for efficient prediction of Casimir interactions among arbitrary materials in arbitrary geometries". In: *Physical Review A* 88.2 (2013), 022514.

[12] Alejandro Rodriguez et al. "Virtual photons in imaginary time: Computing exact Casimir forces via standard numerical electromagnetism techniques". In: *Physical Review A* 76.3 (2007), 032106.

[13] Jonathan Richard Shewchuk et al. *An introduction to the conjugate gradient method without the agonizing pain*. 1994.

[14] V. B. Svetovoy et al. "Optical properties of gold films and the Casimir force". In: *Physical Review B* 77 (3 Jan. 2008), 035439. DOI: `10.1103/PhysRevB.77.035439`.

[15] Kane Yee. "Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media". In: *IEEE Transactions on antennas and propagation* 14.3 (1966), pp. 302–307.

[16] Li Zhao and Andreas C Cangellaris. "A general approach for the development of unsplit-field time-domain implementations of perfectly matched layers for FDTD grid truncation". In: *IEEE Microwave and Guided Wave Letters* 6.5 (1996), pp. 209–211.