Contents lists available at ScienceDirect

Journal of Quantitative Spectroscopy & Radiative Transfer

journal homepage: www.elsevier.com/locate/jqsrt

The discrete-dipole-approximation code ADDA: Capabilities and known limitations

Maxim A. Yurkin [a,b,*], Alfons G. Hoekstra [c]

[a] Institute of Chemical Kinetics and Combustion SB RAS, Institutskaya Street 3, 630090 Novosibirsk, Russian Federation
[b] Novosibirsk State University, Pirogova Street 2, 630090 Novosibirsk, Russian Federation
[c] Computational Science Research Group, Faculty of Science, University of Amsterdam, Science Park 904, 1098 XH Amsterdam, The Netherlands

ARTICLE INFO

Available online 1 February 2011

Keywords:
Discrete dipole approximation
Light scattering simulation
Computer program
Very large particles

ABSTRACT

The open-source code ADDA is described, which implements the discrete dipole approximation (DDA), a method to simulate light scattering by finite 3D objects of arbitrary shape and composition. Besides standard sequential execution, ADDA can run on a multiprocessor distributed-memory system, parallelizing a *single* DDA calculation. Hence the size parameter of the scatterer is in principle limited only by total available memory and computational speed. ADDA is written in C99 and is highly portable. It provides full control over the scattering geometry (particle morphology and orientation, and incident beam) and allows one to calculate a wide variety of integral and angle-resolved scattering quantities (cross sections, the Mueller matrix, etc.). Moreover, ADDA incorporates a range of state-of-the-art DDA improvements, aimed at increasing the accuracy and computational speed of the method. We discuss both physical and computational aspects of the DDA simulations and provide a practical introduction into performing such simulations with the ADDA code. We also present several simulation results, in particular, for a sphere with size parameter 320 (100-wavelength diameter) and refractive index 1.05.

1. Introduction

The discrete dipole approximation (DDA) is a general method to calculate scattering and absorption of electromagnetic waves by particles of arbitrary geometry. In this method the volume of the scatterer is divided into small cubical subvolumes ("dipoles"). Dipole interactions are approximated based on the integral equation for the electric field [1]. Initially the DDA (sometimes referred to as the "coupled dipole approximation") was proposed by Purcell and Pennypacker [2] replacing the scatterer by a set of point dipoles (hence the name of the technique). Although the final equations are essentially the same, derivations based on the integral equations give more mathematical insight into the approximation, while the model of point dipoles is physically clearer. For an extensive review of the DDA, including both theoretical and computational aspects, the reader is referred to [1] and references therein.

ADDA is a C implementation of the DDA developed by the authors. The development was conducted by Hoekstra and coworkers [3–6] since 1990 at the University of Amsterdam. From the very beginning the code was intended to run on a multiprocessor system or a multicore processor (parallelizing a *single* DDA simulation). The code was significantly rewritten and improved by Yurkin et al. [7], also at the University of Amsterdam. Since then the authors have been further developing the code. Originally coined "Amsterdam DDA", the code has been

[*] Corresponding author at: Institute of Chemical Kinetics and Combustion SB RAS, Institutskaya Street 3, 630090 Novosibirsk, Russian Federation. Tel.: +7 383 333 3240; fax: +7 383 330 7350.
E-mail address: yurkin@gmail.com (M.A. Yurkin).

officially abbreviated to ADDA to reflect the international nature of its development. ADDA is intended to be a versatile community tool, suitable for a wide variety of applications ranging from interstellar dust and atmospheric aerosols to biological cells and nanoparticles; its applicability is limited only by available computer resources.

ADDA is freely available under the terms of the GNU General Public License at http://code.google.com/p/a-dda, and has been used by the community since 2006.[1] Both the full source code and compiled executables for 32-bit Windows systems are available. The source code is easily compiled under any operating system supporting a C99 compiler and executes on any parallel system supporting the MPI (message passing interface). It is accompanied by extensive documentation, consisting of a user manual and a number of wiki pages. The former focuses on computational and physical aspects of ADDA, while the latter are devoted to more technical issues, ranging from compiling instructions to description on how to add new predefined shapes. ADDA development has gone beyond its original authors and is intended to be an open-source community effort, with (source code) contributions from members of the community.

This paper summarizes the capabilities and limitations of ADDA,[2] and is based on the corresponding manual [8]. The overall goal is to discuss physical and computational aspects of DDA simulations and to provide a practical introduction into performing such simulations using the ADDA code. The paper discusses general applicability of the code (Section 2), system requirements (Section 3), and how to specify a scattering problem, including particle orientation and incident beam (Section 4). Then different DDA formulations, as incorporated into ADDA, are discussed (Section 5) as well as scattering quantities that can be calculated (Section 6). At the end, we discuss computational aspects of the code (Section 7), present a number of sample simulations (Section 8) and a general conclusion (Section 9).

ADDA is a console application without graphical user interface. Its behavior is mostly controlled through the command line, although large sets of parameters (e.g. shape of a scatterer) are supplied through special input files. A brief description of the most important command line options is given in the relevant parts of this paper. The full list can be obtained through the built-in help system (running ADDA with "`-h`" flag) and from the manual [8]. Much more detailed information, in particular, formats of input and output files, is given in the above-mentioned documentation of the code.

Finally, to our knowledge, there are at least three other freely available DDA codes: DDSCAT [9], OpenDDA [10], and DDA-SI toolbox [11]. More DDA codes exist, and some of them are discussed in [12], but these are not freely available to the community. Although certain comparative comments are given in the remainder of the paper, a thorough comparison of ADDA with those other codes lies outside its scope. A detailed albeit slightly outdated comparison of ADDA, DDSCAT, and two other codes was performed by Penttila et al. [12].

[1] See http://code.google.com/p/a-dda/wiki/Publications for a list of journal publications using ADDA.

[2] The descriptions in this paper are based on v.1.0 of the code, released in September 2010.

2. Applicability of the DDA

2.1. General applicability

The principal advantage of the DDA is that it is completely flexible regarding the geometry of the scatterer, being limited only by the need to use a dipole size d small compared to both any structural length in the scatterer and the wavelength λ. A large number of studies devoted to the accuracy of DDA exist, e.g. [9,13–16,7,17–21]. Most of them are reviewed in [1]; here we only give a brief overview.

The rule of thumb for particles with size comparable to the wavelength is: "10 dipoles per wavelength inside the scatterer", i.e. size of one dipole is

$$d = \lambda/10|m|, \tag{1}$$

where m is the refractive index of the scatterer. That is the default for ADDA. The expected accuracy of cross sections is then several percents (for moderate m, see below). With increasing m the number of dipoles that is used to discretize the particle increases; moreover, the convergence of the iterative solver (Section 7.1) becomes slower. Additionally, the accuracy of the simulation with default dipole size deteriorates, and smaller, hence more dipoles must be used to improve it. Therefore, it is accepted that the refractive index should satisfy

$$|m-1| < 2. \tag{2}$$

Higher m can also be simulated accurately. In that case however, the required computer resources rapidly increase with m. Fortunately, state-of-the-art DDA formulations (Section 5) can alleviate this problem and render higher refractive indices accessible to DDA simulations. Note however that the application of the DDA in this large m regime is investigated much less thoroughly than for moderate refractive indices, and therefore warrants further studies.

When considering larger scatterers (volume-equivalent size parameter $x > 10$) the rule of thumb still applies. However, it does not describe well the dependence on m. When employing the rule of thumb, errors do not significantly depend on x, but do significantly increase with m [7]. However, simulation data for large scatterers is also limited; therefore, it is hard to propose any simple method to set the dipole size. The maximum reachable x and m are mostly determined by the available computer resources (Section 3).

The DDA is also applicable to particles smaller than the wavelength, e.g. nanoparticles. In some respects, it is even simpler than for larger particles, since many convergence problem for large m are not present for small scatterers. However, in this regime there is an additional requirement for d—it should allow for an adequate description of the shape of the particle. This requirement is relevant for any scatterer, but for larger scatterers it is usually automatically satisfied by Eq. (1). For instance, for a sphere (or similar compact shape) it is recommended to use at least 10 dipoles along the smallest dimension, no matter how small the particle is. Smaller dipoles are required for irregularly shaped particles and/or large refractive index. The accuracy of the DDA for gold nanoparticles was studied in [22].

To conclude, it is hard to estimate *a priori* the accuracy of DDA simulation for a particular particle shape, size, and refractive index, although the papers cited above do give a hint. If one runs a single DDA simulation, there is no better alternative than to use rule of thumb and hope that the accuracy will be similar to that of the spheres, which can be found in one of the benchmark papers (e.g. [7,20,15]). However, if one plans a series of simulations for similar particles, especially outside of the usual DDA application domain [Eq. (2)], it is highly recommended to perform an accuracy study. For that one should choose a single test particle and perform DDA simulations with different d's, both smaller and larger than proposed by the rule of thumb. The estimate of d required for a particular accuracy can be obtained from a variation of results with decreasing d. Moreover, the estimation can be made much more rigorous by using an extrapolation technique, as proposed by Yurkin et al. [23] and applied in [22,24,25].

Finally, it is important to note that the price paid for versatility of the DDA is its large computational costs, even for "simple" scatterers. Thus, in certain cases other (more specialized) methods will clearly be superior to the DDA. A review of relevant comparative studies is given in [1]. Additionally, it was recently shown that the DDA (and the ADDA code in particular) performs exceptionally well for large index-matching particles (e.g. biological cells in a liquid medium). In this regime the DDA is 10–100 times faster than a (general-purpose) finite-difference time-domain method when required to reach the same accuracy [24], and is comparable in speed to the discrete sources method for red blood cells [26], where the latter method explicitly employs the axisymmetry of the problem.

2.2. Extensions of the DDA

In its original form the DDA is derived for finite particles (or a set of several finite particles) in vacuum. However, it is also applicable to finite particles embedded in a homogeneous non-absorbing dielectric medium (refractive index m_0). To account for the medium one should replace the particle refractive index m by the relative refractive index m/m_0, and the wavelength in vacuum λ by the wavelength in the medium λ/m_0. All the scattering quantities produced by DDA simulations with such modified m and λ are then the correct ones for the initial scattering problem.

ADDA cannot be directly applied to infinite scatterers. In particular, it *cannot* be applied to particles located near an infinite dielectric plane surface or, more generally, particles above or inside a substrate of finite or infinite width. Although a modification of the DDA is known to rigorously solve this problem [11,27–30], it requires principal changes in the DDA formulation and hence in the internal structure of the computer code. However, the current version of ADDA still provides an opportunity to solve this problem. One could consider the substrate only by its influence on the incident field $E^{\text{inc}}(r)$ (by adding a reflected wave $E^{\text{ref}}(r)$). This may be accurate enough if the particle is far from the substrate and the contrast between the substrate and the upper medium is small. Another approach is to take a large computational box around the particle, and explicitly discretize the substrate that falls into it [31]. This is rigorous in the limit of infinite size of computational box, but requires much larger computer resources than that for the initial problem. The problem with the latter approach is the diffraction of the plane wave on the edges of the computational domain. It can be alleviated by using a Gaussian beam with a width smaller than the computational domain but larger than all structural features of the problem (particles above or inhomogeneities inside the substrate) [32].

A combination of these two approaches was proposed by D'Agostino et al. [33] to decrease spurious boundary effects. The total near- of far-field $\mathbf{E}(\mathbf{r})$ is replaced by an adjusted field $\mathbf{E}^{\text{adj}}(\mathbf{r})$

$$\mathbf{E}^{\text{adj}}(\mathbf{r}) = \mathbf{E}(\mathbf{r}) - \mathbf{E}^{\text{sub}}(\mathbf{r}) + \mathbf{E}^{\text{inc}}(\mathbf{r}) + \mathbf{E}^{\text{ref}}(\mathbf{r}), \tag{3}$$

where $\mathbf{E}^{\text{sub}}(\mathbf{r})$ is the result of a DDA simulation for the truncated substrate alone (without particles or inhomogeneities). This technique was proposed and tested for nanoparticles above metallic layers, but it could also be useful for other problems that fall under the general description given above. In other words, it is expected that $\mathbf{E}^{\text{adj}}(\mathbf{r})$ will converge to the correct solution with increasing computational domain faster than $\mathbf{E}(\mathbf{r})$.

Another useful extension of the DDA is introduction of periodic boundary conditions [34,35], which is relevant to photonic crystals and similar applications. This requires relatively simple modification of the algorithm and it was recently implemented in the DDSCAT v.7 [36]. However, ADDA does *not* yet support this feature.

3. System requirements

Computational requirements of DDA primarily depend on the size of the computational grid, which in turn depends on the size parameter x and refractive index m of the scatterer. The memory requirements of ADDA depend both on the total number of dipoles in a computational box (N) and the number of real (non-void) dipoles (N_{real}); it also depends on the number of dipoles along the x-axis (n_x) and number of processors or cores used (n_{p}). The total memory requirement M_{tot} (for all processors) is approximately

$$M_{\text{tot}} = [288 + 384n_{\text{p}}/n_x(+192/n_{\text{p}})]N + [271(+144)]N_{\text{real}} \text{ bytes}, \tag{4}$$

where additional memory (in round brackets) proportional to N is required only in parallel mode, and proportional to N_{real} only for the QMR and Bi-CGStab iterative solvers (Section 7.1). It is important to note that *double* precision is used everywhere in ADDA. This requires more memory as compared to single precision, but it helps when convergence of the iterative solver is very slow and machine precision becomes relevant, as is the case for large simulations, or when very accurate results are desired, as in [23].

There is a maximum number of processors, which ADDA can effectively employ, which is equal to n_z. This determines the largest problem size solvable on a given supercomputer with very large number of processors but with a limited amount of memory per processor M_{pp}. For a given problem, setting $n_{\mathrm{p}}=n_z$ leads to the following memory requirements per processor:

$$M_{\mathrm{pp}} = [288n_x + 384n_z + 192n_x/n_z]n_y + [271(+144)]n_{\mathrm{slice}}\text{bytes}, \quad (5)$$

where $n_{\mathrm{slice}} \le n_x n_y$ is the maximum number of real dipoles in a slice parallel to the yz-plane, and number in round parentheses has the same meaning as in Eq. (4).

To have a quick estimate of maximum achievable discretization on a given hardware, one may consider a cube and less-memory-consuming iterative solvers. Then Eqs. (4) and (5) lead to maximum n_x equal to $113[M_{\mathrm{tot}}(\mathrm{GB})]^{1/3}$ and $1067[M_{\mathrm{pp}}(\mathrm{GB})]^{1/2}$ for single-core PC and very large cluster, respectively.

Simulation time consists of two major parts: solution of the linear system of equations and calculation of the scattered fields. The first one depends on the number of iterations to reach convergence, which mainly depends on the size parameter, shape and refractive index of the scatterer, and time of one iteration, which depends only on N as $O(N \ln N)$. Execution time for calculation of scattered fields is proportional to N_{real}, and is usually relatively small if scattering is only calculated in one plane. However, it may be significant when a large grid of scattering angles is used (see Section 7 for details).

For example, on a desktop computer (P4-3.2 GHz, 2 Gb RAM) it was possible to simulate light scattering by spheres up to $x=35$ and 20 for $m=1.313$ and 2.0, respectively (simulation times are 20 and 148 h, respectively). The capabilities of ADDA for simulation of light scattering by spheres using 64 3.4 GHz cores were reported in [7]. In particular, light scattering by a homogeneous sphere with $x=160$ and $m=1.05$ was simulated in only 1.5 h, although the runtime steeply increased with refractive index. Examples of more recent and even larger simulations are presented in Section 8.

4. Defining a scattering problem

4.1. Reference frames

Three different reference frames are used by ADDA: laboratory, particle, and incident wave reference frames. The laboratory reference frame is the default one, and all input parameters and other reference frames are specified relative to it. ADDA simulates light scattering in the particle reference frame, which naturally corresponds to particle geometry and symmetries, to minimize the size of the computational grid, especially for elongated or oblate particles. In this reference frames the computational grid is build along the coordinate axes. The incident wave reference frame is defined by setting the z-axis along the propagation direction. All scattering directions are specified in this reference frame.

The origins of all reference frames coincide with the center of the computational grid. By default, both particle and incident wave reference frames coincide with the laboratory frame. However they can be made different by rotating the particle (Section 4.5) or by specifying a different propagation direction of the incident beam (Section 4.6).

4.2. The computational grid

ADDA embeds a scatterer in a rectangular computational box, which is divided into identical cubes (as required for the FFT-acceleration—Section 7.1). Each cube is called a "dipole"; its size should be much smaller than a wavelength. The flexibility of the DDA method lies in its ability to naturally simulate the scattering of any arbitrarily shaped and/or inhomogeneous scatterer, because the optical properties (refractive index) of each dipole can be set independently. There are a few parameters describing the computational grid: size of one dipole (cube) d, number of dipoles along each axis n_x, n_y, n_z, total size (in μm) of the grid along each axis D_x, D_y, D_z, volume-equivalent radius r_{eq}, and incident wavelength λ. However, they are not independent. ADDA allows one to specify all three grid dimensions n_x, n_y, n_z as arguments to the command line option

```
-grid <nx> [<ny> <nz>]
```

If omitted, n_y, n_z are automatically determined by n_x based on the proportions of the scatterer. When particle geometry is read from a file all grid dimensions are initialized automatically.

One can also specify the size parameter of the entire grid kD_x (with k the free space wave vector) or $x=kr_{\mathrm{ef}}$, using three command line options:

```
-lambda <arg>
-size <arg>
-eq_rad <arg>
```

which specify (in μm) λ, D_x and r_{ef}, respectively. By default $\lambda=2\pi$ μm, then `-size` determines kD_x and `-eq_rad` sets x. The last two are related by

$$x = kD_x \sqrt[3]{3f_{\mathrm{vol}}/4\pi}, \quad (6)$$

where f_{vol} is the ratio of particle to computational grid volumes, which is known analytically for many shapes available in ADDA (see Section 4.4). The size parameter of the dipole is specified by the parameter "dipoles per lambda" (dpl)

$$\mathrm{dpl} = \frac{\lambda}{d} = \frac{2\pi}{kd}, \quad (7)$$

which is given to the command line option

```
-dpl <arg>
```

dpl does not need to be an integer; any real number can be specified.

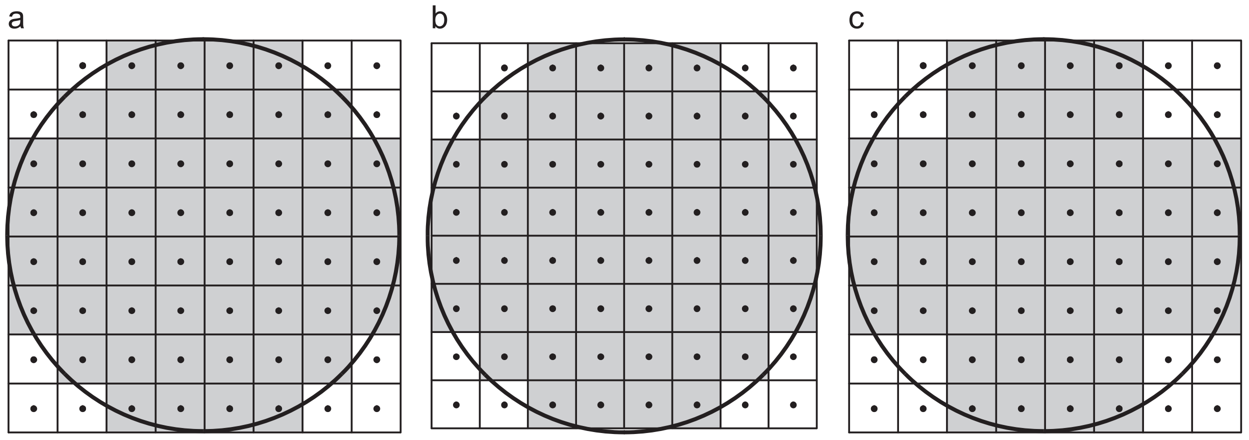

Fig. 1. An example of dipole assignment for a sphere (2D projection). Assigned dipoles are gray and void dipoles are white: (a) initial assignment; (b) after volume correction; and (c) with "`−jagged`" option enabled ($J=2$) and the same total grid dimension.

ADDA will accept at most two parameters from: dpl, n_x, kD_x, and x since they depend on each other by Eq. (6) and

$$kD_x \cdot \mathrm{dpl} = 2\pi \cdot n_x. \tag{8}$$

Moreover, specifying a pair of kD_x and x is also not possible. If any other pair from these four parameters is given on the command line (n_x is also defined if particle geometry is read from file) the other two are automatically determined from Eqs. (6) and (8). If less than two parameters are defined dpl or/and grid dimension are set by default. The default for dpl is $10|m|$ [cf. Eq. (1)], where m is the maximum (by absolute value) refractive index specified by the `−m` option (or the default one). The default for n_x is 16.

4.3. Construction of a dipole set

After defining the computational grid each dipole of the grid should be assigned a refractive index (a void dipole is equivalent to a dipole with refractive index equal to 1). This can be done automatically for a number of predefined shapes or in a very flexible way by specifying scatterer geometry in a separate input file. For predefined shapes the dipole is assigned to the scatterer if and only if its center falls into the particle shape [see Fig. 1(a) for an example]. When the scatterer consists of several domains, e.g. a coated sphere, the same rule applies to each domain. By default, ADDA slightly corrects the dipole size (or equivalently dpl) to ensure that the volume of the dipole representation of the particle is exactly correct [Fig. 1(b)], i.e. exactly corresponds to x. This is believed to increase the accuracy of DDA, especially for small scatterers [9], but it can be turned off by the command line option

```
−no_vol_cor
```

In parallel mode the dipoles are distributed among different processors in slices parallel to the xy-plane. Although parallel performance of ADDA significantly depends on this partition and careful choice of n_p and n_z, we omit this discussion from the paper, see e.g. [8].

To read particle geometry from a file, specify the file name as an argument to the command line option

```
−shape read <filename>
```

This file specifies all the dipoles in the simulation grid that belong to the particle (possibly several domains with different refractive indices). Both ADDA text formats and the DDSCAT 6.1 format [37] are supported.

Sometimes it is useful to describe particle geometry in a coarse way by bigger dipoles (cubes), but then use smaller dipoles for the simulation itself. ADDA enables this by the command line option

```
−jagged <arg>
```

that specifies a multiplier J. Large cubes ($J \times J \times J$ dipoles) are used [Fig. 1(c)] for construction of the dipole set. Cube centers are tested for belonging to a particle's domain. All grid dimensions are multiplied by J. When particle geometry is read from file it is considered to be a configuration of big cubes, each of them is further subdivided into J^3 dipoles.

ADDA includes a granule generator, which can randomly fill any specified domain with granules of a predefined size. It is enabled by the command line option

```
−granul <vol_frac> <diam>
[<dom_number>]
```

which specifies that one particle domain should be randomly filled with spherical granules with specified diameter `<diam>` and volume fraction `<vol_frac>`. The domain number to fill is given by the last optional argument (default is the first domain). The total number of domains is then increased by one; the last is assigned to the granules.

The last parameter to completely specify a scatterer is its refractive index. Refractive indices are given on the command line

```
−m {<m1Re> <m1Im> [...]|<m1xxRe> <m1x-
xIm> <m1yyRe> <m1yyIm> <m1zzRe>
<m1zzIm> [...]}
```

Each pair of arguments specifies the real and imaginary part[3] of the refractive index of the corresponding domain (first pair corresponds to domain number 1, etc.). Command line option

```
−anisotr
```

can be used to specify that a refractive index is anisotropic. In that case three refractive indices correspond to one domain. They are the diagonal elements of the refractive index tensor in the particle reference frame.

Finally, ADDA saves the constructed dipole set to a file if the command line option

```
−save_geom [<filename>]
```

is specified, where `<filename>` is an optional argument. The format is determined by the command line option

```
−sg_format {text|text_ext|ddscat}
```

The first two are ADDA default formats for single- and multi-domain particles, respectively. DDSCAT 6.1 format corresponds to its shape option `FRMFIL` and output of the `calltarget` utility [37].

4.4. *Predefined shapes*

Predefined shapes are initialized by the command line option

```
−shape <type> [<args>]
```

where `<type>` is a name of the predefined shape. The size of the scatterer is determined by the size of the computational grid (D_x); `<args>` specify different dimensionless aspect ratios or other proportions of the particle shape.

An extensive description of all predefined shape is given in the manual [8], while here only a brief description is provided by Table 1. For multi-domain shapes f_{vol} is based on the total volume of the particle.

4.5. *Orientation of the scatterer*

Any particle orientation with respect to the laboratory reference frame can be specified by three Euler angles (α,β,γ). ADDA uses a notation based on [38], which is also called "*zyz*-notation" or "*y*-convention". In short, coordinate axes attached to the particle are first rotated by the angle α over the z-axis, then by the angle β over the current position of the y-axis (the line of nodes), and finally by the angle γ over the new position of the z-axis (see Fig. 2). These angles are specified in degrees as three arguments to the command line option

```
−orient <alpha> <beta> <gamma>
```

[3] ADDA uses exp($-i\omega t$) convention for time dependence of harmonic electric field, therefore absorbing materials have *positive* imaginary part.

Table 1
Brief description of arguments, symmetries, and availability of analytical value of f_{vol} for predefined shapes. "±" denotes that the value depends on the arguments.

<type>	<args>	dom.[a]	symY[b]	symR[c]	f_{vol}	size[d]
`axisymmetric`	Filename	1	+	+	–	+
`bicoated`	R_{cc}/d, d_{in}/d	2	+	+	+	–
`biellipsoid`	y_1/x_1, z_1/x_1, x_2/x_1, y_2/x_2, z_2/x_2	2	+	±	+	–
`bisphere`	R_{cc}/d	1	+	+	+	–
`box`	[y/x, z/x]	1	+	±	+	–
`capsule`	h/d	1	+	+	+	–
`coated`	d_{in}/d, [x/d, y/d, z/d]	2	±	±	+	–
`cylinder`	h/d	1	+	+	+	–
`egg`	ε, ν	1	+	+	+	–
`ellipsoid`	y/x, z/x	1	+	±	+	–
`line`	–	1	–	–	–	–
`rbc`	h/d, b/d, c/d	1	+	+	–	–
`sphere`	–	1	+	+	+	–
`spherebox`	d_{sph}/D_x	2	+	+	+	–

[a] Number of domains.
[b] Symmetry with respect to reflection over the *xz*-plane.
[c] Symmetry with respect to rotation by 90° over the *z*-axis.
[d] Whether a shape defines absolute size of the particle.

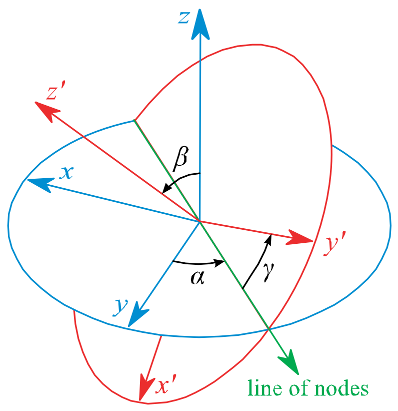

Fig. 2. Transformation of the laboratory reference system *xyz* into the particle reference frame $x'y'z'$ through consecutive rotation by angles α, β, and γ.

Alternatively, the result can be averaged over the orientation, using

```
−orient avg [<filename>]
```

where `<filename>` is an optional argument that specifies a file with parameters of the averaging. Averaging is performed over the Euler angles and rotating over α is equivalent to rotating the scattering plane without changing the orientation of the scatterer relative to the incident radiation. Therefore, averaging over this orientation angle is done with a single computation of internal fields; additional computation time for each scattering plane is comparably small. Averaging over the other two Euler angles is done by independent DDA simulations. Integration points for β are spaced uniformly in values of cos β.

4.6. Incident beam

The direction of propagation of the incident radiation is specified by the command line option

```
−prop <x> <y> <z>
```

where arguments are x, y, and z components of the propagation vector. Normalization (to the unit vector) is performed automatically by ADDA. By default, vector $\mathbf{e}_z$=(0,0,1) is used. Two incident polarizations are used by default: along the x and y axes. Those are perpendicular ($\perp$) and parallel ($||$) polarizations [39], respectively, with respect to the default scattering plane (yz). These polarizations are transformed simultaneously with the propagation vector—all three are rotated by two spherical angles (θ, φ) so that (0,0,1) is transformed into the specified propagation vector. Afterwards, the scattering angles are specified with respect to the incident wave reference frame based on the *new* propagation vector (z) and two *new* incident polarizations (x, y).

Additionally to the default ideal plane wave ADDA supports several types of finite-size incident beams, specified by the command line option

```
−beam <type> [<width> <x> <y> <z>]
```

where `<type>` is one of `plane`, `lminus`, `davis3`, or `barton5`. All beam types except the default plane wave are approximate descriptions of a Gaussian beam. Four arguments specified in the command line specify width (w_0) and x, y, z coordinates of the center of the beam, respectively (all in μm). The coordinates are specified in the laboratory reference plane. `lminus` is the simplest approximation [40], `davis3` [41] and `barton5` [42] are correct up to the third and fifth order of the beam confinement factor ($s=1/kw_0$), respectively. For all beam types we assume unit amplitude of the electric field in the focal point of the beam.

5. DDA formulation

Since its introduction by Purcell and Pennypacker [2] DDA has been constantly developed; therefore, a number of different DDA formulations exist [1]. Here we only provide a short summary, focusing on those that are implemented in ADDA. All formulations are equivalent to the solution of the linear system to determine unknown dipole polarizations $\mathbf{P}_i$

$$\overline{\boldsymbol{\alpha}}_i^{-1}\mathbf{P}_i-\sum_{j\neq i}\overline{\mathbf{G}}_{ij}\mathbf{P}_j=\mathbf{E}_i^{\rm inc}, \tag{9}$$

where $\mathbf{E}_i^{\rm inc}$ is the incident electric field, $\overline{\boldsymbol{\alpha}}_i$ is the dipole polarizability (self-term), $\overline{\mathbf{G}}_{ij}$ is the interaction term, and indices i and j enumerate the dipoles. For a plane wave incidence

$$\mathbf{E}^{\rm inc}(\mathbf{r})=\mathbf{e}^0\exp(\mathrm{i}\mathbf{k}\cdot\mathbf{r}), \tag{10}$$

where $\mathbf{k}=k\mathbf{a}$, $\mathbf{a}$ is the incident direction, and $|\mathbf{e}^0|=1$. The (total) electric field $\mathbf{E}_i$ is the one present in a homogeneous particle modeled by an array of dipoles, also known as macroscopic field [43]. It should be distinguished from the exciting electric field $\mathbf{E}_i^{\rm exc}$ that is a sum of $\mathbf{E}_i^{\rm inc}$ and the field due to all other dipoles, but excluding the field of the dipole i itself. Both total and exciting electric field can be determined once the polarizations are known:

$$\mathbf{P}_i=\overline{\boldsymbol{\alpha}}_i\mathbf{E}_i^{\rm exc}=V\chi_i\mathbf{E}_i, \tag{11}$$

where $V=d^3$ is the volume of a dipole and $\chi_i=(\varepsilon_i-1)/4\pi$ is the susceptibility of the medium at the location of the dipole (ε_i is the relative permittivity). In the following we will also refer to $\mathbf{E}$ as internal fields (those inside the particle) in contrast to near- and far-fields, which are calculated from $\mathbf{E}$ or $\mathbf{P}$ together with other scattering quantities. Below we discuss different formulations for the polarization prescription, interaction term and formulae to calculate scattering quantities.

The distinctive feature of ADDA compared to other DDA codes is the incorporation of many modern DDA formulations, which in certain cases may largely outperform the standard one. Additionally to the published ones, ADDA contains options to use new theoretical improvements that we are developing ourselves. However, we do not discuss them here, since they are still in the early research phase.

5.1. Polarizability prescription

A number of expressions for the polarizability are known [1]. ADDA implements five: the Clausius–Mossotti (CM, [2]), the radiative reaction correction (RR, [44]), the lattice dispersion relation (LDR, [13]), corrected LDR (CLDR, [45]), and the Filtered Coupled Dipoles (FCD, [21]). The CM polarizability is the basic one [2] and given by

$$\alpha_i^{\rm CM}=d^3\frac{3}{4\pi}\frac{\varepsilon_i-1}{\varepsilon_i+2}. \tag{12}$$

RR is a third-order (in kd) correction to CM [44]:

$$\alpha^{\rm RR}=\frac{\alpha^{\rm CM}}{1-(2/3)\mathrm{i}k^3\alpha^{\rm CM}}. \tag{13}$$

LDR adds second-order corrections [13]

$$\alpha^{\rm LDR}=\frac{\alpha^{\rm CM}}{1-(\alpha^{\rm CM}/d^3)[(b_1^{\rm LDR}+b_2^{\rm LDR}m^2+b_3^{\rm LDR}m^2S)(kd)^2+(2/3)i(kd)^3]}, \tag{14}$$

$$b_1^{\rm LDR}\approx 1.8915316,\quad b_2^{\rm LDR}\approx -0.1648469,\quad b_3^{\rm LDR}\approx 1.7700004, \tag{15}$$

$$S=\sum_\mu (a_\mu e_\mu^0)^2, \tag{16}$$

where μ denote vector components. The LDR prescription can be averaged over all possible incident polarizations [13], resulting in

$$S=\frac{1}{2}\left(1-\sum_\mu a_\mu^4\right). \tag{17}$$

Corrected LDR is independent on the incident polarization but leads to a diagonal polarizability tensor instead of scalar [45]

$$\alpha^{\mathrm{CLDR}}_{\mu\nu} = \frac{\alpha^{\mathrm{CM}}\delta_{\mu\nu}}{1-(\alpha^{\mathrm{CM}}/d^3)[(b_1^{\mathrm{LDR}}+b_2^{\mathrm{LDR}}m^2+b_3^{\mathrm{LDR}}m^2a_\mu^2)(kd)^2+(2/3)\mathrm{i}(kd)^3]}, \tag{18}$$

where $\delta_{\mu\nu}$ is the Kronecker symbol. The FCD polarizability is obtained from the value of filtered Green's tensor [Eq. (22)] for zero argument [21], leading to

$$\alpha^{\mathrm{FCD}} = \frac{\alpha^{\mathrm{CM}}}{1-(\alpha^{\mathrm{CM}}/d^3)[(4/3)(kd)^2+(2/3)(i+\ln((\pi-kd)/(\pi+kd))/\pi)(kd)^3]}. \tag{19}$$

Naturally, Eq. (19) is applicable only when $kd<\pi$, i.e. dpl>2. CM, RR, LDR, and FCD can be used together with anisotropic electric permittivity, given by a *diagonal* tensor $\bar{\boldsymbol{\varepsilon}}$. The polarizability is then also a diagonal tensor, calculated by the same formulae [Eqs. (12)–(14)] but separately for each component:

$$\alpha_{\mu\nu} = \delta_{\mu\nu}\alpha(\varepsilon_{\mu\mu}). \tag{20}$$

The choice of the polarization prescription is performed by the command line option

```
−pol <type> [<arg>]
```

where `<type>` is one of the `cm`, `rrc`, `ldr`, `cldr`, `fcd`. `<arg>` is optional flag that can be only `avgpol` and only for LDR specifying that the LDR polarizability should be averaged over incident polarizations. Default is LDR without averaging. It is important to note that this is not the best option for all cases. Our experience shows that LDR may perform particularly badly (as compared to CM or RR) for very large refractive indices, and FCD (together with its interaction term, described below) becomes the best option [25].

Finally, other DDA improvements are known, which modify the polarizability (or permittivity) of the dipoles near the boundary. These improvements include weighted discretization [46] and spectral filtering of the permittivity that was proposed in combination with FCD [21]. These ideas have not yet been implemented in ADDA.

5.2. Interaction term

A few formulations for the interaction term are known [1]. Currently, ADDA can use the simplest one (interaction of point dipoles), FCD (in other words, filtered Green's tensor [21]), the quasistatic version of FCD, and the Integrated Green's Tensor (IGT, [16]). The interaction of point dipoles is described by the Green's tensor

$$\bar{\mathbf{G}}_{ij} = \bar{\mathbf{G}}(\mathbf{r}_i,\mathbf{r}_j) = \frac{\exp(\mathrm{i}kR)}{R}\left[k^2\left(\bar{\mathbf{I}}-\frac{\hat{R}\hat{R}}{R^2}\right)-\frac{1-\mathrm{i}kR}{R^2}\left(\bar{\mathbf{I}}-3\frac{\hat{R}\hat{R}}{R^2}\right)\right], \tag{21}$$

where $\mathbf{r}_i$ is the radius-vector of the dipole center, $\mathbf{R}=\mathbf{r}_j-\mathbf{r}_i$, $R=|\mathbf{R}|$, $\bar{\mathbf{I}}$ is the identity tensor, and $\hat{R}\hat{R}$ is a tensor defined as $\hat{R}\hat{R}_{\mu\nu}=R_\mu R_\nu$. The filtered Green's tensor is defined [21] as

$$\bar{\mathbf{G}}^{\mathrm{FCD}}_{ij} = \bar{\mathbf{I}}\left(k^2 g_{\mathrm{F}}(R)+\frac{g'_{\mathrm{F}}(R)}{R}+\frac{4\pi}{3}h_{\mathrm{r}}(R)\right)+\frac{\hat{R}\hat{R}}{R^2}\left(g''_{\mathrm{F}}(R)-\frac{g'_{\mathrm{F}}(R)}{R}\right), \tag{22}$$

where h_{r} is filter impulse response

$$h_{\mathrm{r}}(R) = \frac{\sin(k_{\mathrm{F}}R)-k_{\mathrm{F}}R\cos(k_{\mathrm{F}}R)}{2\pi^2R^3}, \tag{23}$$

$k_{\mathrm{F}}=\pi/d$ is the wavenumber corresponding to the grid, and g_{F} is the filtered scalar Green's function

$$g_{\mathrm{F}}(R) = \frac{1}{\pi R}\{\sin(kR)[\pi\mathrm{i}+\mathrm{Ci}((k_{\mathrm{F}}-k)R)-\mathrm{Ci}((k_{\mathrm{F}}+k)R)] + \cos(kR)[\mathrm{Si}((k_{\mathrm{F}}+k)R)+\mathrm{Si}((k_{\mathrm{F}}-k)R)]\}. \tag{24}$$

To apply this formulation k_{F} must be larger than k, i.e. dpl>2. Quasistatic FCD is obtained in the limit $kR\to 0$, which leads to a simpler expression [47]

$$\bar{\mathbf{G}}^{\mathrm{FCD,st}}_{ij} = -\frac{2}{3\pi R^3}\left(\bar{\mathbf{I}}-3\frac{\hat{R}\hat{R}}{R^2}\right)[3\mathrm{Si}(k_{\mathrm{F}}R)+k_{\mathrm{F}}R\cos(k_{\mathrm{F}}R)-4\sin(k_{\mathrm{F}}R)]. \tag{25}$$

Since FCD was originally designed for high refractive indices, we recommend using it especially in this regime. However, it also works fine for moderate refractive index, generally not worse than the standard approach of point dipoles [25]. Additional computational time for using FCD is comparable to a single iteration of the iterative solver, which is negligible in most cases.

The IGT directly accounts for the finiteness of the cubical dipole, by integrating over its volume V_j

$$\bar{\mathbf{G}}^{\mathrm{IGT}}_{ij} = \frac{1}{V_j}\int_{V_j} \mathrm{d}^3r'\bar{\mathbf{G}}(\mathbf{r}_i,\mathbf{r}'). \tag{26}$$

Implementation of the IGT in ADDA is based on the Fortran code kindly provided by IGT's original authors [16]. The IGT is known to perform very good for small scatterers with large and almost real refractive indices [16]. The choice of the interaction term is performed by the command line option

```
−int <type> [<arg1> [<arg2>]]
```

where `<type>` is one of the `poi`, `fcd`, `fcd_st`, `igt`. Two optional arguments are relevant only for `igt`. `<arg1>` is the maximum distance (in dipole sizes), for which integration is performed (for larger distances simpler equation (21) is used). The default formulation for interaction term is that of point dipoles (`poi`); however, it is expected to be inferior to `fcd` or `igt` in many cases. However, the latter two have been studied in much less details.

5.3. Calculating scattering quantities

The simplest way to calculate scattering quantities is to consider a set of point dipoles with known polarizations, as summarized by Draine [44]. The scattering amplitude **F** for any scattering direction **n** is given as

$$\mathbf{F}(\mathbf{n}) = -\mathrm{i}k^3(\bar{\mathbf{I}}-\hat{n}\hat{n})\sum_i \mathbf{P}_i \exp(-\mathrm{i}k\mathbf{r}_i\cdot\mathbf{n}). \quad (27)$$

The amplitude and Mueller scattering matrices for direction **n** are determined from **F**(**n**) calculated for two incident polarizations [39]. Scattering cross section C_{sca} and asymmetry vector **g** is determined by integration of **F**(**n**) over the whole solid angle:

$$C_{\mathrm{sca}} = \frac{1}{k^2}\oint \mathrm{d}\Omega \left|\mathbf{F}(\mathbf{n})\right|^2, \quad (28)$$

$$\mathbf{g} = \frac{1}{k^2 C_{\mathrm{sca}}}\oint \mathrm{d}\Omega\, \mathbf{n}\left|\mathbf{F}(\mathbf{n})\right|^2. \quad (29)$$

Extinction and absorption cross section (C_{ext} and C_{abs}) are determined directly from $\mathbf{P}_i$

$$C_{\mathrm{ext}} = 4\pi k \sum_i \mathrm{Im}(\mathbf{P}_i \cdot \mathbf{E}_i^{\mathrm{inc}*}), \quad (30)$$

$$C_{\mathrm{abs}} = 4\pi k \sum_i [\mathrm{Im}(\mathbf{P}_i \cdot \mathbf{E}_i^{\mathrm{exc}*}) - (2/3)k^3 |\mathbf{P}_i|^2]. \quad (31)$$

Variations of Eq. (31) (which is used by default) are possible, for instance [16]:

$$C_{\mathrm{abs}} = 4\pi k \sum_i \mathrm{Im}(\mathbf{P}_i \cdot \mathbf{E}_i^{*}), \quad (32)$$

which is based on considering radiation correction of a *finite* dipole instead of a *point* dipole used in Eq. (31). There is no difference between these two expressions for the CM, the RR, and the FCD polarizability formulations and also for real refractive indices [1]. The choice between these two options is performed by the command line option

```
-scat <type>
```

where `<type>` is `dr`, `fin`. Draine's classical formulation (`dr`) corresponds to Eqs. (27)–(31). Finite dipole correction (`fin`) uses Eq. (32) to calculate C_{abs}, and C_{ext} is obtained as Eq. (30) plus Eq. (32) minus Eq. (31). In other words, C_{ext} is corrected by the same amount as C_{abs} to ensure exact compliance with the optical theorem, which is discussed below.

C_{sca} can be determined as $C_{\mathrm{ext}}-C_{\mathrm{abs}}$, which is faster than Eq. (28). However, this issue needs further clarifying. Draine noted [44] that C_{sca} calculated by integration over the solid angle can be more accurate than $C_{\mathrm{ext}}-C_{\mathrm{abs}}$, due to loss of significant digits when the latter two cross sections have close to equal values. Moreover, it has been suggested [48] that the difference between C_{sca} calculated by these two methods can be used as an internal measure of DDA accuracy. However, we stress that this difference is a measure of convergence of the iterative solver and accuracy of the integration over the solid angle, but not of the physical approximation itself. In other words, the difference may be very small, while the values themselves are very inaccurate (compared to the exact solution). To prove this, one may consider a supplementary disconnected particle consisting of a set of point dipoles with the same positions and polarizabilities as dipoles representing the original (connected) particle. For this supplementary particle DDA equations (9) involve no approximations, since they directly follow from the constitutive equations of a point dipole. Therefore, their exact solution will satisfy the optical theorem, which can be formulated as $C_{\mathrm{sca}}=C_{\mathrm{ext}}-C_{\mathrm{abs}}$. Hence, the only possible reasons for the latter to be violated are inaccurate solution of Eq. (9) or inaccurate calculation of Eq. (28). And the difference between the original and supplementary particles, which is the error of the DDA itself, is not relevant. The simulations comply with this conclusion (data not shown).

6. What scattering quantities are calculated

6.1. Mueller and amplitude scattering matrices

ADDA calculates the complete Mueller scattering matrix [39] for a set of scattering angles (polar θ and azimuthal φ), which are specified with respect to the incident wave (Section 4.6). By default, scattering in the *yz*-plane is calculated. The range of [0°,180°] is equally divided into N_θ intervals, the latter is defined trough the command line option

```
-ntheta <arg>
```

If the particle is not symmetric and orientation averaging is not used, the range is extended to 360°. To calculate the Mueller matrix in one scattering plane ADDA simulates two incident polarizations, except when the particle is symmetric with respect to the rotation by 90° over the propagation vector of incident radiation. In the latter case one incident polarization and two scattering planes are used instead [7].

More advanced options are available to calculate scattering at any set of angles. If any of the two command line options

```
-store_scat_grid
-phi_integr <arg>
```

is specified, the Mueller matrix is calculated for a set of angles, specified in a special file. The first flag indicates that values of the Mueller matrix for all calculated angles should be saved to a file, while the second flag turns on the integration of Mueller matrix over φ with multipliers selected from 1, cos(2φ), sin(2φ), cos(4φ), and sin(4φ). For each multiplier a separate output file is produced.

ADDA can also calculate the amplitude scattering matrix [39] through the command line option

```
-scat_matr {muel|ampl|both|none}
```

which allows one to specify whether the Mueller matrix (`muel`, the default choice), the amplitude matrix (`ampl`), `both`, or `none` should be calculated and saved to file. The set of angles to calculate the amplitude matrix are determined the same way as for the Mueller matrix. However, no averaging of the amplitude matrix is performed—neither over orientation nor over the azimuthal scattering angle.

6.2. Integral scattering quantities

All scattering quantities described in this section are saved to several files, corresponding to each of two incident polarizations and to orientation-averaged results. ADDA always calculates C_{ext} and C_{abs} (together with corresponding efficiencies Q_{ext}, Q_{abs}). Optionally, it can calculate scattering cross section C_{sca} (and efficiency Q_{sca}) and normalized and non-normalized asymmetry vectors—$\mathbf{g}$ and $\mathbf{g}C_{\text{sca}}$, respectively (the z-component of $\mathbf{g}$ is the usual asymmetry parameter $<\cos\theta>$). Values of cross sections are in units of μm^2. All the efficiencies are calculated by dividing the corresponding cross section over the area of the geometrical cross section of the sphere with volume equal to that of dipole representation of the particle. Optional calculation of C_{sca}, $\mathbf{g}C_{\text{sca}}$ and $\mathbf{g}$ can be enabled, respectively, by command line options

```
−Csca
−vec
−asym
```

The calculation of $\mathbf{g}$ and C_{sca} is performed by integration over the whole solid angle using a grid of scattering angles.

In most cases C_{sca} can be accurately calculated as C_{ext}–C_{abs}, without using −Csca option. As discussed in Section 5.3, accuracy of this method can be improved when $C_{\text{sca}} \ll C_{\text{abs}}$ by using smaller $\varepsilon_{\text{iter}}$ (Section 7.1). Whether this is more computationally effective than using −Csca or not, depends on the particular problem. For instance, C_{ext} and C_{abs} may coincide in (almost) all digits for particles much smaller than the wavelength, leaving integration of scattered fields over the solid angle as the only viable option. However, in this case the integration can be done analytically, using trivial angular dependence of the Mueller matrix.

Radiation force for the whole scatterer and for each dipole can also be calculated by ADDA, using the command line options

```
−Cpr_mat
−store_force
```

However, the latter features are still under development. In particular, the FFT-acceleration of radiation-force calculation [6] has not yet been implemented, limiting its applicability to relatively small number of dipoles.

6.3. Internal and near-fields

ADDA can save internal electric fields $\mathbf{E}_i$ and/or dipole polarizations $\mathbf{P}_i$ at each dipole (see Section 5) using command line options

```
−store_int_field
−store_dip_pol
```

These options allow one to study in details the accuracy of the DDA (see e.g. [19]) or the physics behind it.

Currently, ADDA cannot calculate the field near the particle in a completely convenient manner. However, Fabio Della Sala and Stefania D'Agostino [33] have contributed a package `near_field`, which adds this functionality using the dipole polarizations calculated by ADDA. This package is distributed in the `misc/` folder, and is accompanied by all necessary instructions.

7. Computational issues

7.1. Solving linear system

The main computation of a DDA simulation, usually taking the major part the execution time, is finding a solution of a large system of linear equations. We use an alternative form of Eq. (9)

$$\sum_j \mathbf{A}_{ij}\mathbf{x}_j = \overline{\boldsymbol{\beta}}_i\mathbf{E}_i^{\text{inc}}, \quad \text{where } \overline{\boldsymbol{\alpha}}_i = \overline{\boldsymbol{\beta}}_i^T\overline{\boldsymbol{\beta}}_i,$$

$$\mathbf{x}_i = \overline{\boldsymbol{\beta}}_i\mathbf{E}_i = \overline{\boldsymbol{\beta}}_i^{-T}\mathbf{P}_i, \; \overline{\mathbf{A}}_{ij} = \overline{\mathbf{I}}\delta_{ij} - \overline{\boldsymbol{\beta}}_i\overline{\mathbf{G}}_{ij}\overline{\boldsymbol{\beta}}_j^T. \tag{33}$$

Decomposition of $\overline{\boldsymbol{\alpha}}_i$ is possible if and only if it is complex-symmetric. This is sufficient for the current version of ADDA, which supports only diagonal polarizability tensors. Since $\overline{\mathbf{G}}_{ij} = \overline{\mathbf{G}}_{ji}$, the interaction matrix $\mathbf{A}$ is complex-symmetric and Jacobi-preconditioned (has unity diagonal). ADDA incorporates four different iterative methods for solution of Eq. (33): conjugate gradient applied to normalized equations with minimization of the residual norm (CGNR) [49], Bi-conjugate gradient (Bi-CG) [50,51], Bi-CG stabilized (Bi-CGStab) [49] and quasiminimal residual (QMR) [50]. Bi-CG and QMR employ the complex-symmetric property of $\mathbf{A}$ to reduce the number of matrix–vector products per iteration by a factor of two [50].

Our experience suggests that QMR is usually the fastest iterative solver of these four, however in some cases Bi-CGStab may be faster. Performance of Bi-CG is comparable to that of the QMR, but convergence behavior of the former is erratic, similar to that of Bi-CGStab. While being the slowest of the four, CGNR however is very simple and its convergence is guaranteed to be monotonic [49]. QMR and Bi-CGStab require about 20% more RAM (for additional intermediate vectors, Section 3) as compared to CGNR and Bi-CG. Hence, Bi-CG may be preferred when memory is sparse.

The iterative solver is chosen by the command line option

```
−iter <type>
```

where <type> is one of: `cgnr`, `bicg`, `bi-cgstab`, `qmr`. By default QMR is used. The stopping criterion for iterative solvers is the relative norm of the residual. The process stops when this norm is less than $\varepsilon_{\text{iter}}$. The latter can be specified by the command line option

```
−eps <arg>
```

where $\varepsilon_{\text{iter}}=10^{-<\text{arg}>}$. By default, $\varepsilon_{\text{iter}}=10^{-5}$.

The iterative method only needs the interaction matrix for calculating matrix–vector products. This can be done in $O(N \ln N)$ operations (where N is the total number of dipoles) using the FFT [52]. 3D (parallel) FFT is used in ADDA by an explicit decomposition into a set of 1D FFTs, reducing calculations since only part of the array, on which FFT is performed, is actually used (see [5] for details). 1D FFTs are performed using standard libraries – two are implemented in ADDA: a routine by Temperton (`CFFT99`, [53]), or the more advanced package FFTW 3 [54]. The latter is generally significantly faster, but requires separate installation of the package. Symmetry of the DDA interaction matrix is used to reduce the storage space for the Fourier-transformed matrix.

7.2. Various other issues

ADDA can run on a multiprocessor system, parallelizing a single DDA simulation. It uses the message passing interface (MPI) for communication routines. This feature can be used both to accelerate the computations and to circumvent the single-computer limit of the available memory, thus enabling simulations with a huge number of dipoles. We do not discuss the parallel efficiency of ADDA, which largely depends on the hardware. However, in our experience it was usually larger than 70% on modern computer clusters.

To facilitate very long simulations ADDA incorporates a checkpoint system, which can be used to break a single simulation into smaller parts. The system is controlled by command line options:

```
−chpoint <time>
−chp_type <type>
```

where `<time>` is time in format "`#d#h#m#s`", and `<type>` is one of `normal`, `regular`, `always`. The latter command line option allows one to save the state of the iterative solver after the specified time elapses, in regular time intervals, or always after the completion of the iterative solver, respectively. Simulation is restarted from a checkpoint when

```
−chp_load
```

is used in the command line. It is important to note that currently only the state of the iterative solver is saved at a checkpoint, therefore it is suitable only for simulations for a single incident polarization.

Integration is performed in several parts of ADDA: orientation averaging, integration of the Mueller matrix over the azimuthal angle, and integration of the scattered field over the whole solid angle. The same routine is used for all these purposes, which is based on the one- or two-dimensional Romberg integration [55]. This is a high-order technique that may be used in adaptive mode (automatically performing only the necessary number of function evaluations to reach a prescribed accuracy). The latter is relevant for orientation averaging, where each function evaluation is a complete DDA simulation. Romberg integration also provides an estimate of the integration error, which is reliable for "sufficiently nice" functions [55]. The drawback is that argument values must be uniformly spaced and their total number is limited to be 2^n+1 (n is any integer).

ADDA automatically produces timing results for any simulation, consisting of about 15 time values covering all major parts of ADDA execution, which can be used for different optimization purposes. In parallel mode communication time (between different processors) is shown separately for some tasks, which can be used to estimate the parallel efficiency. Moreover, ADDA can be compiled is a special mode ("precise timing") to produce very detailed timing results for the matrix–vector product, which is the computationally most expensive part of an iterative solver.

8. Sample simulations

This section provides three ADDA simulation examples, highlighting some of its prominent features. Benchmarking of the code was reported in our previous publications [7,12,22–26,56], as well as in many papers by other researchers, who actually use ADDA for practical applications, e.g. [57–60]. Moreover, we maintain two wiki pages devoted to the largest ADDA simulations[4] and comparison with other codes and methods.[5]

The first example is a sphere with $x=320$ (diameter larger than 100 wavelengths) and $m=1.05$ (representative for biological particles in a watery environment). The simulation was run on MareNostrum,[6] using 512 processors (IBM Power PC 970MP, 2.3 GHz) and 700 GB of memory, and took 7.5 h. We used 1024 dipoles per diameter of a sphere resulting in a total number of occupied dipoles of half a billion. The extinction efficiency (equal to 1.963) was calculated with a relative error 0.02%. The comparison of simulated S_{11} with the Mie solution is shown in Fig. 3 and shows excellent agreement. To the best of our knowledge, this is the largest particle ever simulated with the DDA, and arguably also with any other rigorous volume-discretization method. We should note that highly optimized surface-based codes are capable of simulating at least 4 times larger *homogeneous* particles on comparable hardware [61]. However, such large homogeneous particles can be accurately and much faster solved with asymptotic (geometric-optics-based) methods [62].

The second example aims to demonstrate ADDA's capability in simulating scattering of a Gaussian beam. We simulated light scattering by a sphere with $x=5$ and $m=1.5$, illuminated by a Gaussian beam with a wavelength of 1 μm, a width of 2 μm, and with the position of the beam center at (1,1,1) μm. The beam propagates along the z-axis. We used 64 dipoles per diameter of the sphere and calculated $S_{11}(\theta)$ in the yz-plane. Results were compared with that of the multiple multipole program (MMP),[7] which were kindly provided by Roman Schuh and Thomas Wriedt (see Fig. 4). The agreement is excellent.

[4] http://code.google.com/p/a-dda/wiki/LargestSimulations
[5] http://code.google.com/p/a-dda/wiki/ComparisonOtherCodes
[6] http://www.bsc.es/plantillaA.php?cat_id=5
[7] http://www.scattport.org/index.php/programs-menu/generalized-multipole-menu/49-3d-mmp

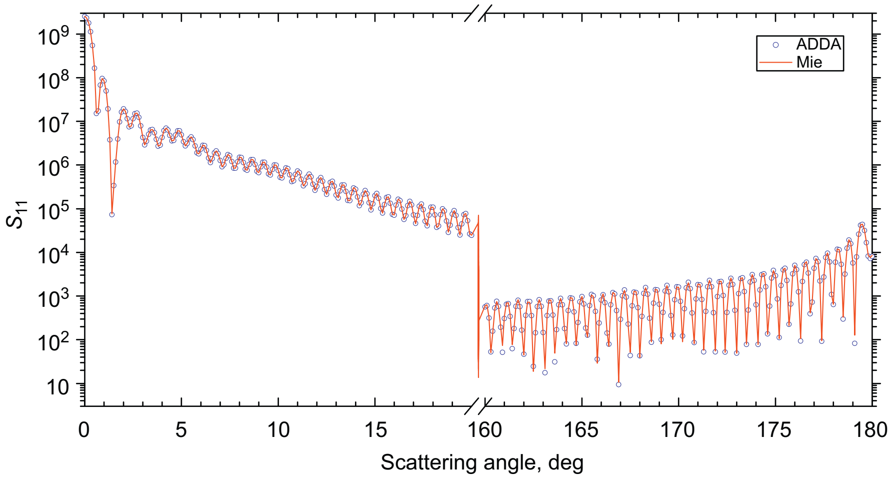

Fig. 3. Comparison of ADDA and Mie results of S_{11} near forward and backward directions for a sphere with $x=320$ and $m=1.05$ in logarithmic scale.

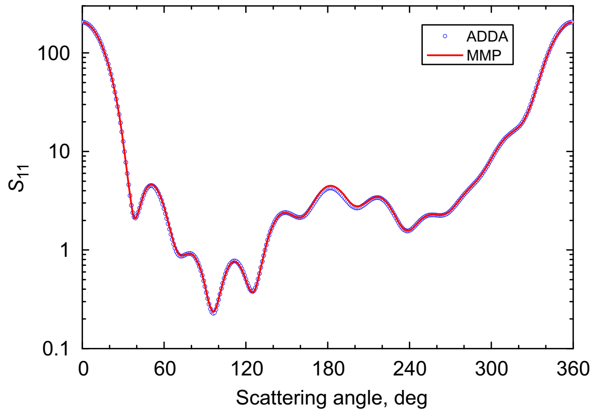

Fig. 4. Comparison of ADDA and MMP results of S_{11} for sphere with $x=5$ and $m=1.5$ in logarithmic scale. The sphere is illuminated by tightly focused off-center Gaussian beam (see text).

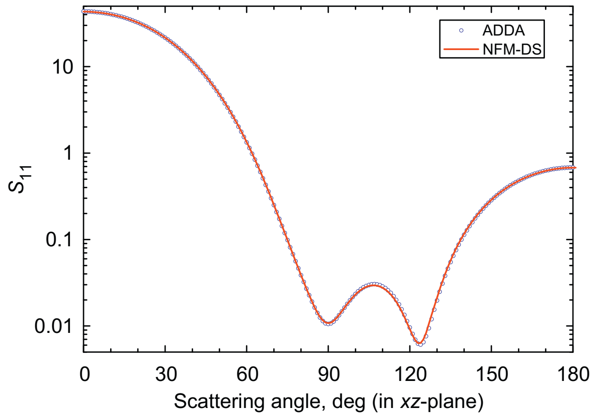

Fig. 5. Comparison of ADDA and NFM-DS results of S_{11} for ellipsoid with semi-axes (a,b,c) $ka=2$, $kb=3$, $kc=4$ and anisotropic refractive index $m_x=1.5$, $m_y=m_z=1.1$ in logarithmic scale.

The third example deals with anisotropic scatterers. We consider an ellipsoid with semi-axes (a,b,c) $ka=2$, $kb=3$, $kc=4$ and refractive index $m_x=1.5$, $m_y=m_z=1.1$. Incident wave propagates along the z-axis and S_{11} is calculated in xz-plane. Comparison of ADDA results (with default discretization) to that of null-field method with discrete sources (NFM-DS, [63]) is given in Fig. 5. The latter data was kindly provided by Vladimir Schmidt. Again, the agreement is excellent.

9. Conclusion

ADDA is a mature parallelized code applicable to a wide variety of problems related to interaction of electromagnetic waves with arbitrary shaped inhomogeneous particles. Its main features are parallel execution and incorporation of several state-of-the-art DDA formulations, which allow achieving high accuracy and speed in many different applications. The code is completely open-source with a transparent development process and a slowly increasing number of contributors. Moreover, a community of researchers is building up around the code, producing thorough discussions on different related topics.

ADDA has large development plans, exemplified by almost 90 open issues in the issue tracker.[8] They range from making some of the existing features fully operational to incorporating of completely new DDA formulations and improving ADDA efficiency on modern hardware (adding support e.g. for OpenMP and GPUs). We invite all researchers interested in DDA simulations to participate in the discussion group[9] and to consider contributing to ADDA.

Acknowledgments

We are grateful to many people who contributed their code to ADDA, directly or indirectly. Some of them are mentioned in the main text of this paper; the full list is maintained at http://code.google.com/p/a-dda/wiki/Acknowledgements.

[8] http://code.google.com/p/a-dda/issues

[9] http://groups.google.com/group/adda-discuss

All users of ADDA are acknowledged for positive feedback and numerous discussions. We also thank Roman Schuh, Thomas Wriedt, and Vladimir Schmidt for results of benchmark simulations with other methods.

M.Yu. acknowledges support of the program of the Russian Government "Research and educational personnel of innovative Russia" (Contract nos. P422, P1039, P2497 and 14.740.11.0729), Siberian Branch of the Russian Academy of Sciences (Grant nos. 2009-37 and 2009-7), Presidium of the Russian Academy of Sciences (Grant no. 2009-27-15), and program of the Russian President for the support of leading scientific schools (Grant no. NSh-65387.2010.4).

References

[1] Yurkin MA, Hoekstra AG. The discrete dipole approximation: an overview and recent developments. J Quant Spectrosc Radiat Transfer 2007;106:558–89.

[2] Purcell EM, Pennypacker CR. Scattering and adsorption of light by nonspherical dielectric grains. Astrophys J 1973;186:705–14.

[3] Hoekstra AG, Sloot PMA. New computational techniques to simulate light-scattering from arbitrary particles. Part Part Syst Charact 1994;11:189–93.

[4] Hoekstra AG, Sloot PMA. Coupled dipole simulations of elastic light scattering on parallel systems. Int J Mod Phys C 1995;6:663–79.

[5] Hoekstra AG, Grimminck MD, Sloot PMA. Large scale simulations of elastic light scattering by a fast discrete dipole approximation. Int J Mod Phys C 1998;9:87–102.

[6] Hoekstra AG, Frijlink M, Waters LBFM, Sloot PMA. Radiation forces in the discrete-dipole approximation. J Opt Soc Am A 2001;18: 1944–53.

[7] Yurkin MA, Maltsev VP, Hoekstra AG. The discrete dipole approximation for simulation of light scattering by particles much larger than the wavelength. J Quant Spectrosc Radiat Transfer 2007;106: 546–57.

[8] Yurkin MA, Hoekstra AG. User manual for the discrete dipole approximation code ADDA v.1.0 [Internet]. 2010 [cited 2010 November 17]. Available from: 〈http://a-dda.googlecode.com/svn/tags/rel_1.0/doc/manual.pdf〉.

[9] Draine BT, Flatau PJ. Discrete-dipole approximation for scattering calculations. J Opt Soc Am A 1994;11:1491–9.

[10] McDonald J, Golden A, Jennings SG. OpenDDA: a novel high-performance computational framework for the discrete dipole approximation. Int J High Perform Comput Appl 2009;23:42–61.

[11] Loke VLY, Menguc MP. Surface waves and atomic force microscope probe-particle near-field coupling: discrete dipole approximation with surface interaction. J Opt Soc Am A 2010;27:2293–303.

[12] Penttila A, Zubko E, Lumme K, Muinonen K, Yurkin MA, Draine BT, et al. Comparison between discrete dipole implementations and exact techniques. J Quant Spectrosc Radiat Transfer 2007;106:417–36.

[13] Draine BT, Goodman JJ. Beyond Clausius–Mossotti:wave propagation on a polarizable point lattice and the discrete dipole approximation. Astrophys J 1993;405:685–97.

[14] Draine BT. The discrete dipole approximation for light scattering by irregular targets. In: Mishchenko MI, Hovenier JW, Travis LD, editors. Light scattering by nonspherical particles, theory, measurements, and applications. New York: Academic Press; 2000. p. 131–45.

[15] Piller NB. Coupled-dipole approximation for high permittivity materials. Opt Commun 1999;160:10–4.

[16] Chaumet PC, Sentenac A, Rahmani A. Coupled dipole method for scatterers with large permittivity. Phys Rev E 2004;70:036606.

[17] Singham SB. Theoretical factors in modeling polarized light scattering by arbitrary particles. Appl Opt 1989;28:5058–64.

[18] Hoekstra AG, Sloot PMA. Dipolar unit size in coupled-dipole calculations of the scattering matrix elements. Opt Lett 1993;18: 1211–3.

[19] Hoekstra AG, Rahola J, Sloot PMA. Accuracy of internal fields in volume integral equation simulations of light scattering. Appl Opt 1998;37:8482–97.

[20] Ayranci I, Vaillon R, Selcuk N. Performance of discrete dipole approximation for prediction of amplitude and phase of electromagnetic scattering by particles. J Quant Spectrosc Radiat Transfer 2007;103:83–101.

[21] Piller NB, Martin OJF. Increasing the performance of the coupled-dipole approximation: a spectral approach. IEEE Trans Antennas Propag 1998;46:1126–37.

[22] Yurkin MA, de Kanter D, Hoekstra AG. Accuracy of the discrete dipole approximation for simulation of optical properties of gold nanoparticles. J Nanophoton 2010;4:041585-15.

[23] Yurkin MA, Maltsev VP, Hoekstra AG. Convergence of the discrete dipole approximation, II: an extrapolation technique to increase the accuracy. J Opt Soc Am A 2006;23:2592–601.

[24] Yurkin MA, Hoekstra AG, Brock RS, Lu JQ. Systematic comparison of the discrete dipole approximation and the finite difference time domain method for large dielectric scatterers. Opt Express 2007;15:17902–11.

[25] Yurkin MA, Min M, Hoekstra AG. Application of the discrete dipole approximation to very large refractive indices: filtered coupled dipoles revived. Phys Rev E 2010;82:036703-12.

[26] Gilev KV, Eremina E, Yurkin MA, Maltsev VP. Comparison of the discrete dipole approximation and the discrete source method for simulation of light scattering by red blood cells. Opt Express 2010;18:5681–90.

[27] Martin OJF. Efficient scattering calculations in complex backgrounds. AEU-Int J Electr Commun 2004;58:93–9.

[28] Schmehl R, Nebeker BM, Hirleman ED. Discrete-dipole approximation for scattering by features on surfaces by means of a two-dimensional fast Fourier transform technique. J Opt Soc Am A 1997;14:3026–36.

[29] Mackowski DW. A generalization of image theory to predict the interaction of multipole fields with plane surfaces. J Quant Spectrosc Radiat Transfer 2010;111:802–9.

[30] Wijers C. Rayleigh scattering from single-site polysylane adsorbed on silicon: theory. Surf Sci 1986;168:816–22.

[31] Parviainen H, Lumme K. Scattering from rough thin films: discrete-dipole-approximation simulations. J Opt Soc Am A 2008;25:90–7.

[32] Mackowski DW. Direct simulation of scattering and absorption by particle deposits. In: Videen G, Mishchenko M, Menguc MP, editors. Proceedings of the 10th conference on electromagnetic and light scattering, June 17, Bodrum, Turkey; 2007. p. 113–6.

[33] D'Agostino S, Pompa PP, Chiuri R, Phaneuf RJ, Britti DG, Rinaldi R, et al. Enhanced fluorescence by metal nanospheres on metal substrates. Opt Lett 2009;34:2381–3.

[34] Chaumet PC, Rahmani A, Bryant GW. Generalization of the coupled dipole method to periodic structures. Phys Rev B 2003;67:165404.

[35] Wijers CMJ, Emmett KME. Structural contribution to the anisotropic reflection from the Si (110) surface. Phys Scr 1988;38:435–40.

[36] Draine BT, Flatau PJ. Discrete-dipole approximation for periodic targets: theory and tests. J Opt Soc Am A 2008;25:2693–703.

[37] Draine BT, Flatau PJ. User guide for the discrete dipole approximation code DDSCAT 6.1 [Internet]. 2006 [cited 2010 November 17]. Available from: arXiv:ph/0409262v2.

[38] Mishchenko MI. Calculation of the amplitude matrix for a nonspherical particle in a fixed orientation. Appl Opt 2000;39:1026–31.

[39] Bohren CF, Huffman DR. In: Absorption and scattering of light by small particles. New York: Wiley; 1983.

[40] Gouesbet G, Maheu B, Grehan G. Light-scattering from a sphere arbitrarily located in a Gaussian beam, using a Bromwich formulation. J Opt Soc Am A 1988;5:1427–43.

[41] Davis LW. Theory of electromagnetic beams. Phys Rev A 1979;19: 1177–9.

[42] Barton JP, Alexander DR. Fifth-order corrected electromagnetic field components for a fundamental Gaussian beam. J Appl Phys 1989;66:2800–2.

[43] Jackson JD. In: Classical electrodynamics. 3rd ed. New York: Wiley; 1998.

[44] Draine BT. The discrete dipole approximation and its application to interstellar graphite grains. Astrophys J 1988;333:848–72.

[45] Gutkowicz-Krusin D, Draine BT. Propagation of electromagnetic waves on a rectangular lattice of polarizable points [Internet]. 2004 [cited 2010 November 17]. Available from: arXiv:ph/0403082.

[46] Piller NB. Influence of the edge meshes on the accuracy of the coupled-dipole approximation. Opt Lett 1997;22:1674–6.

[47] Gay-Balmaz P, Martin OJF. A. library for computing the filtered and non-filtered 3D Green's tensor associated with infinite homogeneous space and surfaces. Comput Phys Commun 2002;144: 111–20.

[48] Zubko E, Muinonen K, Shkuratov Y, Videen G, Nousiainen T. Scattering of light by roughened Gaussian random particles. J Quant Spectrosc Radiat Transfer 2007;106:604–15.

[49] Barrett R, Berry M, Chan TF, Demmel J, Donato J, Dongarra J, et al. In: Templates for the solution of linear systems: building blocks for iterative methods. 2nd ed. SIAM; 1994.
[50] Freund RW. Conjugate gradient-type methods for linear systems with complex symmetrical coefficient matrices. SIAM J Sci Statist Comput 1992;13:425–48.
[51] van der Vorst HA, Melissen JBM. A. Petrov–Galerkin type method for solving *Ax*=*b*, where *A* is symmetric complex. IEEE Trans Magn 1990;26:706–8.
[52] Goodman JJ, Draine BT, Flatau PJ. Application of fast-Fourier-transform techniques to the discrete-dipole approximation. Opt Lett 1991;16:1198–200.
[53] Temperton C. Self-sorting mixed-radix fast Fourier transforms. J Comput Phys 1983;52:1–23.
[54] Frigo M, Johnson SG. The design and implementation of FFTW3. IEEE Proc 2005;93:216–31.
[55] Davis PJ, Rabinowitz P. In: Methods of numerical integration. New York: Academic Press; 1975.
[56] Yurkin MA, Maltsev VP, Hoekstra AG. Convergence of the discrete dipole approximation, I: theoretical analysis. J Opt Soc Am A 2006;23:2578–91.
[57] Priezzhev AV, Nikitin SY, Lugovtsov AE. Ray-wave approximation for the calculation of laser light scattering by transparent dielectric particles, mimicking red blood cells or their aggregates. J Quant Spectrosc Radiat Transfer 2009;110:1535–44.
[58] Grynko Y, Pulbere S. Discrete dipole approximation simulations of light reflection from flat non-transparent particles with rough surfaces. J Quant Spectrosc Radiat Transfer 2009;110:1382–91.
[59] Bi L, Yang P, Kattawar GW, Kahn R. Modeling optical properties of mineral aerosol particles by using nonsymmetric hexahedra. Appl Opt 2010;49:334–42.
[60] Teschl F, Randeu WL, Teschl R. Microwave scattering from ice crystals: how much parameters can differ from equal volume spheres. Adv Geosci 2010;25:127–33.
[61] Taboada JM, Landesa L, Obelleiro F, Rodriguez JL, Bertolo JM, Araujo MG, et al. High scalability FMM-FFT electromagnetic solver for supercomputer systems. IEEE Antennas Propag Mag 2009;51:20–8.
[62] Bi L, Yang P, Kattawar GW, Kahn R. Single-scattering properties of triaxial ellipsoidal particles for a size parameter range from the Rayleigh to geometric-optics regimes. Appl Opt 2009;48:114–26.
[63] Schmidt V, Wriedt T. *T*-matrix method for biaxial anisotropic particles. J Quant Spectrosc Radiat Transfer 2009;110:1392–7.