depSolver v2.0

Document Revision 1.0
June 1, 2008

Carlos Rosales Fernández
carlos@ihpc.a-star.edu.sg

Heterogeneous Coupled Systems Team
Large-Scale Complex Systems
Institute of High Performance Computing
1 Science Park Road, #01-01 The Capricorn
Science Park II, Singapore 117528

Preface

`depSolver` is a Boundary Element Method solver for Laplace's equation in 3D AC electrostatic problems. I originally wrote this program to study the characteristics of different dielectrophoretic trap designs. A big advantage of using this code to calculate dielectrophoretic forces is that the user can specify which approximation to use for the calculation of the force. In this manner, without having to change anything in the code, one can use a simple dipolar or quadrupolar approximation to quickly explore the force generated on a sphere over a large volume of space, or use the more general calculation based on the integration of the Maxwell's stress tensor over an arbitrarily shaped particle. An example of research work done with this code is reference [1].

`depSolver` is very convenient for dielectrophoretic force calculations, but it is in no way limited to them. It can be used to solve a wide variety of electrostatic problems as long as they involve conductors were the electric potential is specified and all dielectrics in the system can be considered as piecewise homogeneous. The source code is written in C and uses separate functions for most tasks in order to make modifications easier.

Changes in Version 2.0 of depSolver are related to the code structure and the output options:

1. More sane header files in the Bjarne Stroustrup style. This should make modification of the code easier.
2. Similar functions are now bundled in common files, all of them shorter than 1000 lines. This is also to facilitate code modification.
3. A new output option is available with a format accepted by Paraview[2]. Visualization of the potential and the field is now much easier.

This manual consists of three parts: Part I describes how to prepare the inputs necessary for the program and how to post-process the solution; Part II describes the indirect formulation of the BEM for those interested in the details, and Part III gives details on the implementation. In order to use the code you striclty only need to read Part I, but I recommend anyone using the program to calculate dielectrophoretic forces to go through Chapter 9 in Part III of this document. Part II has been

included because most people are not familiar with the indirect formulation of the BEM, but you can safely ignore it if this does not apply to you.

The Institute of High Performance Computing (IHPC) distributes `depSolver` under a dual licensing policy. We believe in open source software for pushing the frontiers of science and engineering, and we encourage everyone to publish open source software under the GPL License. If you are developing and distributing open source applications under the GPL License, then you are free to use any element of `depSolver` under the GPL License. For OEMs, ISVs, and VARs who distribute any element of `depSolver` with their products, and do not license and distribute their source code under the GPL, IHPC provides a flexible OEM Commercial License.

Finally, although the GPL License does not require you to contact us, I would appreciate if you could send an email telling me that you are using `depSolver` and what application you are using the code for.

The complete package can be downloaded from `software.ihpc.a-star.edu.sg`.

Carlos Rosales Fernández
Singapore, June 1, 2008

Contents

Part I

User Guide

Chapter 1

Introduction and general remarks

`depSolver` is distributed as a series of C function files, a make file for compilation, a Patran [3] script for mesh generation called `bem_save.pcl`, and tools to modify the output files to set them in formats which are easily plotted in Matlab [4] or gnuplot [5]. The complete list of functions is collected in Appendix A.

1.1 Quick Install Guide

Unzip the `depSolver-1.0.tar.gz` file in a suitable directory. This can be done in a linux system by typing on the command line:

```
$ tar -zxvf depSolver-2.0.tar.gz
```

or alternatively using two separate commands:

```
$ gunzip depSolver-2.0.tar.gz
$ tar -xvf depSolver-2.0.tar
```

The following directories will be created:

`depSolver`	: Main program directory
`depSolver/bin`	: Directory where the binary files are stored after compilation
`depSolver/docs`	: Directory containing the code documentation
`depSolver/examples`	: Directory with examples to test the compiled code
`depSolver/src`	: Directory containing the C source code
`depSolver/utils`	: Directory with utilities for pre- and post-processing
`utils/depBreak`	: Utility to break data files into several units
`utils/fieldPost`	: Utility to format the field values for plotting in matlab
`utils/forcePost`	: Utility to format the force values for plotting in matlab
`utils/gplotFormat`	: Utility to format standard output files for 3D plotting in gnuplot with p
`utils/meshgen`	: Utility to generate planes of points for post-processing
`utils/meshgen-vtk`	: Utility to generate points for post-processing with Paraview
`utils/patran2bem`	: Utility to transform mesh from Patran format
`utils/potPost`	: Utility to format the potential values for plotting in matlab

In order to install and run `depSolver` all the `.c` and `.h` listed in appendix A are necessary. Make sure all the mentioned files are inside the `depSolver/src` directory. Then proceed with the following steps.

1.1.1 Compilation

Two versions of the code can be compiled. The basic version is called `depSolver`, and an additional version called `depSolver-sft` is also provided. Both versions of the code are identical except that in `depSolver-sft` it is assumed that a particle - which can be of any shape - is created in the geometry and the program will shift it as a rigid body following a vector given in the input file. Notice that this shift assumes that the particle is created at the center of coordinates. This version is useful to produme Maxwell Stress Tensor calculations of the dielectrophoretic force without having to create a new mesh for each particle position. It can be abused to work without a particle or using a zero displacement, but the basic `depSolver` is less confusing to use in those cases.

A makefile is provided for the compilation of the code. Inside `depSolver/src` type:

```
$ make depSolver
```

or

```
$ make depSolver-sft
```

This will compile using gcc with the flag -O3 and several other performance optimization flags. If for some reason you don't like this, or you would like to add an architecture-related flag simply change the makefile. The executable files are automatically saved to `depSolver/bin`.

To clean the directory of object files after the compilation, type the following:

```
$ make mrMonk
```

1.1.2 Running the solver

In order to run `depSolver` certain input files are needed. These files have the extension `.bem` and the ones needed for every run are:

`input.bem`	: Main input file
`nodes.bem`	: File containing the coordinates of the nodes in the mesh
`elems.bem`	: File containing the element connectivity of the mesh
`bcs.bem`	: File containing the boundary conditions at every node

There are also two optional files files which are only used for some calculations:

`forcepoints.bem`	: [opt] File containing the points where the force is calculated when using the multipolar approximation
`internal.bem`	: [opt] File containing the internal points where the potential or field (or both) are required

Notice that the names of all these files are read from `input.bem` and can be changed at will. Only the main input file `input.bem` must keep this name as it is hard coded. Once these files have been properly set – see the following section for details on the format and contents –, one can simply run `depSolver` in the background, since all output is directed to files and there is no interaction with the program while it runs. The output files generated by the program are also divided in those that are produced on every run:

`bem.log`	: Main log file, logs program advance and execution time
`solution.dat`	: Solution in the format `x y z Re[s] Im[s]`

And those that are produced only for certain input options:

`gmres.log`	: [opt] GMRES log file, registers the error per iteration
`field.dat`	: [opt] Contains the electric field at the required points
`force-mp.dat`	: [opt] Contains the DEP force (multipolar approximation)
`formce-mst.dat`	: [opt] Contains the DEP force (Maxwell's stress tensor method)
`potential.dat`	: [opt] Contains the potential at the required points

Chapter 2

Pre-Processing

This chapter describes how to set up the necessary input files.

2.1 Mesh Generation Using Patran

To generate the mesh and the boundary conditions using Patran, simply generate the geometry using a structural model and ensuring that the normals are outward-facing (otherwise go to *Elements* and use *Modify→Element→Reverse*).

Then go to *Loads* and use the *Displacement* type to set the boundary conditions in the conductors in the system as:

```
< 1 Re[V] 0 >
< 1 Im[V] 0 >
```

The first number (1) indicates the potential is given, and the last number can be anything, because it is not used.

Next, use the *Force* type to set the boundary conditions in the interfaces as:

```
< 0 0 interfaceID >
< 0 0 interfaceID >
```

In this case it is important that the first two values are zero, since they are the type

of boundary condition - the value of `vBCType[]` in the code - and the right hand side of the boundary equation - the value of `vB[]` in the code.

Once this is done run the command `!!input bem_save.pcl` in Patran's command line, and make sure that you receive a message saying that the compilation has been successful (this should be instantaneous). Then run `bem_save()` in the same command line of Patran. This saves the nodes, elements and boundary conditions in the files `nodes.out`, `elems.out` and `bcs.out`, and also stores information about the mesh - number of nodes, elements and nodes per element - in the file `mesh_info.out`.

2.2 Converting Patran output to the right format

In order to get the input files in the exact format needed for the `depSolver` executable a further step is necessary. Go to the directory called `depSolver/utils/p2b` and run:

```
$ ./p2b nodeNumber elemNumber elemType scaling bcsNumber reOrder
```

This needs the output files from Patran, `nodes.out`, `elems.out` and `bcs.out`, and yields the correctly formatted `nodes.bem`, `elems.bem` and `bcs.bem`.

This last step seems a bit unnecessary, but Patran uses a different order for the quadratic elements than `depSolver`, and setting the parameter `reOrder` to one transforms the element connectivity to the appropriate format for the program. It is also handy when one needs to test the same system but scaled at different sizes, because no re-meshing is necessary. Additionally this filter takes care of repeated nodes in the mesh left by Patran, and updates the corresponding references in the element connectivity file, resulting in a more stable system of equations (otherwise there would be repeated equations in the coefficient matrix, making the system not solvable!).

Full help can be obtained on screen by calling the program as: `./p2b -h`.

2.3 Generating evaluation points file

2.3.1 Standard Output

Inside `depSolver/utils/meshgen` there is an executable file that generates sets of points in constant planes. This is useful to generate the `internal.bem` and `forcepoints.bem` files. It is called as:

```
$ ./meshgen a nx ny nz n rmin rmax r lineNumber
```

This generates a regular 2D mesh with limits [(`xmin`,`xmax`):(`ymin`,`ymax`)] in the plane `constantPlane` = `r`, where `constantPlane` takes the values `x`, `y` or `z`. The parameter `lineNumber` can take values 0 or 1:

`lineNumber` = 0 indicates the line number will not be included in the file
`lineNumber` = 1 indicates the line number will be recorded in the file

Example of use:

```
$ ./mesh 1 z 100 20 -50 50 -10 10 0.0 1
```

Produces a mesh in the plane z = 0.0 in the range x = (-50,50), y = (-10,10), with 100 points in the x direction and 20 in the y direction. The line number is saved to the ouput file starting with 1.

The output file is called 'mesh.dat' for convenience. It can be called sucesively in order to produce a single file with all the necessary points, as long as 'a' contains the correct number of the first element of the plane for each call. If a 21x21 set of points is being generated the first call will be done with `a` = 1, the second with `a` = 442, etc ...

Full help can be obtained on screen by calling the program as: `./mesh -h`.

2.3.2 Paraview Formatted Output

Inside `depSolver/utils/meshgen-vtk` there is an executable file that generates sets of points in a given volume. This is useful to generate the `internal.bem` files with a VTK format, which can be read by Paraview. It is called as:

```
$ ./meshgen-vtk nx ny nz xmin xmax ymin ymax zmin zmax
```

This generates a regular 3D mesh with limits [(`xmin`,`xmax`):(`ymin`,`ymax`):(`zmin`,`zmax`)]

Example of use:

```
$ ./mesh 20 20 10 -50 50 -25 25 0 10
```

Produces a regular cartesian grid with 4000 points in the range x = (-50,50), y = (-25,25), z = (0,10). The line number is saved to the ouput file starting with 1. The two first lines of the file will contain the number of nodes and the spacing between points in each direction respectively.

The output file is called 'mesh.dat' for convenience.

Full help can be obtained on screen by calling the program as: `./meshgen-vtk -h`.

Chapter 3

Input File Format

The main input file, `input.bem`, is divided in several sections, each of them with a title to increase readability. C++ style comments are allowed in the file, either occupying a line of their own or situated after the data in a line. The same kind of comments may be included in any of the other `.bem` files. Blank lines can be used to make the file easier to read.

3.1 Nodes Section

The title `NODES` is typically used for this section of the file, as it contains the total umber of nodes in the mesh and the name of the file with the nodes positions and indexes. This section should look like:

```
NODES
nodeNumber
nodeFilename
```

The data file containing the nodes can have any name (up to 32 characters long), such as the mentioned `nodes.bem`, and must be written in the form:

```
nodeID x y z
```

And the nodes must be sequentially numbered from 1 to `nodeNumber`.

3.2 Elements Section

The title `ELEMENTS` is usually given to this section, that must contain the total number of elements in the mesh, the type of elements used, and the name of the file containing the element connectivity information. This section should look like:

```
ELEMENTS
elemNumber
elemType
elemFilename
```

Where `elemType` can be one of the following:

```
tria3   : Linear interpolation in triangles (3-noded triangles)
tria6   : Quadratic interpolation in triangles (6-noded triangles)
```

The actual name of the element type can be in uppercase, lowercase, or a mixture of the two, as long as the spelling is correct!

The file containing the element connectivity information must have the following structure:

```
elemID nodeID1 nodeID2 ...  nodeIDM
```

Where `M` is 3 for linear interpolation in triangles and 6 for quadratic interpolation. The elemID must range from 1 to `elemNumber`. Note that all numbers in this file should be integers.

3.3 Materials Section

This section contains the number of materials used in the simulation and the values of the electric conductivity and relative permittivity of each of them, usually under the title `MATERIALS`. This section should look like:

```
MATERIALS
matID sigma eps
```

Where `sigma` is the conductivity of the material, and `eps` its relative permittivity.

3.4 Interfaces Section

This sections carries the title `INTERFACES`, and contains the information about the interfaces between different materials in the following format:

```
INTERFACES
interfaceNumber
interfaceID mat1 mat2
```

Where `mat1` and `mat2` are the two materials that interface. No support for three material interfaces is provided in the code or the input files.

3.5 Problem Section

This section is named `PROBLEM` and contains the frequency of the external field and the name of the file that has the boundary conditions. It looks like this:

```
PROBLEM
frequency
bcsFilename
```

The file that contains the boundary conditions must be in the following format:

```
nodeID bcType value1Re value2Re
...
nodeID bcType value1Im value2Im
...
```

A dielectric interface is indicated by a `bcType` of 0, a `value1` of 0, and a `value2` equal to the `interfaceID` where the node belongs. All three values must be the same for the real and imaginary parts.

If the dielectric interface belongs to a particle and the force is calculated using the Maxwell Stress tensor method the `bcType` must be set to 6 in order to differentiate this surface from any other dielectric interfaces in the system. `value1` and `value2` must be set to 0 and the `interfaceID` where the node belongs as for any other dielectric interface.

A dirichlet boundary condition - in this case, potential given on a conductor - is indicated by a `bcType` of 1, a `value1Re` equal to the real part of the potential, and a `value1Im` equal to the imaginary part of the potential. The `value2` is not used in the code and must be omited from the file.

Notice that in the boundary conditions file the real part of the boundary conditions is written first, so that for N nodes there are 2N rows, the first N with the real components, the last N with the imaginary ones.

The values 2 to 5 for the boundary condition type are not used by the program and may be assigned to other boundary condition types at a later stage.

3.6 Re-positioning Section

This section must only be used with `depSolver-sft`. It contains the first node corresponding to the particle and the displacement vector for the shift:

```
REPOSITION
pNode
dx dy dz
```

where `pNode` is the last node of the mesh before the particle nodes (if the nodes in the electrodes go from 1 to 536 and the nodes in the particle from 537 to 752 we would use `pNode` = 536. In Patran the particle must be meshed last to use this option correctly. (`dx`,`dy`,`dz`) are teh displacements in each of the spatial directions.

3.7 Analysis Section

This is the most complicated section in the input file. It is usually titled `ANALYSIS`, and contains al the information relative to which solver to use, and what postprocessing to do with the solution. This section must follow the format below:

```
ANALYSIS
solver preCond nInit
analysisType
pointNumber a b c
forceFilename
```

where `forceFilename` is the file containing the points where the force is calculated (when using the multipolar approximation), and has the following format:

```
X1 Y1 Z1
...
Xpn Ypn Zpn
```

It is not always necessary to include all these parameters in the section. The particular set of them necessary for a calculation depends on the `solver` and `analysisType` requested. Let us examine the section line by line in more detail.

`solver` can be specified as:

`gaussBksb`	: Gauss elimination solver with partial pivoting (columns only), does not require the `preCond` and `nInit` parameters
`gmres`	: GMRES solver, requires the parameters `preCond` and `nInit`

The name of the solver can be in lowercase or uppercase. The parameters `preCond` and `nInit` must be used only with the GMRES solver. `preCond` takes value 0 if no pre-conditioning is required and value 1 for Jacobi pre-conditioning (recommended). `nInit` takes the value 0 if no initial guess is given for the solution, and the number of nodes where the solution is provided otherwise. The file containing the solution must be named `solution.init`, and its format must be:

```
x y z Re[s] Im[s]
```

Where `Re[s]` and `Im[s]` are the real and imaginary parts of the solution at the point `(x,y,z)`, and the file has `nInit` rows.

A good practice is to solve the problem of an empty trap first specifying `nInit` as zero, and then use the solution as initial guess for a trap containing the particle. Note that due to the way the program works one should use exactly the same mesh (apart from the addition of the particle) in both cases. Normally one would generate the mesh for the empty trap, run it, then generate the mesh for the trap with te particle but taking care of meshing in the same order as before, leaving the mesh

on the particle surface to the end. This guarantees the program will work correctly. If in doubt set `nInit` to zero.

The `analysisType` is an integer that can take the following values:

0 : Calculate only the potential ϕ
1 : Calculate only the electric field $\vec{E}$
2 : Calculate ϕ and $\vec{E}$
3 : Calculate ϕ, $\vec{E}$, and $\vec{F}_{\mathrm{DEP}}$ using Maxwell's stress tensor method
4 : Calculate only $\vec{F}_{\mathrm{DEP}}$ using Maxwell's stress tensor method
5 : Calculate ϕ, $\vec{E}$, and $\vec{F}_{\mathrm{DEP}}$ using a dipolar approximation on a sphere
6 : Calculate ϕ, $\vec{E}$, and $\vec{F}_{\mathrm{DEP}}$ using a quadrupolar approx. on a sphere
7 : Calculate ϕ, $\vec{E}$, and $\vec{F}_{\mathrm{DEP}}$ using an octupolar approx. on a sphere
8 : Not available (reserved for multipolar approx. of order 4 on a sphere)
9 : Not available (reserved for multipolar approx. of order 5 on a sphere)
10 : Calculate ϕ, $\vec{E}$, and $\vec{F}_{\mathrm{DEP}}$ using a dipolar approx. for a homogeneous ellipsoid
11 : Calculate ϕ, $\vec{E}$, and $\vec{F}_{\mathrm{DEP}}$ using a dipolar approx. for a single-shelled ellipsoid

A quadrupolar approximation in this context indicates that dipole + quadrupole terms are included in the calculation, while an octupolar approximation indicates the inclusion of dipolar + quadrupolar + octupolar. When `analysisType` takes values [0–4] the parameters `(a,b,c)` do not need to be included in the section. When the value is between 5 and 9 only `a` has to be specified, and it represents the radius of the spherical particle under consideration. If the calculation of the force is done for an ellipsoid and options 10 or 11 are chosen, all `(a,b,c)` parameters must be specified, and they represent the three semiaxes of the ellipsoid.

The parameter `pointNumber` is only used with options [5–11] and indicates on how many positions of the centre of the sphere/ellipsoid should the force be calculated. The particular positions to use are indicater in the following lines by `(x,y,z)`. If options [1–4] are chosen for the type of analysis neither `pointNumber` nor these last lines should not be included in the file.

3.8 Internal Points Section

This section is optional, and only necessary when the potential or the electric field are going to be calculated. It is usually title `INTERNALPOINTS` and has the following

format:

```
INTERNALPOINTS
internalPointsNumber
internalPointsFilename
```

Where `internalPointsNumber` is the total number of rows in the file `internalPointsFilename`, which has the following structure:

```
pointID x y z
```

And the prefered structure is to keep z fixed, y fixed, x fixed, because in this way slices at different constant z are kept separated and are easy to postprocess later on.

3.9 Columns Section

This section is provided in order to be able to calculate properly the electric field and temperature in a channel with cylindrical or square electrode columns. The format is simply:

```
COLUMNS
colType
nColumns
X1 Y1 R1
...
XnCol YnCol RnCol
```

where `nColumns` is the total number of columns in the domain, and `colType` is the type of column (1 for cylindrical columns, 2 for square columns).

In the case of cylindrical columns `Xi` and `Yi` are the positions of the ith column centre and `Ri` is the radius of the ith column.

In the case of square cross-section columns `Xi` and `Yi` are the positions of the ith column centre and `Ri` is half the side length of the ith column.

If `nColumns` is zero the rest of the section will be ignored by the program.

Chapter 4

Post-Processing

This section describes how to use the tools in the `utils` folder in order to manipulate the program's output. There are four main utilities called `break`, `gplotForm`, `pot-post` and `field-post`.

4.1 Break

This program breaks a single output data file into several independent files. This is useful when the output includes calculations of the potential and the field in several planes and it is necessary to have the results from each plane in an independent file in order to plot them. It resides in `depSolver/utils/breaker` and must be called as:

```
$ ./break fileName fileNumber colNumber rowNumber
```

where `fileName` is the name of the file to break up, `fileNumber` is the number of output files, `colNumber` is the number of columns in the input file, and `rowNumber` is the number of rows in each of the output files. The output files will be named `data1`, `data2`, ..., `datafileNumber`. The input file is not modified.

If called as `./break -h` it will print a short help to the screen.

4.2 gplotForm

In the directory `depSolver/utils/gplotFormat` there is an executable called `gplotForm` that allows to separate any given data file with an arbitratry number of rows into blocks of a fixed size. This can be used for plotting a set of data corresponding to a plane, for example $z = 0$, in gnuplot with the pm3d option. The utility is called as:

```
$ ./gplotForm filename nRows nBlock nCols
```

Where `filename` is the name of the file to separate into blocks, `nRows` is the total number of rows in the file, `nBlock` is the size of the blocks to make, and nCols the number of columns in the file. This produces as output the file `temp.gnu` with the block-separated data that has a blank line every nBlock lines of the original file.

Let's assume that we generated the internal points file using the utility `meshgen` described in section [2.3], and that we asked meshgen to generate 100 points in the x direction and 20 in the y direction as in the example of use given. Because the total number of points is $100 \times 20 = 2000$ and we only have 4 columns (x,y,z,T) in the temperature data file we call separate as:

```
$ ./gplotForm potential.dat 2000 20 4
```

The output file can be now used in gnuplot to produce a density plot. First run gnuplot by calling it from the command line:

```
$ gnuplot
```

Once inside gnuplot type:

```
gnuplot> set pm3d
gnuplot> splot 'temp.gnu' u 1:2:3:4 w pm3d
```

if you only want a xy surface plot then use instead:

```
gnuplot> set pm3d map
gnuplot> splot 'temp.gnu' u 1:2:4 w pm3d
```

If called as `./gplotForm -h` it will print a short help to the screen.

4.3 FieldPost

To process the file `field.dat` you must go into directory `depSolver/utils/fieldPost` and run:

```
$ ./fieldPost nx ny nxi nyi
```

Where `nx` and `ny` are the number of points in the plane in the x and y directions –or xz, yz– and `nxi` and `nyi` are the number of interpolation points desired in each direction. The input file must be called `field.dat` and can be copied directly from the main program's output without changes. If several planes of data were calculated it is necessary to first call `break` –see section [4.1]– in order to get the independent files for each plane, and then name one of this files as `field.dat` before running `fieldpr`. The outputs from this will be:

`E.dat` : File containing the modulus of the electric field as a `nx` by `ny` matrix
`X.dat` : File containing the x position vector of size `nx` (monotonically increasing)
`Y.dat` : File containing the y position vector of size `ny` (monotonically increasing)
`XI.dat` : File containing the interpolated x position vector of size `nxi`
`YI.dat` : File containing the interpolated y position vector of size `nyi`

This is useful for later post-processing in matlab, since the vectors can be imported and then an interpolation done with the commands:

```
>> [X,Y] = meshgrid(X,Y);
>> [XI,YI] = meshgrid(XI,YI);
>> EI = interp2(X,Y,E,XI,YI,'cubic');
```

And then plotted using:

```
>> surf(XI,YI,VI)
>> contourf(XI,YI,EI,25)
```

Notice that the parser will get rid of the constant valued coordinate but will not store it anywhere, so the user must be aware of what constant plane he is using and rename the output files correspondingly.

Full help can be obtained on screen by calling the program as: `./fieldPost -h`.

4.4 ForcePost

To process the file `force.dat` you must go into directory `depSolver/utils/forcePost` and run:

```
$ ./forcePost nx ny nxi nyi step
```

Where `nx` and `ny` are the number of points in the x and y directions respectively, and `nxi` and `nyi` are the number of required interpolated values. The input `step` is used to control how many of the points are included in the files for the arrow plot data. Using `step` = 1 saves all data, using `step` = 2 saves 1 point of every 4 (divides by two in both x and y) and so on. This is useful because too many points produce very messy arrow plots.

The program produces the following outputs:

`X.dat`	: File containing the x coordinate vector of size `nxny`/`step`^{2}
`Y.dat`	: File containing the y coordinate vector of size `nxny`/`step`^{2}
`FX.dat`	: File containing F_x as a vector of size `nxny`/`step`^{2}
`FY.dat`	: File containing F_y as a vector of size `nxny`/`step`^{2}
`FZ.dat`	: File containing F_z as a vector of size `nxny`/`step`^{2}
`XX.dat`	: File containing the x coordinate vector of size `nx`
`YY.dat`	: File containing the y coordinate vector of size `ny`
`XXI.dat`	: File containing the interpolated x coordinate vector of size `nxi`
`YYI.dat`	: File containing the interpolated y coordinate vector of size `nyi`
`FXX.dat`	: File containing F_x as a vector of size `nxny`
`FYY.dat`	: File containing F_y as a vector of size `nxny`

The files `X.dat`, `Y.dat`, `FX.dat`, `FY.dat`, and `Fz.dat` can be used directly in order to obtain arrow plots in matlab by using the following commands:

```
>> quiver(X,Y,FX,FY,3,'k')
```

where 3 is the arrow size and `'k'` indicates that the arrows are black.

A nice figure is made by superimposing an arrow plot of the force over a density plot of the electric field intensity. This is achieved by typing (if `X.dat` and `Y.dat` correspond to the electric field values):

```
>> [X,Y] = meshgrid(X,Y);
>> [XI,YI] = meshgrid(XI,YI);
>> EI = interp2(X,Y,E,XI,YI,'cubic);
>> contourf(XI,YI,EI,35)
>> hold on
>> quiver(X,Y,FX,FY,3,'k')
```

Full help can be obtained on screen by calling the program as: `./forcePost -h`.

4.5 PotPost

In order to divide the file `potential.dat` into files containing only a plane you must go into directory `depSolver/utils/potPost` and run:

```
$ ./potPost nx ny nxi nyi
```

Where `nx` and `ny` are the number of points in the plane in the x and y directions –or xz, yz– and `nxi` and `nyi` are the number of interpolation points desired in each direction. The outputs from this will be:

`V.dat`	: File containing the real part of the potential as a `nx` by `ny` matrix
`X.dat`	: File containing the x position vector of size `nx` (monotonically increasing)
`Y.dat`	: File containing the y position vector of size `ny` (monotonically increasing)
`XI.dat`	: File containing the interpolated x position vector of size `nxi`
`YI.dat`	: File containing the interpolated y position vector of size `nyi`

After `potPost` has run we can use matlab to produce figures of the field intensity in the same manner as we did with the electric field. The input file is not modified by the program.

Notice that the parser will get rid of the constant valued coordinate but will not store it anywhere, so the user must be aware of what constant plane he is using and rename the output files correspondingly. The input file is not modified.

Full help can be obtained on screen by calling the program as: `./potPost -h`.

Chapter 5

Example Files

There are four example sets of input and output files in `/depSolver/examples`. These files are provided as a way to become familiar with the input file format as well as a handy test of your executable. After compiling you may want to run one of the examples and compare the outputs you obtain with the outputs provided. Although small differences in the last digits may be due to different compilers / architectures, a significant difference will indicate a problem. Please contact the developer with details of the problem if this happens.

The examples provided are:

capacitor-1: Simple parallel plate capacitor example. Examples for linear and quadratic elements provided.

capacitor-2: Parallel plate capacitor with two different dielectric materials sandwidched between the plates. Examples for linear and quadratic elements provided.

dep-trap-1: Example of an empty dielectrophoretic trap with eight circular electrodes and three different dielectric materials: the suspension fluid, the substrate, and the particle (calculated in the multipolar approximation).

dep-trap-2: Example of a dielectrophoretic trap with a spherical particle in the center. This is a small exampled to use with depSolver-stk.

Part II

The Indirect Boundary Element Formulation

Chapter 6

Indirect Formulation of the Boundary Element Method

In this section the indirect formulation of the boundary element method is derived from the well-known direct formulation. Both Laplace and Poisson equations are treated in detail.

6.1 Laplace Equation in the IBEM Formulation

There is plenty of literature on the direct boundary element method formulation (DBEM), and the interested reader can find good reviews in the classical texts by Brebbia [6] and Wrobel [7]. In this section the equations for the indirect boundary element method (IBEM) will be derived from the well stablished equations for the DBEM.

Let us study a potential ϕ that obeys Laplace equation $\nabla^2\phi = 0$ in all space. Consider a bounded domain Ω enclosed by a smooth boundary Γ, and surrounded by its complementary domain $\overline{\Omega} = \mathbb{R}^3 - \Omega$ as shown in Figure 6.1.

The integral equation describing the potential at any point in the domain Ω is given by:

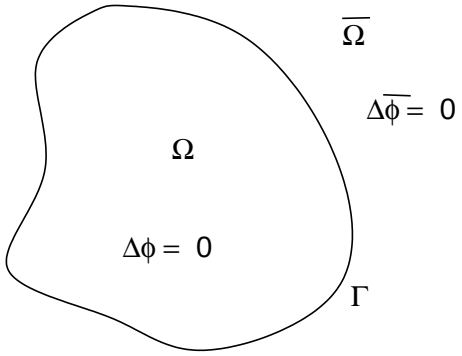

Figure 6.1: Domain used for the definition of the equations in the indirect method.

$$c(\vec{r})\phi(\vec{r}) + \int_\Gamma \phi(\vec{r}')H(\vec{r},\vec{r}')d\Gamma' = \int_\Gamma q(\vec{r}')G(\vec{r},\vec{r}')d\Gamma' \tag{6.1}$$

where $G(\vec{r},\vec{r}')$ is the Green's function of the problem to solve, and $H(\vec{r},\vec{r}')$ its normal derivative in the direction pointing outwards of the domain. $q(\vec{r}')$ is defined as the normal derivative of the potential $\partial_{\mathrm{n}}\phi$ with the normal pointing outwards of the domain. $c(\vec{r})$ is given – for smooth surfaces – by:

$$c(\vec{r}) = \begin{cases} 1 & \text{if } \vec{r} \in \Omega \\ 1/2 & \text{if } \vec{r} \in \Gamma \\ 0 & \text{if } \vec{r} \in \overline{\Omega} \end{cases} \tag{6.2}$$

The Green's function for free space is defined as the solution to the problem:

$$\nabla^2 G(\vec{d}) = -\delta(\vec{d}) \tag{6.3}$$

which has the following forms in two and three dimensions:

$$G^{2D}(\vec{d}) = \frac{1}{2\pi}\ln\left(\frac{1}{d}\right) \tag{6.4}$$

$$G^{3D}(\vec{d}) = \frac{1}{4\pi d} \tag{6.5}$$

where we have defined the vector $\vec{d}$ as the relative distance between two vectors $\vec{r}$ and $\vec{r}'$:

$$\vec{d} = \vec{r} - \vec{r}' \tag{6.6}$$
$$d = |\vec{r} - \vec{r}'| \tag{6.7}$$

The normal derivative of the Green's function is obtained from equations (6.4) and (6.5) as:

$$H^{2D}(\vec{d}) = \frac{1}{2\pi}\frac{\vec{d}\cdot\hat{n}}{d^2} \tag{6.8}$$
$$H^{3D}(\vec{d}) = \frac{1}{4\pi}\frac{\vec{d}\cdot\hat{n}}{d^3} \tag{6.9}$$

Let $\overline{\phi}$ denote the solution to Laplace equation in the complementary domain $\overline{\Omega}$, and $\overline{q}$ be its normal derivative. The potential in the complementary domain must obey the following expression:

$$\overline{c}(\vec{r})\overline{\phi}(\vec{r}) + \int_\Gamma \overline{\phi}(\vec{r}')H(\vec{r},\vec{r}')d\Gamma' = \int_\Gamma \overline{q}(\vec{r}')G(\vec{r},\vec{r}')d\Gamma' \tag{6.10}$$

For any point with position $\vec{r}$ interior to Ω we have $c(\vec{r}) = 1$ and $\overline{c}(\vec{r}) = 0$, so we can re-write equations (6.1) and (6.10) as:

$$\phi(\vec{r}) + \int_\Gamma \phi(\vec{r}')H(\vec{r},\vec{r}')d\Gamma' = \int_\Gamma q(\vec{r}')G(\vec{r},\vec{r}')d\Gamma' \tag{6.11}$$
$$\int_\Gamma \overline{\phi}(\vec{r}')H(\vec{r},\vec{r}')d\Gamma' = \int_\Gamma \overline{q}(\vec{r}')G(\vec{r},\vec{r}')d\Gamma' \tag{6.12}$$

Taking now the sum of these two equations, and using both in q and $\overline{q}$ the normal exterior to Ω, we obtain the following expression:

$$\phi(\vec{r}) + \int_\Gamma \left[\phi(\vec{r}') - \overline{\phi}(\vec{r}')\right] H(\vec{r},\vec{r}')d\Gamma' = \int_\Gamma \left[q(\vec{r}') + \overline{q}(\vec{r}')\right] G(\vec{r},\vec{r}')d\Gamma' \tag{6.13}$$

This is the general expression for the potential in the indirect boundary element method, where the integral on the right hand side can be considered as the contribution of a distribution of sources, and the integral on the left hand side as a the contribution from a distribution of dipoles. Let us define the following source and dipole densities:

$$s(\vec{r}') = \overline{q}(\vec{r}') + q(\vec{r}') \tag{6.14}$$
$$m(\vec{r}') = \overline{\phi}(\vec{r}') - \phi(\vec{r}') \tag{6.15}$$

Substituting now these definitions into the previous expression we obtain a more familiar result:

$$\phi(\vec{r}) = \int_\Gamma m(\vec{r}')H(\vec{r},\vec{r}')d\Gamma' + \int_\Gamma s(\vec{r}')G(\vec{r},\vec{r}')d\Gamma' \tag{6.16}$$

Comparing expressions (6.1) and (6.16) we see that they are formally identical, but that in the case of the IBEM we need to solve for twice as many unknowns. In principle, this would leave us with an underspecified system, but we can get around this problem by espcifying an extra boundary condition on each node.

This additional boundary condition will depend on what physical quantity we are trying to calculate. When the condition specified is the continuity of the potential we find that $m(\vec{r}') = 0$ and the problem is reduced to a *source formulation*:

$$\phi(\vec{r}) = \int_\Gamma s(\vec{r}')G(\vec{r},\vec{r}')d\Gamma' \tag{6.17}$$

When the condition specified is the continuity of the normal derivative of the potential we find that $s(\vec{r}') = 0$ and the problem is reduced to a *dipole formulation*:

$$\phi(\vec{r}) = \int_\Gamma m(\vec{r}')H(\vec{r},\vec{r}')d\Gamma' \tag{6.18}$$

6.2 Poisson Equation in the IBEM Formulation

Let us consider now the case where there is a certain distribution of sources in space, such that the potential obeys the Poisson equation, $\nabla^2\phi = b(\vec{r})$, as shown in Figure 6.2.

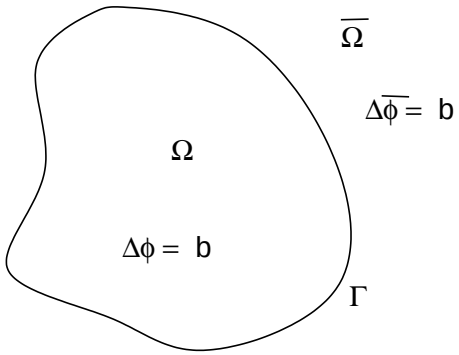

Figure 6.2: The Poisson problem domain for the definition of the IBEM equations.

In this case the equation that must be obeyed by the potential at any point in the domain Ω is given by:

$$c(\vec{r})\phi(\vec{r}) + \int_\Gamma \phi(\vec{r}')H(\vec{r},\vec{r}')d\Gamma' = \int_\Gamma q(\vec{r}')G(\vec{r},\vec{r}')d\Gamma' - \int_\Omega b(\vec{r}')G(\vec{r},\vec{r}')d\Omega' \quad (6.19)$$

For the complementary domain we have a similar expression:

$$\overline{c}(\vec{r})\overline{\phi}(\vec{r}) + \int_\Gamma \overline{\phi}(\vec{r}')H(\vec{r},\vec{r}')d\Gamma' = \int_\Gamma \overline{q}(\vec{r}')G(\vec{r},\vec{r}')d\Gamma' - \int_{\overline{\Omega}} b(\vec{r}')G(\vec{r},\vec{r}')d\Omega' \quad (6.20)$$

Just like inthe previous section, for any point at position $\vec{r}$ interior to Ω we have $c(\vec{r}) = 1$ and $\overline{c}(\vec{r}) = 0$, so we can re-write equations (6.19) and (6.20) as:

$$\phi(\vec{r}) \quad + \quad \int_\Gamma \phi(\vec{r}')H(\vec{r},\vec{r}')d\Gamma' = \int_\Gamma q(\vec{r}')G(\vec{r},\vec{r}')d\Gamma' - \int_\Omega b(\vec{r}')G(\vec{r},\vec{r}')d\Omega' \quad (6.21)$$

$$\int_\Gamma \overline{\phi}(\vec{r}')H(\vec{r},\vec{r}')d\Gamma' = \int_\Gamma \overline{q}(\vec{r}')G(\vec{r},\vec{r}')d\Gamma' - \int_{\overline{\Omega}} b(\vec{r}')G(\vec{r},\vec{r}')d\Omega' \quad (6.22)$$

Preceeding as in section 6.1 we add these two equations, and using both in q and $\overline{q}$ the normal exterior to Ω, we obtain the following expression:

$$\phi(\vec{r}) + \int_\Gamma \left[\phi(\vec{r}') - \overline{\phi}(\vec{r}')\right] H(\vec{r},\vec{r}')d\Gamma' = \int_\Gamma \left[q(\vec{r}') + \overline{q}(\vec{r}')\right] G(\vec{r},\vec{r}')d\Gamma' - \int_{\Omega+\overline{\Omega}} b(\vec{r}')G(\vec{r},\vec{r}')d\Omega' \quad (6.23)$$

Using the definitions (6.14) and (6.15) we can re-write this expression with a source term and a dipole term:

$$\phi(\vec{r}) = \int_\Gamma m(\vec{r}')H(\vec{r},\vec{r}')d\Gamma' + \int_\Gamma s(\vec{r}')G(\vec{r},\vec{r}')d\Gamma' - \int_{\Omega+\overline{\Omega}} b(\vec{r}')G(\vec{r},\vec{r}')d\Omega' \quad (6.24)$$

As expected, this expression is very similar to the one obtained in the case of Laplace equation, with the exception of the domain integral due to the body force term.

Part III

Implementation Details

Chapter 7

Element Library

We have used the following convention for the nodes in the elements:

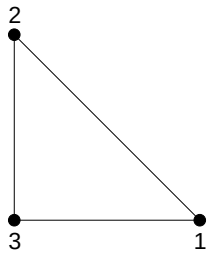

Figure 7.1: Linear interpolation.

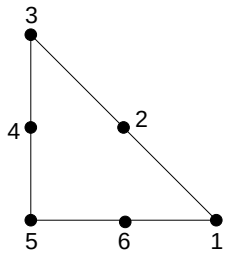

Figure 7.2: Quadratic interpolation.

Using this convention the shape functions for the linear case are simply given by:

$$N_1 = L_1 \tag{7.1}$$
$$N_2 = L_2 \tag{7.2}$$
$$N_3 = 1 - L_1 - L_2 \tag{7.3}$$

Under this convention the quadratic interpolation shape functions are given by:

$$N_1 = L_1(2L_1 - 1) \tag{7.4}$$

$$
\begin{aligned}
N_2 &= 4L_1L_2 && (7.5)\\
N_3 &= L_2(2L_2-1) && (7.6)\\
N_4 &= 4L2(1-L_1-L_2) && (7.7)\\
N_5 &= L_3[1-2(L_1+L_2)] && (7.8)\\
N_6 &= 4L_1(1-L_1-L_2) && (7.9)
\end{aligned}
$$

Chapter 8

Numerical Integration

8.1 Regular integrals

For non-singular integrals the integration is done using gaussian quadrature with NG integration points per element as in:

$$\int_{-1}^{1} F(L_1, L_2) dL_1 dL_2 \approx \sum_{i=1}^{NG} F(L_1^i, L_2^i) w_i \tag{8.1}$$

By default the program uses 7 points per triangular element. Tests where done with 16 and 64 points per element and the accuracy of the results was not affected, so 7 points were kept for speed.

8.2 Weakly singular integrals

For weakly singular integrals the integration is done through a regularization transformation that eliminates the singularity. The element is transformed into a triangle with a singularity in node 1 and then into a degenerate square. In the degenerate square we can use Gauss-Jacobi integration to integrate getting rid of the singularity. See Figure 8.1 for the transformation.

$$\int_{-1}^{1} F(L_1, L_2)(1 + L_2) dL_1 dL_2 \approx \sum_{i=1}^{NG} \sum_{j=1}^{NG} F(L_1^i, L_2^j) w_i^{\text{Gauss}} w_j^{\text{Gauss-Jacobi}} \tag{8.2}$$

Notice that the Gauss-Jacobi integration is necessary in only one of the directions, so in the other the standard Gauss quadrature on a line is used and the product of the two provides the correct integration as indicated by the expression above.

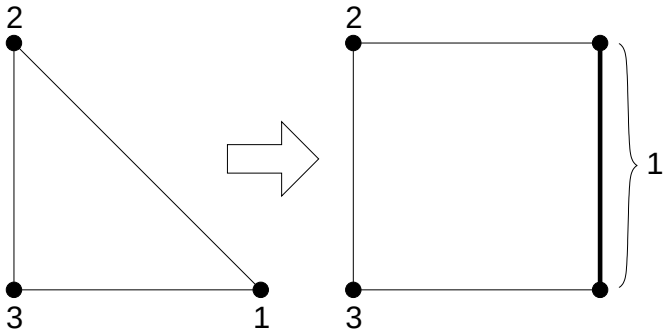

Figure 8.1: Regularization transformation for weakly singular integrals (linear case).

In the case of quadratic triangles when the singular point is on an edge of the triangle rather than on a vertex the triangle is divided in two and then the same regularization procedure is applied to both subtriangles as shown in Figure 8.2.

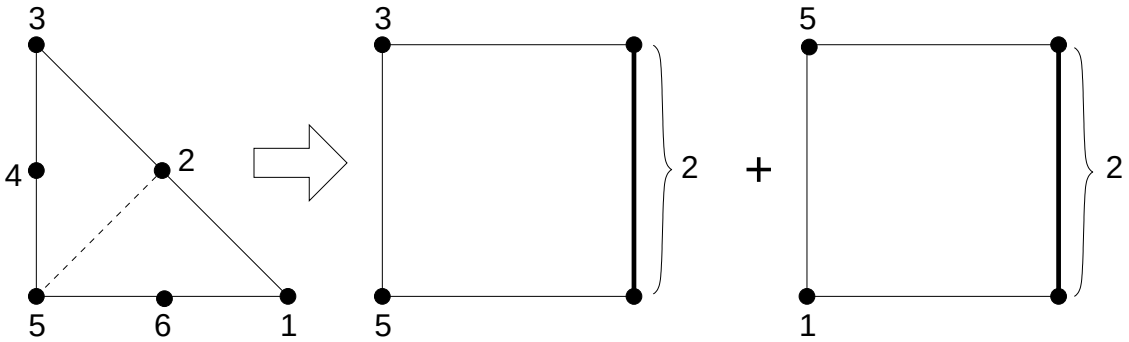

Figure 8.2: Regularization transformation for weakly singular integrals (quadratic case).

8.3 Strongly singular integrals

For strongly singular integrals we subdivide the triangle progressively in up to `NSUBDIVISIONS` –found in file `constants.h`– subsequent divisions, and integrate using standard gausian quadrature in each subtriangle except the closest to the singular point, which is neglected (it can be shown that the integrand goes to zero very close to the singularity point). The subdivision process is ilustrated in Figure 8.3 for a singularity at node 3 in a flat triangle.

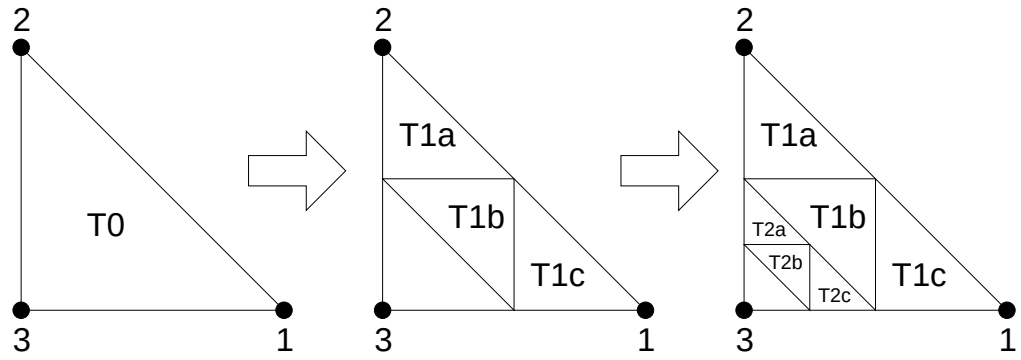

Figure 8.3: Subdivision process for strongly singular integrands with singularity at node 3 (only two consecutive subdivisions shown).

In the cases where the singular point is on the edge of a quadratic element rather than on a vertex the triangle is divided in two and the same subdivision process applied to the resulting subtriangles.

8.4 Changing the number of integration points

The gaussian quadrature data is stored in the file `gaussData.h`, and has two options, one with 7 integration points and another one with 64 integrations points. By default the code uses 7 integration points because increasing this number does not seem to improve accuracy, but this can be changed by following these steps:

1. Change the value of `TNGAUSS` and `TSNGAUSS` to 64 in file `constants.h`

2. Comment out the 7 point `TGauss` and `TSGauss` matrices in file `gaussData.h`
3. Uncomment the 64 point `TGauss` and `TSGauss` matrices in file `gaussData.h`
4. Recompile the source code

The constant values `TNGAUSS` and `TSNGAUSS` are the number of integration points in regular and strongly singular integrals, so it is also possible to use different values for them. Keep in mind that `NSUBDIVISIONS` are made in the case of strongly singular integrals, and therefore many more points are used for the integration even if `TNGAUSS` and `TSNGAUSS` are given the same value.

All integration schemes were chosen so that the integration points are in the interior of the elements, avoiding issues with evaluation points falling exactly on top of mesh nodes. The numerical values of the gaussian integration are reproduced in the following tables.

Table 1: Abscissas and weights for Gauss quadrature on a triangle

NG	x_i	y_i	w_i
7	0.333333333333333	0.333333333333333	0.112500000000000
	0.470142064105115	0.470142064105115	0.066197076394253
	0.059715871789770	0.470142064105115	0.066197076394253
	0.470142064105115	0.059715871789770	0.066197076394253
	0.101286507323456	0.101286507323456	0.062969590272414
	0.797426985353088	0.101286507323456	0.062969590272414
	0.101286507323456	0.797426985353088	0.062969590272414

Table 2: Abscissas and weights for Gauss-Jacobi quadrature on a line

NG	x_i	w_i
8	-0.910732089420060	0.013180765768995
	-0.711267485915709	0.713716106239446
	-0.426350485711139	0.181757278018796
	-0.090373369606853	0.316798397969277
	0.256135670833455	0.424189437743720
	0.571383041208738	0.450023197883551
	0.817352784200412	0.364476094545495
	0.964440169705273	0.178203217446225

Table 3: Abscissas and weights for Gauss quadrature on a line

NG	x_i	w_i
8	-0.960289856497536	0.101228536290370
	-0.796666477413627	0.222381034453376
	-0.525532409916329	0.313706645877887
	-0.183434642495650	0.362683783378362
	0.183434642495650	0.362683783378362
	0.525532409916329	0.313706645877887
	0.796666477413627	0.222381034453376
	0.960289856497536	0.101228536290370

Chapter 9

The electrostatic problem in DEP traps

This chapter describes in detail the equations used by `depSolver` to calculate the electric field and the forces acting in dielectrophoretic traps – the original application `depSolver` was written for.

Dielectrophoretic traps work by exerting a force on a particle due to the interaction of the polarization field in the particle with the gradient of the applied field [8]. A detailed knowledge of the electric field on the surface of the particle is necessary in order to calculate the total force that it experiences at any given moment in time.

This problem consists in a flow chamber that has a certain set of electrodes on the bottom and the ceiling of the channel, forming a dielectrophoretic trap. The chamber encloses a liquid and several particles with different electrical properties, as shown in Figure 9.1.

A fixed (in DC) or alternating (in AC) potential is applied on the electrodes and we are interested in the potential and electric field inside the trap volume. The boundary conditions are continuity of the potential and the normal component of the electric displacement across dielectric boundaries and the given potential at the electrodes. Due to the absence of sources inside the volume (the particles are supposed to be neutral) we only need to solve Laplace's equation in each subdomain and match the boundary conditions at all surfaces.

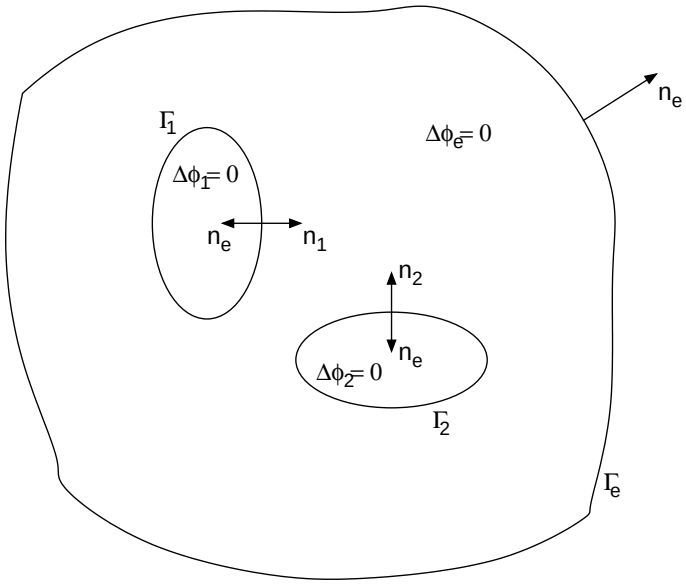

Figure 9.1: Laplace problem with different materials. Note definition of normal vectors.

In order to solve the problem we use the indirect boundary element method. To do this we discretize the system surfaces into elements and then apply the equations obtained in the following sections to get a linear system of equations. The advantage of the indirect method over the direct one is that we need only to apply a single equation on each node in the surface mesh.

9.1 IBEM treatment of the electrostatic problem

As mentioned above, Laplace equation must be must be satisfied in all subdomains (that is, inside each cell as well as in the liquid itself), and the potential and the normal derivative of the electric displacement must be continuous across all dielectric interfaces.

Because of the continuity of the electric potential we have to make the term $m(\vec{r}')$ equal to zero, and thus we can use equation (6.17):

$$\phi(\vec{r}) = \int_{\Gamma} s(\vec{r}')G(\vec{r},\vec{r}')d\Gamma' \tag{9.1}$$

Where Γ represents the total surface of the system, given by:

$$\Gamma = \Gamma_{\mathrm{e}} + \sum_{i=1}^{Np} \Gamma_i \tag{9.2}$$

In electrostatics the normal derivatives of the potential correspond to the normal components of the electric field, so the source $s(\vec{r})$ can be written as:

$$s(\vec{r}) = q(\vec{r}') + \overline{q}(\vec{r}') = E_{\mathrm{n}}(\vec{r}') - \overline{E}_{\mathrm{n}}(\vec{r}') \tag{9.3}$$

Using Gauss' Law jump condition we can write:

$$s(\vec{r}) = E_{\mathrm{n}}(\vec{r}') - \overline{E}_{\mathrm{n}}(\vec{r}') = \frac{\rho_{\mathrm{s}}(\vec{r}')}{\varepsilon_0} \tag{9.4}$$

where ε_0 is the permittivity of free space and ρ_{s} is the total surface charge density in Γ. For convenience we will redefine G and H as:

$$G^{3D}(\vec{r},\vec{r}') = \frac{1}{r} \tag{9.5}$$

$$H^{3D}(\vec{r},\vec{r}') = -\frac{(\vec{r}-\vec{r}')\cdot\hat{n}}{|\vec{r}-\vec{r}'|^3} \tag{9.6}$$

So that the correct expression for the potential results in:

$$\phi(x) = \frac{1}{4\pi\varepsilon_0}\int_{\Gamma} \rho_{\mathrm{s}}(\vec{r}')G(\vec{r},\vec{r}')d\Gamma' \tag{9.7}$$

9.1.1 DC Case

In this case we consider lossless media, and the boundary conditions at the dielectric interfaces are given by:

$$\phi_{\mathrm{e}}|_{\Gamma_{\mathrm{i}}} = \phi_{\mathrm{i}}|_{\Gamma_{\mathrm{i}}} \tag{9.8}$$

$$\varepsilon_{\mathrm{e}} \left.\frac{\partial \phi_{\mathrm{e}}}{\partial n}\right|_{\Gamma_{\mathrm{i}}} = \varepsilon_{\mathrm{i}} \left.\frac{\partial \phi_{\mathrm{i}}}{\partial n}\right|_{\Gamma_{\mathrm{i}}} \tag{9.9}$$

for all dielectric interfaces i.

Taking the derivative of (9.7) at a point not in the surface gives:

$$\frac{\partial \phi(\vec{r})}{\partial n} = \frac{1}{2\alpha\pi\varepsilon_0} \int_{\Gamma} \rho_{\mathrm{s}}(\vec{r}') H(\vec{r}, \vec{r}') d\Gamma' \tag{9.10}$$

When taking this expression to the boundary we must include the jump in the integral due to the discontinuity of the normal derivative of the potential:

$$\frac{\partial \phi_{\mathrm{e}}(\vec{r})}{\partial n} = -\frac{\rho_{\mathrm{s}}(\vec{r})}{2\varepsilon_0} + \frac{1}{2\alpha\pi\varepsilon_0} \int_{\Gamma} \rho_{\mathrm{s}}(\vec{r}') H(\vec{r}, \vec{r}') d\Gamma' \qquad \text{if } \vec{r} \in \Gamma_{\mathrm{i}} \tag{9.11}$$

$$\frac{\partial \phi_{\mathrm{i}}(\vec{r})}{\partial n} = \frac{\rho_{\mathrm{s}}(\vec{r})}{2\varepsilon_0} + \frac{1}{2\alpha\pi\varepsilon_0} \int_{\Gamma} \rho_{\mathrm{s}}(\vec{r}') H(\vec{r}, \vec{r}') d\Gamma' \qquad \text{if } \vec{r} \in \Gamma_{\mathrm{i}} \tag{9.12}$$

Using these expression in the boundary condition (9.9) gives:

$$\varepsilon_{\mathrm{e}} \left[-\frac{\rho_{\mathrm{s}}(\vec{r})}{2\varepsilon_0} + \frac{1}{2\alpha\pi\varepsilon_0} \int_{\Gamma} \rho_{\mathrm{s}}(\vec{r}') H(\vec{r}, \vec{r}') d\Gamma' \right] = \varepsilon_{\mathrm{i}} \left[\frac{\rho_{\mathrm{s}}(\vec{r})}{2\varepsilon_0} + \frac{1}{2\alpha\pi\varepsilon_0} \int_{\Gamma} s(\vec{r}') H(\vec{r}, \vec{r}') d\Gamma' \right] \tag{9.13}$$

Reorganizing terms to isolate the integrals to the right we get:

$$-\frac{(\varepsilon_{\mathrm{i}}+\varepsilon_{\mathrm{e}})\rho_{\mathrm{s}}(\vec{r})}{2\varepsilon_0}=\frac{(\varepsilon_{\mathrm{i}}-\varepsilon_{\mathrm{e}})}{2\alpha\pi\varepsilon_0}\int_\Gamma \rho_{\mathrm{s}}(\vec{r}')H(\vec{r},\vec{r}')d\Gamma' \tag{9.14}$$

And finally, for any point in the boundary Γ_{i} we find:

$$\rho_{\mathrm{s}}(\vec{r})=\frac{\varepsilon_{\mathrm{e}}-\varepsilon_{\mathrm{i}}}{\alpha\pi(\varepsilon_{\mathrm{e}}+\varepsilon_{\mathrm{i}})}\int_\Gamma \rho_{\mathrm{s}}(\vec{r}')H(\vec{r},\vec{r}')d\Gamma' \tag{9.15}$$

So using (9.7) evaluated on te external boundary, and (9.15) on each of the internal boundaries we can solve for σ, and then calculate the potential from (9.7) at any point in the domain. Any derivative of the potential can be calcualted by taking the derivative in (9.7), which will affect only x and therefore leave the σ as constants.

9.1.2 AC case

When real materials are studied the losses have to be included in the calculations, so we must consider not only the dielectric permittivity but also the electric conductivity of the material. It is also the case that the electrical response of the material will appear to have a time lag with respect to the externally applied field. In the case of harmonic applied fields (periodic in time) this time lag is easily expressed as a phase difference between the applied field and the resulting current and polarization. A good reference for this problem is the work by Pohl [9] that we follow in this section.

For a simple (linear) medium if the applied field is harmonic, the charge density, current, polarization, and other descriptive electromagnetic parameters will also be harmonic, but not necessarily in phase with the applied field. If the applied field is given by:

$$\vec{E}(\vec{r},t)=\vec{E}_0(\vec{r})\exp(j\omega t) \tag{9.16}$$

The we also have:

$$\vec{J}(\vec{r},t) \;=\; \vec{J}_0(\vec{r},t)\exp(j\omega t-j\delta_{\mathrm{J}}) \tag{9.17}$$

$$\vec{D}(\vec{r},t) = \vec{D}_0(\vec{r},t)\exp(j\omega t - j\delta_{\rm D}) \quad (9.18)$$
$$\vec{P}(\vec{r},t) = \vec{P}_0(\vec{r},t)\exp(j\omega t - j\delta_{\rm P}) \quad (9.19)$$
$$\rho(\vec{r},t) = \rho_0(\vec{r},t)\exp(j\omega t - j\delta_{\rho}) \quad (9.20)$$
$$\vec{B}(\vec{r},t) = \vec{B}_0(\vec{r},t)\exp(j\omega t - j\delta_{\rm B}) \quad (9.21)$$
$$\vec{H}(\vec{r},t) = \vec{H}_0(\vec{r},t)\exp(j\omega t - j\delta_{\rm H}) \quad (9.22)$$

And for homogeneous, isotropic media we can write:

$$\vec{D} = \varepsilon\vec{E} \quad (9.23)$$
$$\vec{J} = \sigma\vec{E} \quad (9.24)$$

Where both ε and σ are complex quantities. Maxwell's equations are given by:

$$\nabla\vec{E} = \frac{\rho}{\varepsilon} \quad (9.25)$$
$$\nabla\times\vec{E} = -\frac{\partial\vec{B}}{\partial t} \quad (9.26)$$
$$\nabla\cdot\vec{B} = 0 \quad (9.27)$$
$$\nabla\times\vec{H} = \vec{J} + \frac{\partial\vec{D}}{\partial t} \quad (9.28)$$

which in this case, and using (9.23) and (9.24) reduce to:

$$\nabla\vec{E} = \frac{\rho}{\varepsilon} \quad (9.29)$$
$$\nabla\times\vec{E} = -j\omega\vec{B} \quad (9.30)$$
$$\nabla\cdot\vec{B} = 0 \quad (9.31)$$
$$\nabla\times\vec{H} = [\sigma + i\omega\varepsilon]\,\vec{E} \quad (9.32)$$

Now, the boundary conditions at a dielectric interface between materials 1 and 2 are given by:

$$\varepsilon_1 E_{1\mathrm{n}} - \varepsilon_2 E_{2\mathrm{n}} = -\rho_{\mathrm{free}} \tag{9.33}$$

$$\left(\hat{n} \times \vec{E}_1\right) - \left(\hat{n} \times \vec{E}_2\right) = 0 \tag{9.34}$$

$$\left(\hat{n} \times \vec{H}_1\right) - \left(\hat{n} \times \vec{H}_2\right) = -\vec{I}_{\mathrm{free}} \tag{9.35}$$

$$B_{1\mathrm{n}} - B_{2\mathrm{n}} = 0 \tag{9.36}$$

with ρ_{free} the surface density of free charge, and $\vec{I}_{\mathrm{free}}$ the surface density of free current, which is different from zero only in perfect conductors.

The equations for the conservation of charge in a harmonic field are:

$$\varepsilon \nabla \vec{E} + j\omega\rho_{\mathrm{free}} = 0 \tag{9.37}$$

$$\nabla \cdot \vec{I}_{\mathrm{free}} + j\omega\rho_{\mathrm{free}} - \hat{n} \cdot \left(\sigma_1 \vec{E}_1 - \sigma_2 \vec{E}_2\right) = 0 \tag{9.38}$$

Using (9.29) into (9.37) we get:

$$\nabla \left[(\varepsilon - i\sigma/\omega)\vec{E}\right] = 0 \tag{9.39}$$

everywhere except at the boundaries, where $\nabla(\varepsilon - i\sigma/\omega) \neq 0$. Combining now (9.24) and (9.38) for a dielectric material ($I_{\mathrm{free}} = 0$):

$$(\varepsilon_1 - j\sigma_1/\omega)E_{1\mathrm{n}} = (\varepsilon_2 - j\sigma_2/\omega)E_{2\mathrm{n}} \tag{9.40}$$

If we define now the *complex permittivity* of a material as $\tilde{\varepsilon} = \varepsilon - j\sigma/\omega$, we can re-write the previous equations in terms of the potential:

$$\nabla \left(\tilde{\varepsilon}\vec{E}\right) = \nabla^2 \phi = 0 \tag{9.41}$$

$$\tilde{\varepsilon}_1 \frac{\partial \phi_1}{\partial n} = \tilde{\varepsilon}_2 \frac{\partial \phi_2}{\partial n} \tag{9.42}$$

These are exactly the same equations that we used for the DC case, with the only difference that now both the permittivity and the potential are complex quantitites.

Call the total surface charge density $\rho_{\mathrm{s}}(\vec{r})$ the equations to solve in the indirect boundary element method are:

$$\phi(\vec{r}) = \frac{1}{2\alpha\pi\varepsilon_0}\int_\Gamma \rho_{\mathrm{s}}(\vec{r}')G(\vec{r},\vec{r}')d\Gamma' \qquad \text{for a conductor} \tag{9.43}$$

$$\rho_{\mathrm{s}}(\vec{r}) = \frac{(\tilde{\varepsilon}_{\mathrm{e}}-\tilde{\varepsilon}_{\mathrm{i}})}{\alpha\pi(\tilde{\varepsilon}_{\mathrm{e}}+\tilde{\varepsilon}_{\mathrm{i}})}\int_\Gamma \rho_{\mathrm{s}}(\vec{r}')H(\vec{r},\vec{r}')d\Gamma' \qquad \text{for a dielectric interface} \tag{9.44}$$

with the boundary conditions:

$$\phi_{\mathrm{e}}|_{\Gamma_{\mathrm{i}}} = \phi_{\mathrm{i}}|_{\Gamma_{\mathrm{i}}} \tag{9.45}$$

$$\tilde{\varepsilon}_{\mathrm{e}}\left.\frac{\partial\phi_{\mathrm{e}}}{\partial n}\right|_{\Gamma_{\mathrm{i}}} = \tilde{\varepsilon}_{\mathrm{i}}\left.\frac{\partial\phi_{\mathrm{i}}}{\partial n}\right|_{\Gamma_{\mathrm{i}}} \tag{9.46}$$

Expanding $\tilde{\varepsilon}$ in equation (9.44):

$$\rho_{\mathrm{s}}(\vec{r}) = \frac{[(\sigma_{\mathrm{e}}+j\omega\varepsilon_{\mathrm{e}})-(\sigma_{\mathrm{i}}+j\omega\varepsilon_{\mathrm{i}})]}{\alpha\pi[(\sigma_{\mathrm{e}}+j\omega\varepsilon_{\mathrm{e}})-(\sigma_{\mathrm{i}}+j\omega\varepsilon_{\mathrm{i}})]}\int_\Gamma \rho_{\mathrm{s}}(\vec{r}')G(\vec{r},\vec{r}')d\Gamma' \tag{9.47}$$

which can be re-written as:

$$\rho_{\mathrm{s}}(\vec{r}) = \frac{[(\sigma_{\mathrm{e}}-\sigma_{\mathrm{i}})+j\omega(\varepsilon_{\mathrm{e}}-\varepsilon_{\mathrm{i}})]}{\alpha\pi[(\sigma_{\mathrm{e}}+\sigma_{\mathrm{i}})+j\omega(\varepsilon_{\mathrm{e}}+\varepsilon_{\mathrm{i}})]}\int_\Gamma \rho_{\mathrm{s}}(\vec{r}')G(\vec{r},\vec{r}')d\Gamma' \tag{9.48}$$

Dividing now $\rho_{\mathrm{s}}(\vec{r})$ into its real and imaginary parts (denoted as $\rho_{\mathrm{s}}^{\mathrm{R}}(\vec{r})$ and $\rho_{\mathrm{s}}^{\mathrm{I}}(\vec{r})$ respectively) we get:

$$\begin{aligned}&\alpha\pi[\rho_{\mathrm{s}}^{\mathrm{R}}(\vec{r})+j\rho_{\mathrm{s}}^{\mathrm{I}}(\vec{r})][(\sigma_{\mathrm{e}}+\sigma_{\mathrm{i}})+j\omega(\varepsilon_{\mathrm{e}}+\varepsilon_{\mathrm{i}})] = \\ &[(\sigma_{\mathrm{e}}-\sigma_{\mathrm{i}})+j\omega(\varepsilon_{\mathrm{e}}-\varepsilon_{\mathrm{i}})]\int_\Gamma [\rho_{\mathrm{s}}^{\mathrm{R}}(\vec{r}')+j\rho_{\mathrm{s}}^{\mathrm{I}}(\vec{r}')]H(\vec{r},\vec{r}')d\Gamma'\end{aligned} \tag{9.49}$$

Separating all real and imaginary terms we obtain two equations:

$$\alpha\pi[(\sigma_\mathrm{e}+\sigma_\mathrm{i})\rho_\mathrm{s}^\mathrm{R}(\vec{r}) - \omega(\varepsilon_\mathrm{e}+\varepsilon_\mathrm{i})\rho_\mathrm{s}^\mathrm{I}(\vec{r})] = \\ (\sigma_\mathrm{e}-\sigma_\mathrm{i})\int_\Gamma \rho_\mathrm{s}^\mathrm{R}(\vec{r})H(\vec{r},\vec{r}')d\Gamma' - \omega(\varepsilon_\mathrm{e}-\varepsilon_\mathrm{i})\int_\Gamma \rho_\mathrm{s}^\mathrm{I}(\vec{r})H(\vec{r},\vec{r}')d\Gamma' \qquad (9.50)$$

$$\alpha\pi[(\sigma_\mathrm{e}+\sigma_\mathrm{i})\rho_\mathrm{s}s^\mathrm{I}(\vec{r}) + \omega(\varepsilon_\mathrm{e}+\varepsilon_\mathrm{i})\rho_\mathrm{s}^\mathrm{R}(\vec{r})] = \\ \omega(\varepsilon_\mathrm{e}-\varepsilon_\mathrm{i})\int_\Gamma \rho_\mathrm{s}^\mathrm{R}(\vec{r})H(\vec{r},\vec{r}')d\Gamma' + (\sigma_\mathrm{e}-\sigma_\mathrm{i})\int_\Gamma \rho_\mathrm{s}^\mathrm{I}(\vec{r})H(\vec{r},\vec{r}')d\Gamma' \qquad (9.51)$$

We can isolate the real part of $\rho_\mathrm{s}(\vec{r})$ on the left by adding (9.50)$\times(\sigma_\mathrm{e}+\sigma_\mathrm{i})$ and (9.51)$\times\omega(\varepsilon_\mathrm{e}+\varepsilon_\mathrm{i})$:

$$\rho_\mathrm{s}^\mathrm{R}(\vec{r}) = A\int_\Gamma \rho_\mathrm{s}^\mathrm{R}(\vec{r}')H(\vec{r},\vec{r}')d\Gamma' + B\int_\Gamma \rho_\mathrm{s}^\mathrm{I}(\vec{r}')H(\vec{r},\vec{r}')d\Gamma' \qquad (9.52)$$

And we can isolate the imaginary part of $\rho_\mathrm{s}(x)$ on the left by subtracting (9.50)$\times\omega(\varepsilon_\mathrm{e}+\varepsilon_\mathrm{i})$ from (9.51)$\times(\sigma_\mathrm{e}+\sigma_\mathrm{i})$:

$$\rho_\mathrm{s}^\mathrm{I}(\vec{r}) = A\int_\Gamma s^\mathrm{I}(\vec{r}')H(\vec{r},\vec{r}')d\Gamma' - B\int_\Gamma \rho_\mathrm{s}^\mathrm{R}(\vec{r}')H(\vec{r},\vec{r}')d\Gamma' \qquad (9.53)$$

Where the factors A and B are given by:

$$A = \frac{(\sigma_\mathrm{e}^2-\sigma_\mathrm{i}^2)+\omega^2(\varepsilon_\mathrm{e}^2-\varepsilon_\mathrm{i}^2)}{\alpha\pi[(\sigma_\mathrm{e}+\sigma_\mathrm{i})^2+\omega^2(\varepsilon_\mathrm{e}+\varepsilon_\mathrm{i})^2]} \qquad (9.54)$$

$$B = \frac{2\omega(\sigma_\mathrm{e}\varepsilon_\mathrm{i}-\sigma_\mathrm{i}\varepsilon_\mathrm{e})}{\alpha\pi[(\sigma_\mathrm{e}+\sigma_\mathrm{i})^2+\omega^2(\varepsilon_\mathrm{e}+\varepsilon_\mathrm{i})^2]} \qquad (9.55)$$

Equations (9.52–9.55) are used in nodes situated in a dielectric boundary. For nodes in a conductor a simpler pair of equations is used:

$$\phi^{\mathrm{R}}(\vec{r}) = \frac{1}{2\alpha\pi\varepsilon_0}\int_{\Gamma}\rho_{\mathrm{s}}^{\mathrm{R}}(\vec{r}')G(\vec{r},\vec{r}')d\Gamma' \tag{9.56}$$

$$\phi^{\mathrm{I}}(\vec{r}) = \frac{1}{2\alpha\pi\varepsilon_0}\int_{\Gamma}\rho_{\mathrm{s}}^{\mathrm{I}}(\vec{r}')G(\vec{r},\vec{r}')d\Gamma' \tag{9.57}$$

NOTE: It is common to absorb ε_0 into $\rho_{\mathrm{s}}(\vec{r}')$.

9.2 Calculation of the DEP force

This section describes the two alternative methods of calculating the electrical force in dielectrophoretic traps.

9.2.1 Equivalent multipole model of the DEP force

In this approach the particle under consideration is substituted by a series of multipoles that account for the local changes of the electric field in the region surrounding the particle [10],[11]. This approximation is strictly valid only when the external field is not changing rapidly in the region containing the particle.

Spherical particles

In the simplest possible approximation, the time-averaged DEP force for a spherical particle is given by the expression:

$$\left\langle \vec{F}_{\mathrm{DEP}} \right\rangle = 2\pi R^3 \varepsilon_{\mathrm{f}} Re\left[K\left(\omega\right)\right]\nabla(E^2) \tag{9.58}$$

where R is the particle radius, ε_f is the permittivity of the fluid suspending medium, E is the rms magnitude of the local electric field, and $Re[K(\omega)]$ is the real part of the Clausius-Mossotti factor, given by:

$$K(\omega) = \frac{\tilde{\varepsilon}_{\mathrm{p}} - \tilde{\varepsilon}_{\mathrm{f}}}{\tilde{\varepsilon}_{\mathrm{p}} + 2\tilde{\varepsilon}_{\mathrm{f}}} \tag{9.59}$$

where ω is the angular frequency of the external applied field and $\tilde{\varepsilon}_\mathrm{f}$ and $\tilde{\varepsilon}_\mathrm{p}$ are the complex dielectric permittivities of the fluid and the particle respectively. The complex permittivities are given by $\tilde{\varepsilon}_\mathrm{i} = \varepsilon_\mathrm{i} - j\sigma_\mathrm{i}/\omega$, where ε_i is the dielectric permittivity of medium i, σ_i is the electric conductivity of medium i, and j is $\sqrt{-1}$.

This is the dipolar approximation for the DEP force on a sphere, and it is used extensively in the literature to predict the characteristics of DEP cell traps.

The dipolar approximation is simple and convenient, but in cases where the electric field presents a null or a high gradient its results are not accurate. When designing single-cell traps the electric field gradient must be very high for the electric forces on the particle to be significant, and therefore a better approximation, including higher order multipolar terms, is needed.

The general expression of the DEP force in terms of its multipolar components obtained by Washizu and Jones [12,13] was recently extended to include the magnetic field effects by Washizu [17]. However, in the cases studied in the present work the magnetic effects are negligible, and the expression of the time averaged nth force order contribution is:

$$\left\langle \vec{F}^{(n)}_\mathrm{DEP} \right\rangle = \frac{1}{2} \frac{\mathbf{p}^{(n)} [\cdot]^n \left(\nabla\right)^n \vec{E}}{n!} \tag{9.60}$$

where $[\cdot]^n$ and $\left(\nabla\right)^n$ represent n dot products and gradient operations, and $\mathbf{p}^{(n)}$ is the multipolar induced tensor of order n:

$$\mathbf{p}^{(n)} = \frac{4\pi\varepsilon_\mathrm{f} R^{2n+1} n}{(2n+1)!!} K^{(n)}(\omega) \left(\nabla\right)^{n-1} \vec{E} \tag{9.61}$$

with $K^{(n)}$ the nth order Clausius-Mossotti factor given by:

$$K^{(n)}(\omega) = \frac{\tilde{\varepsilon}_\mathrm{p} - \tilde{\varepsilon}_\mathrm{f}}{n\tilde{\varepsilon}_\mathrm{p} + (n+1)\tilde{\varepsilon}_\mathrm{f}} \tag{9.62}$$

Note that a factor 1/2 is included in (9.60) in order to account for the time average, under the assumption that the external field oscillates harmonically.

A BEM calculation is used to find the external field produced by the electrode arrangement, and its derivatives, in order to find the force using this approximation. Once the field and its derivatives are calculated on the centre of the sphere, a simple application of the formulas above gives the total DEP force on the particle.

Ellipsoidal particles

The induced effective moment for an ellipsoidal particle with principal radii a, b, and c, can be found by examining the limit of the electrostatic potential at a point far from the ellipsoid, where ellipsoidal coordinates degenerate into spherical coordinates. Following Jones [8] the x component of the effective dipole moment due to a dielectric ellipsoid is given by:

$$\mathrm{p_x} = \frac{4\pi \mathrm{abc}}{3} \varepsilon_\mathrm{f} \left[\frac{\varepsilon_\mathrm{p} - \varepsilon_\mathrm{f}}{\varepsilon_\mathrm{f} + (\varepsilon_\mathrm{p} - \varepsilon_\mathrm{f}) \mathrm{L_x}} \right] \mathrm{E_x} \tag{9.63}$$

The other two components of the effective moment, $\mathrm{p_y}$ and $\mathrm{p_y}$, are of similar form. The time-averaged dielectrophoretic force produced by a harmonically oscillating external field on the ellipsoid is then given by the following expression:

$$\left\langle \vec{F}_\mathrm{DEP}^\mathrm{ellipsoid} \right\rangle = \frac{2\pi abc}{3} \varepsilon_\mathrm{f} \left[\frac{(\varepsilon_\mathrm{p} - \varepsilon_\mathrm{f}) E_\mathrm{x} \partial_\mathrm{x}}{\varepsilon_\mathrm{f} + (\varepsilon_\mathrm{p} - \varepsilon_\mathrm{f}) L_\mathrm{x}} + \frac{(\varepsilon_\mathrm{p} - \varepsilon_\mathrm{f}) E_\mathrm{y} \partial_\mathrm{y}}{\varepsilon_\mathrm{f} + (\varepsilon_\mathrm{p} - \varepsilon_\mathrm{f}) L_\mathrm{y}} + \frac{(\varepsilon_\mathrm{p} - \varepsilon_\mathrm{f}) E_\mathrm{z} \partial_\mathrm{z}}{\varepsilon_\mathrm{f} + (\varepsilon_\mathrm{p} - \varepsilon_\mathrm{f}) L_\mathrm{z}} \right] \vec{E} \tag{9.64}$$

Where the depolarization factors L_x, L_y, and L_z, are all positive and interrelated as follows:

$$0 \leq L_i \leq 1, \qquad i = x, y, z \tag{9.65}$$

$$L_\mathrm{x} + L_\mathrm{y} + L_\mathrm{z} = 1 \tag{9.66}$$

The value of L_x is given by an elliptic integral:

$$L_\mathrm{x} = \frac{abc}{2} \int_0^\infty \frac{ds}{(s + a^2)\sqrt{(s + a^2)(s + b^2)(s + c^2)}} \tag{9.67}$$

Similar expressions apply for y and z by simply changing the $(s + a^2)$ outside the square root by $(s + b^2)$ or $(s + c^2)$.

Note that this is only a first order approximation and therefore it will predict zero force for any particle position such that the ellipsoid's centre corresponds to a field null.

9.2.2 Maxwell stress tensor derivation of the DEP force

A different approach to the calculation of the DEP force is to use the Maxwell stress tensor formulation and integrate the stress tensor $\mathbf{T}$ over the surface of the particle:

$$\vec{F}(t)_{\mathrm{DEP}}^{\mathrm{MST}} = \oint (\mathbf{T} \cdot \vec{n})\, dA \tag{9.68}$$

where $\vec{n}$ is the unit vector normal to the surface and t is time. This is regarded as the most rigurous approach to derive field-induced forces.

The general expression for the DEP force obtained by Wang et al [12] is used in order to find out how precise the multipolar approximation is when compared to this more rigurous calculation.

The time-averaged net DEP force on a particle using MST is given by:

$$\left\langle \vec{F}_{\mathrm{DEP}}^{\mathrm{MST}} \right\rangle = \frac{\varepsilon_{\mathrm{f}}}{4} \oint \left\{ \left[\left(\vec{E}_{\mathrm{f}} \vec{E}_{\mathrm{f}}^{*} + \vec{E}_{\mathrm{f}}^{*} \vec{E}_{\mathrm{f}} \right) - |\vec{E}_{\mathrm{f}}|^2 \mathbf{I} \right] \cdot \vec{n} \right\} dA \tag{9.69}$$

Note that in this case the presence of the particle is included directly in the calculations, and that no assumptions are made regarding the external field homogeneity. This means that even when strong field inhomogeneities are present, the values of the DEP force obtained using this method will be correct. It is expected that the results obtained using the multipolar approximation described in the previous section will worsen as the field gradient increases, and will depart from the values predicted by the MST method. It is of interest to find out how significant is the difference between the two methods for different particle sizes and positions inside a dielectrophoretic trap.

Appendix A

Function Listing

Directory	: depSolver/src
Source Files	: 87
Compilation Script	: make depSolver / make depSolver-sft

comFilter.c
constants.h
depolarization.c
depolarization.h
depSolver.c
depSolver.h
depSolver-shift.c
dotProd.c
doubleMatrix.c
doublePointer.c
doubleVector.c
electricFormA_tria3.c
electricFormA_tria3.h
electricFormA_tria6.c
electricFormA_tria6.h
electricFormFullA_tria3.c
electricFormFullA_tria6.c
elemType.c
errorHandler.c
field_tria3.c
forceMST_tria6.c
forceMST_tria6.h
forceMultipole_tria3.c
forceMultipole_tria3.h
forceMultipole_tria6.c
forceMultipole_tria6.h
freeDoubleMatrix.c
freeDoublePointer.c
freeUintMatrix.c
gaussBksb.c
gaussData.h
getLocalNormal_tria3.c
getLocalNormal_tria6.c
getNormal_tria3.c
getNormal_tria6.c
initRes_el.c
initRes_el.h
intDE_tria3.c
intDE_tria6.c
integral_tria3.h
intSingularH_tria3.c
intSingularH_tria6.c
iterGMRES_el.c
iterGMRES_el.h
L2Norm.c
makefile
matVectProd_el.c
postProcessCol_tria3.c
postProcessCol_tria3.h
postProcessCol_tria6.c
postProcessCol_tria6.h
postProcess_tria3.c
postProcess_tria3.h
postProcess_tria6.c
postProcess_tria6.h
potential_tria3.c
potential_tria3.h
potential_tria6.c
potential_tria6.h
shape_line2.c

field_tria3.h
field_tria6.c
field_tria6.h
forceEllipsoid_tria3.c
forceEllipsoid_tria3.h
forceEllipsoid_tria6.c
forceEllipsoid_tria6.h
forceMST_tria3.c
forceMST_tria3.h
integral_tria6.h
intF_tria3.c
intF_tria6.c
intG_tria3.c
intG_tria6.c
intH_tria3.c
intH_tria6.c
intSingularG_tria3.c
intSingularG_tria6.c
shape_tria3.c
shape_tria6.c
solverGMRES_el.c
solverGMRES_el.h
uintMatrix.c
uintVector.c
X2L_line2.c
X2L_tria3.c
X2L_tria6.c

Directory	: depSolver/utils/break
Source Files	: 2
Compilation Script	: breakComp

break.c
errorHandler.c

Directory	: depSolver/utils/fieldPost
Source Files	: 4
Compilation Script	: fieldPostComp

doubleMatrix.c
errorHandler.c
freeDoubleMatrix.c
fieldPost.c

Directory	: depSolver/utils/forcePost
Source Files	: 4
Compilation Script	: forcePostComp

doubleMatrix.c
errorHandler.c
freeDoubleMatrix.c
forcePost.c

Directory	: depSolver/utils/gplotFormat
Source Files	: 2
Compilation Script	: gplotComp

gplotForm.c errorHandler.c

Directory	: depSolver/utils/meshgen
Source Files	: 2
Compilation Script	: meshComp

meshgen.c errorHandler.c

Directory	: depSolver/utils/p2b
Source Files	: 5
Compilation Script	: p2bComp

bem_save.pcl doubleMatrix.c errorHandler.c
freeDoubleMatrix.c p2b.c

Notice that the file bem_save.pcl must be in the same directory from which Patran is executed, so copy the file over as necessary.

Directory	: depSolver/utils/potPost
Source Files	: 4
Compilation Script	: thPostComp

doubleMatrix.c errorHandler.c freeDoubleMatrix.c
potPost.c

Bibliography

[1] C. Rosales and K. M. Lim. Numerical comparison between maxwell stress method and equivalent multipole approach for calculation of the dielectrophoretic force in single-cell traps. *Electrophoresis*, 26:2057–2065, 2005.

[2] Paraview is an open source multi-platform visualization application that provides options relevant to the scientific community such as parallel rendering facilities for large data sets. For more information visit the official web site at `http://www.paraview.org/`.

[3] Patran is an open-architecture, 3-d computer aided-engineering software package from msc.software corporation. for more information visit their web site at `http://www.mscsoftware.com/products/patran.cfm`.

[4] Matlab (*matrix laboratory*) is an interactive programming environment from The MathWorks that provides tools for data visualization, data analysis, and numeric computation. for more information visit their web site at `http://www.mathworks.com/products/matlab/description1.html`.

[5] Gnuplot is a command-line driven interactive data and function plotting utility distributed as copyrighted but free software. for more information visit the official gnuplot web site `http://www.gnuplot.info/`.

[6] C. A. Brebbia. *The boundary element method for engineers*. Pentech Press, London (UK), 1978.

[7] L. C. Wrobel. The Boundary Element Method, 2002.

[8] T. B. Jones. *Electromechanics of Particles*. Cambridge University Press, Cambridge (UK), 1995.

[9] H. Pohl. *Dielectrophoresis*. Cambridge University Press, Cambridge (UK), 1978.

[10] T. B. Jones and M. Washizu. Multipolar dielectrophoretic force calculation. *Journal of Electrostatics*, 33:187–198, 1994.

[11] T. B. Jones and M. Washizu. Multipolar dielectrophoretic and electrorotation theory. *Journal of Electrostatics*, 37:121–134, 1996.

[12] X. Wang, X.-B. Wang, and P. R. C. Gascoyne. General expressions for dielectrophoretic force and electrorotational torque derived using the maxwell stress tensor method. *Journal of Electrostatics*, 39:277–295, 1997.