

NUMERICAL CAPACITANCE EXTRACTION FOR LARGE-AREA SYSTEMS

by

Hoan Huu Pham

A thesis

presented to the University of Waterloo

in fulfilment of the

thesis requirement for the degree of

Doctor of Philosophy

in

Electrical Engineering

Waterloo, Ontario, Canada, 1998

©Hoan Huu Pham 1998



**National Library
of Canada**

**Acquisitions and
Bibliographic Services**

**395 Wellington Street
Ottawa ON K1A 0N4
Canada**

**Bibliothèque nationale
du Canada**

**Acquisitions et
services bibliographiques**

**395, rue Wellington
Ottawa ON K1A 0N4
Canada**

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-32852-X

The University of Waterloo requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

Abstract

This thesis describes the development of a new computational approach for rapid and accurate evaluation of the three-dimensional potential field and its gradient. Efficient evaluation of the potential field is an essential requirement for simulation of large ensembles of particles in many applications including astrophysics, plasma physics, fluid dynamics, molecular dynamics, VLSI systems, and micro-electro-mechanical systems. Current methods use multipole expansion of spherical harmonics for the potential field, which is computationally expensive in terms of running time and memory requirements when a high degree of accuracy is desired. The mathematical background for the approach proposed in this thesis stems from an exponential integral representation of Green's function $\frac{1}{r}$ and an approximation to the integral using Gauss quadratures. The translations are simple in structure, error-free, and independent of the approximation, which enables the overall accuracy and computational performance to be controlled externally via the approximation. In addition, the gradient of the potential can be readily retrieved as a by-product of the computational process. More importantly, the memory requirement is independent of the desired degree of accuracy. The technique presented here opens new possibilities for efficient distributed computing and parallel processing of large-scale simulation of particle systems.

The research described in this thesis makes the following key contributions:

- (i) A randomized algorithm is devised for generating exponential expansion of the Green's function. Given an intended error, this probabilistic-type scheme constructs the Gauss quadratures for the expansions with sizes as small as possible.

It makes use of the available Gauss-Legendre quadratures and does not require solving non-linear equations. (ii) Employing the exponential expansion, a theory is developed for efficient evaluation of the potential field and its gradient. The main motivation is to provide an alternative to the current methods for N-body problems involving millions of particles. (iii) A new formulation is proposed for the charge density problem. It yields integral equations of the second kind, for which efficient numerical algorithms are available. The resultant discretization matrices have improved conditioning and the convergence rate is faster. (iv) A simulator is designed and implemented for numerically extracting capacitance of multi-layered dielectric multi-conductor systems. It is used for studying capacitance associated with amorphous silicon thin-film transistors and imaging arrays.

Acknowledgements

I am deeply grateful to my thesis supervisor, Professor Arokia Nathan, for his thoughtfulness and supervision during the course of my training at the University of Waterloo. Without his guidance, this work would not have existed.

My experiences have been made enjoyable by the encouragement and advice from the PhD examination committee, Professors R. I. Hornsey, V. Karanassios, and A. Vannelli, to all of whom I owe special thanks.

It is a great honor to have Professor J. G. Korvink, Institute for Microsystems, University of Freiburg, Germany, as the external examiner. I wish to express my sincere appreciation for his presence.

I would like to thank the Microtechnology Research Laboratory and the Department of Electrical and Computer Engineering for providing help, support, and a friendly environment. Many thanks go to Dr. R.V.R. Murthy for help with the fabrication. The scholarships from the Natural Sciences and Engineering Research Council of Canada (NSERC), the Institute for Computer Research (ICR), and Faculty of Engineering are also gratefully acknowledged.

On this occasion, I would like to express my deep gratitude to my foster parents and their family for their constant nurturing, great care and kindness, and for their tireless effort to save me from the ordeals and to bring me to Canada, a land of freedom, generosity and opportunity.

*In memory of my mother and father,
who planted this seed
but had to wait long
for the tree to blossom and bear fruit.*

Contents

1	Introduction	1
1.1	Capacitance and charge density	2
1.1.1	Differential-equation formulation	5
1.1.2	Integral-equation formulation	6
1.2	Evaluation of three-dimensional potential field	9
1.3	Multipole-expansion-based methods	12
1.3.1	Multipole expansion	12
1.3.2	Translations and computational cost	14
1.4	Difficulties and motivations	16
1.5	Organization of thesis	18
2	Generating Exponential Expansion	20
2.1	Exponential expansion	20
2.2	Summary of randomized approach	24
2.3	From infinite-interval to finite-interval integrals	27
2.4	Gauss quadratures for outer loop	30
2.4.1	Randomized algorithm I	31

2.5	Gauss quadratures for inner loop	32
2.5.1	An integral representation for Bessel function $J_0(t)$	32
2.5.2	Gauss quadrature	33
2.5.3	Randomized algorithm II	35
2.6	Numerical results	36
3	Exponential-expansion Method	43
3.1	Mathematical background	44
3.1.1	Six directions	45
3.1.2	Approximation and scaling	48
3.1.3	Approximation: accuracy and efficiency	49
3.2	Translations	53
3.2.1	Properties of exponential distances	53
3.2.2	Vector: boundedness and direction	54
3.2.3	Theorem for translations	55
3.2.4	Remarks	59
3.2.5	Illustration	61
3.3	Computational elements	64
3.3.1	Clustering: source points and source centres	64
3.3.2	Shifting: from source centres to target centres	66
3.3.3	Redistributing: target centres and target points	70
3.3.4	Potential Φ and its gradient $\nabla\Phi$	72
3.4	Salient features	76
3.4.1	Simplicity	76

3.4.2	Accuracy	77
3.4.3	Speed	77
3.4.4	Memory requirement	78
3.4.5	Retrieval of gradient of potential	80
3.4.6	Distributed and parallel processing	80
3.5	Algorithms	83
3.5.1	Common computational steps	83
3.5.2	Algorithm I (for $a = 1, b \geq 2^n$)	85
3.5.3	The scaling problem	86
3.5.4	Algorithm II (for $a = 1, b = 2^2$)	88
3.5.5	Algorithm III (for $a = 1, 2^2 \leq b = 2^m \leq 2^n$)	90
3.5.6	Algorithms: comparison	92
3.6	Numerical results	93
3.6.1	Approximation and error	93
3.6.2	Computational time $O(\mathbf{S}_{\text{app}}\mathbf{N})$	97
3.6.3	Algorithms I, II, and III : comparison	97
4	Solving the Charge Density Problem	106
4.1	Discretization	107
4.2	Matrix-vector multiplication	109
4.2.1	Charge to source (Q2S) translation	111
4.3	Preconditioning	130
4.4	Stopping criteria	132
4.5	Multi-level multi-precision iteration	134

4.5.1	Multi-level iteration	137
4.5.2	Multi-precision iteration	138
4.6	Numerical results	141
4.6.1	Illustrative examples	141
4.6.2	Computational time and memory storage	147
5	Integral Equations of Second Kind for Charge Density Problem	150
5.1	Existing approach	151
5.1.1	Integral equations	151
5.1.2	Difficulties	153
5.2	Integral equations of the second kind	156
5.2.1	Integral equations	156
5.2.2	Choices of parameters	157
5.3	Numerical experiments	160
5.3.1	Simulation procedure	161
5.3.2	Results	161
5.3.3	Observation	162
6	Numerical Extraction of Capacitance in a-Si TFTs and Imaging Arrays	167
6.1	Simulation vs. measurement	168
6.2	Simulation vs. parallel-plate approximation	171
6.2.1	Geometric overlap capacitance in a-Si TFTs	171
6.2.2	Interconnect crossover capacitance	174
6.3	Extraction of overlap length	176

7 Conclusions	180
Bibliography	182

List of Tables

1.1	Computational cost of translations in the FMA.	15
2.1	Absolute-error approximation: sample Gauss quadratures.	38
2.2	Relative-error approximation: sample Gauss quadratures.	39
2.3	Approximation and errors: $b = 2^2$	40
2.4	Approximation and errors: $b = 2^3$	41
2.5	Approximation and errors: $b = 2^4$	42
3.1	Number of s2T translations per cube	71
3.2	Computational cost of translations in EE method	78
3.3	Cost in memory independent of degree of accuracy	79
3.4	Memory requirement per cube in the two approaches: MP vs. EE.	80
3.5	Cost of operations per cube or per particle in TreeTraversal for all six directions	85
3.6	Cost of operations in Algorithm I	86
3.7	Cost of operations in Algorithm II	89
3.8	Cost of operations in Algorithm III	92

3.9	Maximum relative errors: approximation (3.1) and calculated potentials.	96
3.10	Running time as function of sample sizes in tree-like hierarchies of different levels.	98
3.11	Running time (s) for Algorithms I and II.	101
3.12	Average relative error of Algorithms I and II.	103
3.13	Running time (s) for Algorithms II and III.	103
3.14	Average relative error of Algorithms II and III.	105
4.1	Change in residual and associated relative error in calculated capacitance. Here C denotes the exact value of capacitance and \tilde{C} its computed value. The result is stabilized for $r \leq 2.5 \times 10^{-4}$. $\ V\ = 2.2 \times 10^1$	135
4.2	Change in residual and associated relative error in calculated capacitance. The result is stabilized for $r \leq 10^{-8}$. $\ V\ = 2.2 \times 10^1$	135
4.3	Relative change in calculated capacitance versus residual.	136
4.4	Number of iterations and cost ratio: with and without multi-level iteration.	138
4.5	Number of iterations and cost ratio: with and without multi-precision iteration.	140
4.6	Error in capacitance, charge density, and charge in Example 1.	145
4.7	Convergence rate in terms of number of iterations with different preconditioning in Example 1. The absolute error tolerance $\epsilon_{\text{abs}}^{\text{tol}} = 1.0 \times 10^{-6}$	145

4.8	Error in capacitance, charge density, and charge in Example 2. . . .	146
4.9	Convergence rate in terms of number of iterations with different preconditioning in Example 2. The absolute error tolerance $\epsilon_{\text{abs}}^{\text{tol}} = 1.0 \times 10^{-6}$	146
4.10	Computational time as a function of number of panels and iterations.	148
4.11	Computational memory storage as a function of number of panels. .	148
5.1	Convergence rate in terms of number of iterations and relative error in capacitance in Example 1.	163
5.2	Error in charge, potential and its normal derivative in Example 1.	164
5.3	Convergence rate in terms of number of iterations and relative error in capacitance in Example 2.	164
5.4	Error in charge, potential and its normal derivative in Example 2.	165
5.5	Effect of mismatch of integral equations of the first and second kinds due to scaling in Example 2. Scaling factor $s = 1.0 \times 10^{-6}$. Accuracy of potential approximation is 10^{-8}	165
5.6	Without preconditioning. Effect of mismatch of the integral equations of the first and second kinds due to scaling in Example 2. Scaling factor $s = 1.0 \times 10^6$. Accuracy of potential approximation is 10^{-8}	166
5.7	With preconditioning. Effect of mismatch of the integral equations of the first and second kinds due to scaling in Example 2. Scaling factor $s = 1.0 \times 10^6$. Accuracy of potential approximation is 10^{-8} .	166
6.1	Thickness of various layers (see Fig. 1.2) and their dielectric constants.	169

6.2	Overlap capacitance in a-Si TFTs as a function of overlap length: values shown are obtained from measurement, simulation, and the parallel-plate approximation.	169
6.3	Geometric overlap capacitances from parallel-plate formula (C_{approx}) and numerical simulation ($C_{\text{simulation}}$).	174
6.4	Crossover capacitances (fF) from parallel-plate approximation(C_{approx}) and numerical simulation ($C_{\text{simulation}}$).	176

List of Figures

1.1	(a) Schematic diagram of a pixel in a-Si imaging array and (b) interconnect addressing lines.	2
1.2	Schematic diagram of an a-Si TFT: (a) cross section and (b) geometric quasi-static coupling capacitances.	3
1.3	Multiconductor system and equivalent circuit.	4
1.4	Surface charge σ and potential Φ	7
3.1	A translation from Q to P through centres C_1 and C_2 : $E(QP) = E(QC_1) \times E(C_1C_2) \times E(C_2P)$	60
3.2	Translations for efficient approximation of the potential field at $N_2 = 3$ target points in Cube_2 due to $N_1 = 4$ source points in Cube_1	63
3.3	Translations in the EE method.	72
3.4	Running time for evaluation of potential only (dashed), and both potential and its gradient (solid). The overhead in evaluating both potential and its partial derivatives ranges from 1.06 to 1.15.	81

3.5	Computational time as a function of number of particles, over a range from 0.3×10^5 to 4×10^5 particles in tree-like hierarchy of (a) four levels and (b) five levels.	99
3.6	Computational time as a function of approximation size (S_{app}) for different sample sizes: (a) 5000, (b) 10000, (c) 25000, and (d) 40000.	100
3.7	Algorithms I and II: (a) running time and (b) average relative error.	102
3.8	Algorithms II and III: (a) running time and (b) average relative error.	104
4.1	Q2S translation for continuous charge distribution on a triangle ΔABC to a source center S : (i) all charge Q is translated to the centroid G and (ii) G is translated to S	112
4.2	Local coordinate system $\mathcal{B} = (e_x, e_y, e_z)$ associated with panel Π and transformation T of a triangle (ΔHUV) into a square ($[0, 1] \times [0, 1]$): $T(\mathbf{H}Q) = T(u\mathbf{H}U + v\mathbf{U}V) = (u, v)$	122
4.3	Computational time per iteration as a function of number of panels. Approximation size $S_{app} = 31$	149
6.1	Fabricated test structures: (a) layout and (b) photograph.	170
6.2	(a) Overlap capacitance (fF) as a function of overlap area (μm^2): measurement(+), parallel-plate approximation (*), and simulation (o). (b) Overlap capacitance (fF) as function of overlap length (μm): parallel-plate approximation (dashed) and simulation (solid); corresponding overlap areas are from $-500\mu m^2$ to $500\mu m^2$	172
6.3	Fringing factor $\alpha = \frac{C_{approx}}{C_{simulation}}$ as a function of overlap length L_{ov} . The substrate is glass.	175

6.4	Fringing factor $\alpha = \frac{C_{\text{approx}}}{C_{\text{simulation}}}$ as a function of crossover area. The substrate is glass.	177
6.5	(a) Overlap area and (b) crossover area.	178
6.6	Capacitance (fF) as function of overlap length (μm).	179

List of Symbols

A	Discretization matrix in equation $A\sigma = b$
A_s	Submatrix of matrix A
\overline{ABC}	Signed area of triangle ΔABC
a-Si	Amorphous silicon
$B = (e_x, e_y, e_z)$	Local coordinate system for a panel
b	Parameter associated with domain of approximation D_b Right-hand side of system of linear equations $A\sigma = b$
C	Set of child cubes (see (3.53))
C_d	Cost for direct pairwise particle-to-particle potential calculation
C_e, C_m	Cost for direct and indirect exponential distance calculation
C_{ij}	Coupling capacitance between conductors S_i and S_j
C_{new}, C_{old}	Capacitance obtained in iterative process
D_b	Domain of approximation for exponential expansion
$Dir(V)$	Direction of a vector V
DirSet	Set of six directions
dir	Direction associated with exponential distance $E[k, j](x, y, z)$
dS_X, dS_Y	Surface integral variable for X and Y
EE	Exponential expansion
$E[k, j](x, y, z)$	Exponential distance associated with indices $[k, j]$ of vector (x, y, z)
E_{AvgRel}, E_{MaxRel}	Average and maximum relative errors in potential calculation

FMA	Fast multipole algorithm
F_U	Finite-interval integral for infinite-interval integral for $\frac{1}{r}$
$G(X, Y)$	Green's function $\frac{1}{r}$, $r = \ X - Y\ $
$g^{dir}[k, j]$	See Definition 3.1.1
\mathcal{G}_i	Nested grids
H	See (4.9)
I	Identity matrix, identity integral operator
I_l, I_l^{dir}	Set of indices in interaction list (see (3.60))
$I_{D,l}^{dir}, I_F^{dir}, I_H^{dir}$	
I_1, I_2, I_3	Components of surface integral of $\frac{1}{r}$
I_{kj}^{dir}	Imaginary part of exponential distance
I_Π, I_{UV}	Surface and line integrals associated with exponential distance
$I_r(P, \Pi)$	Surface integral of $\frac{1}{r}$
I_{xg}, I_{yg}, I_{zg}	Surface integral of partial derivatives of $\frac{1}{r}$ in global coordinate system
I_x, I_y, I_z	Surface integral of partial derivatives of $\frac{1}{r}$ in local coordinate system
$J(T)$	Jacobian matrix of a linear transformation T (see (4.43))
$J_0()$	Bessel function of the first kind with order 0
K_c, K_d, K_f	Integral operators
L_m^n	Local expansion coefficient
L2P	Translation from local centre to particle/point
M_m^n	Multipole expansion coefficient

M2L	Translation from multipole centre to local centre
M2M	Translation between multipole centres
M2X	Translation from multipole coefficient to exponential coefficient
N	Number of points or unknowns of interest
$\mathbf{n}(X)$	Unit normal vector at a point X on a surface
n_c	Number of nodes in Chebyshev-Gauss integration
n_l	Order of Lagrange-Gauss integration
$O()$	Big O -notation
P_{pc}	Average number of particles per cube
P_l, P_r	Left or right preconditioner
P_{sr}	Submatrix of right preconditioner P_r
p	Order of multipole expansion
q	Charge or charge distribution
Q2M	Translation from charge point to multipole centre
Q2S	Translation from charge point to source centre
R_{kj}^{dir}	Real part of exponential distance
r	Distance between two points
	Radius
	Residual $\ A\sigma^i - b\ $
S	Source centre
	Union of all conductor surfaces and interfaces

s2S, s2T	Translation from source centre to source or target centre
S_{app}	Approximation size of exponential expansion
S_c	Union of all conductor surfaces
S_d	Interface surface between dielectric layers
S_i	Source centre Conductor surface
$S_{inner}(k)$	Number of nodes for inner loop corresponding to node k^{th} of outer loop
S_{outer}	Number of nodes in outer loop of exponential integral representation of $\frac{1}{r}$
$S(x, y, z)$	Exponential expansion for $\frac{1}{r}$, $r = \sqrt{x^2 + y^2 + z^2}$
s	Scaling factor
s_{c_j}	Node in Chebyshev-Gauss integration
T2P	Translation from target centre to particle for potential
TFT	Thin-film transistor
U	Finite value for ∞ in randomized algorithm
V	Vector of applied voltages (equal to b in $A\sigma = b$)
V_i	Applied voltage on conductor surface S_i
x2L	Translation from exponential expansion coefficient to local coefficient
x2X	Translation between exponential expansion coefficients
X_i	Point or particle
$X_i X_j$	Vector connecting points X_i and X_j
$Y_m^n(\theta, \phi)$	Spherical harmonics of angles θ and ϕ

α	Coefficient for integral equation of second kind in new formulation
	Fringing factor
α_o	Pre-factor
α_{kj}	Nodes for inner loop
β	Displacement angle
	Parameter in integral equation of second kind in new formulation
Δ_m, Δ_n	See § 3.5.4 (Algorithm II)
$\epsilon_c^+, \epsilon_c^-$	Dielectric constants
ϵ^+, ϵ^-	
$\epsilon_{abs}, \epsilon_{rel}$	Intended absolute and relative error for exponential expansion
ϵ_o	Permittivity of air ($= 8.854 \times 10^{-12} \text{F/m}$)
ϵ_{EE}	Error in potential approximation using exponential expansion
ϵ_{rel}^{cap}	Relative error in calculation of capacitance matrix
$\epsilon^{tol}, \epsilon_{abs}^{tol}, \epsilon_{rel}^{tol}$	Tolerance (accepted error) in solution of $A\sigma = b$
γ	Parameter in integral equation of second kind in new formulation
γ_I, γ_O	Relative contribution of error to inner and outer loops, respectively
λ_k, λ'_k	Nodes for outer loop
$\nabla\Phi(X)$	Gradient of potential Φ at point X
ω_{c_j}	Weight in Chebyshev-Gauss integration
ω_k	Weight in Gauss quadratures for $\frac{1}{r}$
$\omega_k^o, \omega_k^{o'}$	Weight in Gauss quadratures for outer loop of $\frac{1}{r}$

$\Phi(X)$	Potential at a point X
Φ^D, Φ^T	Components of potential
Φ_z^D, Φ_z^T	Components of the derivative of potential in z-direction
ρ	$\sqrt{x^2 + y^2}$
σ, σ_{ij}	Charge density

Chapter 1

Introduction

With the rapid increase in both component density and operating frequency, the parasitic coupling capacitance associated with interconnects poses serious design issues in high density and high performance integrated circuits. These issues are also common to large-area high-resolution amorphous silicon (a-Si) arrays for applications in X-ray imaging or active-matrix liquid-crystal displays. The parasitic capacitances increase with increasing component density [1]. The interconnect capacitance due to the crossover of the gate and data lines (see Fig. 1.1) introduces electronic noise, thus undermining the quality of the image generated [2]. The capacitive coupling due to geometric overlapping between the gate and source/drain electrodes in the TFT (see Fig. 1.2) can give rise to charge feedthrough thus shifting the pixel voltage [3].

To gain insight into the effect of parasitic coupling capacitance on the overall array performance, one needs to be able to extract the capacitance with a high degree of accuracy and in an efficient manner. In addition, this aids in further

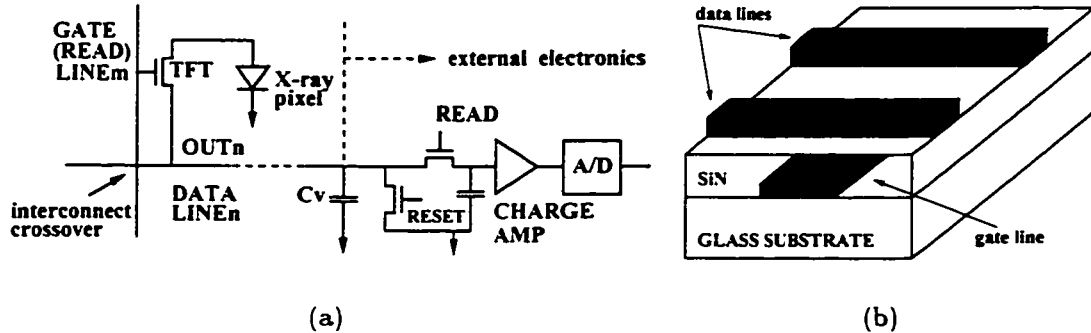


Figure 1.1: (a) Schematic diagram of a pixel in a-Si imaging array and (b) interconnect addressing lines.

development of equivalent circuit models for effective SPICE-like simulations for sensitivity analysis and design optimization at the system level.

1.1 Capacitance and charge density

The capacitance is calculated from the distribution of charge density on the surface of the conductors. In general, given a conductor system S_1, S_2, \dots, S_n with the applied voltages V_1, V_2, \dots, V_n , we need to determine the charge density σ on these conductor surfaces. Here, we assume that the conductors are embedded in homogeneous or multiple dielectric media. The surface S of interest consists of the conductor surfaces (denoted as $S_c = \bigcup S_i$) and the interfaces (denoted as S_d) between dielectric layers. Figure 1.3 shows a system of three conductors and its equivalent circuits. Here, each non-ground node represents a conductor and the element connecting two nodes is the associated coupling capacitance. The capacitance C_{ij} between two conductors S_i and S_j is defined as the surface integral of

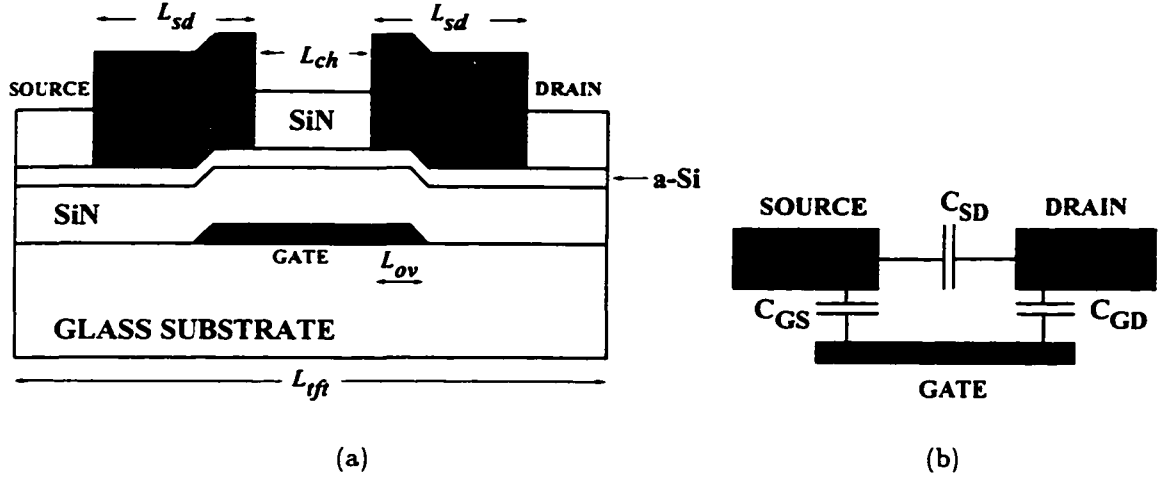


Figure 1.2: Schematic diagram of an a-Si TFT: (a) cross section and (b) geometric quasi-static coupling capacitances.

charge density σ on S_j [4]

$$C_{ij} \triangleq \int_{S_j} \sigma_{ij}(X) dS_X. \quad (1.1)$$

Here, $\sigma_{ij}(X)$ is the charge density on conductor S_j when conductor S_i is applied a voltage of $1V$ (i.e., $V_i = 1$) and other conductors are grounded (i.e., $V_k = 0, k \neq i$). The capacitances C_{ij} form a matrix C , called *capacitance matrix*, with which the relationship between the charge vector Q and the voltage vector V of the conductors are expressed:

$$Q = CV \quad (1.2)$$

In theory, the capacitance matrix is symmetric. However, in real computation, due to amplification by different values of the permittivity (see eqs. (1.18) and (1.20)),

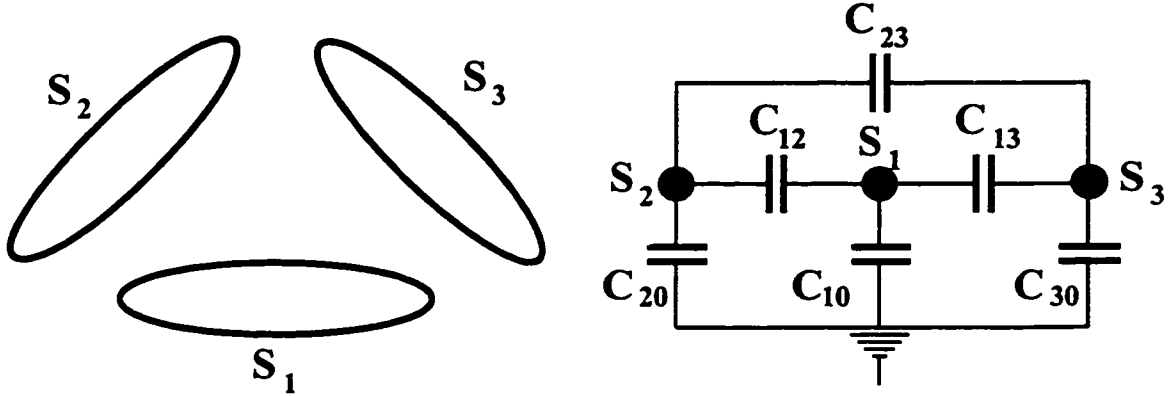


Figure 1.3: Multiconductor system and equivalent circuit.

the errors in the computed charge density can result in $C_{ij} \neq C_{ji}$. It is noted that $C_{ii} > 0$ and $C_{ij} < 0$ for $j \neq i$; furthermore, the coupling capacitance C_{i0} between conductor S_i and the ground at infinity is $C_{i0} = \sum_{j=1}^n C_{ij} \geq 0$ (see [5, 6]).

The relationship between the charge density on a conductor and the potential can be expressed as [4]:

$$\sigma(X) = \epsilon_c^-(X) \frac{d\Phi^-(X)}{dn} - \epsilon_c^+(X) \frac{d\Phi^+(X)}{dn}, \quad X \in S_c. \quad (1.3)$$

Here, $\epsilon_c^+(X)$ and $\epsilon_c^-(X)$ (with $\epsilon_c^+(X) \leq \epsilon_c^-(X)$) denote the permittivities of the media separated by the conductor surface. The unit normal vector $\mathbf{n}(X)$ points to the medium with dielectric constant ϵ_c^+ . The derivatives $\frac{d\Phi^+(X)}{dn}$ and $\frac{d\Phi^-(X)}{dn}$ are taken along the direction of normal vector \mathbf{n} at points $(X + 0\mathbf{n})$ and $(X - 0\mathbf{n})$, respectively. For a thick conductor, we have $\frac{d\Phi^-(X)}{dn} = 0$ and the above expression

is reduced to

$$\sigma(X) = -\epsilon_c^+(X) \frac{d\Phi^+(X)}{dn}. \quad (1.4)$$

There are two main approaches for describing the charge density: differential-equation formulation [7] and integral-equation formulation [8,9]. The former leads to differential equations with the potential as unknown. The latter yields integral equations with the charge density or capacitance as unknown.

1.1.1 Differential-equation formulation

With the equations (1.3) and (1.4), the charge density problem can be described in terms of the potential. In the absence of external source charges, the potential satisfies the Laplace equation:

$$\nabla^2\Phi = \frac{\partial^2\Phi}{\partial x^2} + \frac{\partial^2\Phi}{\partial y^2} + \frac{\partial^2\Phi}{\partial z^2} = 0. \quad (1.5)$$

The boundary conditions are

$$\Phi(X) = V_i, \quad X \in S_i \quad (1.6)$$

$$\epsilon^+ \frac{d\Phi^+(X)}{dn} = \epsilon^- \frac{d\Phi^-(X)}{dn}, \quad X \in S_d \quad (1.7)$$

$$\Phi(\infty) = 0. \quad (1.8)$$

Here, ϵ^- and ϵ^+ denote the permittivities of the respective dielectric layers. The boundary conditions (1.6) and (1.8) signify that each conductor surface or the infinity is equipotential while the discontinuity of the electric field across the interface

between two dielectric layers is described in the boundary condition (1.7).

The Laplace equation (1.5) together with the boundary conditions (1.6)–(1.8) can be solved using finite difference or finite element methods (see, for example, [10]). The resultant matrix is sparse thanks to local coupling but large due to discretization of the whole volume. The main advantage is that there are efficient methods for solving sparse systems (see, for example, [11] and the references therein). The main problem is that the unbounded domain (1.8) needs to be replaced by a bounded one at the cost of loss of accuracy. In this case, at the boundary S_{bd} of the bounded domain, the Neumann condition is often employed:

$$\frac{d\Phi(X)}{dn} = 0, \quad X \in S_{bd}. \quad (1.9)$$

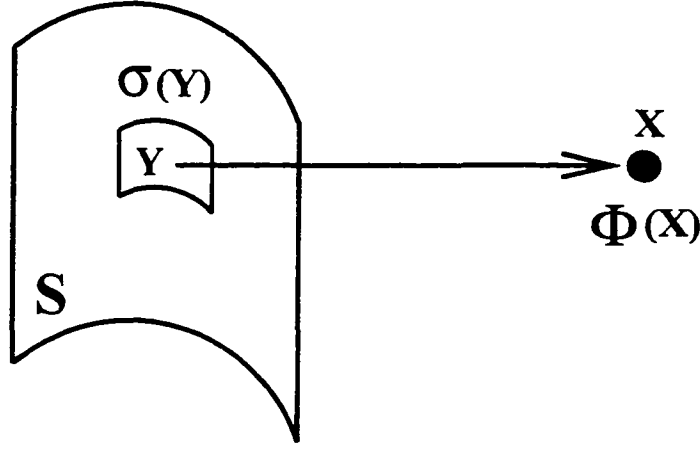
1.1.2 Integral-equation formulation

Instead of solving the Laplace equation, we can express the potential Φ in terms of the charge density (see Fig. 1.4):

$$\Phi(X) = \int_S G(X, Y) \sigma(Y) dS_Y. \quad (1.10)$$

where $G(X, Y)$ is the associated Green's function $\frac{1}{r}$:

$$G(X, Y) = \frac{1}{4\pi \|X - Y\|}. \quad (1.11)$$

Figure 1.4: Surface charge σ and potential Φ .

With the jump relation [8], the derivatives $\frac{d\Phi^+(X)}{dn}$ and $\frac{d\Phi^-(X)}{dn}$ can be expressed in terms of the charge density as:

$$\frac{d\Phi^+(X)}{dn} = \int_S \frac{dG(X, Y)}{dn} \sigma(Y) dS_Y - \frac{1}{2} \sigma(X) \quad (1.12)$$

$$\frac{d\Phi^-(X)}{dn} = \int_S \frac{dG(X, Y)}{dn} \sigma(Y) dS_Y + \frac{1}{2} \sigma(X) \quad (1.13)$$

With the equations (1.12) and (1.13), the discontinuity of the electric field at the interface (eq. (1.7)) can now be described in terms of an integral equation:

$$\sigma(X) + 2\lambda \nabla \Phi(X) \cdot \mathbf{n}(X) = 0, \quad X \in S_d, \quad (1.14)$$

where

$$\lambda = \frac{\epsilon^- - \epsilon^+}{\epsilon^- + \epsilon^+} \quad (1.15)$$

Assuming that $\epsilon^- > \epsilon^+$ and $\mathbf{n}(X)$ points towards the dielectric medium with smaller permittivity (i.e., ϵ^+), we have $\lambda \in [0, 1]$. The model equations for the charge density are the systems of (1.6) and (1.14) with $\Phi(X)$ expressed explicitly in (1.10) rather than (1.5).

$$\Phi(X) = V_i, \quad X \in S_i \quad (1.16)$$

$$\sigma(X) + 2\lambda \nabla \Phi(X) \cdot \mathbf{n}(X) = 0, \quad X \in S_d. \quad (1.17)$$

Note that the Neumann condition (1.9) can also be simulated as a special case of the interface equation (1.14). For $\frac{d\Phi^+(X)}{dn} = 0$ or $\frac{d\Phi^-(X)}{dn} = 0$, we can choose $\lambda = -1$ or $\lambda = 1$, respectively.

With the charge density σ obtained from the above model equations, in order to calculate the capacitance described in (1.1), we need to scale σ accordingly. The reason is that when describing the Green's function $\frac{1}{r}$, we do not include the permittivity. For thick conductors, the formula for C_{ij} in (1.1) now reads:

$$C_{ij} = \int_{S_j} \epsilon_c^+(X) \sigma_{ij}(X) dS_X. \quad (1.18)$$

For infinitesimally thin conductors, using eqs. (1.3), and the jump relation (1.12) and (1.13), we obtain a formula for C_{ij} :

$$\begin{aligned} C_{ij} &= \int_{S_j} \left(\frac{[\epsilon^-(X) + \epsilon^+(X)]}{2} \sigma_{ij}(X) + \right. \\ &\quad \left. [\epsilon^-(X) - \epsilon^+(X)] \int_S \frac{dG(X, Y)}{dn} \sigma_{ij}(Y) dS_Y \right) dS_X \quad (1.19) \\ &= \int_{S_j} \left(\frac{[\epsilon^-(X) + \epsilon^+(X)]}{2} \sigma_{ij}(X) + [\epsilon^-(X) - \epsilon^+(X)] \nabla \Phi(X) \cdot \mathbf{n}(X) \right) dS_X \quad (1.20) \end{aligned}$$

If the the medium contacting the conductor is homogeneous (i.e., $\epsilon^-(X) = \epsilon^+(X)$), the formula (1.20) is reduced to (1.18).

In discretization of the model equations (1.17), the resultant matrix is full due to global coupling but smaller thanks to discretization of the surfaces S only. The main advantage of the integral-equation formulation is that we can avoid the unbounded domain (1.8) and need to deal with the surfaces of conductors and interfaces only. The main problem is that the discretization matrix is dense. The challenge is how to deal with such dense systems with millions of unknowns. Any scheme that requires storage for the whole matrix is practically ruled out since the memory requirement is of the order $O(N^2)$. Here, N denotes the number of unknowns. Furthermore, any scheme that needs computational time of the order $O(N^2)$ for matrix-vector multiplication can be considered too expensive. It is one of the motivations of this work to meet this challenge.

The overall computational accuracy and performance in solving the system (1.16)–(1.17) lies in the evaluation of the potential $\Phi(X)$ and its gradient $\nabla\Phi(X)$ at all N points of interest.

1.2 Evaluation of three-dimensional potential field

In a more general context, efficient methods for rapid evaluation of the potential field and its gradient in three dimensions are essential in many practical problems in applied mathematics, physics, chemistry, and engineering. In these problems, given an ensemble of particles located at points X_i , $i = 1..N$ in R^3 and a set of associated real values q_i , $i = 1..N$, we need to rapidly and accurately evaluate the potential and its gradient at each location X_i . The potential and its gradient are

given by

$$\Phi(X_i) = \sum_{j=1, j \neq i}^N \frac{q_j}{\|X_i X_j\|} \quad (1.21)$$

$$\nabla \Phi(X_i) = \sum_{j=1, j \neq i}^N \nabla \left(\frac{q_j}{\|X_i X_j\|} \right). \quad (1.22)$$

The particles can represent different entities. In astrophysics, they can be masses: in chemistry, they can be atoms: in electrical engineering, they can be charges: in magnetism, they can be magnetic dipoles. In this thesis, we will adopt the terminology used in electrostatics, *viz.* charge and potential, and the terms particle and point will be used interchangeably. At first glance, the expressions (1.21) and (1.22) are simple to calculate. However, for a large system, it may take too much computing resource to perform the calculation at all those N points. It is another motivation of this work to tackle this problem.

In computation, the potential (1.21) can be expressed as

$$\Phi = \Phi_{near} + \Phi_{far} + \Phi_{external}. \quad (1.23)$$

where Φ_{near} denotes the near-field potential (e.g. Van der Waals potential), Φ_{far} the far-field potential (e.g. coulombic or gravitational potential), and $\Phi_{external}$ the contribution to potential from external sources which are not part of the ensemble. The evaluation of Φ_{near} is of the order $O(N)$ because Φ_{near} decays rapidly and each particle interacts significantly only with neighbours of close proximity. The evaluation of $\Phi_{external}$ is also of the order $O(N)$ because its contribution to the

potential of each particle is calculated only once. However, the evaluation of Φ_{far} , which belongs to a more general class of problems known as the N -body problem, presents a challenge. If the evaluation is performed through direct pairwise particle-to-particle interactions, the computational time is of the order $O(N^2)$, which is prohibitively expensive for large N .

State-of-the-art algorithms have managed to evaluate the potential (with some loss of accuracy) in computational time of only $O(N \log N)$ or even $O(N)$ [12, 13]. The main idea behind these algorithms is to replace a cluster of particles by a pseudo-particle or centre, and subsequent processes need only to deal with the pseudo-particle (centre of the cluster) instead of each of the individual particles that form the cluster. These algorithms rely on two main steps:

- (i) Decomposing the ensemble of particles into clusters. This is usually realized using a tree-like hierarchy of cubes, in which the root cube is a bounding box containing all of the particles. Such a tree-like hierarchy has some important features. It offers an efficient method for the systematic clustering of particles and their associated hierarchical processing. It also provides a means to determine quickly whether a pair of particles are close to each other without the need to calculate the distance between them.
- (ii) Starting with the finest level in the tree-like hierarchy, we replace all particles in each (leaf) cube with an equivalent pseudo-particle located at the cube's centre. This operation is called *translation*. The process is then continued at each level until the root cube is reached. Then, starting from the root cube, a similar process progresses down to the finest level, and the potential and its

gradient are obtained.

The above processes can be performed using different methods. Among these methods are the popular Barnes-Hut algorithm [12] and the Greengard-Rokhlin Fast Multipole Algorithm (FMA) [13–15]. The mathematical background for translation of centres comes from the theory of spherical harmonics [16] and notably the *addition theorem*. The expansion of potential in terms of spherical harmonics is called *multipole expansion* and the associated methods can be classified as multipole-expansion-based (MP) methods. These methods have so far been the choice for evaluation of the potential field. In electrical engineering, for example, they are used for capacitance extraction [17, 18] and for electrostatic analysis in micro-electro-mechanical systems [19].

1.3 Multipole-expansion-based methods

In this section, we briefly describe multipole expansion for the potential, which was developed a century ago for solution of the Laplace equation, and its use for computation of the potential field of a particle system in linear time, which was realized in the last decade.

1.3.1 Multipole expansion

According to the theory of spherical harmonics, any harmonic function, Φ , which satisfies the Laplace equation, can be expanded in the form

$$\Phi = \sum_{n=0}^{\infty} \sum_{m=-n}^n \left(L_m^n r^n + \frac{M_m^n}{r^{n+1}} \right) Y_m^n(\theta, \phi). \quad (1.24)$$

Here, (r, θ, ϕ) are the spherical coordinates of the point at which the harmonic function Φ is evaluated. $Y_m^n(\theta, \phi)$ are spherical harmonics, and M_m^n and L_m^n are called multipole and local expansion coefficients, respectively. The relation (1.24) is also referred to as multipole expansion. In numerical computation, the expansion is truncated and the outer sum is evaluated for $0 \leq n \leq p$. Here, p is some non-negative integer, which is called the order of the expansion. A truncated version of (1.24) is

$$\Phi = \sum_{n=0}^p \sum_{m=-n}^n \left(L_m^n r^n + \frac{M_m^n}{r^{n+1}} \right) Y_m^n(\theta, \phi). \quad (1.25)$$

With the truncation, the function Φ can be determined approximately by spherical harmonics $Y_m^n(\theta, \phi)$, and the multipole and local expansion coefficients M_m^n and L_m^n for $0 \leq n \leq p$ and $-n \leq m \leq n$. The order of expansion p is the important parameter that characterizes the overall performance of the MP method. The larger the value of p , the more accurate is the result. However, the computing process becomes more costly in terms of computational time and memory requirements. A significant numerical effort in the MP method lies in obtaining the expansion coefficients efficiently through a sequence of translations from source points to multipole centres (Q2M), from multipole centres to multipole centres (M2M), from multipole centres to local centres (M2L), from local centres to local centres (L2L), and from local centres to the target points for evaluation of the potentials (L2P). The interested reader is referred to the work of Steinborn & Ruedenberg [20] and Greengard [13] for detailed discussions of these translations. For the purpose of comparison, we briefly discuss some computational aspects of the translations. We focus especially

on the original FMA [13] and its latest version [14, 15]. The FMA is an ingenious combination of the ideas from the addition theorem in spherical harmonics and the hierarchical algorithms (see, for example, [12]) to approximate the potential with virtually any degree of accuracy in the order of only $O(N)$ computational time.

1.3.2 Translations and computational cost

A rough estimation of computational cost of the FMA in terms of running time and memory requirement is summarized in Table 1.1.

Original version

In the original FMA, the cost for each of the translations M2M, M2L, and L2L is of the order $O(p^4)$. For a given order of expansion p , there are $\frac{(p+1)(p+2)}{2}$ coefficients for each multipole centre and each local centre. These must be stored explicitly so that coefficients for new centres can be generated. The evaluation of each new coefficient involves a double sum of the order $O(p^2)$. Therefore, the cost in computational time is $O(p^4)$, and the cost in memory is $O(p^2)$. Furthermore, for each cube, we need about 875 M2L translations, which makes the translation of multipole centres to local centres the most expensive operation in the FMA. With the given dependence on p , it becomes very expensive in terms of computational time and memory to achieve a high degree of accuracy. An approach to reduce the overall cost of the FMA is to reduce the number of M2L translations by allowing M2L translations between cubes that are 1 cube apart, instead of 2 cubes apart as in the original FMA. The number of M2L translations for each cube can be reduced to 189. However, such translations require a larger p to achieve the same degree of accuracy.

Version	Time	Memory	Reference
Original	$O(p^4)$	$O(p^2)$	[13]
New	$O(p^3)$	$O(p^2) + O(6p^2)$	[14, 15]

Table 1.1: Computational cost of translations in the FMA.

New version

Recently, multipole and exponential expansions were combined to reduce the cost of M2L translations significantly [14, 15, 21, 22]. The cost of translations using exponential expansion (X2X) is roughly $O(p^2)$ and the number of M2L is 189. The combination of both expansions requires recomputation of coefficients to deal with the conversion of multipole coefficients to exponential coefficients (M2X). This is then followed by a reverse conversion from exponential coefficients to local coefficients (X2L). Both of these conversions require $O(p^3)$. There are also rotations prior to an M2X and posterior to an X2L. These rotations also require $O(p^3)$. More specifically, at each cube, we need 5 rotations of M , 6 M2X conversions, 189 X2X translations, 6 X2L conversions, and 5 rotations of L . Thus, the total cost for M2L translation at each cube can be estimated as $O(22p^3 + 189p^2)$, which is a significant reduction from $O(875p^4)$ of the original version. With optimization, this cost can be further reduced. In terms of memory requirement, the new version needs $O(6p^2)$ extra units of memory for storing the exponential expansion coefficients in the six associated directions for cubes whose M2L translations have not been completed. Furthermore, the use of combined expansions may lead to additional loss of accuracy.

1.4 Difficulties and motivations

In studying the problems associated with the computation of capacitance, charge density, and general three-dimensional potential, we identify the following difficulties.

From a computational standpoint, the main difficulties in the numerical extraction of capacitance in a-Si TFTs and imaging arrays include:

1. The extreme device geometries, in which the ratio of thin film thickness to other physical dimensions (such as the width and length) can well be of the order of 10^{-3} .
2. The floating potential of the glass substrate: the substrate now needs to be included as part of the computational domain.
3. Treatment of multi-dielectric media where the electric field is discontinuous across dielectric interfaces. This requires computation of the electric field (see eq. (1.14)).

The extreme geometry of metal lines and active devices, and the presence of the glass substrate require a large number N of panels (or mesh elements) to barely resolve the surfaces of interconnects and the interfaces between two dielectric media. This leads to a very large system of equations. Furthermore, to gain sufficient accuracy on the electric field, we may need to evaluate the potential with a higher degree of accuracy. Accurate evaluation of the potential is also needed to ensure that the errors in the computed charge density are not due to the approximation of the potential and its gradient.

With the MP methods, to evaluate the potential with accuracy of $O(\frac{1}{2^p})$, where p is the order of the multipole expansion, the computational time is of $O(p^3 N)$ and the associated memory requirement is of $O(p^2 N)$ (see Table 1.1). For accurate large-scale simulation (i.e., with high p and large N in the range of hundred thousands or millions), this memory requirement of $O(p^2 N)$ can be problematic and can impose a severe constraint on computing resources.

For a large number N of unknowns, due to the mismatch between the integral equations (1.16) (first kind) and (1.17) (second kind), the discretization matrix can become increasingly ill-conditioned. As a result, the matrix-vector multiplication is very sensitive to error in the solution, leading to slow convergence rate and even incorrect solution.

The above difficulties led us to the development of a new approach for evaluation of the potential field in three dimensions and a new formulation for the charge density problem. With the latter, we obtain integral equations of the second kind; the discretization matrix has improved conditioning, leading to a faster convergence rate. The former, which is referred to as the exponential-expansion-based (EE) method, employs an exponential integral representation of the Green's function. With the EE method, the potential and its gradient can be evaluated in linear time with memory requirements independent of the desired degree of accuracy and with different forms of parallelism available for remotely-distributed networks and closely-coupled parallel systems.

The proposed approach and formulation are documented in detail in a series of technical reports [23–28]. The EE method was first briefly reported in [29] and the first comprehensive treatment of the method is presented in [30].

1.5 Organization of thesis

In the chapter that follows, we describe a randomized algorithm for generating the exponential expansions for the Green's function. Starting with an exponential integral representation of the Green's function, we discuss how to construct the Gauss quadratures for the expansion with the number of nodes kept as small as possible for a given intended error.

The fundamentals of the EE method are presented in chapter 3. We discuss the relationship between the accuracy of the exponential expansion and that of the computed potential and its gradient. Here, we establish the main theorem for performing translations. The computational elements associated with the translations are described and the salient features are identified. Three algorithms together with the cost analysis and numerical results are presented.

Chapter 4 is devoted to the discussion of solving the charge density problem and the employment of the EE in this process to achieve linear running time and to facilitate any desired degree of accuracy. This includes a description of the panel method with the exponential expansion, preconditioning techniques, stopping criteria, and our proposed multi-level multi-precision iterative process. Illustrative examples for computing the charge density on spherical conductors are also given.

In chapter 5, we report on a new formulation for the charge density problem. We identify the mismatch between integral equations of the first and second kinds and its effect on the convergence rate. With the proposed formulation, the charge density is described in terms of integral equations of the second kind. The choice of related parameters in the integral equations are discussed together with numerical

results.

The numerical extraction of capacitance associated with a-Si TFTs and interconnect crossover of addressing lines in imaging arrays is discussed in chapter 6. The simulation results are compared with those from the parallel-plate formula and measurements. The last chapter summarizes the work and outlines future research extensions.

Chapter 2

Generating Exponential Expansion

The underlying approximation of the Green's function $\frac{1}{r}$ used in the EE method is the exponential expansion. It is the quadrature formula of Gaussian type of an exponential integral representation of the Green's function. Two important parameters associated with the expansion are the intended error and the size of the expansion. They determine the overall accuracy and computational performance of the evaluation of the potential. In this chapter, we present a probabilistic-type approach for generating Gauss quadratures for the integral that maintains the desired exponential forms and at the same time, for a given intended error, keeps the size of the expansion as small as possible.

2.1 Exponential expansion

The Green's function $\frac{1}{r}$ can be expressed in an exponential integral form as:

$$\frac{1}{r} = \frac{1}{\sqrt{x^2 + y^2 + z^2}} = \int_0^\infty e^{-\lambda z} J_0(\rho\lambda) d\lambda, \quad (2.1)$$

where $z > 0$, $\rho = \sqrt{x^2 + y^2}$, and J_0 is Bessel function of the first kind with order 0. The validity of the above expression can be verified with the Laplace transform of the Bessel function J_0 (see. e.g., [31]):

$$\mathcal{L}\{J_0\} \equiv F(s) = \int_0^\infty e^{-st} J_0(ct) dt \quad (2.2)$$

$$= \frac{1}{\sqrt{s^2 + c^2}}. \quad (2.3)$$

The proof can be found in [32, 33]. By replacing $s = z$ and $c = \rho$, we obtain exactly the exponential integral representation of $\frac{1}{r}$. Furthermore, the Bessel function J_0 can also be expressed in an exponential integral form as (see §2.5):

$$J_0(\rho\lambda) = \frac{1}{2\pi} \int_0^{2\pi} e^{i\lambda(x \cos \alpha + y \sin \alpha)} d\alpha. \quad (2.4)$$

$$= \text{real} \left[\frac{1}{\pi} \int_0^\pi e^{i\lambda(x \cos \alpha + y \sin \alpha)} d\alpha \right] \quad (2.5)$$

In our problem, for a given intended error $\epsilon \in \{\epsilon_{rel}, \epsilon_{abs}\}$, we seek an approximation of the following form for the integral (2.1):

$$\frac{1}{r} \approx \text{real}[S(x, y, z)] \quad (2.6)$$

where the desired exponential expansion $S(x, y, z)$ is of the form:

$$S(x, y, z) = \sum_{k=1}^{S_{outer}} \omega_k \sum_{j=1}^{S_{inner}(k)} E[k, j](x, y, z) \quad (2.7)$$

$$E[k, j](x, y, z) = e^{\lambda_k(-z + i(x \cos \alpha_{kj} + y \sin \alpha_{kj}))}. \quad (2.8)$$

The domain of approximation D_b is designed with the EE method in mind for far field translations:

$$D_b = \{(x, y, z) | 1 \leq z \leq b, \max(-b, -z - 2) \leq x, y \leq \min(b, z + 2)\}. \quad (2.9)$$

Here, b is an integer of the form 2^m , $m \geq 2$. The quantity b indicates how far in the $+z$ -direction a point (x, y, z) can be in D_b in order for the approximation (2.6) to be valid. The quantities to be determined are the weights ω_k , nodes λ_k and α_{kj} for the outer integral (2.1) and inner integral (2.5), as well as the respective numbers of nodes S_{outer} and $S_{\text{inner}}(k)$.

An important parameter for characterizing the efficiency of the expansion (2.7) is its size:

$$S_{\text{app}} = \sum_{k=1}^{S_{\text{outer}}} \sum_{j=1}^{S_{\text{inner}}(k)} 1 = \sum_{k=1}^{S_{\text{outer}}} S_{\text{inner}}(k). \quad (2.10)$$

Note that S_{app} depends on both the domain of approximation D_b and the accuracy of the approximation (2.6), i.e., the intended relative error ϵ_{rel} or absolute error ϵ_{abs} .

The integral (2.1) is a special case of the following two general forms:

$$I_1(x, y, z) = \int_0^\infty e^{-\lambda z} f(\rho\lambda) d\lambda \quad (2.11)$$

$$I_2(x, y, z) = \int_0^d g(\lambda) J_0(\rho\lambda) d\lambda. \quad (2.12)$$

Here, d is a positive number, finite or infinite. There are several approaches to generate Gauss quadratures for the above two forms.

1. The simplest approach to evaluate (2.11) would be to use the classical Laguerre-Gauss integration. However, conventional methods of approximating $f(\rho\lambda)$ by a low order polynomial may not be applied here for the Bessel function $J_0(\rho\lambda)$. The order required for Laguerre-Gauss quadrature is large, which gives large S_{outer} . Furthermore, the nodes λ_k in Laguerre-Gauss integration are also large. This makes it more difficult to approximate the Bessel function $J_0(\rho\lambda_k)$, and hence $S_{\text{inner}}(k)$ also becomes large. As the result, the size of the approximation S_{app} turns out to be too large.
2. Another approach to evaluate (2.11) is to find orthogonal polynomials for the weight function $W(\lambda) = e^{-\lambda z}$ [34]. However, this may have the same drawbacks as using Laguerre-Gauss integration. Furthermore, the procedure involved has to deal with solving a system of non-linear equations.
3. The exponentially decaying behavior of $e^{-\lambda z}$ for large λ as well as the oscillatory behavior of Bessel function $J_0(\rho\lambda)$ in (2.12) are made use of in [35–37]. However, the desired exponential form $E[k, j]$ is lost.
4. Rokhlin and Yarvin [38] give a numerical scheme for the design of generalized Gaussian quadratures. The method can be applied to a wide range of functions, including all of the above cases. However, the scheme needs other numerical tools for approximation and singular value decomposition, and has to deal with solving a system of non-linear equations.

In light of the above issues, we develop a new probabilistic-type approach [23.39]. The randomized approach bears some resemblance to the Monte Carlo method in that it needs a sequence of random points (x_i, y_i, z_i) . However, unlike the Monte Carlo method, which tries to evaluate the integral, the proposed method attempts to find a suitable order n for the Gauss quadratures for the integral. Once the order n is obtained, weights and nodes can be generated from the well-known Legendre-Gauss and Chebyshev-Gauss quadratures. Unlike the methods in [34.38], the method proposed here does not involve solution of non-linear equations. While taking advantage of the exponentially decaying behavior of $e^{-\lambda z}$ and the boundedness of the Bessel function $J_0(\rho\lambda)$, the proposed approach always maintains the desired exponential terms $E[k, j](x, y, z)$.

The rest of the chapter is organized as follows. In § 2.2, we give a summary of the new approach. In § 2.3, we discuss how to approximate, for a given relative error ϵ_{rel} , the infinite-interval integral (2.1) using the Gauss quadratures generated for a finite-interval integral. We present in § 2.4 a randomized approach to obtain a suitable order of the Gauss quadratures for the finite-interval integral. Section § 2.5 is devoted for generating the Gauss quadratures for the Bessel function $J_0(\rho\lambda)$. In § 2.6, we present some numerical results.

2.2 Summary of randomized approach

Let γ_o and γ_I be numbers such that:

$$\gamma_o + \gamma_I = 1, \quad \gamma_o > 0, \quad \gamma_I > 0. \quad (2.13)$$

The numbers γ_o and γ_l are used to indicate the relative contribution of the outer and inner approximations, respectively, to the overall relative error ϵ_{rel} . The proposed procedure for generating the exponential expansion can be summarized as follows:

Step 1 By taking advantage of the exponentially decaying behavior of $e^{-\lambda z}$ and the boundedness (by unit) of the Bessel function $J_0(\rho\lambda)$, for a given relative error ϵ_{rel} , we replace the infinite interval $[0, \infty]$ with a finite interval $[0, U]$ such that for some number $\gamma_{o1} \in (0, 1)$:

$$\frac{\left| \int_0^\infty e^{-\lambda z} J_0(\rho\lambda) d\lambda - \int_0^U e^{-\lambda z} J_0(\rho\lambda) d\lambda \right|}{\frac{1}{r}} \leq \gamma_{o1} \gamma_o \epsilon_{rel}. \quad (2.14)$$

Any Gauss quadratures that can approximate the finite-interval integral:

$$\frac{\left| \int_0^U e^{-\lambda z} J_0(\rho\lambda) d\lambda - \sum_{k=1}^{S_{outer}} \omega_k^o e^{-\lambda_k z} J_0(\rho\lambda_k) \right|}{\frac{1}{r}} < (1 - \gamma_{o1}) \gamma_o \epsilon_{rel}. \quad (2.15)$$

can also approximate the infinite-interval integral:

$$\frac{\left| \int_0^\infty e^{-\lambda z} J_0(\rho\lambda) d\lambda - \sum_{k=1}^{S_{outer}} \omega_k^o e^{-\lambda_k z} J_0(\rho\lambda_k) \right|}{\frac{1}{r}} < \gamma_o \epsilon_{rel}. \quad (2.16)$$

Hence, this step makes it possible to use Gauss quadratures for the finite-interval integral in (2.15) just as the Gauss quadratures for the infinite-interval integral in (2.16).

Step 2 Using a sequence of random points (x_i, y_i, z_i) , we find a suitable order n_l for Legendre-Gauss quadrature for the finite-interval integral $\int_0^U e^{-\lambda z} J_0(\rho\lambda) d\lambda$

that probably satisfies the following condition for all points (x, y, z) in D_b :

$$\frac{\left| \int_0^\infty e^{-\lambda z} J_0(\rho\lambda) d\lambda - \sum_{k=1}^{S_{outer}} \omega_k^o e^{-\lambda_k z} J_0(\rho\lambda_k) \right|}{\frac{1}{r}} < \gamma_o \epsilon_{rel}. \quad (2.17)$$

Note that in the above expression, we deal with the infinite-interval integral directly. The reason is that its value is known, namely $\frac{1}{r}$. There is no need to take the intermediate step (2.15).

Step 3 For each node λ_k , using the same randomized procedure in step 2, we find a suitable number of nodes $S_{inner}(k)$ for the Chebyshev-Gauss or midpoint quadrature for the integral representation of Bessel function $J_0(\rho\lambda_k)$ in (2.5) that probably satisfies the following condition for all points (x, y, z) in D_b :

$$\frac{\left| \omega_k^o e^{-\lambda_k z} J_0(\rho\lambda_k) - \sum_{j=1}^{S_{inner}(k)} \omega_k E[k, j](x, y, z) \right|}{\frac{1}{r}} < \gamma_{I_k} \gamma_t \epsilon_{rel}. \quad (2.18)$$

where

$$\omega_k = \frac{\omega_k^o}{S_{inner}(k)} \quad (2.19)$$

$$\sum_{k=1}^{S_{outer}} \gamma_{I_k} = 1, \quad \gamma_{I_k} > 0, \quad k = 1..S_{outer}. \quad (2.20)$$

The more stringent relative error $\gamma_{I_k} \gamma_I \epsilon_{rel}$ and the requirement (2.20) is to ensure that:

$$\frac{\left| \sum_{k=1}^{S_{outer}} \omega_k^\rho e^{-\lambda_k z} J_0(\rho \lambda_k) - \sum_{k=1}^{S_{outer}} \sum_{j=1}^{S_{inner}(k)} \omega_k E[k, j](x, y, z) \right|}{\frac{1}{r}} < \gamma_I \epsilon_{rel}, \quad (2.21)$$

and hence, with (2.13) and (2.17):

$$\frac{\left| \int_0^\infty e^{-\lambda z} J_0(\rho \lambda) d\lambda - \sum_{k=1}^{S_{outer}} \sum_{j=1}^{S_{inner}(k)} \omega_k E[k, j](x, y, z) \right|}{\frac{1}{r}} < \epsilon_{rel}. \quad (2.22)$$

Step 4 Putting the coefficients together, and using another sequence of random points, we verify that the approximation (2.6) is valid. This test is expected not to fail. If it does fail then either step 2 and/or step 3 is repeated for the particular point that causes the approximation to break down.

2.3 From infinite-interval to finite-interval integrals

We denote the finite-interval integral in (2.14) as $F_U(x, y, z)$:

$$F_U(x, y, z) = \int_0^U e^{-\lambda z} J_0(\rho \lambda) d\lambda. \quad (2.23)$$

For a given relative error ϵ_{rel} , we wish to determine the values of U so that the approximation (2.14) is satisfied. The following proposition gives a formula for U :

Proposition 2.3.1 *If $U \geq -\ln\left(\frac{\gamma_{01}\gamma_0}{\sqrt{19}}\epsilon_{rel}\right)$ then the approximation (2.14) is always valid for any point in D_b .*

Proof. For $(x, y, z) \in D_b$ we have:

$$\frac{r}{z} = \frac{\sqrt{x^2 + y^2 + z^2}}{z} \tag{2.24}$$

$$\leq \frac{\sqrt{(z+2)^2 + (z+2)^2 + z^2}}{z} \tag{2.25}$$

$$\leq \sqrt{3 + \frac{8}{z} + \frac{8}{z^2}} \tag{2.26}$$

$$\leq \sqrt{19} \quad (\text{since } z \geq 1). \tag{2.27}$$

Taking advantage of the boundedness (by unit) of Bessel function $J_0(\rho\lambda)$ and the expression (2.27), we can obtain an upper bound for the left hand side of (2.14):

$$\frac{\left| \int_0^\infty e^{-\lambda z} J_0(\rho\lambda) d\lambda - F_U(x, y, z) \right|}{\frac{1}{r}} = r \left| \int_U^\infty e^{-\lambda z} J_0(\rho\lambda) d\lambda \right| \tag{2.28}$$

$$= r \int_U^\infty e^{-\lambda z} d\lambda \quad (|J_0(\rho\lambda)| \leq 1) \tag{2.29}$$

$$\leq \frac{r}{z} e^{-U} \tag{2.30}$$

$$\leq \sqrt{19} \frac{\gamma_{01}\gamma_0}{\sqrt{19}} \epsilon_{rel} \tag{2.31}$$

$$\leq \gamma_{01}\gamma_0 \epsilon_{rel}. \tag{2.32}$$

■

With Proposition 2.3.1, we have a means to find U . However, the value of U may not be an optimal one. In general, too small a value of U can result in (2.16) not being satisfied at all and too large a value of U may unnecessarily require a high

order Gauss quadrature. The above estimate of U can fall into the latter case.

In order to use the available Legendre-Gauss quadratures, we change variable λ to λ' so that the interval $[0, U]$ is transformed to $[-1, 1]$: $\lambda = \frac{U}{2}(\lambda' + 1)$.

$$F_U(x, y, z) = \frac{U}{2} \int_{-1}^1 e^{-\frac{U}{2}(\lambda'+1)z} J_0\left(\frac{U}{2}(\lambda'+1)\rho\right) d\lambda' \quad (2.33)$$

With the above form, for a given point $(x, y, z) \in D_b$, there are Legendre-Gauss quadratures with weights $\omega_k^{o'}$ and nodes λ'_k that can approximate $F_U(x, y, z)$ and hence the infinite-interval integral. Our requirement is that such weights and nodes must be independent of the points (x, y, z) . More specifically, we need to find $\omega_k^{o'}$ and λ'_k such that the following conditions is always satisfied for *all* points (x, y, z) in D_b .

$$\frac{\left| F_U(x, y, z) - \frac{U}{2} \sum_{k=1}^{S_{outer}} \omega_k^{o'} e^{-\frac{U}{2}(\lambda'_k+1)z} J_0\left(\rho \frac{U}{2}(\lambda'_k+1)\right) \right|}{\frac{1}{r}} < (1 - \gamma_{o1}) \gamma_{o} \epsilon_{rel}. \quad (2.34)$$

Once $\omega_k^{o'}$ and λ'_k are available, we can obtain ω_k^o and λ_k using the following expressions:

$$\omega_k^o = \frac{U}{2} \omega_k^{o'} \quad (2.35)$$

$$\lambda_k = \frac{U}{2} (\lambda'_k + 1) \quad (2.36)$$

In practice, the condition (2.34) is not used and is replaced by (2.16), which is rewritten here for convenience:

$$\frac{\left| \int_0^\infty e^{-\lambda z} J_0(\rho \lambda) d\lambda - \sum_{k=1}^{S_{\text{outer}}} \omega_k^o e^{-\lambda_k z} J_0(\rho \lambda_k) \right|}{\frac{1}{r}} < \gamma_o \epsilon_{\text{rel}}. \quad (2.37)$$

It is easier to verify (2.37) than (2.34) because the value of the infinite-interval integral is known, which is $\frac{1}{r}$, while the value of the finite-interval integral is not readily available. The main use of U is to obtain ω_k^o and λ_k from $\omega_k^{o'}$ and λ_k' via (2.35) and (2.36). In the next section, we discuss a randomized approach for obtaining these weights and nodes.

2.4 Gauss quadratures for outer loop

In order to obtain weights and nodes ω_k^o and λ_k , respectively, so that the condition (2.37) is satisfied, we find a suitable order n_l for Legendre-Gauss quadrature $\omega_k^{o'}$ and λ_k' . We first start with an initial order $n_l = n_{lo}$. With the n_l known, we find Legendre-Gauss quadrature $\omega_k^{o'}$ and λ_k' (from, e.g., standard handbooks) and then calculate ω_k^o and λ_k using (2.35) and (2.36). We use a sequence of random points $(x_i, y_i, z_i) \in D_b$ and test whether the condition (2.37) is satisfied for all of the points. If there is a point in the sequence at which the condition is not satisfied, we find the next order n_l of the available Gauss-Legendre quadratures and repeat the whole process. Otherwise, we accept n_l as a suitable order and use the corresponding weights ω_k and nodes λ_k as Gauss quadrature for the infinite-interval integral.

Like other probabilistic approaches, there is no absolute guarantee that for a given relative error ϵ_{rel} , the inequality (2.37) is always satisfied. However, for a

sufficiently long sequence of random points. it is very unlikely that the condition fails. If so, such a break down would have occurred at some point in the sequence and would have been detected and fixed. The approach is summarized in the following algorithm. The input to the algorithm is the value U , the initial order $n_l = n_{l_0}$, the relative error ϵ_{rel} , the length N of a sequence of random points in D_b . The output of the algorithm is a suitable order n_l for Legendre-Gauss quadrature ω_k and λ_k .

2.4.1 Randomized algorithm I

- Step 1** With n_l , find the standard Legendre-Gauss weights $\omega_k^{o'}$ and nodes λ_k' for F_U in (2.33).
- Step 2** With the given U and the known $\omega_k^{o'}$ and λ_k' , we calculate ω_k^o and λ_k using (2.35) and (2.36).
- Step 3** Generate a sequence of N random points in D_b that should contain points on the boundary of D_b , i.e., points with $z = 1$, $z = b$, or $1 < z < b$ and one of the equalities in the definition of D_b (2.9).
- Step 4** For each point (x_i, y_i, z_i) in the sequence, test whether the condition (2.16) is satisfied. If the test fails, we find the next order n_l of the available Legendre-Gauss quadratures with which the condition is satisfied at the point where the test failed and repeat the whole process. If the test passes all of the points in the sequence, then we return the current value of n_l as a suitable order.

Having obtained all the weights ω_k^o and nodes λ_k , we next try to obtain Gauss quadratures for an integral representation of $J_0(\rho\lambda_k)$ for each λ_k . Again, our requirement is that such quadratures should be independent of points in D_b and at the

same time the condition (2.18) be always satisfied. We employ the same strategy as above to deal with this problem.

2.5 Gauss quadratures for inner loop

2.5.1 An integral representation for Bessel function $J_0(t)$

We use a generic variable t for the variable of Bessel function J_0 . In fact, we mean $t = \rho\lambda = \lambda\sqrt{x^2 + y^2}$. We have the following three forms of integral representation of the Bessel function $J_0(t)$ [33]:

$$J_0(t) = \frac{1}{2\pi} \int_0^{2\pi} \cos(t \cos \alpha) d\alpha \quad (2.38)$$

$$J_0(t) = \frac{1}{\pi} \int_0^\pi \cos(t \cos \alpha) d\alpha \quad (2.39)$$

$$J_0(t) = \frac{1}{\pi} \int_{-1}^1 \frac{\cos(ts)}{\sqrt{1-s^2}} ds \quad (2.40)$$

We use (2.38) so that we can express the integrand in terms of a complex exponential form of x and y . We use (2.39) and (2.40) to obtain a quadrature formula for $J_0(t)$. With regard to (2.38), thanks to the periodicity of the cosine and sine functions, we have:

$$\int_0^{2\pi} \cos(t \cos(\alpha - \beta)) d\alpha = \int_0^{2\pi} \cos(t \cos \alpha) d\alpha \quad (2.41)$$

$$\int_0^{2\pi} \sin(t \cos(\alpha - \beta)) d\alpha = 0, \quad (2.42)$$

where β is any displacement angle. If we choose β such that $x = \rho \cos \beta$ and $y = \rho \sin \beta$, then from (2.38), (2.41) and (2.42), we have:

$$J_0(\rho\lambda) = \frac{1}{2\pi} \int_0^{2\pi} \cos(\lambda(x \cos \alpha + y \sin \alpha)) d\alpha \quad (2.43)$$

$$= \frac{1}{2\pi} \int_0^{2\pi} e^{i\lambda(x \cos \alpha + y \sin \alpha)} d\alpha. \quad (2.44)$$

For α in $[\pi, 2\pi]$, the expression $(x \cos \alpha + y \sin \alpha)$ changes sign. Therefore, we can reduce the interval $[0, 2\pi]$ to $[0, \pi]$:

$$J_0(\rho\lambda) = \frac{1}{2\pi} \left(\int_0^\pi e^{i\lambda(x \cos \alpha + y \sin \alpha)} d\alpha + \text{conj} \left[\int_0^\pi e^{i\lambda(x \cos \alpha + y \sin \alpha)} d\alpha \right] \right) \quad (2.45)$$

$$= \frac{1}{2} (I_0(\rho\lambda) + \text{conj}(I_0(\rho\lambda))) \quad (2.46)$$

$$= \text{real}[I_0(\rho\lambda)], \quad (2.47)$$

where,

$$I_0(\rho\lambda) = \frac{1}{\pi} \int_0^\pi e^{i\lambda(x \cos \alpha + y \sin \alpha)} d\alpha. \quad (2.48)$$

Having an integral representation for $J_0(\rho\lambda)$, we next discuss how to obtain the Gauss quadrature.

2.5.2 Gauss quadrature

From (2.39), (2.47), and (2.48), we can use the same quadrature for both (2.39) and (2.48). The integral form (2.40) suggests Chebyshev-Gauss integration. The

weights and nodes for Chebyshev-Gauss integration with a number of nodes n_c are:

$$\omega_{c_j} = \frac{\pi}{n_c} \quad (2.49)$$

$$s_{c_j} = \frac{2j-1}{2n_c} \pi. \quad (2.50)$$

They are exactly the weights and nodes from the mid-point rule for the integral (2.39). It is for this reason we choose the mid-point rule for the integral (2.48).

More specifically, with a given n_c , we choose

$$\frac{\pi}{S_{\text{inner}}(k)} = \frac{\pi}{n_c} \quad (2.51)$$

$$\alpha_{kj} = \frac{2j-1}{2S_{\text{inner}}(k)} \pi, \quad j = 1..S_{\text{inner}}(k). \quad (2.52)$$

as weights and nodes for the integral (2.48). This provides a mechanism to choose the Gauss quadrature. However, we do not intend to approximate the integral (2.48) with these weights and nodes. Instead, we wish to find n_c so that the condition (2.18) is always satisfied for all points (x, y, z) in D_b and for each λ_k . The condition (2.18) is repeated here:

$$\left| \frac{\omega_k^o e^{-\lambda_k z} J_0(\rho \lambda_k) - \sum_{j=1}^{S_{\text{inner}}(k)} \omega_k e^{\lambda_k (-z + i(x \cos \alpha_{kj} + y \sin \alpha_{kj}))}}{\frac{1}{r}} \right| < \gamma_{I_k} \gamma_{I \in \text{rel}}. \quad (2.53)$$

where $S_{\text{inner}}(k)$ and α_{kj} are defined in (2.51) and (2.52). They both depend on k . But our requirement is that they must be independent of (x, y, z) in D_b . This is the same requirement we have in finding the Gauss quadrature for the outer loop. We

employ here the same strategy. It is summarized in the following algorithm. The input to the algorithm is the set of nodes λ_k of the outer loop, the initial number of nodes $n_c = n_{co}$, the relative error ϵ_{rel} , and the length N of the sequence of random points in D_b . The output of the algorithm is a suitable number of nodes n_c for the weights $\frac{1}{S_{inner}(k)}$ and nodes α_{kj} in (2.51) and (2.52) so that the condition (2.53) is always satisfied for all points $(x, y, z) \in D_b$.

2.5.3 Randomized algorithm II

Repeat for each λ_k :

Step 1 With n_c , calculate $S_{inner}(k)$ and nodes α_{kj} via (2.51) and (2.52).

Step 2 Generate a sequence of N random points in D_b that should contain points on the boundary of D_b , i.e., points with $z = 1$, $z = b$, or $1 < z < b$ and one of the equalities in the definition of D_b (2.9).

Step 3 For each point (x_i, y_i, z_i) in the sequence, test whether the condition (2.53) is satisfied. If the test fails, we increment n_c , repeat step 1 and test again; this process is repeated until the condition is satisfied at the point where the test failed. If the test passes all of the points in the sequence, we return the current value of n_c as a suitable number of nodes and the corresponding weights $\frac{1}{S_{inner}(k)}$ and nodes α_{kj} .

Before concluding this section, we have some remarks on the approximation (2.6). It has two forms: relative-error and absolute-error approximations:

$$\left| \frac{\frac{1}{r} - \text{real}[S(x, y, z)]}{\frac{1}{r}} \right| < \epsilon_{rel} \quad (2.54)$$

$$\left| \frac{1}{r} - \text{real}[S(x, y, z)] \right| < \epsilon_{abs} \quad (2.55)$$

For $(x, y, z) \in D_b$, we have $1 \leq r \leq b\sqrt{3}$, therefore as far as the approximation (2.6) is concerned, the above two forms are equivalent. If $\epsilon_{rel} = b\sqrt{3}\epsilon_{abs}$ then (2.55) implies (2.54). If $\epsilon_{abs} = \epsilon_{rel}$ then (2.54) implies (2.55); in this case, the expansion $S(x, y, z)$ generated for (2.54) can also be used for (2.55). However, we can directly generate $S(x, y, z)$ for (2.55) using the above randomized algorithm. This produces a smaller size S_{app} . The only change needed is to replace all the relative-error approximations (2.14)–(2.18) with the corresponding absolute-error approximations. The value of U chosen in Proposition 2.3.1 is replaced by $U \geq -\log(\gamma_{o1}\gamma_o\epsilon_{abs})$.

2.6 Numerical results

We have implemented the above method of generating Gauss quadratures for the integral representation (2.1) of Green's function $\frac{1}{r}$. The approximations are in either form (2.54) or (2.55). Gauss-Legendre quadratures of order n_l from 6 to 40 that are available in [31] are used. Given a relative error ϵ_{rel} or absolute error ϵ_{abs} , the method generates the weights ω_k , nodes λ_k , the number of nodes S_{outer} and $S_{inner}(k)$ for the outer loop and inner loop, respectively. With these values, the approximation (2.6) is expected to be satisfied for all points in D_b . In the current implementation, the values for the parameters in (2.13) and (2.14) are: $\gamma_o = \gamma_l = 0.5$ and $\gamma_{o1} = 0.5$; the values for parameters in (2.18) are: $\gamma_{l_k} = \frac{1}{S_{outer}}$. In most cases, U is chosen exactly as in Proposition 2.3.1; in some other cases, it is slightly reduced. Tables 2.1–2.2 list samples of the Gauss quadratures generated. A full list of the Gauss quadratures generated is documented in [23].

With the weights and nodes, the approximation (2.6) is tested using a sequence

of $N = 5,000,000$ random points on the boundary and in the interior of D_b . In the verification, no failure is observed. In the tests, at each point (x_i, y_i, z_i) , $i = 1..N$, we calculate $\frac{1}{r_i} = \frac{1}{\sqrt{x_i^2 + y_i^2 + z_i^2}}$, $S(x_i, y_i, z_i)$ in (2.7), the left-hand sides of (2.54) and (2.55). At the end, we obtain (a) the maximum absolute error E_{MaxAbs} and maximum relative error E_{MaxRel} of the approximation of $\frac{1}{r}$ for the test points in D_b (b) the absolute error E_{Abs} , and relative error E_{Rel} of the calculated potential, which is $\tilde{\Phi} = \sum_{i=1}^N S(x_i, y_i, z_i)$:

$$E_{MaxAbs} = \max_{1 \leq i \leq N} \left| \frac{1}{r_i} - S(x_i, y_i, z_i) \right| \quad (2.56)$$

$$E_{MaxRel} = \max_{1 \leq i \leq N} \left| \frac{\frac{1}{r_i} - S(x_i, y_i, z_i)}{\frac{1}{r_i}} \right| \quad (2.57)$$

$$E_{Abs} = \left| \Phi - \tilde{\Phi} \right| \quad (2.58)$$

$$E_{Rel} = \left| \frac{\Phi - \tilde{\Phi}}{\Phi} \right|. \quad (2.59)$$

The potential $\Phi = \sum_{i=1}^N \frac{1}{r_i}$ is in the order of 10^6 . Tables 2.3–2.5 show the results of the test for the case of $b = 2^2, 2^3$, and 2^4 . The first, second, and third columns give information about the approximation: its size and forms. The last four columns show the errors of the approximations of $\frac{1}{r}$ and the calculated potential, respectively. For the absolute-error approximations, we have $E_{MaxAbs} < \epsilon_{abs}$ (second and fourth columns) and for the relative-error approximation, we have $E_{MaxRel} < \epsilon_{rel}$ (third and fifth columns). The results demonstrate that the proposed method yields probabilistically reliable approximations.

There is a sharp contrast between the values of absolute error E_{Abs} and the relative error E_{Rel} of the calculated potential $\tilde{\Phi}$; E_{Abs} is too large due to the accu-

mulative effect and a large value of Φ : and E_{Rel} is small, much smaller than ϵ_{rel} or ϵ_{abs} . It is for this reason, the relative error is often used to estimate the error of the calculated potentials. An explanation for the fact that E_{Rel} is much smaller than the intended error ϵ_{rel} or ϵ_{abs} is that the intended errors are meant for the worst case scenario in order to account for all points in D_b . It should be emphasized that due to the probabilistic nature of the approach, for the given relative or absolute error, there is no absolute guarantee that the approximation (2.54) or (2.55) is always satisfied for all points in D_b . However, it can be expected such a breakdown is unlikely to occur. Furthermore, if a breakdown does occur, it is rare and probably has only a small effect on the overall accuracy.

$b = 2^2, \epsilon_{abs} = 10^{-1}, S_{outer} = 5, S_{app} = 18$ $\omega_k = \frac{\omega_k^o}{S_{inner}(k)}, \alpha_{kj} = \frac{2^{j-1}}{2S_{inner}(k)}\pi$		
Nodes (λ_k)	Weights (ω_k^o)	$S_{inner}(k)$
3.20497155410784	0.39835901561247	2
2.58671834468409	0.80474635024563	6
1.68135843054705	0.95650612937788	5
0.77599851641000	0.80474635024563	3
0.15774530698625	0.39835901561247	2

Table 2.1: Absolute-error approximation: sample Gauss quadratures.

$b = 2^2, \quad \epsilon_{rel} = 10^{-1}, \quad S_{outer} = 8, \quad S_{app} = 31$ $\omega_k = \frac{\omega_k^o}{S_{inner}(k)}, \quad \alpha_{kj} = \frac{2j-1}{2S_{inner}(k)}\pi$		
Nodes (λ_k)	Weights (ω_k^o)	$S_{inner}(k)$
3.24917390948062	0.28805787962091	7
2.79308840683894	0.60656951226190	7
2.08256281075830	0.78673103866423	7
1.28015405033579	0.78673103866423	5
0.56962845425515	0.60656951226190	3
0.11354295161347	0.28805787962091	2

Table 2.2: Relative-error approximation: sample Gauss quadratures.

Approximation			Errors			
S_{app}	ϵ_{abs}	ϵ_{rel}	E_{MaxAbs}	E_{MaxRel}	E_{Abs}	E_{Rel}
18	10^{-1}		3.5×10^{-2}	1.2×10^{-1}	8.1×10^3	6.1×10^{-3}
22	10^{-1}		2.4×10^{-2}	1.3×10^{-1}	3.6×10^3	2.7×10^{-3}
25	10^{-1}		2.4×10^{-2}	7.7×10^{-2}	3.6×10^2	2.8×10^{-4}
30	10^{-1}		1.6×10^{-2}	5.0×10^{-2}	1.3×10^3	1.0×10^{-3}
31		10^{-1}	3.5×10^{-2}	3.5×10^{-2}	7.2×10^2	5.5×10^{-4}
32	10^{-1}		2.3×10^{-2}	9.9×10^{-2}	7.9×10^2	6.0×10^{-4}
38		10^{-1}	1.1×10^{-2}	5.5×10^{-2}	1.3×10^3	9.6×10^{-4}
44		10^{-1}	1.1×10^{-2}	1.4×10^{-2}	5.2×10^1	4.0×10^{-5}
50		10^{-1}	1.1×10^{-2}	1.2×10^{-2}	1.3×10^2	9.8×10^{-5}
51		10^{-1}	1.1×10^{-2}	1.8×10^{-2}	8.6×10^1	6.6×10^{-5}
60	10^{-2}		3.6×10^{-3}	2.1×10^{-2}	5.1×10^1	3.9×10^{-5}
64	10^{-2}		4.1×10^{-3}	2.5×10^{-2}	1.1×10^2	8.2×10^{-5}
73	10^{-2}		1.6×10^{-3}	6.7×10^{-3}	1.1×10^2	8.3×10^{-5}
87		10^{-2}	3.5×10^{-3}	3.5×10^{-3}	2.4×10^1	1.8×10^{-5}
102		10^{-2}	1.1×10^{-3}	1.4×10^{-3}	1.2×10^1	9.1×10^{-6}
116	10^{-3}		7.2×10^{-4}	4.3×10^{-3}	6.0×10^0	4.5×10^{-6}
146	10^{-3}		1.4×10^{-4}	5.9×10^{-4}	7.2×10^{-1}	5.5×10^{-7}
179		10^{-3}	6.9×10^{-5}	2.7×10^{-4}	1.7×10^0	1.3×10^{-6}
180	10^{-3}		1.1×10^{-4}	1.3×10^{-4}	8.5×10^{-1}	6.5×10^{-7}
208	10^{-4}		6.7×10^{-5}	4.0×10^{-4}	2.2×10^0	1.6×10^{-6}
243	10^{-4}		1.8×10^{-5}	7.8×10^{-5}	5.8×10^{-1}	4.4×10^{-7}
289	10^{-4}		1.1×10^{-5}	1.5×10^{-5}	2.1×10^{-2}	1.6×10^{-8}
294		10^{-4}	3.5×10^{-5}	3.5×10^{-5}	3.2×10^{-1}	2.5×10^{-7}
322	10^{-5}		8.1×10^{-6}	4.7×10^{-5}	5.1×10^{-2}	3.9×10^{-8}
358	10^{-5}		1.1×10^{-6}	3.4×10^{-6}	5.9×10^{-3}	4.5×10^{-9}
411		10^{-5}	7.2×10^{-7}	4.4×10^{-6}	1.2×10^{-2}	9.4×10^{-9}
425		10^{-5}	1.1×10^{-6}	1.6×10^{-6}	2.6×10^{-3}	2.0×10^{-9}
661		10^{-6}	6.9×10^{-8}	6.9×10^{-8}	3.6×10^{-4}	2.8×10^{-10}
672	10^{-7}		1.1×10^{-8}	5.0×10^{-8}	5.2×10^{-4}	3.9×10^{-10}
772		10^{-7}	1.1×10^{-8}	1.8×10^{-8}	2.0×10^{-5}	1.5×10^{-11}
1052		10^{-9}	6.9×10^{-10}	6.9×10^{-10}	4.8×10^{-7}	3.6×10^{-13}
1076	10^{-9}		1.1×10^{-10}	2.9×10^{-10}	1.4×10^{-6}	1.1×10^{-12}
1220		10^{-9}	1.1×10^{-10}	3.5×10^{-10}	2.6×10^{-7}	2.0×10^{-13}

Table 2.3: Approximation and errors: $b = 2^2$

Approximation			Errors			
S_{app}	ϵ_{abs}	ϵ_{rel}	E_{MaxAbs}	E_{MaxRel}	E_{Abs}	E_{Rel}
22	10^{-1}		4.8×10^{-2}	4.0×10^{-1}	7.6×10^3	3.8×10^{-3}
26	10^{-1}		3.0×10^{-2}	3.3×10^{-1}	3.6×10^4	1.8×10^{-2}
31	10^{-1}		2.0×10^{-2}	2.5×10^{-1}	1.5×10^4	7.8×10^{-3}
32	10^{-1}		2.3×10^{-2}	1.1×10^{-1}	2.0×10^3	9.9×10^{-4}
37	10^{-1}		1.4×10^{-2}	9.4×10^{-2}	5.0×10^2	2.5×10^{-4}
44	10^{-1}		1.6×10^{-2}	5.0×10^{-2}	9.9×10^2	5.0×10^{-4}
55		10^{-1}	1.1×10^{-2}	4.1×10^{-2}	1.0×10^3	5.2×10^{-4}
66		10^{-1}	1.1×10^{-2}	1.1×10^{-2}	1.7×10^2	8.4×10^{-5}
75	10^{-2}		4.7×10^{-3}	5.4×10^{-2}	1.2×10^3	6.0×10^{-4}
103	10^{-2}		1.3×10^{-3}	5.6×10^{-3}	5.3×10^1	2.7×10^{-5}
137		10^{-2}	1.1×10^{-3}	3.7×10^{-3}	5.5×10^1	2.8×10^{-5}
176		10^{-2}	1.1×10^{-3}	1.9×10^{-3}	6.2×10^0	3.1×10^{-6}
186	10^{-3}		1.2×10^{-4}	1.5×10^{-3}	1.7×10^0	8.5×10^{-7}
295	10^{-4}		3.8×10^{-5}	4.9×10^{-4}	2.2×10^0	1.1×10^{-6}
312		10^{-3}	5.7×10^{-5}	1.6×10^{-4}	9.4×10^{-1}	4.8×10^{-7}
582		10^{-5}	1.1×10^{-6}	7.4×10^{-6}	1.6×10^{-2}	7.9×10^{-9}

Table 2.4: Approximation and errors: $b = 2^3$

Approximation			Errors			
S_{app}	ϵ_{abs}	ϵ_{rel}	E_{MaxAbs}	E_{MaxRel}	E_{Abs}	E_{Rel}
26	10^{-1}		3.3×10^{-2}	9.0×10^{-1}	1.8×10^5	5.7×10^{-2}
31	10^{-1}		2.1×10^{-2}	5.7×10^{-1}	5.5×10^4	1.8×10^{-2}
33	10^{-1}		2.3×10^{-2}	2.6×10^{-1}	9.9×10^3	3.1×10^{-3}
38	10^{-1}		1.4×10^{-2}	2.4×10^{-1}	2.8×10^4	8.8×10^{-3}
44	10^{-1}		1.6×10^{-2}	1.8×10^{-1}	9.2×10^3	2.9×10^{-3}
91		10^{-1}	1.1×10^{-2}	1.9×10^{-2}	3.4×10^2	1.1×10^{-4}
104	10^{-2}		3.6×10^{-3}	9.9×10^{-2}	2.9×10^3	9.2×10^{-4}
129	10^{-2}		1.4×10^{-3}	1.8×10^{-2}	1.7×10^2	5.4×10^{-5}
214		10^{-2}	1.1×10^{-3}	1.8×10^{-3}	2.4×10^1	7.7×10^{-6}
227	10^{-3}		4.0×10^{-4}	1.1×10^{-2}	1.1×10^2	3.6×10^{-5}
395		10^{-3}	1.1×10^{-4}	1.3×10^{-4}	6.0×10^{-1}	1.9×10^{-7}
402	10^{-4}		4.3×10^{-5}	1.2×10^{-3}	7.5×10^0	2.4×10^{-6}

Table 2.5: Approximation and errors: $b = 2^4$

Chapter 3

Exponential-expansion Method

Based on the exponential expansion generated for the Green's function, which is discussed in Chapter 2, we develop the EE method [24, 29, 30]. The method belongs to a more general class of hierarchical algorithms (tree codes), in which the domain of computation is partitioned into a tree-like hierarchy of cubes and each of these cubes can be considered as a cluster of all its particles. The main difference with other treecodes is that the underlying approximation of the Green's function is the exponential expansion as opposed to the multipole expansion. As it will be demonstrated later in this chapter, the EE method provides a computationally efficient alternative to the MP methods for evaluation of the three-dimensional potential and its gradient, opening new avenues for accurate large-scale simulation of particle systems and the related applications.

In §3.1, we introduce the notion of *six directions* to extend the domain of approximation D_b (2.9). We also discuss the relationship between the exponential expansion and the computed potential in terms of computational accuracy and effi-

ciency. The translations with EE are presented in § 3.2. Here, we establish the main theorem underlying all the translations that constitute the computational procedure for the EE method. Section § 3.3 is devoted to description of the computational elements associated with the translations (clustering, shifting, and declustering). We particularly give full analysis of the far-field translations (shifting) required at a given cube and illustrate how exponential expansion is used in the evaluation of the potential and its gradient [40]. The salient features of the EE method are articulated in § 3.4 together with qualitative comparisons with the MP methods. We also show a sample pseudo code suggesting how parallel processing can be employed for the EE method. In § 3.5, a computational framework for realization of the EE method is proposed with three EE-based $O(N)$ algorithms along with their cost analysis. Each of these algorithms deals with a different type of domain of approximation. Section § 3.6 presents numerical results, which demonstrate a wide range of accuracy in the calculated potential and a running time of the order $O(N)$.

3.1 Mathematical background

In this section, we describe the underlying mathematics of the proposed method, starting with the exponential expansion described in the previous chapter. With the notion of six directions and by scaling, we extend the domain of approximation D_b to virtually cover the whole three-dimensional space R^3 . We study the relationship between the intended error of the approximation (expansion) and the actual error of the calculated potential. We also identify an important parameter associated with the approximation that affects the overall computational performance of the proposed method.

3.1.1 Six directions

The expressions (2.1) and (2.6)–(2.9) are biased towards the $+z$ -direction while there exists no such bias in $\frac{1}{r}$. To reflect this bias towards the $+z$ -direction, we add the superscript Up to the expressions (2.6)–(2.9). They are now rewritten as:

$$\frac{1}{r} \approx \text{real} [S^{Up}(x, y, z)] . \quad (3.1)$$

$$S^{Up}(x, y, z) \triangleq \sum_{k=1}^{S_{\text{outer}}} \omega_k \sum_{j=1}^{S_{\text{inner}}(k)} E^{Up}[k, j](x, y, z). \quad (3.2)$$

$$E^{Up}[k, j](x, y, z) \triangleq e^{g^{Up}[k, j](x, y, z)} \quad (3.3)$$

$$g^{Up}[k, j](x, y, z) \triangleq \lambda_k(-z + i(x \cos \alpha_{kj} + y \sin \alpha_{kj})) \quad (3.4)$$

$$D_b^{Up} \triangleq \{(x, y, z) \mid a = 1 \leq z \leq b, \max(-b, -z - 2) \leq x, y \leq \min(b, z + 2)\} \quad (3.5)$$

The function $E^{Up}[k, j](x, y, z)$ is called *exponential distance* associated with direction Up and indices $[k, j]$ of a vector (x, y, z) .

By switching the bias towards $-z$ -, $+x$ -, $-x$ -, $+y$ -, and $-y$ -directions, we obtain five more expressions similar to (3.1)–(3.5) with the same α_{kj} , λ_k , ω_k , $S_{\text{inner}}(k)$, and S_{outer} . We refer to $+z$ -, $-z$ -, $+x$ -, $-x$ -, $+y$ -, and $-y$ - directions as Up , $Down$, $East$, $West$, $North$, and $South$. The set of the six directions is denoted by DirSet . Corresponding to (3.4), we have the following definitions for the functions g 's:

Definition 3.1.1

$$\begin{aligned}
g^{Up}[k, j](x, y, z) &\triangleq \lambda_k(-z + i(x \cos \alpha_{kj} + y \sin \alpha_{kj})) \\
g^{Down}[k, j](x, y, z) &\triangleq \lambda_k(z + i(x \cos \alpha_{kj} + y \sin \alpha_{kj})) \\
g^{East}[k, j](x, y, z) &\triangleq \lambda_k(-x + i(z \cos \alpha_{kj} + y \sin \alpha_{kj})) \\
g^{West}[k, j](x, y, z) &\triangleq \lambda_k(x + i(z \cos \alpha_{kj} + y \sin \alpha_{kj})) \\
g^{North}[k, j](x, y, z) &\triangleq \lambda_k(-y + i(x \cos \alpha_{kj} + z \sin \alpha_{kj})) \\
g^{South}[k, j](x, y, z) &\triangleq \lambda_k(y + i(x \cos \alpha_{kj} + z \sin \alpha_{kj})).
\end{aligned}$$

In correspondence to (3.2) and (3.3), we have the following definitions for the exponential distances and the sums:

Definition 3.1.2 For each direction $dir \in \text{DirSet}$.

$$E^{dir}[k, j](x, y, z) \triangleq e^{g^{dir}[k, j](x, y, z)} \quad (3.6)$$

$$S^{dir}(x, y, z) \triangleq \sum_{k=1}^{S_{outer}} \omega_k \sum_{j=1}^{S_{inner}(k)} E^{dir}[k, j](x, y, z). \quad (3.7)$$

Corresponding to (3.5), we have the following definitions for the sets D_b 's:

Definition 3.1.3

$$\begin{aligned}
D_b^{Up} &= \{(x, y, z) \mid a = 1 \leq z \leq b, \max(-b, -z - 2) \leq x, y \leq \min(b, z + 2)\} \\
D_b^{Down} &= \{(x, y, z) \mid a = 1 \leq -z \leq b, \max(-b, z - 2) \leq x, y \leq \min(b, -z + 2)\} \\
D_b^{East} &= \{(x, y, z) \mid a = 1 \leq x \leq b, \max(-b, -x - 2) \leq z, y \leq \min(b, x + 2)\} \\
D_b^{West} &= \{(x, y, z) \mid a = 1 \leq -x \leq b, \max(-b, x - 2) \leq z, y \leq \min(b, -x + 2)\} \\
D_b^{North} &= \{(x, y, z) \mid a = 1 \leq y \leq b, \max(-b, -y - 2) \leq x, z \leq \min(b, y + 2)\} \\
D_b^{South} &= \{(x, y, z) \mid a = 1 \leq -y \leq b, \max(-b, y - 2) \leq x, z \leq \min(b, -y + 2)\}.
\end{aligned}$$

With the above definitions, we can express (3.1) and (3.5) in a more general form, in which the bias towards +z-direction is eliminated.

Proposition 3.1.1 *The expressions (3.1) and (3.5) will imply that for each direction $dir \in \text{DirSet}$ and each point $(x, y, z) \in D_b^{dir}$.*

$$\frac{1}{r} \approx \text{real}[S^{dir}[k, j](x, y, z)]. \quad (3.8)$$

Proof. Permute the variables $x, y,$ and z in (2.1) and (2.6)–(2.9). ■

If no specific direction is given, it can be omitted. If no specific indices $[k, j]$ are given, they can also be omitted. The approximation (3.8) is rewritten as

$$\frac{1}{r} \approx \text{real}[S(x, y, z)], \quad \forall (x, y, z) \in D_b. \quad (3.9)$$

Note that in order for (3.9) to be valid, the directions associated with $S(x, y, z)$ and D_b must be the same. Furthermore, approximation (3.9) only holds for $(x, y, z) \in D_b$. If (x, y, z) is not in any of the D_b 's, then some form of scaling is needed.

3.1.2 Approximation and scaling

If we have a scaling factor $s > 0$ such that $(sx, sy, sz) \in D_b$, then we can obtain approximations that are similar to (2.54) and (2.55).

Proposition 3.1.2 *Let $s > 0$ be a scaling factor. The approximation (2.54) or (2.55) will imply the following respective approximations. $\forall (s \times x, s \times y, s \times z) \in D_b$,*

$$\left| \frac{\frac{1}{r} - \text{real}[s \times S(s \times x, s \times y, s \times z)]}{\frac{1}{r}} \right| < \epsilon_{rel}, \quad \text{or} \quad (3.10)$$

$$\left| \frac{1}{r} - \text{real}[s \times S(s \times x, s \times y, s \times z)] \right| < s \times \epsilon_{abs}. \quad (3.11)$$

Proof. Let $r_s = \sqrt{(sx)^2 + (sy)^2 + (sz)^2} = s \times \sqrt{x^2 + y^2 + z^2} = s \times r$. Because $(sx, sy, sz) \in D_b$, we can apply the approximation (2.54) and (2.55) for $\frac{1}{r_s}$ to obtain (3.10) and (3.11), respectively. ■

From the above proposition, we have two observations: (a) Approximations with absolute errors are sensitive to scaling while those with relative errors are not. It is for this reason, the relative-error approximation (2.54) is preferred over the absolute-error approximation (2.55). (b) For the relative-error approximation (2.54), if we replace the weights ω_k with $s \times \omega_k$, then the new weights can also be used in the new coordinate system in which all coordinates are scaled by s . With this flexibility, the relative-error approximation (2.54) can be applied, with corresponding scaling of the weights ω_k 's, to any set $s^{-1}D_b$ defined as

$$s^{-1}D_b \triangleq \{(x, y, z) \mid (sx, sy, sz) \in D_b\}. \quad (3.12)$$

The union of all these sets with the associated six directions covers the whole three-dimensional space R^3 , except the point $(0, 0, 0)$.

Having established a general form for the approximation to Green's function $\frac{1}{r}$, we now discuss the accuracy and efficiency of the approximation. For the latter, we identify a parameter associated with the approximation that can be used to model the overall performance of the proposed method. For the former, we describe relationships between the intended error of the approximation and the actual error of the calculated potential.

3.1.3 Approximation: accuracy and efficiency

Absolute error vs. relative error

Suppose that Q_i , $i = 1..N$, are source points of charges q_i and P is a target point such that vectors $\mathbf{Q}_i P(x - x_i, y - y_i, z - z_i)$ are all in D_b . Let $\Psi = \sum_{i=1}^N \frac{q_i}{r_i}$, where $r_i = \|\mathbf{Q}_i P\|$, and $\tilde{\Psi} = \sum_{i=1}^N \text{real}[q_i S(x - x_i, y - y_i, z - z_i)]$, which is an approximation of Ψ using exponential expansion. Let $\Psi = \Psi^+ - \Psi^-$, where Ψ^+ and Ψ^- are the contribution to Ψ by positive charges and absolute value of negative charges, respectively. Similarly, let $\tilde{\Psi} = \tilde{\Psi}^+ - \tilde{\Psi}^-$. The error in the calculated potential $\tilde{\Psi}$ can be described in terms of the relative error ϵ_{rel} (2.54) or absolute error ϵ_{abs} (2.55) as follows:

Proposition 3.1.3

$$(i) \quad |\Psi - \tilde{\Psi}| < (\sum_{i=1}^N |q_i|) \epsilon_{abs}.$$

(ii) *If all the charges are non-negative, the relative error in approximating Ψ is*

$$\epsilon_{rel}.$$

(iii) The relative errors in approximating Ψ^+ and Ψ^- by $\tilde{\Psi}^+$ and $\tilde{\Psi}^-$, respectively, are both ϵ_{rel} .

(iv) For a scaling factor $s > 0$ such that $s\mathbf{Q}_i\mathbf{P}$, $i = 1..N$, are all in D_b , with reference to the case of no scaling, the calculated potential has the same relative error, while its absolute error is multiplied by s .

Proof. For (i), we use triangle inequality; for (ii), we apply (2.54) for each $\frac{1}{r_i}$:

$$\left| \frac{q_i}{r_i} - \text{real}[q_i S(x - x_i, y - y_i, z - z_i)] \right| < \frac{q_i}{r_i} \epsilon_{rel} \quad (3.13)$$

$$|\Psi - \tilde{\Psi}| < \left(\sum_{i=1}^N \frac{q_i}{r_i} \right) \epsilon_{rel} \quad (3.14)$$

$$< \Psi \epsilon_{rel}. \quad (3.15)$$

For (iii), we apply (ii) for each of $\tilde{\Psi}^+$ and $\tilde{\Psi}^-$. For (iv), we use (3.10) and (3.11) in Proposition 3.1.2. ■

From the above result, it is clear that the relative-error approximation (2.54) provides more reliable information about the error (due to the approximation of the potential using exponential expansion) compared to the absolute-error approximation (2.55).

In addition to accuracy, the cost to evaluate the sum $S(x, y, z)$ is another important factor that determines the overall computational efficiency of the proposed method. The question of how such cost is modeled and reduced will be discussed next.

Efficiency

The computational cost of evaluating each exponential distance E is the same. Therefore, the total cost to evaluate $S(x, y, z)$ can be modelled as the number of terms in the double sum (3.7) (see (2.10)). As it will be shown later, for a given error ϵ , the proposed algorithms are of the order $O(S_{app}(\epsilon, b)N)$. Therefore, reducing S_{app} without compromising accuracy is a way to improve the computational performance.

Note that in (2.54) and (2.55), $\frac{1}{r}$ is a real number and the sum $S(x, y, z)$ as defined in (3.7) is a complex number. However, we do not insist that the imaginary part of $S(x, y, z)$ be zero. A justification for this is as follows. We can rewrite (3.9) as

$$\frac{1}{r} \approx S_1(x, y, z), \quad \forall (x, y, z) \in D_b. \quad (3.16)$$

Here $S_1(x, y, z) = \frac{1}{2}(S(x, y, z) + \text{conj}(S(x, y, z)))$, where $\text{conj}(S(x, y, z))$ denotes the conjugate of $S(x, y, z)$. The imaginary part of $S_1(x, y, z)$ is zero and the real part of $S_1(x, y, z)$ is the same as $S(x, y, z)$, which approximates $\frac{1}{r}$. Hence (3.16) is valid. If we extend the symbol $E[k, j](x, y, z)$ to include negative indices such that $E[k, -j](x, y, z) = \text{conj}(E[k, j](x, y, z))$, then we can express S_1 as

$$S_1(x, y, z) = \sum_{k=1}^{S_{outer}} \frac{\omega_k}{2} \sum_{j=-S_{inner}(k), j \neq 0}^{S_{inner}(k)} E[k, j](x, y, z). \quad (3.17)$$

$S_1(x, y, z)$ has the same form as $S(x, y, z)$; they are both sums of the exponential distances $E[k, j](x, y, z)$. Suppose that we process the $E[k, j](x, y, z)$'s individually

and add them up to get the approximated value for $\frac{1}{r}$. If we use S_1 then no extra work is needed. If we use $S(x, y, z)$, then we need to take the real part of $S(x, y, z)$ to obtain $\frac{1}{r}$, which costs little. The results in both cases are the same. However, the cost of evaluating $S(x, y, z)$ is only half the cost of evaluating $S_1(x, y, z)$. The reason is that the number of terms in $S(x, y, z)$ is only half that of $S_1(x, y, z)$. Driven by the need for approximations with sizes as small as possible (see (2.10)), we prefer $S(x, y, z)$ to $S_1(x, y, z)$. For convenience and with a slight change of notation, we restate (3.9) as

$$\frac{1}{r} \approx S(x, y, z), \quad \forall (x, y, z) \in D_b. \quad (3.18)$$

We have discussed the approximation of Green's function $\frac{1}{r}$ using one of its exponential integral representations as an intermediate step. Starting with (2.1) and using Gauss quadratures, we obtain (3.8). In the following sections, we describe how the approximation is used for efficient and rapid evaluation of the potential field in three dimensions. As it was mentioned in the introduction, the basis for state-of-the-art $O(N \ln N)$ and $O(N)$ algorithms for evaluation of the potential field lies in the capability of replacing a cluster of particles by a pseudo-particle or centre through translations. The theory of spherical harmonics provides the mathematical background to perform such translations. Current $O(N \ln N)$ or $O(N)$ methods for evaluating the potential field are, in one form or another, based on the theory of spherical harmonics. However, the exponential expansion employed here provides a much more flexible mechanism for efficient translations.

3.2 Translations

In this section, we discuss how translations are described using exponential expansion, along with an illustrative example. We discuss important properties of exponential distances $E[k, j](x, y, z)$. We introduce the notion of boundedness and direction of a vector, which is used to describe the condition for translations. We establish the main theorem underlying all the translations that constitute the computational procedure of the proposed method.

3.2.1 Properties of exponential distances

The important properties of the exponential distance $E[k, j]$ are summarized in the following proposition.

Proposition 3.2.1

$$(i) \quad E^{dir}[k, j](-x, -y, -z) = \text{conj}(E^{-dir}[k, j](x, y, z)).$$

$$(ii) \quad E^{dir}[k, j](x_1 + x_2, y_1 + y_2, z_1 + z_2) = E^{dir}[k, j](x_1, y_1, z_1) \times E^{dir}[k, j](x_2, y_2, z_2).$$

$$(iii) \quad E^{dir}[k, j](s \times x, s \times y, s \times z) = \{E^{dir}[k, j](x, y, z)\}^s.$$

$$(iv) \quad E^{dir}[k, j](-x, -y, -z) = \frac{1}{E^{dir}[k, j](x, y, z)}.$$

Here, $-dir$ denotes the opposite direction of dir (e.g., $-Up \equiv Down$).

Proof. The proof is straightforward and based on Definitions 3.1.1 and 3.1.2. ■

In Proposition 3.2.1, (i) and (iv) suggest that $E^{dir}[k, j](-x, -y, -z)$ can be indirectly obtained by taking the conjugate of $E^{-dir}[k, j](x, y, z)$ or the multiplicative inverse of $E^{dir}[k, j](x, y, z)$ without resorting to an expensive direct calculation; (ii)

can be considered as the *addition theorem* for exponential distances; (iii) provides the means to calculate $E[k, j]$ with different scaling factors; and (iv) is a special case of (iii) in which $s = -1$.

In order to describe translations in a more compact form, we introduce additional terminology.

3.2.2 Vector: boundedness and direction

Definition 3.2.1 For a given closed interval $[c, d]$ with endpoints c, d as positive integers, a vector $V(x, y, z)$ is said to be bounded in $[c, d]$ if

$$c \leq \max(|x|, |y|, |z|) \leq d$$

The geometric motivation behind the above definition is that if vector V connecting the centres of two cubes of unit length is bounded in $[c, d]$, then these two cubes are separated by at least $(c - 1)$ and at most $(d - 1)$ cubes. We shall work with the case of $c = a + 1$ and $d = b - 1$, which is in association with the approximation (3.8).

Proposition 3.1.1 suggests that the approximation (3.8) can be applied in any direction provided that the point or vector¹ (x, y, z) belongs to the corresponding set D_b . Therefore, in order to use the approximation, we need (a) to determine to which set D_b the point belongs and (b) to have a self-consistent criterion to choose a set D_b if the point belongs to more than one set. Through the notion of the direction of a vector, we indeed have an efficient method to satisfy both of the above requirements.

¹As far as coordinates are concerned, the terms point and vector are used interchangeably.

Definition 3.2.2 *The direction of a vector $V(x, y, z)$ is defined as*

$$\text{Dir}(V) = \begin{cases} \text{Up} & \text{if } z = \max(|x|, |y|, |z|) \\ \text{Down} & \text{else if } z = -\max(|x|, |y|, |z|) \\ \text{East} & \text{else if } x = \max(|x|, |y|, |z|) \\ \text{West} & \text{else if } x = -\max(|x|, |y|, |z|) \\ \text{North} & \text{else if } y = \max(|x|, |y|, |z|) \\ \text{South} & \text{else if } y = -\max(|x|, |y|, |z|). \end{cases}$$

With the above definition, the direction of a vector is uniquely determined. There appears to be mild bias toward some directions; this, however, occurs only when at least two of the three coordinates have the maximum absolute value.

3.2.3 Theorem for translations

The mathematical background for translations in the EE method is the following theorem:

Theorem 3.2.1 *Let Cube_1 and Cube_2 denote cubes of unit length centred at C_1 and C_2 . Let Q and P be points in Cube_1 and Cube_2 , respectively. The necessary and sufficient condition for vector C_1C_2 to be bounded in $[a+1, b-1]$ is that there exists a direction $\text{Dir} \in \text{DirSet}$ such that any vector QP is in D_b^{Dir} . Furthermore, under this condition and the approximation (3.1),*

$$E^{\text{Dir}}[k, j](QP) = E^{\text{Dir}}[k, j](QC_1) \times E^{\text{Dir}}[k, j](C_1C_2) \times E^{\text{Dir}}[k, j](C_2P) \quad (3.19)$$

$$\frac{1}{\|QP\|} \approx \text{real}[S^{\text{Dir}}(QP)]. \quad (3.20)$$

Proof. First we observe that $|z_{QC_1}| \leq \frac{1}{2}$ and $|z_{C_2P}| \leq \frac{1}{2}$. There are similar inequalities for the x - and y - coordinates. Hence,

$$z_{C_1C_2} - 1 \leq z_{QP} \leq z_{C_1C_2} + 1 \quad (3.21)$$

$$y_{C_1C_2} - 1 \leq y_{QP} \leq y_{C_1C_2} + 1 \quad (3.22)$$

$$x_{C_1C_2} - 1 \leq x_{QP} \leq x_{C_1C_2} + 1 \quad (3.23)$$

$$z_{QP} - 1 \leq z_{C_1C_2} \leq z_{QP} + 1. \quad (3.24)$$

1. Necessary condition: suppose that vector C_1C_2 is bounded in $[a + 1, b - 1]$. Let Dir denote the direction of C_1C_2 , as defined in Definition 3.2.2. We will be proving that any vector QP is in D_b^{Dir} . For simplicity, we assume that $Dir = U_P$, that is

$$a + 1 \leq z_{C_1C_2} \leq b - 1 \quad (3.25)$$

$$-z_{C_1C_2} \leq x_{C_1C_2}, y_{C_1C_2} \leq z_{C_1C_2}. \quad (3.26)$$

With these assumptions, we need to show that $a \leq z_{QP} \leq b$ and $\max(-b, -z_{QP} - 2) \leq x_{QP}, y_{QP} \leq \min(b, z_{QP} + 2)$.

For z_{QP} , from (3.21) and (3.25), we have

$$a \leq z_{QP} \leq b. \quad (3.27)$$

For y_{QP} , from (3.22), (3.26), (3.24), and (3.25), we have

$$-z_{QP} - 2 \leq y_{QP} \leq z_{QP} + 2 \quad (3.28)$$

$$-b \leq y_{QP} \leq b. \quad (3.29)$$

Similarly for x_{QP} , from (3.23), (3.26), (3.24), and (3.25), we have

$$-z_{QP} - 2 \leq x_{QP} \leq z_{QP} + 2 \quad (3.30)$$

$$-b \leq x_{QP} \leq b. \quad (3.31)$$

Expressions (3.27)–(3.31) show that QP is in D_b^{Dir} .

2. Sufficient condition: suppose that there exists a direction Dir such that any vector QP (with Q in Cube_1 and P in Cube_2) is in D_b^{Dir} . We will be proving that vector C_1C_2 is bounded in $[a+1, b-1]$. Again, without loss of generality, we assume that $\text{Dir} = Up$, that is

$$a \leq z_{QP} \leq b \quad (3.32)$$

$$-z_{QP} - 2 \leq x_{QP}, y_{QP} \leq z_{QP} + 2. \quad (3.33)$$

We need to show $(a+1) \leq z_{C_1C_2} \leq (b-1)$ and $|x_{C_1C_2}|, |y_{C_1C_2}| \leq z_{C_1C_2}$.

For the former, by an appropriate choice of Q and P , we obtain from (3.21), $z_{QP} = z_{C_1C_2} + 1$. From the right hand side of (3.32), it follows that $z_{C_1C_2} \leq (b-1)$. Similarly, with another appropriate choice of Q and P , we can obtain from (3.21), $z_{QP} = z_{C_1C_2} - 1$. From the left hand side of (3.32), it follows

that $(a + 1) \leq z_{C_1 C_2}$. Thus,

$$(a + 1) \leq z_{C_1 C_2} \leq (b - 1). \quad (3.34)$$

For the latter, we use the same procedure but with (3.21), (3.22), and (3.33).

A choice of Q and P such that $z_{QP} = z_{C_1 C_2} - 1$ and $y_{QP} = y_{C_1 C_2} + 1$ (or $y_{QP} = y_{C_1 C_2} - 1$) will lead to $y_{C_1 C_2} \leq z_{C_1 C_2}$ (or $-z_{C_1 C_2} \leq y_{C_1 C_2}$). The treatment of $x_{C_1 C_2}$ is analogous. Thus,

$$-z_{C_1 C_2} \leq y_{C_1 C_2} \leq z_{C_1 C_2} \quad (3.35)$$

$$-z_{C_1 C_2} \leq x_{C_1 C_2} \leq z_{C_1 C_2}. \quad (3.36)$$

Expressions (3.34)–(3.36) show that $C_1 C_2$ is bounded in $[a + 1, b - 1]$.

The expression (3.19) follows from Proposition 3.2.1(ii) and from the fact that $QP = QC_1 + C_1 C_2 + C_2 P$. Finally, with the direction Dir identified, the expression (3.20) comes from (3.8). ■

Note that in the sufficient condition, if we order the directions as *Up*, *Down*, *East*, *West*, *North*, and *South*, and if we choose Dir as the first direction such that any QP is in D_b^{Dir} , then it follows from the above proof that $Dir \equiv Dir(C_1 C_2)$. This is consistent with the necessary condition.

Theorem 3.2.1 is the cornerstone of the EE method. In what follows, we discuss some of its implications. Here, we assume that the condition in the theorem is satisfied; that is $C_1 C_2$ is bounded in $[a + 1, b - 1]$.

3.2.4 Remarks

- Choice of direction: with $\text{Dir}(\mathbf{C}_1\mathbf{C}_2)$ as the common direction for all pairs of points (Q, P) , we have a self-consistent criterion and an efficient mechanism to choose the direction to use in approximation (3.8). The choice $\text{Dir}(\mathbf{C}_1\mathbf{C}_2)$ leads to (3.19) and (3.20). For each \mathbf{QP} , there is no need to test to which D_b it belongs and which D_b should be used in (3.8). From Theorem 3.2.1, we know that it is the set $D_b^{\text{Dir}(\mathbf{C}_1\mathbf{C}_2)}$.
- Domain of approximation: for given values of b and ϵ , the approximation (3.8) can only be used for $\frac{1}{\|\mathbf{QP}\|}$ when the vector connecting their centres $\mathbf{C}_1\mathbf{C}_2$ is bounded in $[a + 1, b - 1]$. Otherwise, the approximation may not hold and some form of scaling (see Proposition 3.1.2) is needed to transform $\mathbf{C}_1\mathbf{C}_2$ to a new vector that is bounded in $[a + 1, b - 1]$. All of the three algorithms to be presented later make use of this observation.
- Geometrical meaning of D_b : now it is clear that D_b is exactly the set of all vectors \mathbf{QP} with Q in Cube_1 and P in Cube_2 . In other words, D_b contains no redundant points and none of \mathbf{QP} is missing. Redundant points may unnecessarily increase the size of the approximation S_{app} ; an example of this case is when we choose D_b^{Up} as the rectangle box $[-b, b] \times [-b, b] \times [a, b]$. At the missing points, approximation (3.1) may be invalid. Therefore, D_b is the optimal domain of approximation.
- The quantity b and the number of levels in the tree-like hierarchy: for a tree-like hierarchy of n levels, the maximum number of cubes along one of the x -, y -, and z -directions is 2^n . Therefore, if we choose the length of the smallest

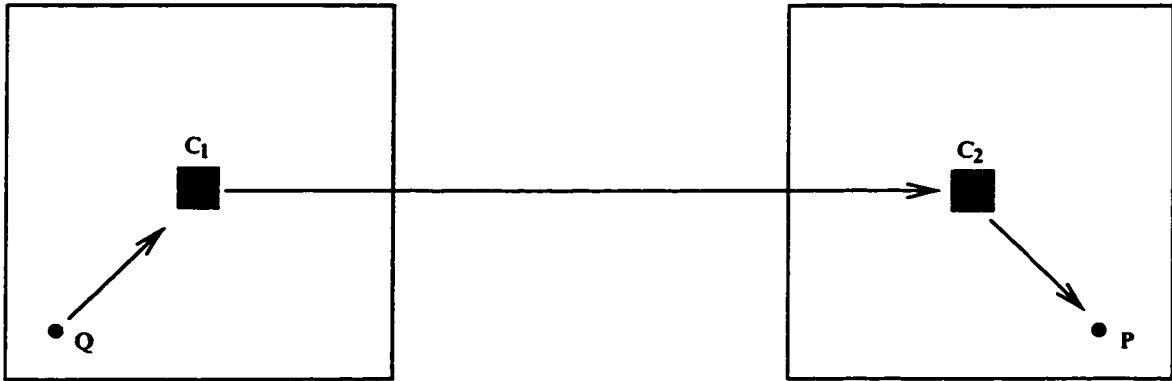


Figure 3.1: A translation from Q to P through centres C_1 and C_2 : $E(QP) = E(QC_1) \times E(C_1C_2) \times E(C_2P)$.

cube as the unit length and if $b \geq 2^n$, then any vector connecting two cube centres that are at least $a = 1$ cube apart will be bounded in $[a + 1, b - 1]$. Hence Theorem 3.2.1 can be applied. This forms the basis for Algorithm I. to be presented later. It turns out that even with b as small as 2^2 , we can, in light of Theorem 3.2.1 and with appropriate scaling, perform all the required translations. This will be presented in Algorithm II.

- Translations as an indirect evaluation of $E[k, j](QP)$: expression (3.19) suggests that we can evaluate $E[k, j](QP)$ indirectly via $E[k, j](QC_1)$, $E[k, j](C_1C_2)$, and $E[k, j](C_2P)$. The indirect evaluation can be depicted in terms of a sequence of three translations: (a) from Q to C_1 , (b) from C_1 to C_2 , and (c) from C_2 to P . This is illustrated in Fig. 3.1.

With Theorem 3.2.1, we now illustrate how the translations work in the EE method. The translations are detailed in the next section.

3.2.5 Illustration

Let Cube_1 and Cube_2 denote cubes of unit length centred at C_1 and C_2 . Assume that the vector C_1C_2 is bounded in $[a + 1, b - 1]$; that is, we can invoke Theorem 3.2.1. Let $\text{Dir} = \text{Dir}(C_1C_2)$. Suppose that Cube_1 contains N_1 source points Q whose charges are denoted by $q(Q)$ and that Cube_2 contains N_2 target points P at which we wish to find the potential due to the charges in Cube_1 . For each target point P in Cube_2 , the potential at P is

$$\Phi(P) = \sum_{Q \in \text{Cube}_1} \frac{q(Q)}{\|QP\|}. \quad (3.37)$$

From (3.20), the potential $\Phi(P)$ can be approximated as

$$\Phi(P) \approx \sum_{Q \in \text{Cube}_1} q(Q) S^{\text{Dir}}(QP) \quad (3.38)$$

$$\approx \sum_{Q \in \text{Cube}_1} q(Q) \sum_{k=1}^{S_{\text{outer}}} \omega_k \sum_{j=1}^{S_{\text{inner}}(k)} E^{\text{Dir}}[k, j](QP) \quad (3.39)$$

$$\approx \sum_{k=1}^{S_{\text{outer}}} \omega_k \sum_{j=1}^{S_{\text{inner}}(k)} \sum_{Q \in \text{Cube}_1} q(Q) E^{\text{Dir}}[k, j](QP). \quad (3.40)$$

We now describe the procedure for efficient evaluation of the innermost sum using translations, i.e. using (3.19) for indirect calculation of $E^{\text{Dir}}[k, j](QP)$ for each P .

$$\Phi(P)^{\text{Dir}}[k, j] \triangleq \sum_{Q \in \text{Cube}_1} q(Q) E^{\text{Dir}}[k, j](QP) \quad (3.41)$$

$$= \sum_{Q \in \text{Cube}_1} q(Q) E[k, j](QC_1) E[k, j](C_1C_2) E[k, j](C_2P) \quad (3.42)$$

$$= \left\{ \left(\sum_{Q \in \text{Cube}_1} q(Q) E[k, j](QC_1) \right) E[k, j](C_1C_2) \right\} E[k, j](C_2P) \quad (3.43)$$

Note that in (3.42) and (3.43) the superscript *Dir* is omitted for ease of presentation. The expression (3.43) suggests the following procedure for translating the effect of Q 's to each P . This is illustrated in Fig. 3.2.

Step 1. Clustering all source points Q 's to centre C_1 :

$$E(C_1)^{\text{Dir}}[k, j] = \sum_{Q \in \text{Cube}_1} q(Q) E^{\text{Dir}}[k, j](QC_1). \quad (3.44)$$

Step 2. Shifting centre C_1 to centre C_2 :

$$E(C_2)^{\text{Dir}}[k, j] = E(C_1)^{\text{Dir}}[k, j] \times E^{\text{Dir}}[k, j](C_1C_2). \quad (3.45)$$

Step 3. Redistributing from centre C_2 to each target point P :

$$E(P)^{\text{Dir}}[k, j] = E(C_2)^{\text{Dir}}[k, j] \times E^{\text{Dir}}[k, j](C_2P). \quad (3.46)$$

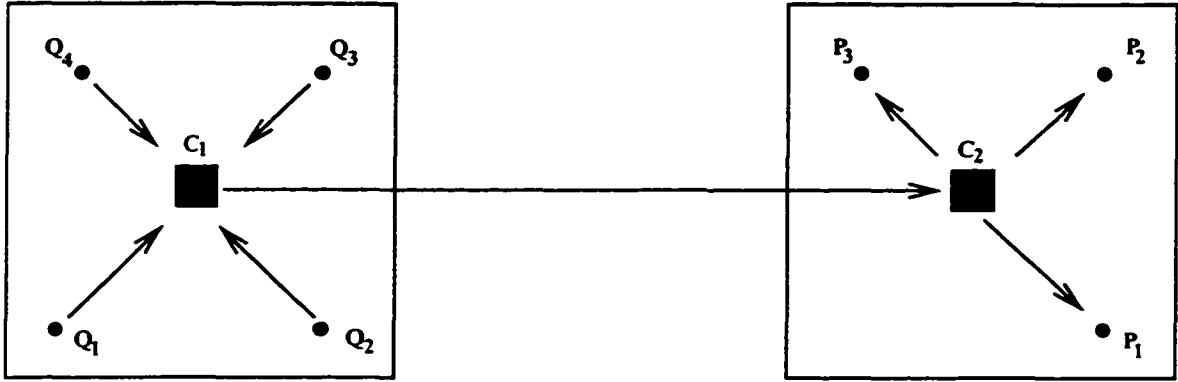


Figure 3.2: Translations for efficient approximation of the potential field at $N_2 = 3$ target points in Cube_2 due to $N_1 = 4$ source points in Cube_1 .

The result $E(P)^{\text{Dir}}[k, j]$ is the innermost sum $\Phi^{\text{Dir}}[k, j](P)$ defined in (3.41). The cost of evaluating the innermost sum is only $O(N_1 + 1 + N_2)$ and therefore, the total cost for approximating $\Phi(P)$ using (3.40) is $O(S_{app}(N_1 + 1 + N_2))$, as opposed to $O(N_1 N_2)$ in direct pairwise particle-to-particle calculation. This suggests that with exponential expansion, we can have $O(N)$ algorithms for evaluation of the potential field. In addition, the translations here are much simpler and more efficient than those based on spherical harmonics. This is further elucidated in the next section.

Having presented the basis, we now describe the elements of the EE method. They are source points (Q) with associated charges (q), source centres (S), target centres (T), and target points (P) with associated potentials Φ and gradients of potentials $\nabla\Phi$. The input is source points, their charges, and target points; the output is the potential and/or the gradient of the potential at the target points. Source points are clustered to form source centres. Source centres are clustered to form new source centres or are shifted to target centres. Target centres are

redistributed either to form new target centres or to target points for obtaining the far-field potential and its gradient. The clustering, shifting, and redistributing translations all have the same structure. In clustering and redistributing, all six directions are involved; in shifting source centres to target centres only one direction is involved. The direction used in the shifting is determined by Theorem 3.2.1.

3.3 Computational elements

In this section, we describe the elements of the proposed method in the context of translations. We assume that the domain of interest is contained in a cube, called the root cube: the root cube is recursively decomposed into a tree-like hierarchy of cubes: the cubes at the lowest level in the hierarchy are called leaf cubes. We assume that the leaf cubes are of equal length² l . We choose the scaling factor s in such a way that the leaf cubes become unit cubes:

$$s = \frac{1}{l} \quad (3.47)$$

3.3.1 Clustering: source points and source centres

A source centre of a cluster (e.g., a cube) represents all the source points in the cluster. The source centre is located at centre of the cluster. The process of obtaining the equivalent charge at the source centre is referred to as a Q2S translation. The equivalent charge is expressed in terms of an exponential distance defined below.

²In adaptive algorithms, l can be chosen as the length of the leaf cubes of smallest size.

Definition 3.3.1 (Q2S) For a collection of n_q source points $\{Q_i, i = 1..n_q\}$ of charges $\{q_i, i = 1..n_q\}$, in a leaf cube centred at S , we define the source at S associated with a pair of indices $[k, j]$ and a direction dir as

$$E^{source.dir}[k, j](S) = s \times \omega_k \sum_{i=1}^{n_q} q_i E^{dir}[k, j](Q_i S). \quad (3.48)$$

Here, the weight ω_k and the exponential distance $E^{dir}[k, j]$ come from approximation (3.1). Note that if each source point Q_i is replaced by a continuous distribution of charge density σ_i over an area A_i or a volume V_i , then the equivalent charge (3.48) is expressed as

$$E^{source.dir}[k, j](S) = s \times \omega_k \sum_{i=1}^{n_q} \iint_{A_i} \sigma_i(Q) E^{dir}[k, j](Q S) dA_Q \quad (3.49)$$

$$E^{source.dir}[k, j](S) = s \times \omega_k \sum_{i=1}^{n_q} \iiint_{V_i} \sigma_i(Q) E^{dir}[k, j](Q S) dV_Q. \quad (3.50)$$

Practical problems in which charges are of continuous distributions are common in electrical engineering. Here, the method of moments [41] for field computation is often employed in these problems. In chapter 4, we will present a formulation for the integral (3.49) in terms of the exponential distance.

A cluster of source centres can be replaced, in a similar way, with a new source centre. This process is referred to as an s2S translation. This new source centre represents all the source points associated with the constituent source centres. Its equivalent charge is also expressed in terms of an exponential distance defined below.

Definition 3.3.2 (s2s) For a cube centred at S that has n_c child cubes centred at $S_1..S_{n_c}$, we define the source at S associated with a pair of indices $[k, j]$ and a direction dir as

$$E^{source,dir}[k, j](S) = \sum_{i=1}^{n_c} E^{source,dir}[k, j](S_i) \times E^{dir}[k, j](S_i S). \quad (3.51)$$

The clustering process starts at the leaf cubes, progressing upwards in the tree-like hierarchy, and ends at the root cube. Once the clustering process is completed, the next step is to shift the source centres to target centres for efficient evaluation of the far-field potential, which is the potential due to source points contained in a cube that is at least one cube apart from the cube containing the target points.

3.3.2 Shifting: from source centres to target centres

A target centre of a cluster represents all target points in the cluster in receiving the far-field potential. The target centre is located at the centre of the cluster. The shifting of a source centre to a target centre is the transformation of the global effect of the source centre to a local accumulative effect at the target centre. The effect is accumulative because the target centre may also be under the influence from other source centres as well as other target centres. The effect is local because it is propagated (redistributed) only to the target points in its cluster. This process is referred to as an s2t translation. The equivalent potential at the target centre, which represents the effect of all source points outside its cluster and the surroundings, is also expressed in terms of an exponential distance defined below.

Definition 3.3.3 (s2T) For a cube centred at T and for cubes centred at $\{S_i, i = 1..n_s\}$ with vectors $S_i T$ being bounded in $[a+1, b-1]$ and having the same direction Dir , we define the target at T associated with a pair of indices $[k, j]$ and the direction Dir as

$$E^{target, Dir}[k, j](T) \quad += \quad \sum_{i=1}^{n_s} E^{source, Dir}[k, j](S_i) \times E^{Dir}[k, j](S_i T). \quad (3.52)$$

Here, we use the symbol $+=$ to indicate that the sources S_i in s2T partially contribute to the potential at the target T ; the other sources of contribution come from other source centres (also via an s2T) or from other targets via a T2T translation (to be described in the next subsection).

The s2T has the same structure as the other translations. However, for a given cube, the number of s2T translations necessary to account for all far-field interactions is much larger and hence more expensive. In the following, we provide an estimate of the total number of s2T translations needed for a given cube.

- Cubes and directions: let P be a parent cube with eight children cubes $\mathcal{C} = \{C_l, l = 0..7\}$. We encode P with a triple (k, j, i) of non-negative integers that are bounded by the number of cubes on the same level along the z -, y -, and x -directions, respectively. Each C_l can be also be encoded as $(2k + \Delta_k^l, 2j + \Delta_j^l, 2i + \Delta_i^l)$, where $\Delta_k^l, \Delta_j^l, \Delta_i^l \in \{0, 1\}$. The set \mathcal{C} can be decomposed into

pairs of subsets of size 4:

$$\mathcal{C} = \begin{cases} \mathcal{C}^{Up} \cup \mathcal{C}^{Down} \\ \mathcal{C}^{East} \cup \mathcal{C}^{West} \\ \mathcal{C}^{North} \cup \mathcal{C}^{South}, \end{cases} \quad (3.53)$$

which are defined as

$$\mathcal{C}^{Up} \triangleq \{C_l \in \mathcal{C} \mid \Delta_k^l = 1\} \quad (3.54)$$

$$\mathcal{C}^{Down} \triangleq \{C_l \in \mathcal{C} \mid \Delta_k^l = 0\} \quad (3.55)$$

$$\mathcal{C}^{East} \triangleq \{C_l \in \mathcal{C} \mid \Delta_i^l = 1\} \quad (3.56)$$

$$\mathcal{C}^{West} \triangleq \{C_l \in \mathcal{C} \mid \Delta_i^l = 0\} \quad (3.57)$$

$$\mathcal{C}^{North} \triangleq \{C_l \in \mathcal{C} \mid \Delta_j^l = 1\} \quad (3.58)$$

$$\mathcal{C}^{South} \triangleq \{C_l \in \mathcal{C} \mid \Delta_j^l = 0\}. \quad (3.59)$$

- Interaction list: for each cube C_l , the associated interaction list, which is denoted as \mathcal{I}_l , is the list of all cubes C at the same level as C_l such that $C_l C$ is bounded in $[2, 3]$. It is the list of C to which we perform the S2T translation from C_l in accordance with Theorem 3.2.1. The size of each \mathcal{I}_l is 189. The interaction list is decomposed into sublists associated with six directions:

$$\mathcal{I}_l = \bigcup_{dir \in \text{DirSet}} \mathcal{I}_l^{dir}, \quad (3.60)$$

which are defined as

$$\mathcal{I}_l^{dir} \triangleq \{C \in \mathcal{I}_l \mid Dir(C_l C) = dir\}. \quad (3.61)$$

Each of these sublists can be further partitioned into smaller sublists:

$$\mathcal{I}_l^{dir} = \begin{cases} \mathcal{I}_F^{dir} \cup \mathcal{I}_{D,l}^{dir} & \text{if } C_l \in \mathcal{C}^{dir} \\ \mathcal{I}_H^{dir} \cup \mathcal{I}_F^{dir} \cup \mathcal{I}_{D,l}^{dir} & \text{if } C_l \in \mathcal{C}^{-dir}. \end{cases} \quad (3.62)$$

Here, \mathcal{I}_F^{dir} and \mathcal{I}_H^{dir} denote the lists of cubes C that belong to all the interaction lists \mathcal{I}_l^{dir} with $C_l \in \mathcal{C}^{dir}$ or belong to only the interaction lists \mathcal{I}_l^{dir} with $C_l \in \mathcal{C}^{-dir}$, respectively. These lists are independent of l . The symbol $\mathcal{I}_{D,l}^{dir}$ denotes the remaining cubes of each \mathcal{I}_l^{dir} , $l = 0..7$.

- Forms of s2T: the above decomposition allows the s2T translations to be performed efficiently. For each $C \in \mathcal{I}_H^{dir}$ and all $C_l \in \mathcal{C}^{-dir}$, the s2T from C_l to C can be carried out by first (a) translating (via an s2S) from C_l to P and then (b) translating (via an s2T) from P to C . For all $C_l \in \mathcal{C}$, the s2T from C_l to each $C \in \mathcal{I}_F^{dir}$ can be performed similarly. These two forms of s2T are referred to as the *half-list* s2T and *full-list* s2T, respectively. The main difference between them is that the latter is applied for all cubes $C_l \in \mathcal{C}$ while the former is only for cubes $C_l \in \mathcal{C}^{-dir}$. The subscripts H and F in the respective lists \mathcal{I}_F^{dir} and \mathcal{I}_H^{dir} are used to distinguish these two forms of s2T. For $C \in \mathcal{I}_{D,l}^{dir}$, we resort to the direct s2T translations from C_l without using P . This form of s2T is referred to as the *direct-list* s2T, which is indicated

by the subscript D in the associated list $\mathcal{I}_{D,l}^{dir}$. The procedure to perform s2T translations for all cubes $C_l \in \mathcal{C}$ is summarized as follows:

Procedure 3.3.1 [s2T(*dir*)]

Step 1 Perform s2S from each $C_l \in \mathcal{C}^{-dir}$ to P .

Step 2 Perform s2T from P to each $C \in \mathcal{I}_H^{dir}$.

Step 3 Perform s2S from each $C_l \in \mathcal{C}^{dir}$ to P .

Step 4 Perform s2T from P to each $C \in \mathcal{I}_H^{dir}$.

Step 5 For each $C_l \in \mathcal{C}$ and for each $C \in \mathcal{I}_{D,l}^{dir}$, perform direct s2T from C_l to C .

- Number of s2T translations: the size of the sublists \mathcal{I}_H^{dir} , \mathcal{I}_F^{dir} , and $\mathcal{I}_{D,l}^{dir}$ for each C_l and each direction is listed in Table 3.1. The number of s2T translations per cube is 89, which is less than half the size of the initial 189-cube interaction list. Furthermore, as a by-product, the s2S translations from \mathcal{C} to P are also obtained (steps 1 and 3) with no extra cost.

Once the target centre has accumulated all the effects from the sources that are outside its cluster and the surroundings, its equivalent potential can then be redistributed to target centres of its subclusters or directly to its target points.

3.3.3 Redistributing: target centres and target points

Redistributing from a target centre to new target centres is the reverse operation of clustering source centres to form new source centres. It is the redistribution of the equivalent potential of one target centre to new target centres of its subclusters. This process is referred to as a T2T translation. The equivalent potentials of the new target centres are also expressed in terms of exponential distances defined below.

<i>dir</i>	\mathcal{I}_H^{dir}	\mathcal{I}_F^{dir}	$\mathcal{I}_{D,i}^{dir}$								s2s
			C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7	
up	36	16	9	9	9	9	9	9	9	9	8
down	36	16	9	9	9	9	9	9	9	9	8
east	24	8	13	7	13	7	13	7	13	7	8
west	24	8	7	13	7	13	7	13	7	13	8
north	16	4	14	14	5	5	14	14	5	5	8
south	16	4	5	5	14	14	5	5	14	14	8
total	152	56	57	57	57	57	57	57	57	57	48

Table 3.1: Number of s2T translations per cube
 (There are 89 ($= 57 + \frac{152+56+48}{8}$) s2T translations for each cube)

Definition 3.3.4 (T2T) For a cube centred at T that has n_c child cubes centred at $T_1..T_{n_c}$, we define the target at T_i , $i = 1..n_c$, associated with a pair of indices $[k, j]$ and a direction *dir* as

$$E^{target.dir}[k, j](T_i) \quad += \quad E^{target.dir}[k, j](T) \times E^{dir}[k, j](TT_i). \quad (3.63)$$

A target point can be viewed as a special target centre of a one-point cluster. In this case, the T2T is referred to as a T2P translation. It redistributes the equivalent potential from a target centre to a target point in the cluster. The equivalent potential contains all the necessary information of the far-field potential and its gradient at the target point.

Definition 3.3.5 (T2P) Let T be the centre of a leaf cube. Let P be a target point in the cube. The equivalent potential at P associated with indices $[k, j]$ and a

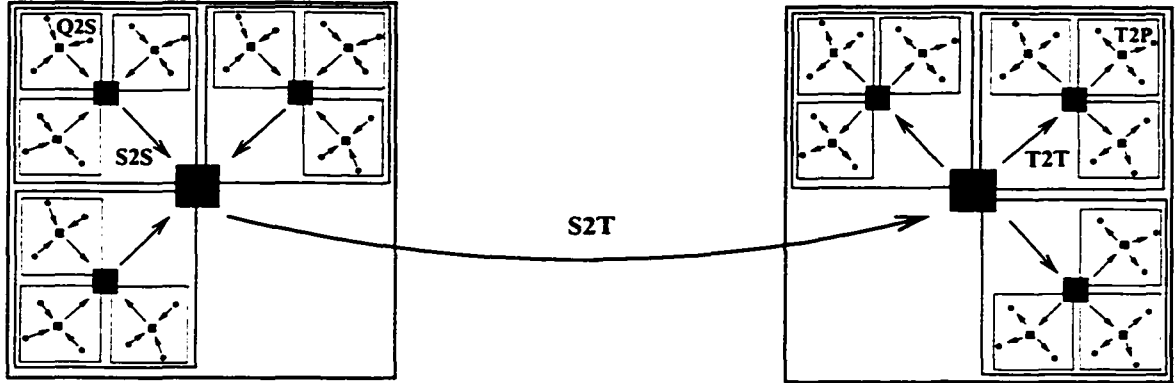


Figure 3.3: Translations in the EE method.

direction dir is defined in terms of exponential distance as

$$E^{target.dir}(P) = E^{target.dir}[k, j](T) \times E^{dir}[k, j](TP) \quad (3.64)$$

$$= \frac{E^{target.dir}[k, j](T)}{E^{dir}[k, j](PT)}. \quad (3.65)$$

Relation (3.65) follows from (3.64) and Proposition 3.2.1(iv). Here, instead of calculating $E^{dir}[k, j](TP)$ directly, we can obtain it via the multiplicative inverse of $E^{dir}[k, j](PT)$, which has already been calculated in the Q2S translation.

The translations in the EE method are illustrated in Fig. 3.3. After a sequence of translations Q2S, S2S, S2T, T2T, and T2P, from source points to target points, the information about the far-field potential and its gradient at target points is available for retrieval.

3.3.4 Potential Φ and its gradient $\nabla\Phi$

Potential Φ

The potential $\Phi(P)$ of a target point P can be expressed as a sum of two

potentials: $\Phi^T(P)$ and $\Phi^D(P)$. The former is due to all far-field interactions, which are covered by the S2T translations. The latter is due to interactions with source points of close proximity. In a tree-like hierarchy, it includes all points in the leaf cube containing P and in the cubes that are neighbours of the leaf cube. For a given target point P , we denote this set as $\text{Neighbour}(P)$.

$$\Phi(P) \approx \Phi^T(P) + \Phi^D(P). \quad (3.66)$$

For $\Phi^D(P)$, we resort to direct pairwise particle-to-particle potential calculation (1.21).

$$\Phi^D(P) = \sum_{Q \in \text{Neighbour}(P)} \frac{q(Q)}{\|QP\|}. \quad (3.67)$$

For $\Phi^T(P)$, we express it in terms of the potential due to translations associated with each pair of indices $[k, j]$, which we denote as $\Phi[k, j](P)$.

$$\Phi^T(P) = \sum_{k=1}^{S_{\text{outer}}} \sum_{j=1}^{S_{\text{inner}}(k)} \Phi[k, j](P). \quad (3.68)$$

Each $\Phi[k, j]$ can be expressed in terms of the potential due to translations associated with six directions, which we denote as $\Phi^{\text{dir}}[k, j](P)$, $\text{dir} \in \text{DirSet}$.

$$\Phi[k, j](P) = \sum_{\text{dir} \in \text{DirSet}} \Phi^{\text{dir}}[k, j](P). \quad (3.69)$$

Gradient $\nabla\Phi = (\Phi_x, \Phi_y, \Phi_z)$

The gradient $\nabla\Phi$ of the potential is a vector whose components are the partial derivatives Φ_x , Φ_y , and Φ_z of the potential Φ in the x -, y -, and z -direction, respec-

tively. The expressions for these partial derivatives can be obtained in correspondence to (3.66)–(3.69). For example, the expressions for Φ_z read

$$\Phi_z(P) = \Phi_z^T(P) + \Phi_z^D(P) \quad (3.70)$$

$$\Phi_z^D(P) = \sum_{Q \in \text{Neighbour}(P)} \frac{-q(Q)z_{QP}}{\|QP\|^3} \quad (3.71)$$

$$\Phi_z^T(P) = s \times \sum_{k=1}^{S_{\text{outer}}} \lambda_k \sum_{j=1}^{S_{\text{inner}}(k)} \Phi_z[k, j](P) \quad (3.72)$$

$$\Phi_z[k, j](P) = \sum_{\text{dir} \in \text{DirSet}} \Phi_z^{\text{dir}}[k, j](P). \quad (3.73)$$

The expressions for Φ_x and Φ_y can be constructed similarly. Note that in (3.72), we have included the scaling factor s introduced in (3.47) and λ_k in (3.4).

In order to have a complete expression for the potential and its gradient, we now describe $\Phi^{\text{dir}}[k, j](P)$, $\Phi_x^{\text{dir}}[k, j](P)$, $\Phi_y^{\text{dir}}[k, j](P)$, and $\Phi_z^{\text{dir}}[k, j](P)$ for each direction $\text{dir} \in \text{DirSet}$. These values can be retrieved from the equivalent potential at the target point $E^{\text{target.dir}}[k, j](P)$ in (3.64).

Proposition 3.3.1 *Let P be a target point with the equivalent potential as a complex number $E^{\text{target.dir}}[k, j](P) = R_{kj}^{\text{dir}} + iI_{kj}^{\text{dir}}$, $\text{dir} \in \text{DirSet}$. The potential at P due to all the translations associated with indices $[k, j]$ and direction dir is*

$$\Phi^{\text{dir}}[k, j] = R_{kj}^{\text{dir}}. \quad (3.74)$$

The corresponding partial derivatives are

$$\Phi_x^{dir}[k, j] = \begin{cases} -\cos \alpha_{kj} I_{kj}^{dir} & dir \in \{Up, Down, North, South\} \\ -R_{kj}^{East} & dir = East \\ +R_{kj}^{West} & dir = West \end{cases} \quad (3.75)$$

$$\Phi_y^{dir}[k, j] = \begin{cases} -\sin \alpha_{kj} I_{kj}^{dir} & dir \in \{Up, Down, East, West\} \\ -R_{kj}^{North} & dir = North \\ +R_{kj}^{South} & dir = South \end{cases} \quad (3.76)$$

$$\Phi_z^{dir}[k, j] = \begin{cases} -R_{kj}^{Up} & dir = Up \\ +R_{kj}^{Down} & dir = Down \\ -\cos \alpha_{kj} I_{kj}^{dir} & dir \in \{East, West\} \\ -\sin \alpha_{kj} I_{kj}^{dir} & dir \in \{North, South\}. \end{cases} \quad (3.77)$$

Proof. Let n be the number of levels in the tree-like hierarchy. We assume that $b = 2^m \geq 2^n$ so that all far-field interactions are covered by the S2T translations and therefore, no extra scaling is required. We shall discuss the case in which $b = 2^m < 2^n$ in §§ 3.5.3 and 3.5.4.

In § 3.2.5, using (3.41), we obtain the equivalent potential associated with one direction using a sequence of clustering, shifting and redistributing operations. In the general case, all of the six directions should be taken into account. This explains the sum in (3.69) and (3.73). Note that in (3.8) only the real part of $S(x, y, z)$ is taken. This explains why only the real part of $E^{target, dir}[k, j](P)$ is used in (3.74).

The partial derivatives (3.75)–(3.77) are in fact the partial derivatives of the equivalent potential $E^{target, dir}[k, j](P)$ (excluding the factor $s \times \lambda_k$). Because the

equivalent potential is only an approximation of the far-field potential, the results in (3.75)–(3.77) may be less accurate than the result in (3.74). This is due to approximation (3.1), and not the translations. Note that the accuracy of the computed gradient can be improved using more accurate approximations (3.1) [25]. ■

The above five Definitions 3.3.1–3.3.5 for translations in the EE method are reminiscent of the Q2M, M2M, M2L, L2L, and L2P translations in the MP method. The translations here, however, have a distinct character. This is discussed in what follows.

3.4 Salient features

In this section, we describe some of the salient features of the proposed approach with respect to simplicity, accuracy, speed, memory requirement, retrieval of gradient of potential, and possibilities for efficient distributed computing and parallel processing. The salient features stem from the simplicity of the translations and the independence between the approximation and the translations.

3.4.1 Simplicity

The translations in the EE method are tidy. They all have the same structure: the value at destination is equal to or accumulative of the value at origin (with the same indices and same direction) multiplied by the exponential distance E between origin and destination (see (3.48)–(3.65)). The translations can be expressed individually for each pair of indices $[k, j]$ and each direction (see (3.68)–(3.69) and (3.72)–(3.73)). It is this kind of simplicity that makes it easier for comprehension and implementation. In contrast, the translations in the MP method have different

structures: one for M2M, another for M2L, and yet another for L2L. Furthermore, they involve a double sum (original version) or a single sum (new version) with tight coupling between the multipole and local expansion coefficients M_m^n 's and L_m^n 's.

3.4.2 Accuracy

All translations in the EE method are exact. They are based purely on the equality given in (3.19) and hence no additional error is introduced. The computational accuracy depends solely on the approximation of $\frac{1}{r}$ (3.1) in the associated set D_b^{Up} (3.5). Proposition 3.1.3 describes the relationships between the intended error of the approximation and the actual error of the calculated result. This shows that the accuracy of the EE method is controlled externally. If a more accurate approximation in (3.1) is available without increasing the approximation size S_{app} , then this automatically translates into an improvement in the accuracy of the computed potential without undermining the computational performance. In stark contrast, the accuracy of the MP method inherently depends on the order of the expansion p and on how the M2L translations are performed. Discussions in the literature (see, for example, [42.43]) on error estimation of the MP method indicates the associated complexity.

3.4.3 Speed

All translations in the EE method are of the order $O(S_{app})$. In Q2S, the calculation involves exponent, cosine, and sine operations. In other translations, only multiplication and addition operations in complex numbers are involved. In all of the translations, except S2T, the calculations for all six directions are needed. In S2T, which has the largest number of translations, calculation for only one direction is

Translation	Cost	Object
Q2S	$6 \times S_{app} \times C_e$	per particle
S2S	$6 \times S_{app} \times C_m$	per cube
S2T	$1 \times S_{app} \times C_m$	per translation
	$89 \times S_{app} \times C_m$	per cube
T2T	$6 \times S_{app} \times C_m$	per cube
T2P	$6 \times S_{app} \times C_m$	per particle

Table 3.2: Computational cost of translations in EE method

needed. Let C_e and C_m denote the costs to perform calculations involving exponent and multiplication operations, respectively. The cost of each of the above translations is summarized in Table 3.2. Here, we observe that all translations are of the order $O(S_{app})$. Hence, the efficiency of the translations is also controlled externally via S_{app} , the size of the approximation (3.1). A smaller approximation size, without compromising the accuracy of approximation (3.1), will improve computational speed without loss of accuracy in the computed potential. This is in contrast to the MP method, whose computation time inherently depends on the order of the expansion, $O(p^3)$ in the new version or $O(p^4)$ in the original version, and on the trade-off between computational time and memory in choosing the interaction list for M2L translations. With both accuracy and speed being controlled externally, the performance of the EE method can be improved by having more accurate approximations with smaller sizes.

3.4.4 Memory requirement

The memory requirement in the EE method is independent of the size of the approximation and hence the desired degree of accuracy. For each pair of indices $[k, j]$

Object	Variable	Memory unit for complex number
particle	E	1 (optional)
cube	E^{source}	1 (and/or)
	E^{target}	1

Table 3.3: Cost in memory independent of degree of accuracy

and for each direction $dir \in \text{DirSet}$, the value at destination depends only on the value at origin with the same indices and same direction. As a result, there is no need to store all of the $E[k, j]$'s. Nor is there need to store the values of $E[k, j]$ for all of the six directions. One E can be used for all six $E^{dir}[k, j]$'s, $dir \in \text{DirSet}$. More specifically, the memory requirement in the proposed method is listed in Table 3.3. The memory unit is the number of bytes to store a complex number or two real numbers. The variables E , E^{source} , and E^{target} are used for all six directions and for all pair of indices $[k, j]$. In contrast, the multipole and local expansion coefficients M_m^n and L_m^n in the MP method have to be stored explicitly; the memory requirement is $O(p^2)$. In the new version of the FMA, the memory requirement is even more demanding, with $O(6p^2)$ extra memory units for storing the exponential expansion coefficients at cubes which are being affected by M2L translations. The improvement in the new version of the FMA is, to some degree, offset by this high demand for memory. The memory requirement per cube in the two approaches, MP versus EE, are summarized in Table 3.4. From this, it is clear that the EE method proposed here is far more efficient in terms of memory requirement. Thus, it is now possible to simulate larger ensembles of particles with higher degrees of accuracy, given the same computing resources.

Approaches	Memory requirement
multipole expansion (MP)	$\frac{(p+1)(p+2)}{2}$
exponential expansion (EE)	1
ratio $\frac{MP}{EE}$	$\frac{(p+1)(p+2)}{2}$

Table 3.4: Memory requirement per cube in the two approaches: MP vs. EE.

If it turns out that there is unused memory, the overhead due to processing each individual $E[k, j]$ can be reduced by storing $E[k, j]$ for more than one direction as well as for more than one pair of indices, and performing translations on them in the same tree traversal. However, it should be emphasized that this is an implementation option, not an inherent requirement imposed by the EE method.

3.4.5 Retrieval of gradient of potential

In the EE method, the gradient of potential can be obtained as a by-product from the computational process of the potential. The retrieval of the gradient is accomplished using expressions (3.75)–(3.77), the overhead of which is small (see Fig. 3.4). This is applied to both types of charge distributions: discrete charge distribution (3.48) and continuous charge distributions (3.49)–(3.50). In contrast, the formulation and the evaluation of the gradient of the potential in MP method is quite complex (see, for example, [17]).

3.4.6 Distributed and parallel processing

The EE method renders itself naturally to distributed computing and parallel processing. For a given pair of indices $[k, j]$ and a direction $dir \in \text{DirSet}$, the exponential distance $E^{dir}[k, j]$ can be processed independently. This makes it possible

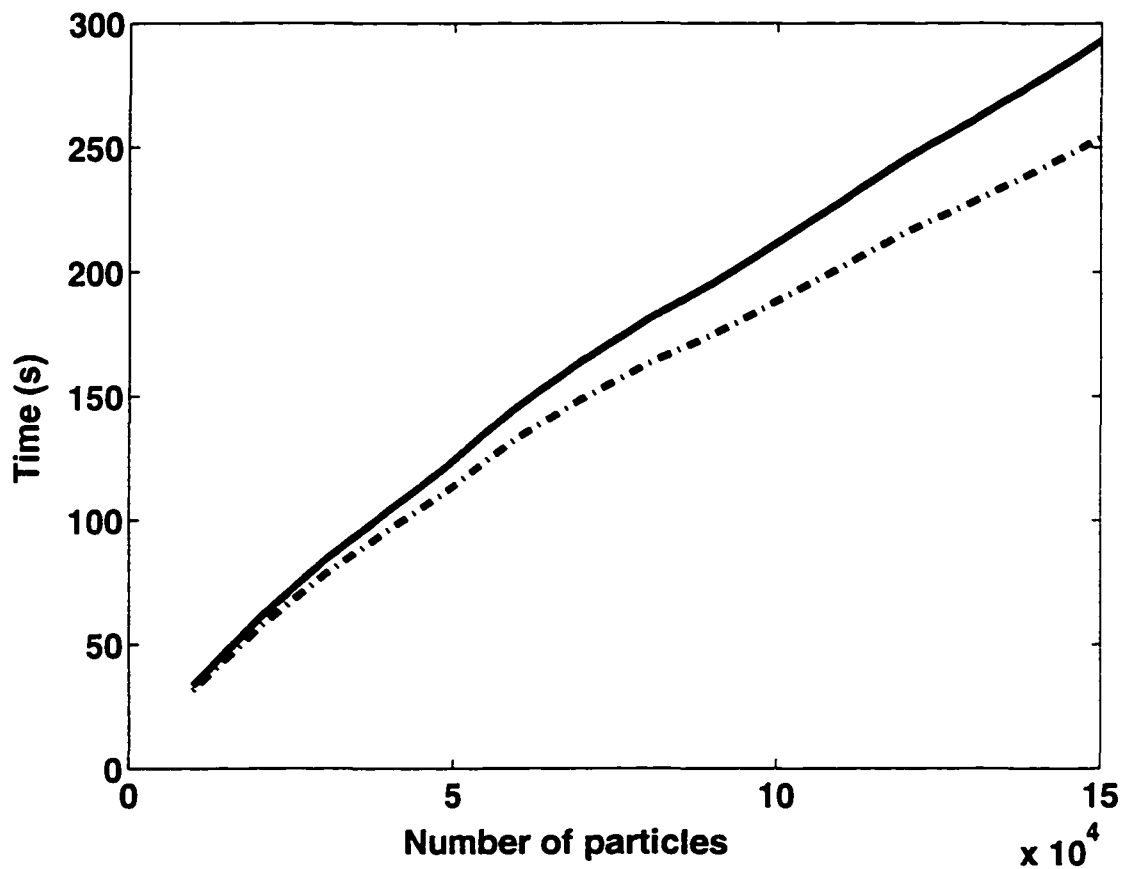


Figure 3.4: Running time for evaluation of potential only (dashed), and both potential and its gradient (solid). The overhead in evaluating both potential and its partial derivatives ranges from 1.06 to 1.15.

for large-scale simulation of particle systems using remotely-distributed computer networks. Furthermore, for each $E^{dir}[k, j]$, parallel or distributed schemes for performing the required translations can also be employed here. Therefore, with the EE method, there are two forms of parallelism: parallelism among the $E^{dir}[k, j]$'s and parallelism among translations for each $E^{dir}[k, j]$. The following typifies sequential and parallel codes:

Sequential Code	Parallel Code
For each pair $[k, j]$ and	In parallel,
For each direction	For each pair $[k, j]$ and
in sequence,	each direction
perform translations	in parallel,
End	perform translations
End	End

Following the above discussion on the attributes of the EE method, we next discuss the associated algorithms. We propose three algorithms using the translation (3.19) and approximation (3.20) for efficient evaluation of the potential field in an ensemble of N particles. Algorithm I handles the case of $a = 1$ and $b \geq 2^n$, where n denotes the number of levels in the tree-like hierarchy of cubes. The framework is similar to Greengard's FMA. However, all the translations are now performed on the exponential distances $E^{dir}[k, j]$'s, which can be processed independently. Furthermore, there is no need to perform the s2S translations directly; they are already available from the s2T translations. Algorithm II handles the case of $a = 1$ and $b = 4$. Here, due to the scaling problem (to be discussed later), we employ a

very different framework without the efficient S2S and T2T translations. Algorithm III is a generalization of Algorithms I and II. It handles the case of $a = 1$ and $b = 2^m \leq 2^n$, where m is an integer between 2 and n .

3.5 Algorithms

The three algorithms have many computational steps in common. They all have the same input and output structure; they have the same structure for S2T and T2P translations: and they have the same procedure for direct pairwise short-range particle-to-particle potential calculation. Before describing the specific algorithms, we first discuss the computational steps they have in common.

3.5.1 Common computational steps

Input and output

In all of the three algorithms, the input is (a) the number of levels n of the tree-like hierarchy, (b) the approximation (3.1) for a given error ϵ , and (c) an ensemble of particles $\{X_i, i = 1..N\}$ whose charges are $\{q_i, i = 1..N\}$. The output is the potential $\Phi(X_i)$ and/or its gradient $\nabla\Phi(X_i)$ at each location X_i . Note that a location X_i represents both a source particle and a target particle. The preprocessing step involves (a) precomputing some frequently used values of E , such as $E(\pm 0.5, \pm 0.5, \pm 0.5)$ for S2S, $E(\pm x, \pm y, \pm z)$ with $x, y, z \in \{0.5, 1.5, 2.5, 3.5\}$ for half-list and full-list S2T, and $E(\pm x, \pm y, \pm z)$ for $x, y, z \in \{0, 1, 2, 3\}$ for direct-list S2T; (b) precomputing $\sin \alpha_{kj}$ and $\cos \alpha_{kj}$ for Q2S; (c) constructing the tree-like hierarchy of cubes; and (d) allocating memory for the variable E at each particle (optional), E^{source} and/or E^{target} at each cube.

Direct pairwise short-range particle-to-particle potential calculation**Procedure 3.5.1 [Direct]**

For each particle P , perform direct pairwise short-range particle-to-particle potential calculation using (3.67) on the set $\text{Neighbour}(P)$.

Cost analysis: let P_{pc} denote the average number of particles per leaf cube. For each particle, $27 \times P_{pc}$ such calculations are needed. Let C_d denote the cost of each direct pairwise particle-to-particle potential calculation. The total cost of all pairwise particle-to-particle potential calculation is

$$\text{Cost}(\text{DirectCalculation}) = 27 \times P_{pc} \times C_d \times N. \quad (3.78)$$

TreeTraversal(dir, n_{start}, n_{end})

The following procedure shows how the translations S2S, S2T, T2T, and T2P, in all of the three algorithms, are linked in order to obtain the potential. The only missing translation is the Q2S translation. Each algorithm handles Q2S in a different manner. The three input parameters are the direction dir and the levels at which the tree traversal starts (n_{start}) and ends (n_{end}), with $n_{start} > n_{end}$. All translations associated with the cubes at the levels between n_{start} and n_{end} are performed, with the exception of the Q2S translation. It is assumed that the source centres at the level n_{start} have been calculated and the length l of cubes at level n_{start} is chosen as the unit length. The scaling factor is given in (3.47). During the tree traversal, the remaining source centres, target centres, and target points are computed.

Step	Operation cost	Particle vs. cube
step 1	$89 \times S_{app} \times C_m$	per cube
step 2	$6 \times S_{app} \times C_m$	per cube
step 3	$6 \times S_{app} \times C_m$	per particle

Table 3.5: Cost of operations per cube or per particle in TreeTraversal for all six directions

Procedure 3.5.2 [TreeTraversal(dir, n_{start}, n_{end})]

Step 1 For level from n_{start} to n_{end} , perform s2T(dir) (see Procedure 3.3.1).

Step 2 For level from n_{end} to $n_{start} - 1$, perform T2T.

Step 3 At level n_{start} , perform T2P and retrieve the potential using (3.69) and (3.74).

Cost analysis: The computational cost of each step in TreeTraversal for all six directions is listed in Table 3.5. For a given cube, there are about 89 s2T translations and the total cost of all translations is $6 \times S_{app} \times C_m \times (1 + \frac{89}{6}) \approx 6 \times S_{app} \times C_m \times 16$. Let $C_{(n_{start}, n_{end})}$ be the total number of cubes from level n_{start} to level n_{end} . The total cost of the procedure TreeTraversal for all of the six directions is estimated as

$$\begin{aligned} \text{Cost}(\text{TreeTraversal}(dir, n_{start}, n_{end})) \\ = 6 \times S_{app} \times C_m \times [16 \times C_{(n_{start}, n_{end})} + N]. \end{aligned} \quad (3.79)$$

3.5.2 Algorithm I (for $a = 1, b \geq 2^n$)

Procedure 3.5.3 [Algorithm I]

For each pair indices $[k, j]$ and each direction $dir \in \text{DirSet}$,

Step	Operation cost	Particles vs. cubes
step 1	$6 \times S_{app} \times C_e \times N$	all particles
step 2	$6 \times S_{app} \times C_m \times [\frac{17}{P_{pc}} + 1] \times N$	all cubes and particles
step 3	$27 \times P_{pc} \times C_d \times N$	all particles

Table 3.6: Cost of operations in Algorithm I

Step 1 At the finest level (n^{th} level), perform Q2S.

Step 2 Perform $\text{TreeTraversal}(dir, n, 2)$.

Step 3 Perform direct pairwise short-range particle-to-particle potential calculation (Procedure 3.5.1).

Cost analysis: the total number of non-leaf cubes is about $\frac{1}{7}$ the number of leaf cubes. Therefore, the total number of cubes in the tree-like hierarchy is about $C_{(n_{start}=n, n_{end}=1)} = \frac{8}{7} \frac{N}{P_{pc}}$. The computational cost of each step is summarized in Table 3.6. The cost of step 2 is obtained from (3.79) by replacing $C_{(n_{start}, n_{end})}$ with $\frac{8}{7} \frac{N}{P_{pc}}$. The cost of step 3 is obtained from (3.78). The total cost is estimated as

$$\begin{aligned} \text{Cost}(\text{Algorithm I}) &= \left(6S_{app}(C_e + [\frac{17}{P_{pc}} + 1]C_m) + 27P_{pc}C_d \right) \times N. \end{aligned} \quad (3.80)$$

If the cost of direct pairwise short-range particle-to-particle potential calculation is relatively small, the total cost can be considered as $O(N)$, or more precisely as $O(S_{app}N)$.

3.5.3 The scaling problem

In the approximation (3.1)–(3.5) with $b < 2^n$, not all s2T translations are possible. The reason is that according to Theorem 3.2.1 and Definition 3.3.3, an s2T

translation is only valid if the vector connecting the centres containing the source and the target is bounded in $[a + 1, b - 1]$. Because $b < 2^n$, there exist source centres and target centres (such as those at the two ends of the root cube) whose vectors are not bounded in $[a + 1, b - 1]$. The case $b = 2^2$ is an extreme case in which this condition is never satisfied when we move either upwards or downwards to the next level. One way to deal with this difficulty is by scaling. Here, we choose the length of the current cubes as the unit length (as opposed to that of the leaf cubes). Due to this new scaling, all of the Q2S's have to be recomputed. A direct recomputation would be too expensive because it involves exponent, cosine, and sine operations. To avoid unnecessary recomputation, we need the following proposition, which provides an efficient method to obtain the Q2S to a child cube from the Q2S to the parent cube. We assume that Q2S is performed with the length of the cube (containing the source) chosen as unit length.

Proposition 3.5.1 *Let S be the centre of a cube and S_P the centre of its parent cube. Let Q be a source particle in cube S , and hence also in cube S_P . The Q2S from Q to S can be obtained from the Q2S from Q to S_P as follows:*

$$\begin{aligned} E^{dir}[k, j](QS_{unit(S)}) \\ = \left\{ E^{dir}[k, j](QS_{Punit(S_P)}) \right\}^2 E^{dir}[k, j](S_P S_{unit(S)}). \end{aligned} \quad (3.81)$$

Here, the subscripts $unit(S)$ and $unit(S_P)$ are used to indicate that the lengths of the respective cubes S and S_P are chosen as unit length in the calculation of E . The cost to perform Q2S from Q to S for each direction is now $2 \times C_m$, which is much cheaper than a direct Q2S, which costs C_e .

Proof.

$$\begin{aligned} E^{dir}[k, j](QS_{unit(S)}) \\ = E^{dir}[k, j](QS_{P_{unit(S)}}) E^{dir}[k, j](S_P S_{unit(S)}) \end{aligned} \quad (3.82)$$

$$= E^{dir}[k, j](2QS_{P_{unit(S_P)}}) E^{dir}[k, j](S_P S_{unit(S)}) \quad (3.83)$$

$$= \left\{ E^{dir}[k, j](QS_{P_{unit(S_P)}}) \right\}^2 E^{dir}[k, j](S_P S_{unit(S)}). \quad (3.84)$$

■

Note that we can also obtain the Q2S to parent from the Q2S to children in a similar way. But this is not preferable because it is more costly in view of the square root operation involved.

In approximation (3.1) and the associated set D_b^{Up} (3.5) with $a = 1$ and $b = 4$, we process the cubes level by level, starting from level 2. At each level, we consider the associated cubes as leaf cubes. The sequence of translations is as follows: Q2S, S2T, and T2P. The translations S2S and T2T are used as part of T2P.

3.5.4 Algorithm II (for $a = 1$, $b = 2^2$)

Procedure 3.5.4 [Algorithm II]

For each pair indices $[k, j]$ and each direction $dir \in \text{DirSet}$,

level = 2

Step 1 Perform direct Q2S.

Step 2 Perform TreeTraversal(dir , $level$, 2).

level = 3 to n step 1

Step 3 Perform Q2S using (3.81) for $E^{dir}[k, j](QS)$.

Step	Operation cost	Particles vs. cubes
step 1	$6 \times S_{app} \times C_e \times N$	all particles
step 2,4	$6 \times S_{app} \times C_m \times [\frac{17}{P_{pc}} + (n - 1)] \times N$	all cubes
step 3	$6 \times S_{app} \times C_m \times 2 \times (n - 2) \times N$	all cubes and particles
step 5	$27 \times P_{pc} \times C_d \times N$	all particles

Table 3.7: Cost of operations in Algorithm II

Step 4 Perform `TreeTraversal(dir, level, level)`.

Direct

Step 5 Perform direct pairwise short-range particle-to-particle potential calculation (Procedure 3.5.1).

Cost analysis: the computational cost of each step is summarized in Table 3.7. In steps 2 and 4, we need $(n - 1)$ T2P translations for each particle. In step 3, we need $(n - 2)$ Q2S translations using (3.81) of Proposition 3.5.1. The cost of each subsequent Q2S translation for each E is $2 \times C_m$. The total cost is estimated as

$$\begin{aligned} & \text{Cost}(\text{Algorithm II}) \\ &= \left(6S_{app}(C_e + [\frac{17}{P_{pc}} + (3n - 5)]C_m) + 27P_{pc}C_d \right) \times N. \end{aligned} \quad (3.85)$$

If the cost of direct pairwise short-range particle-to-particle potential calculation is relatively small, the total cost can also be considered as $O(N)$, or more precisely as $O(S_{app}N)$. It is noted that the total cost (3.85) is also a function of the number of levels n . A comparison between the algorithms will be discussed later.

3.5.5 Algorithm III (for $a = 1, 2^2 \leq b = 2^m \leq 2^n$)

We have discussed the two extreme cases in which the condition of boundedness in $[a + 1, b - 1]$ (of vectors connecting two cube centres that are at least one cube apart) is either always satisfied or never satisfied when one moves from one level to another in the tree-like hierarchy. Now we discuss the general case in which $4 = 2^2 \leq b = 2^m \leq 2^n$, where n is the number of levels in the tree-like hierarchy and m is an integer between 2 and n . Let $\Delta m = m - 1$ and $\Delta n = \lfloor \frac{n-2}{\Delta m} \rfloor$; $(\Delta m - 1)$ can be considered as the number of levels we can move through without violating the condition of boundedness in $[a + 1, b - 1]$; Δn is the number of re-works on Q2S each time the condition is violated. While moving from one level to another, if the condition of boundedness in $[a + 1, b - 1]$ is still satisfied, we can apply the strategy in Algorithm I by just continuing with S2T or T2T translations; otherwise, we use the strategy in Algorithm II by reworking on the Q2S translations. In Algorithm II, the rework on Q2S is based on Proposition 3.5.1. The following proposition extends Proposition 3.5.1 to a more general case.

Proposition 3.5.2 *Let S be the centre of a cube and S_A be the centre of its ancestor cube that is Δm levels higher. Let Q be a source particle in cube S and hence also in cube S_A . The Q2S from Q to S can be obtained from the Q2S from Q to S_A as follows:*

$$\begin{aligned} E^{\text{dir}}[k, j](QS_{\text{unit}(S)}) \\ = \left\{ E^{\text{dir}}[k, j](QS_{A\text{unit}(S_A)}) \right\}^{2^{\Delta m}} E^{\text{dir}}[k, j](S_A S_{\text{unit}(S)}). \end{aligned} \quad (3.86)$$

Proof. Similar to the proof of Proposition 3.5.1. ■

The cost of Q2S from Q to S is now $(\Delta m + 1) \times C_m$. Because Δm is small (1, 2, or 3), the cost here is expected to be smaller than the cost of calculating Q2S directly. Note that if Δm is large, by combining (3.86) with (3.81), we can avoid overflow and underflow in (3.86).

Procedure 3.5.5 [Algorithm III]

For each pair indices $[k, j]$ and each direction $dir \in \text{DirSet}$,

initLevel = $n - \Delta n \times \Delta m$

Step 1 Perform direct Q2S.

Step 2 Perform $\text{TreeTraversal}(dir, \text{initLevel}, 2)$.

level = $\text{initLevel} + \Delta m$ to n step Δm

Step 3 Perform Q2S using (3.86) for $E^{dir}[k, j](QS)$.

Step 4 Perform $\text{TreeTraversal}(dir, \text{level}, \text{level} - \Delta m + 1)$.

Direct

Step 5 Perform direct pairwise short-range particle-to-particle potential calculation (Procedure 3.5.1).

Cost analysis: the computational cost of each step is summarized in Table 3.8. In steps 2 and 4, we need $(\Delta n + 1)$ T2P translations for each particle. In step 3, we need (Δn) Q2S translations using (3.86) of Proposition 3.5.2; the cost of such Q2S translation for each E is $(\Delta m + 1) \times C_m$. The total cost is estimated as

Step	Operation cost	Particles vs. cubes
step 1	$6 \times S_{app} \times C_e \times N$	all particles
step 2,4	$6 \times S_{app} \times C_m \times [\frac{17}{P_{pc}} + (\Delta n + 1)] \times N$	all cubes and particles
step 3	$6 \times S_{app} \times C_m \times [\Delta n \times (\Delta m + 1)] \times N$	all particles
step 5	$27 \times P_{pc} \times C_d \times N$	all particles

Table 3.8: Cost of operations in Algorithm III

Cost(Algorithm III)

$$= \left(6S_{app}(C_e + [\frac{17}{P_{pc}} + \Delta n(\Delta m + 2) + 1]C_m) + 27P_{pc}C_d \right) \times N. \quad (3.87)$$

Again, if the cost of direct pairwise short-range particle-to-particle potential calculation is relatively small, the total cost can be considered as $O(N)$, or more precisely as $O(S_{app}N)$. The cost estimates (3.80) and (3.85) are special cases of (3.87). For Algorithm I, we have $\Delta m = n - 1$ and $\Delta n = 0$; for Algorithm II, we have $\Delta m = 1$ and $\Delta n = n - 2$.

3.5.6 Algorithms: comparison

At first glance, the difference in the costs (3.80) and (3.85) appears to be the terms in bracket: $[\frac{17}{P_{pc}} + 1]$ versus $[\frac{17}{P_{pc}} + (3n - 5)]$. However, an important factor that needs to be taken into account lies in the sizes of the approximations used in the two algorithms; they may be significantly different. With the same degree of accuracy, Algorithm I needs a larger approximation size while Algorithm II requires the indirect re-work on Q2S using Proposition 3.5.1. With the approximations available, if the cost of the indirect re-work on Q2S is relatively small, which is often the case for particles of discrete distributions (3.48), then Algorithm II is

faster than Algorithm I. This is in spite of the dependence of the cost (3.85) on the number of levels n in the tree-like hierarchy. However, for particles of continuous distributions (3.49)–(3.50), the rework can be more involved, and therefore, this may not be the case. In both cases, the overall performance can be improved by obtaining better approximations in (3.1), i.e., more accurate approximations with smaller sizes. Algorithm III is a hybrid version of Algorithms I and II, which inherits both the strengths and weaknesses from the two algorithms.

3.6 Numerical results

All of the three Algorithms I, II, and III are implemented in and tested on uniform random distributions with unit charges. The results demonstrates that with the EE method, the potential can be approximately evaluated with a wide range of accuracy and with a running time of the order $O(N)$, or more precisely $O(S_{app}N)$. In this section, we discuss these results.

3.6.1 Approximation and error

Maximum relative error and average relative error

For the purpose of comparison, we define the following two error norms: maximum relative error E_{MaxRel} and average relative error E_{AvgRel} .

$$E_{MaxRel}(\Phi) \triangleq \max_{1 \leq i \leq N} \left| \frac{\Phi(X_i) - \tilde{\Phi}(X_i)}{\Phi(X_i)} \right| \quad (3.88)$$

$$E_{AvgRel}(\Phi) \triangleq \sqrt{\frac{\sum_{i=1}^N |\Phi(X_i) - \tilde{\Phi}(X_i)|^2}{\sum_{i=1}^N |\Phi(X_i)|^2}}. \quad (3.89)$$

Here, $\Phi(X_i)$ and $\tilde{\Phi}(X_i)$ are the potentials calculated using direct particle-to-particle calculation and using the proposed EE method, respectively. For tests with $N > 10,000$ particles, only a subset of 100 random particles are chosen for direct pairwise particle-to-particle potential calculations, and the error norms (3.88) and (3.89) are evaluated on this subset.

The maximum relative error norm is tighter than the average relative error norm. It can be shown that

$$E_{AvgRel}(\Phi) \leq E_{MaxRel}(\Phi). \quad (3.90)$$

In the EE method, the two error norms above can be estimated using Proposition 3.1.3. If all of the charges are non-negative then $E_{MaxRel}(\Phi) < \epsilon_{rel}$ and hence from (3.90), $E_{AvgRel}(\Phi) < \epsilon_{rel}$. In general, we have

$$E_{MaxRel}(\Phi^+) < \epsilon_{rel} \quad (3.91)$$

$$E_{MaxRel}(\Phi^-) < \epsilon_{rel}, \quad (3.92)$$

and hence from (3.90),

$$E_{AvgRel}(\Phi^+) < \epsilon_{rel} \quad (3.93)$$

$$E_{AvgRel}(\Phi^-) < \epsilon_{rel}. \quad (3.94)$$

In the MP method, the error analysis becomes quite complicated, therefore, the average relative error norm E_{AvgRel} is commonly used for error estimation.

Intended errors ($\epsilon_{\text{abs}}, \epsilon_{\text{rel}}$) and actual errors ($E_{\text{MaxRel}}, E_{\text{AvgRel}}$)

The expressions (3.91)–(3.94) indicate that the actual errors (maximum relative error and average relative error) are bounded by the intended relative error ϵ_{rel} . The test results show that the actual errors are much smaller. For an intended k digits of accuracy in relative error, we can expect between $(k + 2)$ and $(k + 4)$ digits of accuracy in the calculated potential. The reason is that the intended error ϵ_{rel} is meant for the worst case scenario in approximation (2.54) to account for all points in D_b , especially those on the boundary. In general, the value on the left hand side of (2.54) is much smaller than the value ϵ_{rel} on the right hand side. Table 3.9 lists both the maximum relative error E_{MaxRel} and average relative error E_{AvgRel} of the calculated potentials using the exponential expansion listed in Table 2.2. The first three columns describe the approximation (3.1): its size and relative error (2.54) or absolute error (2.55). The remaining three pairs of columns show the maximum relative error E_{MaxRel} (3.88) and average relative error E_{AvgRel} (3.89) of the calculated potential in three sample sizes: $N = 1000, 10000, 50000$. On each row, both of the error norms are much smaller than the corresponding relative or absolute error for the approximation. This confirms (3.91) and (3.92). Furthermore, the average relative error is smaller than the maximum relative error. This supports (3.90). From the test results, it can be concluded that, with the approximations available, a wide range of accuracy, from a few digits to the machine precision, can be achieved in the evaluation of the potential field using the proposed EE method.

Approximation			$E_{MaxRel}(\Phi)$ and $E_{AvgRel}(\Phi)$					
$b = 2^2$			sample size N					
size	ϵ_{abs}	ϵ_{rel}	1000		10000		50000	
			E_{MaxRel}	E_{AvgRel}	E_{MaxRel}	E_{AvgRel}	E_{MaxRel}	E_{AvgRel}
18	10^{-1}		2.4×10^{-3}	3.5×10^{-4}	3.8×10^{-3}	4.3×10^{-4}	1.8×10^{-3}	4.0×10^{-4}
22	10^{-1}		1.7×10^{-3}	2.2×10^{-4}	2.9×10^{-3}	2.8×10^{-4}	1.6×10^{-3}	3.2×10^{-4}
25	10^{-1}		4.4×10^{-3}	1.1×10^{-3}	6.1×10^{-3}	1.4×10^{-3}	4.0×10^{-3}	1.4×10^{-3}
30	10^{-1}		2.3×10^{-3}	6.0×10^{-4}	3.0×10^{-3}	7.3×10^{-4}	2.8×10^{-3}	7.7×10^{-4}
31		10^{-1}	2.8×10^{-4}	6.0×10^{-5}	3.0×10^{-4}	5.1×10^{-5}	1.7×10^{-4}	4.6×10^{-5}
32	10^{-1}		1.3×10^{-3}	3.0×10^{-4}	1.8×10^{-3}	3.9×10^{-4}	1.3×10^{-3}	3.9×10^{-4}
38		10^{-1}	1.9×10^{-4}	2.9×10^{-5}	5.1×10^{-4}	4.1×10^{-5}	1.8×10^{-4}	3.7×10^{-5}
44		10^{-1}	1.1×10^{-4}	1.5×10^{-5}	7.7×10^{-5}	1.2×10^{-5}	3.7×10^{-5}	1.2×10^{-5}
50		10^{-1}	7.1×10^{-5}	1.3×10^{-5}	4.7×10^{-5}	9.1×10^{-6}	2.6×10^{-5}	7.8×10^{-6}
51		10^{-1}	6.7×10^{-4}	1.8×10^{-4}	9.5×10^{-4}	2.2×10^{-4}	9.3×10^{-4}	2.2×10^{-4}
60	10^{-2}		3.6×10^{-5}	7.0×10^{-6}	5.6×10^{-5}	9.6×10^{-6}	3.7×10^{-5}	8.8×10^{-6}
64	10^{-2}		6.1×10^{-5}	6.3×10^{-6}	7.2×10^{-5}	9.3×10^{-6}	5.1×10^{-5}	1.1×10^{-5}
73	10^{-2}		2.1×10^{-5}	2.3×10^{-6}	4.3×10^{-5}	2.6×10^{-6}	1.4×10^{-5}	2.4×10^{-6}
87		10^{-2}	1.3×10^{-6}	2.4×10^{-6}	1.0×10^{-5}	1.3×10^{-6}	4.6×10^{-6}	8.3×10^{-7}
102		10^{-2}	5.5×10^{-6}	1.1×10^{-6}	6.4×10^{-6}	1.2×10^{-6}	3.8×10^{-6}	1.1×10^{-6}
116	10^{-3}		6.6×10^{-6}	5.4×10^{-7}	1.4×10^{-5}	6.1×10^{-7}	3.7×10^{-6}	6.1×10^{-7}
146	10^{-3}		1.3×10^{-6}	2.3×10^{-7}	1.4×10^{-6}	2.5×10^{-7}	8.0×10^{-7}	2.5×10^{-7}
179		10^{-3}	4.5×10^{-7}	4.9×10^{-8}	5.5×10^{-7}	6.8×10^{-8}	3.5×10^{-7}	7.0×10^{-8}
180	10^{-3}		4.2×10^{-7}	5.5×10^{-8}	7.1×10^{-7}	5.9×10^{-8}	1.3×10^{-7}	5.1×10^{-8}
208	10^{-4}		5.8×10^{-7}	4.8×10^{-8}	4.6×10^{-7}	7.0×10^{-8}	2.7×10^{-7}	6.6×10^{-8}
243	10^{-4}		1.7×10^{-7}	1.5×10^{-8}	2.1×10^{-7}	2.0×10^{-8}	6.1×10^{-8}	1.9×10^{-8}
289	10^{-4}		7.7×10^{-8}	4.6×10^{-9}	6.3×10^{-8}	3.1×10^{-9}	6.4×10^{-9}	1.8×10^{-9}
294		10^{-4}	1.2×10^{-7}	1.1×10^{-8}	8.7×10^{-8}	1.2×10^{-8}	3.4×10^{-8}	1.0×10^{-8}
322	10^{-5}		5.2×10^{-8}	2.5×10^{-9}	1.2×10^{-7}	3.1×10^{-9}	1.3×10^{-8}	1.6×10^{-9}
358	10^{-5}		9.1×10^{-9}	6.5×10^{-10}	6.6×10^{-8}	1.4×10^{-9}	6.7×10^{-9}	7.8×10^{-10}
411		10^{-5}	1.5×10^{-9}	1.8×10^{-10}	4.1×10^{-9}	2.0×10^{-10}	4.3×10^{-10}	8.1×10^{-11}
425		10^{-5}	1.5×10^{-9}	2.2×10^{-10}	5.1×10^{-9}	2.5×10^{-10}	5.4×10^{-9}	4.4×10^{-10}
661		10^{-6}	3.3×10^{-9}	1.1×10^{-10}	8.1×10^{-10}	2.8×10^{-11}	1.1×10^{-10}	1.8×10^{-11}
672	10^{-7}		2.9×10^{-9}	8.7×10^{-11}	1.2×10^{-9}	2.9×10^{-11}	1.8×10^{-10}	1.9×10^{-11}
772		10^{-7}	2.0×10^{-11}	1.5×10^{-12}	1.2×10^{-10}	4.2×10^{-12}	1.8×10^{-11}	1.8×10^{-12}
1,052		10^{-9}	4.7×10^{-12}	1.8×10^{-13}	5.5×10^{-11}	7.8×10^{-13}	4.2×10^{-12}	6.1×10^{-13}
1,076	10^{-9}		1.2×10^{-10}	4.6×10^{-12}	6.0×10^{-11}	1.5×10^{-12}	6.6×10^{-13}	8.9×10^{-14}
1,220		10^{-9}	9.6×10^{-13}	4.4×10^{-14}	7.2×10^{-12}	2.4×10^{-13}	5.8×10^{-13}	7.2×10^{-14}

Table 3.9: Maximum relative errors: approximation (3.1) and calculated potentials.

3.6.2 Computational time $O(S_{app}N)$

$O(N)$ behaviour

Table 3.10 lists the running time of Algorithm II, together with the maximum relative error and average relative error of the calculated potential in simulations with tree-like hierarchies of 4 and 5 levels. Here, N denotes the sample size, $T_{AlgorithmII}$ the running time of Algorithm II, T_{Direct} the extrapolated running time of direct particle-to-particle calculation. The approximation used is described in Table 2.2): $S_{app} = 31$, $\epsilon_{rel} = 10^{-1}$, and $b = 2^2$. The error in the calculated potentials is $E_{MaxRel} < 10^{-3}$ and $E_{AvgRel} < 10^{-4}$. In both cases, when the cost of the direct pairwise short-range particle-to-particle potential calculation is relatively small, the curves are virtually linear. When this cost becomes dominant, it can be made negligible by expanding the tree hierarchy to one level higher (see Fig. 3.5). This is in agreement with the cost estimations (3.80), (3.85), and (3.87), suggesting that the proposed EE method is of the order $O(N)$.

$O(S_{app}N)$ behaviour

Figure 3.6 shows running time as a function of approximation size S_{app} . Again, for each sample size, the curve is virtually linear. This demonstrates that the proposed EE method is of the order $O(S_{app}N)$.

3.6.3 Algorithms I, II, and III : comparison

In this subsection, we compare the running time of the three algorithms proposed in the previous section. In spite of the dependence of the computational time on the number of levels in the tree-like hierarchy, with the available exponential expansions generated [23], Algorithm II is the fastest.

Level	N	$T_{AlgorithmII}$	T_{Direct}	E_{MaxRel}	E_{AvgRel}
4	20,000	37.41	(216)	1.3×10^{-4}	3.8×10^{-5}
	30,000	54.72	(486)	1.1×10^{-4}	3.2×10^{-5}
	50,000	95.06	(1,350)	1.0×10^{-4}	3.3×10^{-5}
	100,000	227.71	(5,400)	1.3×10^{-4}	3.2×10^{-5}
	110,000	260.17	(6,534)	1.3×10^{-4}	3.8×10^{-5}
	120,000	293.40	(7,776)	1.4×10^{-4}	3.7×10^{-5}
	130,000	329.29	(9,126)	1.0×10^{-4}	3.6×10^{-5}
	140,000	367.60	(10,584)	1.1×10^{-4}	3.3×10^{-5}
	150,000	405.22	(12,150)	1.4×10^{-4}	3.4×10^{-5}
160,000	446.18	(13,824)	1.4×10^{-4}	3.9×10^{-5}	
5	100,000	247.22	(5,400)	1.1×10^{-4}	3.3×10^{-5}
	110,000	261.36	(6,534)	1.2×10^{-4}	3.7×10^{-5}
	120,000	279.46	(7,776)	1.2×10^{-4}	3.3×10^{-5}
	130,000	295.10	(9,126)	1.2×10^{-4}	3.7×10^{-5}
	140,000	317.81	(10,584)	1.3×10^{-4}	3.5×10^{-5}
	150,000	332.94	(12,150)	1.4×10^{-4}	3.8×10^{-5}
	160,000	350.49	(13,824)	1.3×10^{-4}	3.7×10^{-5}
	200,000	426.23	(21,600)	1.2×10^{-4}	3.1×10^{-5}
	250,000	526.21	(33,750)	9.4×10^{-5}	2.7×10^{-5}
	300,000	631.99	(48,600)	1.2×10^{-4}	3.3×10^{-5}
	350,000	746.35	(66,150)	1.5×10^{-4}	3.5×10^{-5}
400,000	867.70	(86,400)	1.5×10^{-4}	3.9×10^{-5}	

Table 3.10: Running time as function of sample sizes in tree-like hierarchies of different levels.

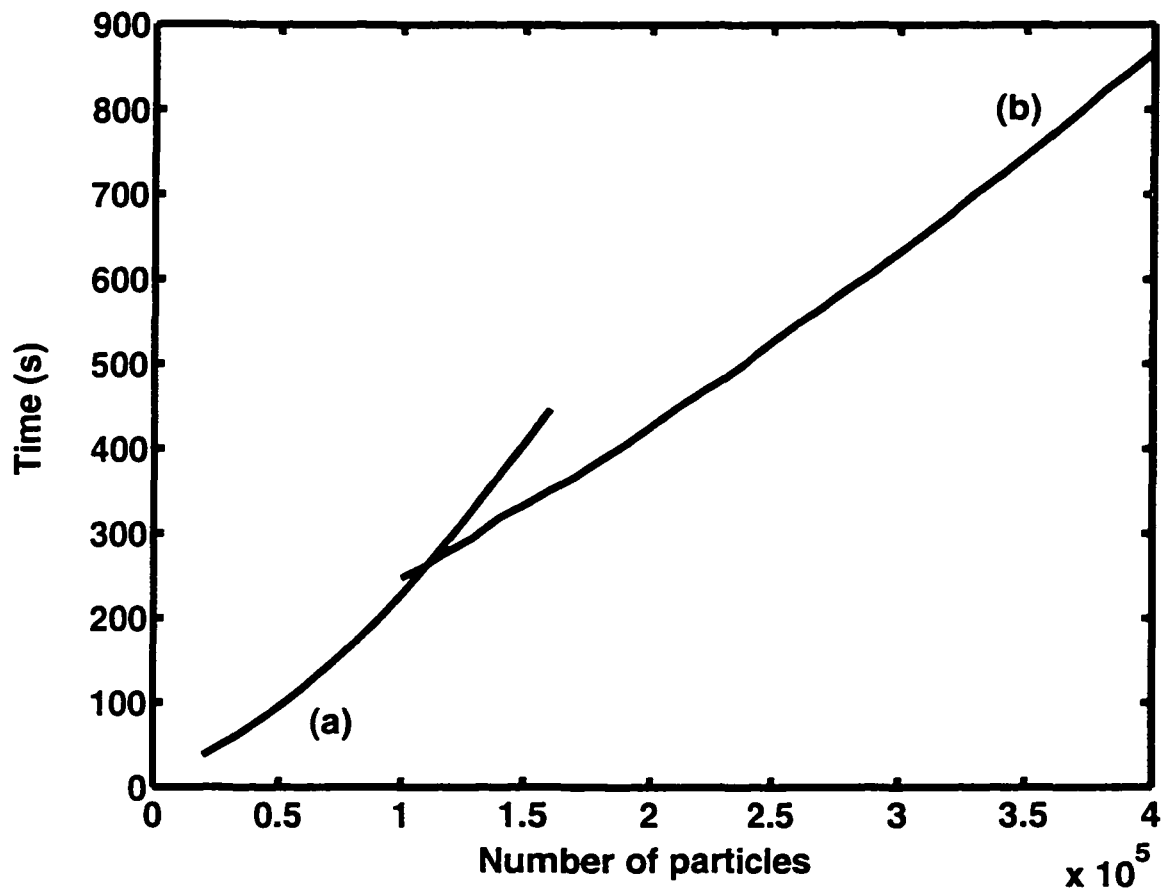


Figure 3.5: Computational time as a function of number of particles, over a range from 0.3×10^5 to 4×10^5 particles in tree-like hierarchy of (a) four levels and (b) five levels.

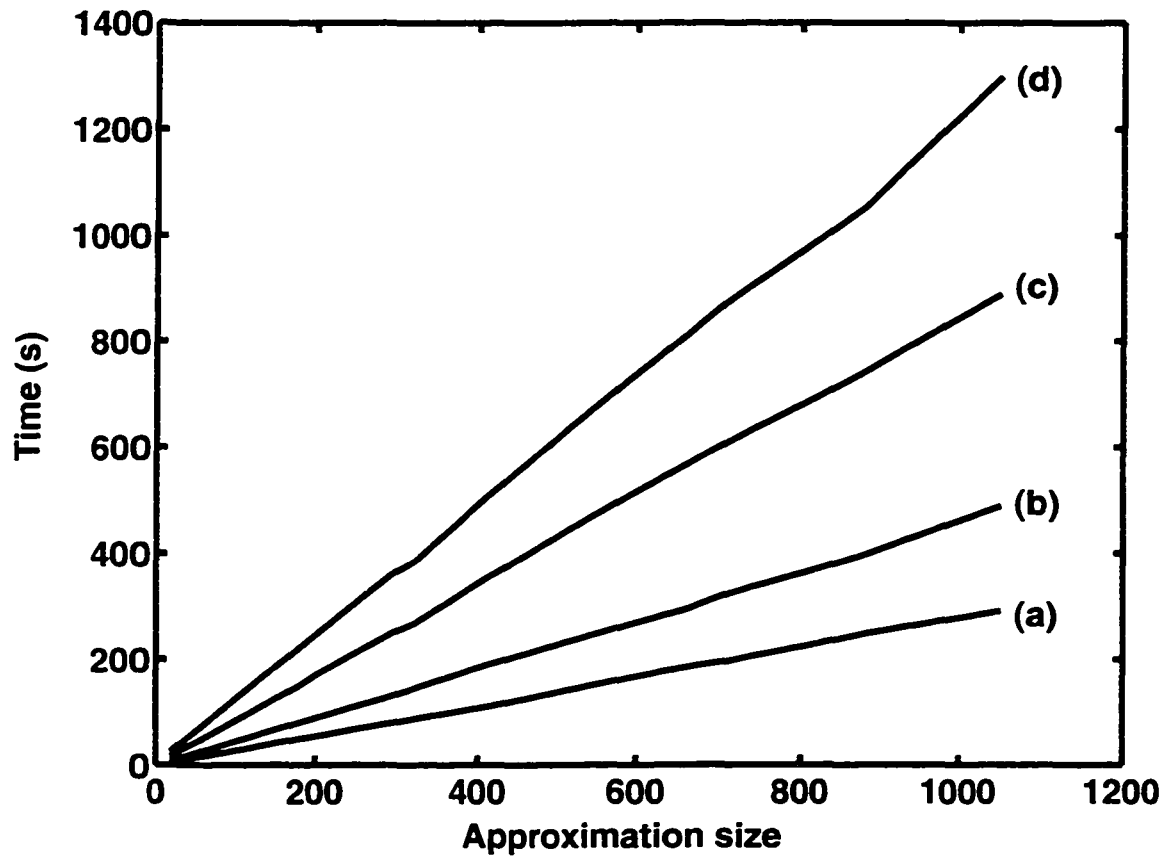


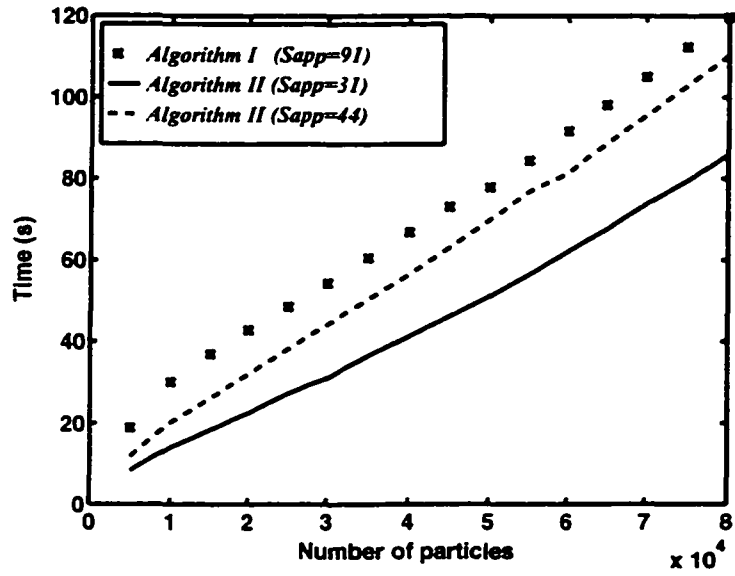
Figure 3.6: Computational time as a function of approximation size (S_{app}) for different sample sizes: (a) 5000, (b) 10000, (c) 25000, and (d) 40000.

Algorithms I and II

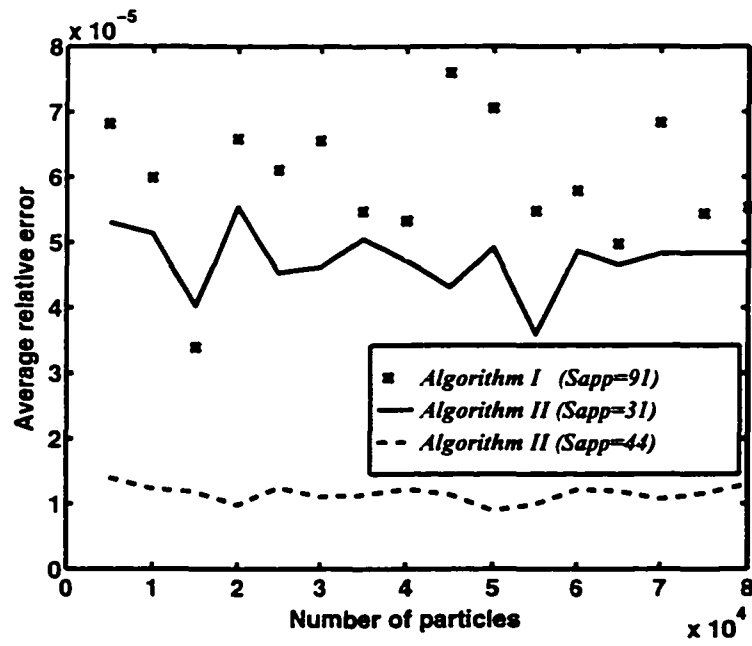
Tables 3.11-3.12 list the running time and average relative error of Algorithms I and II. The sample sizes are 10000, 30000, 50000, and 80000; the number of levels is 4. The approximations used in these algorithms have similar relative error ϵ_{rel} . For each sample size, both the running time and average relative error of Algorithm I are larger than that of Algorithm II. This is illustrated in Figure 3.7. The result here demonstrates that Algorithm I is less efficient than Algorithm II. At first glance, Algorithm I appears to be faster because it has no dependence on the number of level n as it is the case in Algorithm II. However, for a given relative error, the size of the approximation needed for Algorithm I is much larger than that of Algorithm II. The reason for this is that in order to use Algorithm I for tree hierarchies with level 4, we need $b \geq 2^4$, while the requirement for Algorithm II is $b = 2^2$. The larger the value of b , the larger the set D_b is, and hence, the larger the size of the approximation. This trend remains until one has a better method for generating the approximation with large b .

Algorithm	Approximation			Running Time (s)			
	b	ϵ_{rel}	S_{app}	sample size N			
				10000	30000	50000	80000
algorithm I	2^4	10^{-1}	91	30.08	54.44	77.95	119.60
algorithm II	2^2	10^{-1}	44	20.20	44.16	69.85	110.26
algorithm II	2^2	10^{-1}	31	14.02	31.06	51.15	85.81

Table 3.11: Running time (s) for Algorithms I and II.



(a)



(b)

Figure 3.7: Algorithms I and II: (a) running time and (b) average relative error.

Algorithm	Approximation			Average relative error E_{AvgRel}			
	b	ϵ_{rel}	S_{app}	sample size N			
				10000	30000	50000	80000
algorithm I	2^4	10^{-1}	91	5.9×10^{-5}	6.5×10^{-5}	7.0×10^{-5}	5.5×10^{-5}
algorithm II	2^2	10^{-1}	31	5.1×10^{-5}	4.6×10^{-5}	4.9×10^{-5}	4.8×10^{-5}
algorithm II	2^2	10^{-1}	44	1.2×10^{-5}	1.1×10^{-5}	8.9×10^{-6}	1.2×10^{-5}

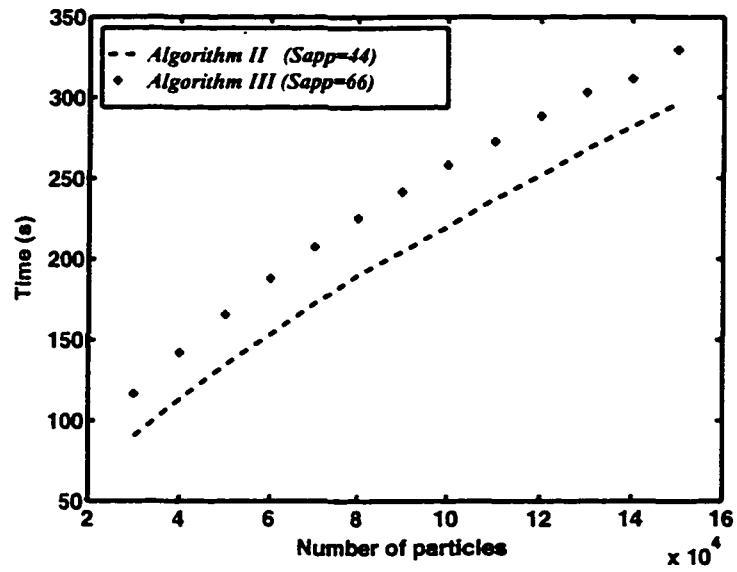
Table 3.12: Average relative error of Algorithms I and II.

Algorithms II and III

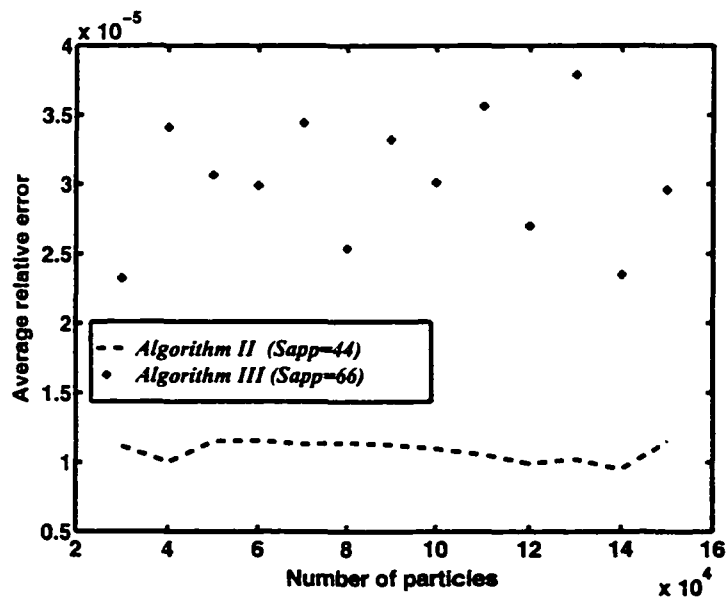
Algorithm III is a generalization based on a compromise between Algorithms I and II. With the same relative error ϵ_{rel} , the size of the approximation needed in Algorithm III is still larger, by a factor 1.5, than Algorithm II. Therefore, in spite of the smaller dependence on number of levels in Algorithm III, Algorithm II is still faster. Tables 3.13-3.14 list the running time and average relative error of Algorithms II and III on sample sizes of 40000, 80000, 120000, and 150000. The number of levels is 5. It can be concluded that with the current available approximation in [23], Algorithm II is the most suitable of the three.

Algorithm	Approximation			Running time (s)			
	b	ϵ_{rel}	S_{app}	sample size N			
				40000	80000	120000	150000
algorithm III	2^3	10^{-1}	66	142.05	225.46	288.64	329.52
algorithm II	2^2	10^{-1}	44	113.41	190.16	251.99	296.42

Table 3.13: Running time (s) for Algorithms II and III.



(a)



(b)

Figure 3.8: Algorithms II and III: (a) running time and (b) average relative error.

Algorithm	Approximation			Average relative error E_{AvgRel}			
	b	ϵ_{rel}	S_{app}	sample size N			
				40000	80000	120000	150000
algorithm III	2^3	10^{-1}	66	3.4×10^{-5}	2.5×10^{-5}	2.7×10^{-5}	3.0×10^{-5}
algorithm II	2^2	10^{-1}	44	1.0×10^{-5}	1.1×10^{-5}	9.9×10^{-6}	1.1×10^{-5}

Table 3.14: Average relative error of Algorithms II and III.

Chapter 4

Solving the Charge Density Problem

In this chapter, we discuss the solution to the charge density problem using the EE method to achieve linear running time and to facilitate any desired degree of accuracy [27, 44]. The discretization of the model equations (1.16)–(1.17), which yields a system of linear equations of some N unknowns, is discussed in §4.1. The linear equations are solved using iterative methods, whose key operation is matrix-vector multiplication. With the EE method, the matrix-vector multiplication can be approximated in computational time of $O(N)$ as opposed to $O(N^2)$. Section §4.2 describes formulae for (3.49) and (3.71) to account for the continuous charge distribution [26]. Preconditioning techniques and stopping criteria in the iterative process are discussed in §§4.3 and 4.4. For capacitance extraction, we have developed a stopping criterion that can reduce the number of iterations significantly without compromising the accuracy of the final solution. In §4.5, we propose a

multi-level multi-precision iterative process, enabling cost-effective generation of a good initial guess, thus improving the overall performance. Numerical results are presented in § 4.6 in terms of two examples, which illustrate the behaviour of convergence rate and the accuracy of the computed charge density and capacitance with respect to the number of panels and the accuracy of the expansion.

4.1 Discretization

In terms of the charge density, the system of equations (1.16)–(1.17) is expressed as:

$$\int_S G(X, Y) \sigma(Y) dS_Y = V_i, \quad X \in S_i, \quad (4.1)$$

$$\sigma(X) + 2\lambda \int_S \frac{\partial G(X, Y)}{\partial n_X} \sigma(Y) dS_Y = 0, \quad X \in S_d, \quad (4.2)$$

These integral equations can be solved numerically by use of a function $\tilde{\sigma}(X)$ that is believed to be close to the true solution $\sigma(X)$. Among the most popular methods to determine $\tilde{\sigma}(X)$ are the Galerkin method and the collocation method (see, for example, [45] and the references therein). We focus on the latter, and in particular, on the constant element collocation method. In this method, the surface areas are meshed into N subareas called panels, $\Pi_1, \Pi_2, \dots, \Pi_N$. On each panel Π_k , the charge density is assumed to be constant and its value σ_k is evaluated at the centroid C_k of the panel: $\sigma_k = \sigma(C_k)$. Here, the main task is to find the charge density σ_k , $k = 1..N$ at the centroids of these panels. The discretization of eqs.

(4.1)–(4.2) leads to a system of linear equations:

$$A\sigma = b, \quad (4.3)$$

where A is an $N \times N$ matrix and b is a column matrix of size N . Their entries are defined below,

$$A(k, j) = \begin{cases} \int_{\Pi_j} G(C_k, Y) dS_Y & \text{if } \Pi_k \subset S_c \\ \begin{cases} 1 & \text{if } k \equiv j \\ 2\lambda \int_{\Pi_j} \frac{\partial G(C_k, Y) dS_Y}{\partial n_{C_k}} & \text{if } k \not\equiv j \end{cases} & \text{if } \Pi_k \subset S_d \end{cases} \quad (4.4)$$

$$b(k) = \begin{cases} V_i & \text{if } \Pi_k \subset S_i \\ 0 & \text{otherwise.} \end{cases} \quad (4.5)$$

If N is small, the system of linear equations (4.3) can be solved using a direct scheme such as Gauss elimination. In practice, however, N is very large, and it becomes prohibitively expensive to use a direct method since the computational time for the solution is $O(N^3)$ and the memory requirement for storing the entries of the matrix is $O(N^2)$. In this case, it is necessary to employ iterative schemes [11], such as GMRES [46]. The main operation in the iterative methods is matrix-vector multiplication. Here, we need to take into account:

- Both accuracy and efficiency (computational time and memory requirement).
A direct multiplication would cost $O(N^2)$ in computational time and $O(N^2)$ in memory requirement.

- Convergence rate.
- Stopping criteria.
- Initial guess.

The above issues are discussed in the next five sections. In the context of our problem, if $x = (x_k)$ is viewed as charge density, then the matrix-vector multiplication Ax can be accelerated using the MP method or the EE method. Here, we focus on the latter. The rate of convergence can be improved using (i) preconditioning, and/or (ii) a high accuracy for the approximation of the potential and/or (iii) reliable initial guess. Alternatively, we can employ a different formulation that yields less ill-conditioned discretization matrix (to be discussed in the next chapter). The stopping criteria needs to take into consideration not only the change in the residual but also the change in the computed charge or capacitance. Additionally, we also need to take into account the accuracy of the approximation on the potential. The initial guess should be as close to the true solution as possible so that the number of iterations can be reduced. Here, we can use solution from coarser grids or from less accurate potential approximations as initial guess for more accurate solution on the finer grid.

4.2 Matrix-vector multiplication

Let $y = (y_k)$ represent Ax , i.e., $y_k = \sum_{j=1}^N A(k,j)x_j$. For a given $k = 1..N$, we examine the evaluation of y_k . There are two cases, depending whether the panel Π_k is on a conductor surface or on an interface. For the first case, according to eq.

(4.4).

$$y_k = \sum_{j=1}^N \int_{\Pi_j} G(c_k, y) x_j dS_y, \quad (4.6)$$

which is the potential at c_k due to the charge density x on S . This suggests the use of exponential expansion for approximating y_k . For the second case,

$$y_k = x_k + 2\lambda \sum_{j=1}^N \int_{\Pi_j} \frac{\partial G(c_k, y)}{\partial n_{c_k}} x_j dS_y \quad (4.7)$$

$$= x_k + 2\lambda \mathbf{H} \cdot \mathbf{n} \quad (4.8)$$

where the vector \mathbf{H} is defined as:

$$\mathbf{H}(c_k) = \sum_{j=1}^N \int_{\Pi_j} \nabla_{c_k} G(c_k, y) x_j dS_y. \quad (4.9)$$

Here, the $\nabla_{c_k} G(c_k, y)$ denotes the gradient of the Green's function $G(x, y)$ at $x = c_k$. From eq. (4.8), the evaluation of y_k is reduced to that of \mathbf{H} , which is the gradient of the potential, and hence can also be approximated using exponential expansion.

The evaluation of y_k can be performed in two steps: one for the near-field interaction and the other for the far-field interaction. For a given k , the set of indices $j = 1..N$ is partitioned into two subsets J_D^k and J_E^k . The subset J_D^k consists of all indices j such that the panel Π_j is a neighbor of panel Π_k , that is, they belong to leaf cubes that have at least one vertex in common. The subset J_E^k consists of the remaining indices j . We can now describe y_k as a sum of two terms.

$$y_k = y_{k,D} + y_{k,E}, \quad (4.10)$$

where

$$y_{k,D} = \sum_{j \in J_D^k} A(k, j) x_j \quad (4.11)$$

$$y_{k,E} = \sum_{j \in J_E^k} A(k, j) x_j. \quad (4.12)$$

The first term $y_{k,D}$, which is referred to as near-field or direct term, is calculated exactly as given in eq. (4.11). However, the second term $y_{k,E}$, which is referred to as far-field or indirect term, is approximated using exponential expansion through a sequence of translations: Q2S, S2S, S2T, and T2P. In the next two subsections, we discuss how to evaluate these terms.

4.2.1 Charge to source (Q2S) translation

For charge distributions that are discrete (particles) or continuous (panels), the translations S2S, S2T, and T2P are identical. In the case of the latter, the Q2S needs an extra step to replace all the charges on each panel with an equivalent discrete charge at the centroid. This is illustrated in Fig. 4.1. More specifically, in terms of exponential distance E , we have

$$E(QS) = E(QG)E(GS), \quad (4.13)$$

and therefore,

$$\int_{\Delta ABC} E(QS) dA_Q = \left(\int_{\Delta ABC} E(QG) dA_Q \right) E(GS) \quad (4.14)$$

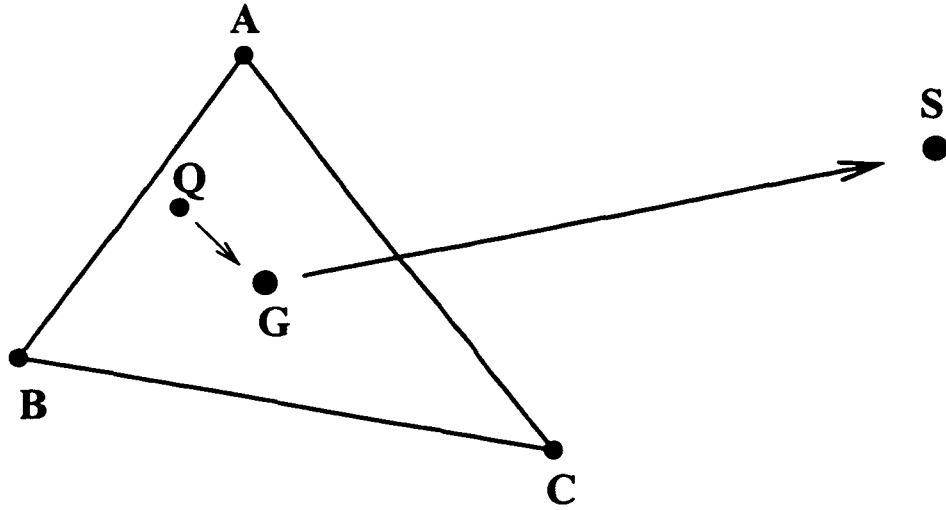


Figure 4.1: Q2S translation for continuous charge distribution on a triangle ΔABC to a source center S : (i) all charge Q is translated to the centroid G and (ii) G is translated to S .

The extra step mentioned above is the computation of the surface integral:

$$I_{\Delta ABC} = \int_{\Delta ABC} E(\mathbf{Q}\mathbf{G})dA_Q. \quad (4.15)$$

In general, we replace the triangle with a planar polygon Π :

$$I_{\Pi} = \int_{\Pi} E(\mathbf{Q}\mathbf{G})dA_Q. \quad (4.16)$$

The above surface integral represents the clustering of the charges on panel Π to its centroid and it is also referred to as exponential distance of a panel. In the rest of this section, we discuss how to express it in terms of the exponential distance at the vertices.

Local coordinate

We use a two-dimensional local coordinate system to describe points on the panel Π . The surface integral is then transformed into line integrals using Green's theorem. Following this, we establish a formulation for the line integral. We choose an orthonormal basis $\mathcal{B} = (\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z)$ as the local coordinate system for the plane π supporting the panel Π . Here, $\|\mathbf{e}_x\| = \|\mathbf{e}_y\| = \|\mathbf{e}_z\| = 1$ and $\mathbf{e}_x \perp \mathbf{e}_y \perp \mathbf{e}_z$. All vectors $QG(x, y)$ with $Q \in \Pi$ can now be described using this local coordinate system: $QG = x\mathbf{e}_x + y\mathbf{e}_y$.

From surface integral to line integral

Let $V_1..V_m$ denote vertices of panel Π (the vertex V_{m+1} is understood as V_1). With respect to the functions g 's in Definition 3.1.1, we need to consider two cases:

Case 1: $g(\mathbf{e}_y) \neq 0$, and

Case 2: $g(\mathbf{e}_x) \neq 0$.

Note that \mathbf{e}_x and \mathbf{e}_y are linearly independent and it can be verified that $g(x, y, z)$ is a linear function with 1-dimensional kernel. Therefore, it is impossible that both $g(\mathbf{e}_x)$ and $g(\mathbf{e}_y)$ are zero. Hence either Case 1 or Case 2 must hold. We have:

$$e^{xg(\mathbf{e}_x)+yg(\mathbf{e}_y)} = \begin{cases} \frac{\partial}{\partial y} \left(\frac{1}{g(\mathbf{e}_y)} e^{xg(\mathbf{e}_x)+yg(\mathbf{e}_y)} \right) & \text{Case 1} \\ \frac{\partial}{\partial x} \left(\frac{1}{g(\mathbf{e}_x)} e^{xg(\mathbf{e}_x)+yg(\mathbf{e}_y)} \right) & \text{Case 2.} \end{cases} \quad (4.17)$$

Applying the Green's theorem, we can express the surface integral (4.16) as a sum of line integrals along the edges of Π

$$I_{\Pi} = \begin{cases} \frac{-1}{g(\mathbf{e}_y)} \oint e^{xg(\mathbf{e}_x)+yg(\mathbf{e}_y)} dx & \text{Case 1} \\ \frac{1}{g(\mathbf{e}_x)} \oint e^{xg(\mathbf{e}_x)+yg(\mathbf{e}_y)} dy & \text{Case 2} \end{cases} \quad (4.18)$$

$$= \begin{cases} \sum_{i=1}^m \frac{-1}{g(\mathbf{e}_y)} \oint_{V_i V_{i+1}} e^{xg(\mathbf{e}_x)+yg(\mathbf{e}_y)} \frac{dx}{dt} dt & \text{Case 1} \\ \sum_{i=1}^m \frac{1}{g(\mathbf{e}_x)} \oint_{V_i V_{i+1}} e^{xg(\mathbf{e}_x)+yg(\mathbf{e}_y)} \frac{dy}{dt} dt & \text{Case 2.} \end{cases} \quad (4.19)$$

Our focus now lies in calculating the line integral on a segment from a point U to another point V :

$$I_{UV} \triangleq \int_{UV} e^{xg(\mathbf{e}_x)+yg(\mathbf{e}_y)} dt. \quad (4.20)$$

The coordinates of vectors QG where Q is on the segment from U to V can be described as:

$$x = x_u - x_{uv}t \quad \text{and} \quad y = y_u - y_{uv}t. \quad (4.21)$$

Here, (x_u, y_u) , (x_v, y_v) , (x_{uv}, y_{uv}) denote the coordinates of the vectors UG , VG , and UV associated with the local coordinate system \mathcal{B} , respectively. The values of t are in the interval $[0, 1]$. The derivatives of x and y with regard to t are:

$$\frac{dx}{dt} = -x_{uv} \quad \text{and} \quad \frac{dy}{dt} = -y_{uv} \quad (4.22)$$

The integral (4.19) can now be expressed in terms of the line integral (4.20) as:

$$I_{\Pi} = \begin{cases} \sum_{i=1}^m \frac{x_{v_i v_{i+1}}}{g(\mathbf{e}_y)} I_{V_i V_{i+1}} & \text{Case 1} \\ \sum_{i=1}^m \frac{-y_{v_i v_{i+1}}}{g(\mathbf{e}_z)} I_{V_i V_{i+1}} & \text{Case 2.} \end{cases} \quad (4.23)$$

With the parametric equation for the segment UV , the line integral (4.20) can be expressed as:

$$I_{UV} = e^{x_{u0}g(\mathbf{e}_z) + y_{u0}g(\mathbf{e}_y)} \int_0^1 e^{-(x_{uv}g(\mathbf{e}_z) + y_{uv}g(\mathbf{e}_y))t} dt \quad (4.24)$$

There are two cases.

$$\text{Case 3: } x_{uv}g(\mathbf{e}_z) + y_{uv}g(\mathbf{e}_y) = 0,$$

$$\text{Case 4: } x_{uv}g(\mathbf{e}_z) + y_{uv}g(\mathbf{e}_y) \neq 0.$$

The value of the line integral (4.24) in each case is

$$\begin{aligned} I_{UV} &= \begin{cases} e^{x_{u0}g(\mathbf{e}_z) + y_{u0}g(\mathbf{e}_y)} & \text{Case 3} \\ \frac{-1}{x_{uv}g(\mathbf{e}_z) + y_{uv}g(\mathbf{e}_y)} (e^{x_{u0}g(\mathbf{e}_z) + y_{u0}g(\mathbf{e}_y)} - e^{x_{v0}g(\mathbf{e}_z) + y_{v0}g(\mathbf{e}_y)}) & \text{Case 4} \end{cases} \quad (4.25) \\ &= \begin{cases} e^{g(\mathbf{UG})} & \text{Case 3} \\ \frac{-1}{g(\mathbf{UV})} (e^{g(\mathbf{VG})} - e^{g(\mathbf{UG})}) & \text{Case 4} \end{cases} \quad (4.26) \\ &= \begin{cases} E(\mathbf{UG}) & \text{Case 3} \\ \frac{-1}{g(\mathbf{UV})} (E(\mathbf{VG}) - E(\mathbf{UG})) & \text{Case 4} \end{cases} \quad (4.27) \end{aligned}$$

With the expressions (4.23) and (4.27), we have established a formulation for calculating the surface integral (4.16). The exponential distance of a panel can be expressed in terms of exponential distance of vectors connecting its vertices and its centroid. Due to the restriction on the set D_b , which is defined in (3.5), scaling is required for points that are not in D_b . In the next subsection, we show that only a slight modification is needed to handle the scaling.

Scaling

For a scaling factor $s > 0$, we replace $g(\mathbf{e}_x)$ and $g(\mathbf{e}_y)$ with $g(s \times \mathbf{e}_x) = s \times g(\mathbf{e}_x)$ and $g(s \times \mathbf{e}_y) = s \times g(\mathbf{e}_y)$. The integrals in (4.23) and (4.27) can now be described as:

$$I_{s \times \Pi} = \int_{\Pi} E(s \mathbf{Q} \mathbf{G}) dA_Q = \begin{cases} \sum_{i=1}^m \frac{x_{v_i v_{i+1}}}{s \times g(\mathbf{e}_y)} I_{s \times v_i v_{i+1}} & \text{Case 1} \\ \sum_{i=1}^m \frac{-y_{v_i v_{i+1}}}{s \times g(\mathbf{e}_x)} I_{s \times v_i v_{i+1}} & \text{Case 2} \end{cases} \quad (4.28)$$

$$I_{s \times UV} = \begin{cases} E(s \times U \mathbf{G}) & \text{Case 3} \\ \frac{-1}{s \times g(\mathbf{e}_y)} (E(s \times V \mathbf{G}) - E(s \times U \mathbf{G})) & \text{Case 4.} \end{cases} \quad (4.29)$$

We have obtained the analytical expression for clustering charges on panel to its centroid via the surface integral (4.16). We have some remarks on the calculation of the surface integral (4.16).

- The choice of the centroid as the source center is to ensure that neither overflow nor underflow occurs.
- The extra cost for dealing with piecewise charge distribution is $O(N^*)$. Here, N^* denotes the total number of vertices of all the panels.

- The exponential distance of a panel depends on the vectors connecting the vertices to the centroid; therefore if two panels are equal, in the sense that one can be obtained from the other through a geometrical translation, then their exponential distances are the same. Taking advantage of this property, we can classify the panels into identical panels and calculate the exponential distance for each class of identical panels. In planar structures, many panels are equal in the above sense, and therefore, the cost in computational time of calculating the surface integral for all panels can be significantly reduced.
- No memory allocation is needed for the exponential distance at each vertex or each edge since the Q2S can be performed for each panel prior to any S2S and S2T. To improve the speed for S2T we may need the memory allocation for the exponential distance at the centroid; this is, however, optional.
- There is no extra cost for calculating the gradient of the potential since it is obtained at the end of the T2P translation. The associated overhead is relatively small, therefore, the computation of the electric field can be considered as a by-product of the computational procedure for the potential. However, for the near-field term $y_{k,D}$, it is still calculated explicitly and exactly as in (4.11). This will be discussed in the next subsection

Integration of Green's function and its gradient

We shall now discuss the calculation of the near-field term $y_{k,D}$ in (4.11). It involves the surface integrals over a panel Π of the Green's function $\frac{1}{r} = \frac{1}{\|QP\|}$ and its partial

derivatives. Here, P is a target point and Q a source point on Π .

$$I_r(P, \Pi) = \iint_{\Pi} \frac{1}{r} dA \quad (4.30)$$

$$I_{xg}(P, \Pi) = \iint_{\Pi} \frac{\partial}{\partial x} \left(\frac{1}{r} \right) dA \quad (4.31)$$

$$I_{yg}(P, \Pi) = \iint_{\Pi} \frac{\partial}{\partial y} \left(\frac{1}{r} \right) dA \quad (4.32)$$

$$I_{zg}(P, \Pi) = \iint_{\Pi} \frac{\partial}{\partial z} \left(\frac{1}{r} \right) dA \quad (4.33)$$

Discussion on evaluation of the integrals (4.30)–(4.33) can be found in [47–49]. However, we will introduce the notion of *signed area* to avoid possible errors arising from intuitive analysis. We demonstrate that the integrals (4.31)–(4.33) can be efficiently obtained as by-products of the computational process for evaluating (4.30); the overhead involved is small.

Signed area

The motivation behind the concept of signed area stems from the following problem. Given a triangle ΔABC on a plane π and a point G on π , we need to find an expression of the area of ΔABC in terms of the areas of ΔGAB , ΔGBC , and ΔGCA . This is an analogy with, and in fact a generalization to, the problem for signed distance. For a segment AB on a line l and a point G on l , the signed distance \overline{AB} can be expressed in terms of the signed distance \overline{GB} and \overline{GA} as $\overline{AB} = \overline{GB} - \overline{GA}$.

Definition 4.2.1 *For a given triangle ΔABC on a plane π , let $A(x_A, y_A)$, $B(x_B, y_B)$, and $C(x_C, y_C)$ denote some local coordinates of A , B , and C associated with π . The*

signed area of triangle ΔABC in the order of vertices A , B , and C is defined as:

$$\overline{ABC} \triangleq \frac{1}{2} \begin{vmatrix} x_B - x_A & y_B - y_A \\ x_C - x_A & y_C - y_A \end{vmatrix} \quad (4.34)$$

$$= \frac{1}{2} [(x_B - x_A)(y_C - y_A) - (y_B - y_A)(x_C - x_A)] \quad (4.35)$$

Some properties of the signed area is summarized in the following proposition.

Proposition 4.2.1

$$(i) \text{ area}(\Delta ABC) = \text{sign}(\overline{ABC})\overline{ABC}$$

$$(ii) \overline{ABC} = \text{sign}(\overline{ABC}) \text{ area}(\Delta ABC)$$

$$(iii) \overline{ABC} = \overline{BCA} = \overline{CAB}$$

$$(iv) \overline{ACB} = -\overline{ABC}$$

$$(v) \overline{ABC} = \overline{GAB} + \overline{GBC} + \overline{GCA}$$

Proof. For (i) and (ii), if we extend the two-dimensional local coordinate system to a three-dimensional coordinate system, preserving the local coordinates, the signed area \overline{ABC} is the third component of the cross product $\mathbf{AB} \times \mathbf{AC}$. The properties (iii) and (iv) follow directly from (4.35). For (v), by expanding both sides and performing the subtraction, we get 0. ■

From property (v) in Proposition 4.2.1, there are four important results.

1. It establishes the consistency and hence validity of Definition 4.2.1. The reason is that in the definition, the value of the signed area \overline{ABC} is defined

in terms of local coordinates. What if one has other local coordinates, i.e., different (x_A, y_A) , (x_B, y_B) , and (x_C, y_C) ? Thanks to (v) , the value of \overline{ABC} is independent of the choice of local coordinates. To verify this, we use (v) with G being the origin of the local coordinate system.

2. It solves the problem we have posed on how to express the area of ΔABC in terms of the area of the triangles ΔGAB , ΔGBC , and ΔGCA . We have

$$\begin{aligned} \text{area}(\Delta ABC) &= \text{sign}(\overline{ABC})(\text{sign}(\overline{GAB}) \text{area}(\Delta GAB) \\ &\quad + \text{sign}(\overline{GBC}) \text{area}(\Delta GBC) \\ &\quad + \text{sign}(\overline{GCA}) \text{area}(\Delta GCA)). \end{aligned} \quad (4.36)$$

3. We can extend the notion of signed area to polygons. If $\Pi = V_1..V_m V_{m+1}$, where $V_{m+1} \equiv V_1$, is a planar polygon on a plane, then the signed area of Π can be defined as:

$$\overline{V_1..V_m V_{m+1}} \triangleq \sum_{i=1}^m \overline{GV_i V_{i+1}} \quad (4.37)$$

The value of $\overline{V_1..V_m V_{m+1}}$ is independent of the choice of G on the plane supporting Π .

4. We can further have the notion of signed surface integral over a planar polygon $\Pi = V_1..V_m V_{m+1}$.

$$\overline{\iint_{\Pi} f dA} \triangleq \text{sign}(\overline{V_1..V_m V_{m+1}}) \iint_{\Pi} f dA. \quad (4.38)$$

With the notion of signed surface integral, we have

$$\overline{\iint_{\Pi} f dA} = \sum_{i=1}^m \overline{\iint_{\Delta GV_i V_{i+1}} f dA} \quad (4.39)$$

Here, we assume that the integrand f is defined and integrable on the triangles $\Delta GV_i V_{i+1}$.

Surface integration of Green's function $\frac{1}{r}$

Having established a mechanism to algebraically decompose the planar polygonal area Π into signed triangular areas, we shall now discuss how to evaluate the surface integrals of the Green's function and its gradient over a triangle ΔHUV , where H is the projection of the target point P on the plane supporting the panel Π and UV is an edge of Π . Corresponding to (4.30), we have

$$I_r(\Delta HUV) = \iint_{\Delta HUV} \frac{1}{r} dA. \quad (4.40)$$

We use the same local coordinate system $\mathcal{B} = (\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z)$ as in the previous subsection. Let Q be a point on ΔHUV . The triangle ΔHUV can be mapped into a square $[0, 1] \times [0, 1]$ with a transformation T defined as (see Fig. 4.2),

$$\mathbf{HQ} = u(\mathbf{HU} + v\mathbf{UV}) \quad (4.41)$$

$$T(\mathbf{HQ}) = (u, v). \quad (4.42)$$

Here, $u, v \in [0, 1]$. The geometrical meaning of (u, v) can be explained as follows. We extend HQ to intersect UV at M and draw QN parallel to UV . We have

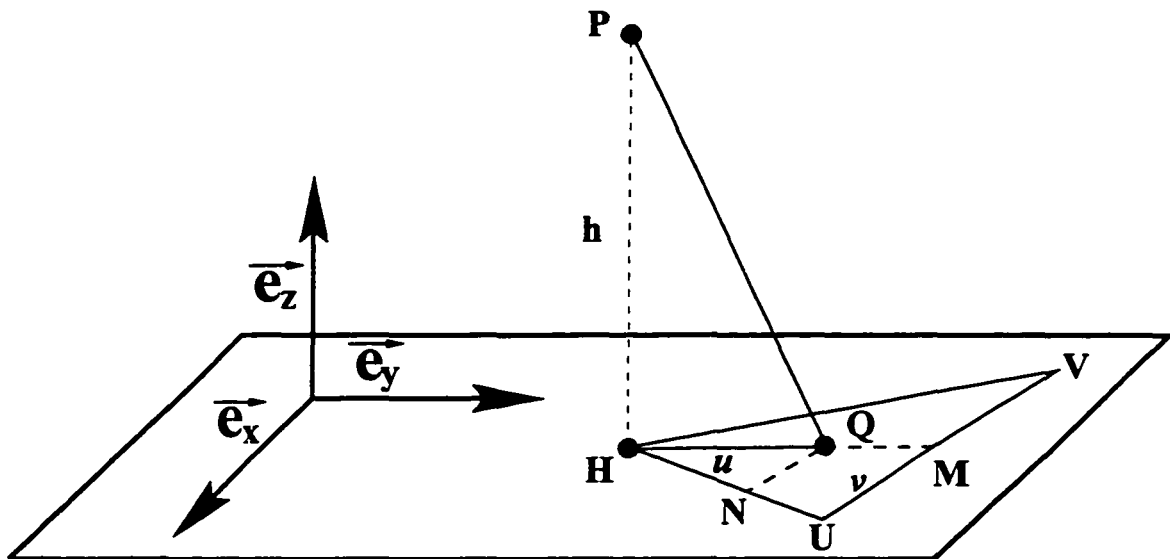


Figure 4.2: Local coordinate system $\mathcal{B} = (\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z)$ associated with panel Π and transformation T of a triangle (ΔHQU) into a square $([0, 1] \times [0, 1])$: $T(\mathbf{HQ}) = T(u(\mathbf{HU} + v\mathbf{UV})) = (u, v)$.

$\mathbf{HQ} = u\mathbf{HM}$. $\mathbf{HM} = \mathbf{HU} + \mathbf{UM}$, and $\mathbf{UM} = v\mathbf{UV}$, thus leading to (4.41).

The Jacobian of the mapping is:

$$J(T) = \frac{\partial \mathbf{HQ}}{\partial (u, v)} = \begin{pmatrix} x_{\mathbf{HU}} + x_{\mathbf{UV}}v & x_{\mathbf{UV}}u \\ y_{\mathbf{HU}} + y_{\mathbf{UV}}v & y_{\mathbf{UV}}u \end{pmatrix} \quad (4.43)$$

The absolute value of the determinant of the matrix is $|\det(J(T))| = 2 \text{area}(\Delta HUV)u =$

δu . Here δ denotes $2 \text{ area}(\Delta HUV)$. With this transformation, we have

$$I_r(\Delta HUV) = \delta \int_0^1 \int_0^1 \frac{u \, du \, dv}{\sqrt{QP^2}} \quad (4.44)$$

$$= \delta \int_0^1 \int_0^1 \frac{u \, du \, dv}{\sqrt{HQ^2 + h^2}} \quad (4.45)$$

$$= \delta \int_0^1 \int_0^1 \frac{u \, du \, dv}{\sqrt{q(v)u^2 + h^2}}, \quad (4.46)$$

where $q(v) = (HU + vUV)^2$. The above double integral can be reduced to three single integrals, which can be evaluated analytically.

$$I_r(\Delta HUV) = \delta \int_0^1 \left[\int_0^1 \frac{d(q(v)u^2 + h)}{2q(v)\sqrt{q(v)u^2 + h^2}} \right] dv \quad (4.47)$$

$$= \delta \int_0^1 \left[\frac{\sqrt{q(v)u^2 + h^2}}{q(v)} \Big|_0^1 \right] dv \quad (4.48)$$

$$= \delta \left[- \int_0^1 \frac{|h|}{q(v)} dv + \int_0^1 \frac{\sqrt{q(v) + h^2}}{q(v)} dv \right] \quad (4.49)$$

$$= \delta \left[- \int_0^1 \frac{|h| dv}{q(v)} + \int_0^1 \frac{h^2 dv}{q(v)\sqrt{q(v) + h^2}} + \int_0^1 \frac{dv}{\sqrt{q(v) + h^2}} \right] \quad (4.50)$$

$$= |h|(I_2 - I_1) + \delta I_3, \quad (4.51)$$

where

$$I_1 = \delta \int_0^1 \frac{dv}{q(v)} \quad (4.52)$$

$$I_2 = \delta \int_0^1 \frac{|h| dv}{q(v)\sqrt{q(v) + h^2}}. \quad (4.53)$$

$$I_3 = \int_0^1 \frac{dv}{\sqrt{q(v) + h^2}} \quad (4.54)$$

We now describe the formulae for these integrals [50]. We have

$$q(v) = (\mathbf{HU} + v\mathbf{UV})^2 = \mathbf{UV}^2 v^2 + 2\mathbf{HU} \cdot \mathbf{UV} v + \mathbf{HU}^2. \quad (4.55)$$

For I_1 , the corresponding indefinite integral is of the form

$$I_1^i = \int \frac{dx}{ax^2 + bx + c} \quad (4.56)$$

$$= \frac{2}{\sqrt{\Delta_{12}}} \arctan \frac{2ax + b}{\sqrt{\Delta_{12}}}. \quad (4.57)$$

Here, $a = \mathbf{UV}^2 > 0$, $b = 2\mathbf{HU} \cdot \mathbf{UV}$, and $c = \mathbf{HU}^2$. We have $\Delta_{12} = -b^2 + 4ac = 4[(\|\mathbf{UV}\| \|\mathbf{HU}\|)^2 - (\mathbf{HU} \cdot \mathbf{UV})^2] = 4\delta^2 > 0$. Therefore,

$$I_1 = \arctan \left(\frac{\mathbf{UV} \cdot \mathbf{HV}}{\delta} \right) - \arctan \left(\frac{\mathbf{UV} \cdot \mathbf{HU}}{\delta} \right). \quad (4.58)$$

For I_2 , the corresponding indefinite integral is

$$I_2^i = \int \frac{dx}{(ax^2 + bx + c)\sqrt{ax^2 + bx + c + h^2}} \quad (4.59)$$

$$= \frac{2}{|h|\sqrt{\Delta_{12}}} \arctan \left(\frac{|h|}{\sqrt{\Delta_{12}}} \frac{2ax + b}{\sqrt{ax^2 + bx + c + h^2}} \right). \quad (4.60)$$

Here, $a = \mathbf{UV}^2 > 0$, $b = 2\mathbf{HU} \cdot \mathbf{UV}$ and $c = \mathbf{HU}^2$. We have $\Delta_{12} = -b^2 + 4ac = 4[(\|\mathbf{UV}\| \|\mathbf{HU}\|)^2 - (\mathbf{HU} \cdot \mathbf{UV})^2] = 4\delta^2 > 0$. Therefore,

$$I_2 = \arctan \left(\frac{|h|}{\delta} \frac{\mathbf{UV} \cdot \mathbf{HV}}{\sqrt{\mathbf{HV}^2 + h^2}} \right) - \arctan \left(\frac{|h|}{\delta} \frac{\mathbf{UV} \cdot \mathbf{HU}}{\sqrt{\mathbf{HU}^2 + h^2}} \right). \quad (4.61)$$

For I_3 , the corresponding indefinite integral is

$$I_3^i = \int \frac{dx}{\sqrt{ax^2 + bx + c + h^2}}. \quad (4.62)$$

Here, $a = UV^2 > 0$, $b = 2HU \cdot UV$ and $c = HU|^2$. We have $\Delta_3 = -b^2 + 4a(c + h^2) = 4[(\|UV\|\|HU\|)^2 + h^2 - (HU \cdot UV)^2] \geq 4\delta^2 \geq 0$. The formula for the indefinite integral I_3^i is

$$I_3^i = \begin{cases} \frac{1}{\sqrt{a}} \ln \left| ax + \frac{b}{2} \right| & \text{if } \Delta_3 = 0 \\ \frac{1}{\sqrt{a}} \ln \left| \left(ax + \frac{b}{2} \right) + \sqrt{a} \sqrt{ax^2 + bx + c + h^2} \right| & \text{if } \Delta_3 > 0 \end{cases} \quad (4.63)$$

Therefore,

$$I_3 = \begin{cases} \frac{1}{\|UV\|} \ln \left| \frac{UV \cdot HV}{UV \cdot HU} \right| & \text{if } \Delta_3 = 0 \\ \frac{1}{\|UV\|} \ln \left| \frac{UV \cdot HV + \|UV\| \sqrt{HV^2 + h^2}}{UV \cdot HU + \|UV\| \sqrt{HU^2 + h^2}} \right| & \text{if } \Delta_3 > 0 \end{cases} \quad (4.64)$$

The formulae (4.58), (4.61), and (4.64) facilitate the evaluation of the surface integral of the Green's function $\frac{1}{r}$ (4.40) using (4.51). They can also be used for the evaluation of the surface integrals of the partial derivatives of $\frac{1}{r}$ (to be discussed in the next subsection). We now describe some special cases.

- $h = 0$: the surface integral (4.51) is reduced to δI_3 . Therefore, we need only to evaluate I_3 .
- $\text{area}(\Delta HUV = 0)$: the surface integral (4.51) is zero. Note that in this case, it is possible that $I_3 \neq 0$.

Surface integration of partial derivatives of Green's function

To determine the surface integrals of the partial derivatives, we first express the partial derivatives using local coordinates, evaluate the integrals using the local coordinates, and then switch back to the global coordinate system for the final result.

Partial derivatives in local coordinates

Let (x_g, y_g, z_g) be the global coordinates of vector QP . Let (x, y) be the local coordinate of QH . We have

$$\begin{pmatrix} x \\ y \\ h \end{pmatrix} = B^{-1} \begin{pmatrix} x_g \\ y_g \\ z_g \end{pmatrix}. \quad (4.65)$$

The partial derivatives of $\frac{1}{r} = \frac{1}{\|QP\|} = \frac{1}{\sqrt{QG^2 + h^2}}$ are:

$$\frac{\partial}{\partial x_g} \left(\frac{1}{r} \right) = \frac{-x_g}{r^3} \quad (4.66)$$

$$\frac{\partial}{\partial y_g} \left(\frac{1}{r} \right) = \frac{-y_g}{r^3} \quad (4.67)$$

$$\frac{\partial}{\partial z_g} \left(\frac{1}{r} \right) = \frac{-z_g}{r^3}. \quad (4.68)$$

With (4.65), these partial derivatives can be expressed in terms of the local coordinates as

$$\begin{pmatrix} \frac{\partial}{\partial x_g} \left(\frac{1}{r} \right) \\ \frac{\partial}{\partial y_g} \left(\frac{1}{r} \right) \\ \frac{\partial}{\partial z_g} \left(\frac{1}{r} \right) \end{pmatrix} = B \begin{pmatrix} \frac{-x}{r^3} \\ \frac{-y}{r^3} \\ \frac{-h}{r^3} \end{pmatrix}. \quad (4.69)$$

We have the same linear relationship between the two corresponding integrals,

$$\begin{pmatrix} I_{xg}(\Pi) \\ I_{yg}(\Pi) \\ I_{zg}(\Pi) \end{pmatrix} = B \begin{pmatrix} I_x(\Pi) \\ I_y(\Pi) \\ I_h(\Pi) \end{pmatrix}, \quad (4.70)$$

where $I_{xg}(\Pi)$, $I_{yg}(\Pi)$, and $I_{zg}(\Pi)$ are defined in (4.31)–(4.33). The integrals $I_x(\Pi)$, $I_y(\Pi)$, and $I_h(\Pi)$ are defined below:

$$I_x(\Pi) = \iint_{\Pi} \frac{-x}{(\sqrt{x^2 + y^2 + h^2})^3} dA \quad (4.71)$$

$$I_y(\Pi) = \iint_{\Pi} \frac{-y}{(\sqrt{x^2 + y^2 + h^2})^3} dA \quad (4.72)$$

$$I_z(\Pi) = \iint_{\Pi} \frac{-h}{(\sqrt{x^2 + y^2 + h^2})^3} dA. \quad (4.73)$$

Evaluation of integrals in local coordinate system

For the integrals (4.71) and (4.72), the integrands are the partial derivatives of $\frac{1}{\sqrt{x^2 + y^2 + h^2}}$ with respect to x and y , respectively; therefore, we can transform these

surface integrals into line integrals similar to (4.23). For the integral (4.73), we use the same approach taken in evaluating (4.30).

$$I_x(\Pi) = \sum_{i=1}^m y_{v_i v_{i+1}} I_{ixy}(v_i v_{i+1}) \quad (4.74)$$

$$I_y(\Pi) = \sum_{i=1}^m -x_{v_i v_{i+1}} I_{ixy}(v_i v_{i+1}) \quad (4.75)$$

$$I_h(\Pi) = \frac{1}{V_1 \dots V_m V_{m+1}} \sum_{i=1}^m \frac{H V_i V_{i+1}}{I_{lh}(\Delta H V_i V_{i+1})} \quad (4.76)$$

where.

$$I_{ixy}(uv) = \int_0^1 \frac{dt}{\|UH - tUV + HP\|} \quad (4.77)$$

$$= \int_0^1 \frac{dt}{\sqrt{(x_{hu} + x_{uv}t)^2 + (y_{hv} + y_{uv}t)^2 + h^2}} \quad (4.78)$$

$$= \int_0^1 \frac{dt}{\sqrt{q(t) + h^2}} \quad (4.79)$$

and

$$I_{lh}(\Delta HUV) = \delta \int_0^1 \int_0^1 \frac{-hu \, du \, dv}{(\sqrt{d(v)u^2 + h^2})^3} \quad (4.80)$$

$$= -h\delta \int_0^1 \left[\int_0^1 \frac{d(q(v)u^2 + h)}{2q(v)\sqrt{q(v)u^2 + h^2}} \right] dv \quad (4.81)$$

$$= -h\delta \int_0^1 \left[\frac{1}{q(v)\sqrt{q(v)u^2 + h^2}} \Big|_0^1 \right] dv \quad (4.82)$$

$$= -\text{sign}(h)\delta \left[\int_0^1 \frac{|h|dv}{q(v)\sqrt{q(v) + h^2}} - \int_0^1 \frac{dv}{q(v)} \right] \quad (4.83)$$

These integrals can also be expressed using the integrals (4.58), (4.61) and (4.64).

We have:

$$I_{l_{xy}}(uv) = I_3 \quad (4.84)$$

$$I_{lh}(\Delta HUV) = -\text{sign}(h)(I_2 - I_1). \quad (4.85)$$

We have obtained formulae for evaluating the integrals of Green's function and its partial derivatives over a planar polygonal area. The computational process is summarized in the following algorithm.

Procedure

Step 1 For each edge UV , calculate the integrals I_1 , I_2 , and I_3 .

Step 2

For Green's function $\frac{1}{r}$, use (4.38), (4.39), and (4.51).

For $\frac{\partial}{\partial x_g}(\frac{1}{r})$, use (4.74) and (4.84).

For $\frac{\partial}{\partial y_g}(\frac{1}{r})$, use (4.75) and (4.84).

For $\frac{\partial}{\partial z_g}(\frac{1}{r})$, use (4.38), (4.39), and (4.85).

Step 3 For partial derivatives, use (4.70).

Remarks

- Most of the effort lies in Step 1. Therefore, the surface integral of the partial derivatives of Green's function $\frac{1}{r}$ can be obtained as by-products of the computational process for evaluating the surface integral of $\frac{1}{r}$.

- In boundary element methods, the above surface integrals are normally calculated using numerical integration, notably, Gauss quadratures. However, the analytical formulation proposed here is preferable. The main reason is that these integrals are meant for near-field interaction and in this case, Gauss quadratures are not as accurate and efficient. It is noted that the far-field interaction is already handled by the translations and therefore, there is no need for direct calculation of these integrals.

4.3 Preconditioning

The discretization matrix A in (4.3) can be ill-conditioned, thereby making the matrix-vector multiplication Ax sensitive to small changes in x , potentially leading to deterioration of the convergence rate or even to unreliable results. Therefore, preconditioning is essential for improvement of the rate of convergence. A detailed discussion of preconditioning can be found in [11]. In preconditioning, we can employ one of two kinds: left preconditioner P_l and right preconditioner P_r . With a left preconditioner, instead of solving equation $A\sigma = V$, we solve $P_l A\sigma = P_l V$, while with a right preconditioner, we solve $AP_r y = V$ and obtain $\sigma = P_r y$. We prefer the right preconditioner because the nature of the residual $r = \|A\sigma - V\|$ is not altered by P_r . It is expected that with either kind of preconditioner, the matrix $P_l A$ or AP_r is less ill-conditioned than the original matrix A . The ideal case is $P_l = A^{-1}$ or $P_r = A^{-1}$; however, for large N , it is impractical to obtain A^{-1} ; therefore, we contend with some form of approximation of A^{-1} .

In constructing P_r , we employ the technique suggested in [51,52], which belongs

to a more general scheme called overlapping block preconditioning. Here, a block represents the equations associated with all the panels in a given leaf cube. Suppose that the given leaf cube has m_1 panels: $\Pi_{k_1}, \dots, \Pi_{k_{m_1}}$ and m_2 neighbor panels: $\Pi_{k_{m_1+1}}, \dots, \Pi_{k_m}$, belonging to its neighboring leaf cubes. Here $m = m_1 + m_2$. Let A_s be an $m \times m$ submatrix of A that contains only the rows and columns corresponding to the m indices: k_1, \dots, k_m . Let P_{sr} be a submatrix (to be determined) of size $m_1 \times m$ of the right preconditioner P_r corresponding to the m_1 panels. We choose $P_{r,s}$ as the corresponding submatrix of $(A_s^{-1})^T$. That is,

$$P_{sr}^T = A_s^{-1} I. \quad (4.86)$$

where I be an $m \times m_1$ matrix whose diagonal entries are 1 and whose other entries are 0. Note that m_1 is usually much smaller than m . Therefore, it is more efficient to use LU decomposition on A_s than to compute A_s^{-1} directly. We solve the following equation for P_{sr} :

$$A_s P_{sr}^T = I \quad (4.87)$$

The memory requirement for preconditioning lies in the storage of matrix P_r , which is the same as that of storing the entries of A for the direct part in eq. (4.11). When N is large, the storage can be significant. If there is insufficient memory, we may resort to using a higher tree hierarchy for reducing m_1 and hence m . In such a case, the preconditioning may be less effective. In terms of computational time, the extra cost of using preconditioning includes calculating the entries of A_s that

do not belong to any near-field term (see (4.6)) and solving eq. (4.87) for each leaf cube. It also includes the extra multiplication $P_r y$ at each iteration step and the restoration of the solution $\sigma = P_r y$ at the end of the iteration process.

4.4 Stopping criteria

In iterative methods, it is crucial to determine the stopping criteria needed which effectively terminate the iteration process without compromising physical plausibility of solutions. The only available indication of the error of the computed charge density is the residual of the solution of eq. (4.3) at each iteration step:

$$r = \|A\sigma^{(i)} - V\|. \quad (4.88)$$

Here, $\sigma^{(i)}$ denotes the charge density obtained at the i^{th} iteration and $V \equiv b$. Note that exact value of r is not available because we only have an approximation of $A\sigma^i$, the accuracy of which is determined by that of the underlying exponential expansion and is affected by the conditioning of the matrix A . Even if the exact value of r is known, it does not reflect the true error of the solution to the problem because the discretization and associated errors are not taken into account. Therefore, in devising meaningful stopping criteria, one should be aware of the above issues.

For a given tolerance $\epsilon^{\text{tol}} \in \{\epsilon_{\text{abs}}^{\text{tol}}, \epsilon_{\text{rel}}^{\text{tol}}\}$, the first stopping criterion is

$$\|A\sigma^{(i)} - V\| < \epsilon_{\text{abs}}^{\text{tol}}, \quad (4.89)$$

or

$$\frac{\|A\sigma^{(i)} - V\|}{\|V\|} < \epsilon_{\text{rel}}^{\text{tol}} \quad (4.90)$$

The overall computational accuracy and efficiency is dependent of the value chosen for the tolerance ϵ^{tol} . The main difficulty lies in the choice of value for ϵ^{tol} to guarantee the desired degree of accuracy with minimal number of iterations. On the one hand, if ϵ^{tol} is too large, equation (4.3) is not satisfied. On the other hand, if ϵ^{tol} is too small, an unnecessarily large number of iterations may be required without corresponding improvement in the accuracy of the computed charge density. Furthermore, it is noted that $A\sigma^{(i)}$ is not exact and therefore, the error in matrix-vector multiplication, which is denoted as ϵ^{EB} , should also be taken into account. It may turn out that the residual r is limited or masked by ϵ^{EB} . In this case, it is not necessary to choose values of ϵ^{tol} small relative to ϵ^{EB} .

For a desired degree of accuracy on the solution σ of eq. (4.3), with some problems, it is sufficient if the values of ϵ^{tol} are in the same range as the expected error ϵ of σ . However, in other problems, it may be necessary that $\epsilon^{\text{tol}} \ll \epsilon$. This is evident from the results shown in Tables 4.1 and 4.2, which show the change in relative error of the calculated capacitance with respect to the residual r . In the first case (Table 4.1), the value of the capacitance is stabilized when r is smaller than 10^{-4} ; while in the second case (Table 4.2), the calculated value stabilizes only when r is smaller than 10^{-8} .

In capacitance extraction, it is observed that when the residual falls below some value, the change in capacitance becomes negligible and iteration beyond this value

is not necessary. This is illustrated in Table 4.3. Here, if one needs four digits of accuracy for the capacitance, the iteration process can be terminated after the 8th iteration, at which point the residual is still only around 10^{-3} . The above observation suggests a more effective stopping criterion:

$$\frac{\|C_{new} - C_{old}\|}{\|C_{new}\|} < \epsilon_{rel}^{cap}. \quad (4.91)$$

Here, C_{new} and C_{old} denote capacitances obtained in the last two iterations and ϵ_{rel}^{cap} the relative error tolerance on the capacitances. Because the values of the computed capacitance are very small, we use the relative error instead of absolute error as the stopping criterion. Furthermore, the values in the capacitance matrix may be different by orders of magnitude, therefore, the above criterion should be applied for each entry in the capacitance matrix. To deal with the transient nature of the error and to improve reliability of the criterion, it may be mandatory that the inequality (4.91) be applied in a number consecutive iterations. Note that with the use of (4.91), the number of iterations required can be reduced significantly. However, the additional cost lies in the evaluation of the capacitance C_{new} , which involves the computation of the charge density in eq. (4.3) at each iteration step.

4.5 Multi-level multi-precision iteration

In addition to efficient matrix-vector multiplication and effective preconditioning, generating reliable initial guesses is an important component of the iterative process for solving the large and dense system of linear equations (4.3). If an initial guess is close to the true solution, the number of iteration required can be reduced

Iteration	$r = \ A\sigma^{(i)} - V\ $	$\frac{\ C - \tilde{C}\ }{\ C\ }$
1	1.5	6.39×10^{-1}
2	1.7×10^{-2}	2.47×10^{-3}
3	2.1×10^{-3}	2.20×10^{-3}
4	2.5×10^{-4}	2.16×10^{-3}
5	3.5×10^{-5}	2.16×10^{-3}

Table 4.1: Change in residual and associated relative error in calculated capacitance. Here C denotes the exact value of capacitance and \tilde{C} its computed value. The result is stabilized for $r \leq 2.5 \times 10^{-4}$. $\|V\| = 2.2 \times 10^1$.

Iteration	$r = \ A\sigma^{(i)} - V\ $	$\frac{\ C - \tilde{C}\ }{\ C\ }$
1	1.8×10^{-2}	6.5×10^{-1}
3	2.8×10^{-3}	6.5×10^{-1}
6	2.3×10^{-4}	6.5×10^{-1}
8	3.9×10^{-5}	6.5×10^{-1}
24	1.0×10^{-8}	6.2×10^{-1}
28	3.3×10^{-7}	6.6×10^{-2}
29	5.9×10^{-8}	1.9×10^{-3}
30	8.0×10^{-9}	1.3×10^{-4}

Table 4.2: Change in residual and associated relative error in calculated capacitance. The result is stabilized for $r \leq 10^{-8}$. $\|V\| = 2.2 \times 10^1$.

Iteration	Residual	Capacitance		Relative change in capacitance
		$\frac{C_{11}}{\epsilon_0}$	$\frac{C_{12}}{\epsilon_0}$	
1	1.0×10^0	7.2744	-3.2830	4.1×10^{-3}
2	4.6×10^{-1}	7.2324	-3.2849	5.8×10^{-3}
3	2.8×10^{-1}	7.2199	-3.2702	4.5×10^{-3}
4	1.7×10^{-1}	7.2175	-3.2695	3.3×10^{-4}
5	8.2×10^{-2}	7.2209	-3.2691	4.7×10^{-4}
6	3.0×10^{-2}	7.2251	-3.2699	5.8×10^{-4}
7	1.4×10^{-2}	7.2264	-3.2699	1.8×10^{-4}
8	5.8×10^{-3}	7.2270	-3.2701	8.6×10^{-5}
9	2.4×10^{-3}	7.2274	-3.2702	4.9×10^{-5}
10	1.4×10^{-3}	7.2275	-3.2702	2.2×10^{-5}
11	6.5×10^{-4}	7.2276	-3.2702	1.2×10^{-5}
12	2.2×10^{-4}	7.2277	-3.2702	1.0×10^{-5}
13	1.5×10^{-4}	7.2277	-3.2702	1.3×10^{-6}
14	5.5×10^{-5}	7.2277	-3.2702	1.9×10^{-6}
15	1.5×10^{-5}	7.2277	-3.2702	5.1×10^{-7}
16	5.2×10^{-6}	7.2277	-3.2702	1.5×10^{-7}

Table 4.3: Relative change in calculated capacitance versus residual.

significantly, thus improving the overall computational performance. In this section, we describe two schemes that can be combined to generate reliable initial guesses: multi-level iteration and multi-precision iteration. The former is based on the general multigrid technique [53] and the latter is realized using the EE method.

4.5.1 Multi-level iteration

The solution associated with a coarse grid can be used as initial guess for iterations with a fine grid. Suppose that the conductor and interface surfaces are meshed into a sequence of nested grids: $\mathcal{G}_0 \subset \mathcal{G}_1 \subset \mathcal{G}_2 \cdots \subset \mathcal{G}_n$. Starting with the initial coarse grid \mathcal{G}_0 , we find the charge density $\sigma^{(0)}$, which can be obtained easily. Using $\sigma^{(0)}$ as initial guess, we find the charge density $\sigma^{(1)}$ on the finer grid \mathcal{G}_1 without requiring too many iterations. This process is repeated until the charge density $\sigma^{(n)}$ on the finest grid \mathcal{G}_n is obtained.

Although there is an extra cost of finding the charge density on coarser grids, the cost is offset by a significant reduction in the number of iterations. Table 4.4 lists the number of iterations and the cost associated with and without the use of multi-level iteration. Here, seven nested grids are used and the corresponding numbers of panels are listed in column 1. Columns 2 and 4 show the number of iterations in each case. Column 5 lists the cost associated with the case of not using multi-level iteration. The cost is defined as the product of the numbers of panels and iterations. Column 3 lists the accumulative cost for finding the charge density on all grids when multi-level iteration is used. The ratio of the costs of the two approaches (columns 5 and 3) is listed in column 6. An improvement by a factor of 1.5 is observed. As

Panel	With multi-level		Without multi-level		Cost ratio
	Iteration	Accumulative cost	Iteration	Cost	
160	10	160	10	160	1.00
996	11	12,556	14	13,944	1.11
2.656	14	49,470	17	45,152	0.91
8.148	16	180,108	24	195,552	1.09
28.132	16	630,220	24	675,168	1.07
102.772	21	2,788,432	32	3,288,704	1.18
393.368	33	15,769,576	61	23,995,448	1.52

Table 4.4: Number of iterations and cost ratio: with and without multi-level iteration.

the grid is refined, the discretization matrix becomes increasingly ill-conditioned, leading to poorer convergence rate. Therefore, good initial guesses become more important in reducing the number of iterations. The multi-level iteration can be employed to generate good initial guesses, thus improving the overall computational performance.

4.5.2 Multi-precision iteration

The EE method provides a flexible mechanism for approximating the matrix-vector multiplication with various degrees of accuracy during the iteration process. The accuracy of the matrix-vector multiplication is determined by that of the underlying exponential expansion for the Greens' function $\frac{1}{r}$, which in turn can be expressed in terms of its approximation size S_{app} . Because each term in the expansion can be processed independently of each other, switching among expansions of different S_{app} incurs negligible overhead. Furthermore, the memory requirement is independent of S_{app} . Therefore, for a given matrix size $N \times N$, the main computational cost

of matrix-vector multiplication is $O(S_{app}N)$ in time, which is proportional to the approximation size S_{app} . For smaller S_{app} , the multiplication Ax is faster at the cost of a less accurate result and the number of iterations is larger due to sensitivity to the conditioning of the matrix A . For larger S_{app} , the multiplication is slower but the results are more accurate and the number of iterations is smaller. If preconditioning is used, the number of iterations is less dependent on S_{app} than it is without preconditioning; with good preconditioning, the variation in number of iterations with S_{app} is insignificant (see Tables 4.7 and 4.9). Furthermore, for an extra digit of accuracy, the approximation size S_{app} needed may increase twofold or more. These observations lead to our proposed multi-precision iteration scheme. In the early stage of the iteration process with preconditioning, we choose a small S_{app} to speed up the matrix-vector multiplication and to obtain a preliminary solution, which can then be used as an initial guess for the next iterative step. When the solution becomes stabilized (e.g., when the residual falls below some value or when there is no significant change in the capacitance), we switch to a larger S_{app} and continue with the iteration process. The number of remaining iterations is expected to be smaller thanks to the improved initial guess previously obtained using a smaller S_{app} .

In a more general description, we choose a sequence of non-decreasing approximation sizes $\{S_{app}^{(k)}\}$. At the i^{th} iteration step, we perform matrix-vector multiplication using exponential expansion of size $S_{app}^{(k_i)}$. Note that the traditional method can be considered as a special case in which all the $S_{app}^{(k)}$'s are the same. With the multi-precision iteration scheme, we can observe the change in the accuracy of the

Panel	With multi-precision			Without multi-precision		Cost ratio
	Iteration		Cost	Iteration $S_{app} = 60$	Cost	
	$S_{app} = 18$	$S_{app} = 60$				
8.148	5	15	990	24	1.440	1.45
28.132	5	15	990	24	1.440	1.45
102.772	7	20	1.326	32	1.920	1.45
393.368	12	35	2.316	65	3.990	1.68

Table 4.5: Number of iterations and cost ratio: with and without multi-precision iteration.

iterative solution with respect to the approximation size S_{app} during the iteration process. It is expected as the iteration process progresses, the solution become stabilized beyond a certain $S_{app}^{k_o}$. At this point, the iteration process can be terminated and we are confident that the accuracy of the solution is adequate with the approximation size $S_{app}^{k_o}$. Here, any discrepancy from the true solution may be attributed to discretization and related errors [54].

Table 4.5 lists the number of iterations and the cost associated with and without the use of multi-precision iteration. Column 1 shows the number of panels. Columns 5 and 6 list the number of iterations and cost associated with the case of not using multi-precision iteration. Here, the cost is defined as the product of numbers of iterations and approximation size S_{app} . Columns 2 and 3 lists the number of iterations corresponding to two approximation sizes when the multi-precision iteration is used; the total cost is shown in column 4. The ratio of the costs of the two approaches (columns 6 and 4) is listed in column 7. Here, an improvement by a factor of 1.5 is observed. Hence, the multi-precision iteration can also be used to generate good initial guesses, thus reducing the number of iterations required.

4.6 Numerical results

In this section, we consider two simple examples for verification of capacitances computed using the EE method. We also show the computational time and memory storage in solving the charge density problem.

4.6.1 Illustrative examples

We consider two simple examples for verification of capacitances computed using the EE method. The examples deal with spherical conductors. Here, the exact value of capacitance can be determined analytically. We examine the convergence rate and accuracy of the computed charge and capacitance in terms of panel number and the approximation accuracy on the potential. Note that the accuracy on the exponential expansion can be characterized by its approximation size S_{app} . Therefore, the accuracy on the potential can also be expressed in terms of S_{app} of the underlying exponential expansion. Additionally, the effect of different preconditioning schemes on convergence rate and accuracy is examined. Unless specified otherwise, the unit of length is $1m$, and the capacitance is normalized in terms of the permittivity of free space $\epsilon_o = 8.854 \times 10^{-12}F/m$.

Example 1

The value of capacitance associated with a spherical conductor of unit radius ($1m$) is $4 \times \pi\epsilon_o = 111.26pF$ [55]. The relative error in the computed capacitance and the L_2 -norm error in charge density ($\|\sigma - \tilde{\sigma}\|$) and charge ($\|q - \tilde{q}\|$) are listed in Table 4.6. Here, σ and q denote the exact values of charge density and charge, respectively, while $\tilde{\sigma}$ and \tilde{q} denote the respective computed values.

For a small number of panels N , the discretization error is dominant and there-

fore a small approximation size S_{app} is sufficient. When N is large, however, the discretization error is reduced for larger N , and therefore, a larger S_{app} (i.e., more accurate approximation on potential) is needed for more accurate charge density, charge, and capacitance. The difference in error due to variations in approximation size becomes evident in the L_2 -norm error in charge density and charge rather than in the relative error in capacitance. The difference in error grows with increasing N .

The effect of preconditioning on convergence rate is shown in Table 4.7. The difference between preconditioning schemes 1 and 2 is that in the case of the former, the average number of panels per leaf cube is smaller. Regardless of preconditioning, the number of iterations decreases as the approximation size S_{app} increases since a more accurate approximation on potential can reduce the effect of ill-conditioning of discretization matrix A . However, the number of iterations increases as the number of panels N increases. This is due to the discretization matrix A becoming more ill-conditioned. This clearly demonstrates the effect of an ill-conditioned matrix on the convergence rate and on the accuracy of matrix-vector multiplication $A\sigma$. From the point of view of computational efficiency, using a larger S_{app} to reduce the number of iterations may not be efficient, since the time taken for each iteration is proportional to S_{app} . The preconditioning technique discussed in section 4.3 significantly reduces the number of iterations. The effectiveness of the preconditioning depends on the number of neighbor panels of a given panel, which can be characterized by the average number of panels per leaf cube. It appears that the preconditioning becomes more effective as the number of panels per leaf cube

increases. However, the associated memory requirement for storage of the extra near-field coefficients as well as the preconditioner's coefficients can be quite high, especially when N is large.

Example 2

We compute the capacitance of a unit sphere conductor embedded in a concentric dielectric sphere of radius 1.5 unit and dielectric constant of 2. Its value is $4.8 \times \pi \epsilon_0 = 133.52 \text{pF}$ [55]. The relative error in the computed capacitance and the L_2 -norm error in charge density ($\|\sigma - \tilde{\sigma}\|$) and charge ($\|q - \tilde{q}\|$) are listed in Table 4.8. The convergence rates with different preconditioning schemes are listed in Table 4.9. The trends with respect to accuracy and convergence rate observed in this example are similar to those in Example 1. For a small number of panels N , the discretization error is dominant; and hence, the use of a small approximation size appears to be sufficient. However, as N increases, the discretization error reduces and therefore, a large approximation size is needed to improve the overall accuracy.

Note that the accuracy in the capacitance alone may not reflect the overall accuracy. The value of capacitance computed from a coarse mesh (2560 panels) appears to be more accurate than that obtained from a finer mesh. But this is most likely due to numerical cancellation. The results shown in Tables 4.6 and 4.8 indicate that as the mesh is refined, the accuracy of the computed charge density and charge, and hence the overall accuracy, are improved.

With regard to convergence rate, as N increases, the number of iterations also increases. With the use of preconditioning or higher approximation accuracy on potential, the convergence rate is improved. For a larger number of panels per leaf

cube, preconditioning appears to be very effective, but at the cost of high memory requirement.

Panel	Capacitance relative error			L_2 -error in charge density			L_2 -error in charge		
	S_{app}			S_{app}			S_{app}		
	18	60	243	18	60	243	18	60	243
512	9.9×10^{-3}	9.0×10^{-3}	9.0×10^{-3}	4.1×10^{-1}	3.7×10^{-1}	3.7×10^{-1}	9.2×10^{-3}	9.0×10^{-3}	9.0×10^{-3}
2,048	2.4×10^{-3}	2.2×10^{-3}	2.2×10^{-3}	5.6×10^{-1}	2.8×10^{-1}	2.8×10^{-1}	3.2×10^{-3}	1.7×10^{-3}	1.7×10^{-3}
8,192	7.9×10^{-4}	5.7×10^{-4}	5.7×10^{-4}	1.4×10^{-0}	2.1×10^{-1}	2.0×10^{-1}	2.1×10^{-3}	3.3×10^{-4}	3.3×10^{-4}
32,768	3.7×10^{-4}	1.5×10^{-4}	1.4×10^{-4}	4.0×10^{-0}	2.0×10^{-1}	1.5×10^{-1}	1.4×10^{-3}	7.9×10^{-5}	6.0×10^{-5}

Table 4.6: Error in capacitance, charge density, and charge in Example 1.

Panel	No preconditioning			Preconditioning 1			Preconditioning 2				
	S_{app}			panels per leaf cube	S_{app}			panels per leaf cube	S_{app}		
	18	60	243		18	60	243		18	60	243
512	15	13	13	2.3	5	4	4	9.1	4	3	3
2,048	22	19	15	2.7	7	6	5	7.5	6	5	5
8,192	31	23	23	2.2	10	9	8	7.5	7	6	6
32,768	43	30	26	2.3	15	13	10	7.8	10	9	9

Table 4.7: Convergence rate in terms of number of iterations with different preconditioning in Example 1. The absolute error tolerance $\epsilon_{abs}^{tol} = 1.0 \times 10^{-6}$.

Panel	Capacitance relative error			L_2 -error in charge density			L_2 -error in charge		
	S_{app}			S_{app}			S_{app}		
	18	60	243	18	60	243	18	60	243
640	1.4×10^{-2}	1.4×10^{-2}	1.4×10^{-2}	8.1×10^{-1}	7.8×10^{-1}	7.8×10^{-1}	6.7×10^{-2}	6.4×10^{-2}	6.4×10^{-2}
2,560	-4.1×10^{-4}	-9.3×10^{-5}	-9.3×10^{-5}	7.3×10^{-1}	5.9×10^{-1}	5.9×10^{-1}	1.6×10^{-2}	1.3×10^{-2}	1.3×10^{-2}
10,240	-2.3×10^{-3}	-1.9×10^{-3}	-1.9×10^{-3}	8.9×10^{-1}	4.6×10^{-1}	4.6×10^{-1}	5.2×10^{-3}	2.5×10^{-3}	2.5×10^{-3}
40,960	-1.8×10^{-3}	-1.4×10^{-3}	-1.4×10^{-3}	2.2×10^{-0}	3.6×10^{-1}	3.6×10^{-1}	3.4×10^{-3}	5.2×10^{-4}	5.2×10^{-4}

Table 4.8: Error in capacitance, charge density, and charge in Example 2.

Panel	No preconditioning			Preconditioning 1			Preconditioning 2				
	S_{app}			panels per leaf cube	S_{app}			panels per leaf cube	S_{app}		
	18	60	243		18	60	243		18	60	243
640	11	8	8	2.0	5	5	5	10.0	4	4	4
2,560	16	13	13	2.2	8	7	6	6.8	6	5	5
10,240	21	18	17	2.0	10	9	8	6.5	8	7	6
40,960	30	23	20	2.0	12	11	9	6.8	9	8	6

Table 4.9: Convergence rate in terms of number of iterations with different preconditioning in Example 2. The absolute error tolerance $\epsilon_{abs}^{tol} = 1.0 \times 10^{-6}$.

4.6.2 Computational time and memory storage

For the purpose of estimating the trends on the computational time and memory storage used in solving the charge density problem, we perform numerical extraction of geometric capacitance associated with a-Si TFTs (to be discussed in Chapter 6). In the simulation, we record the running time and maximum memory storage. There are several factors that can influence these values such as (i) the design and implementation of the software, (ii) the degree of optimization invested, and (iii) the trade of between speed and memory requirement in the choice of near-field/far-field calculation, the preconditioner (see § 4.2), and the number of directions processed in each translation (see § 3.2). However, these values reflect the linear dependence of both the computational time and memory storage on the number of panels N . The former is linearly dependent on both the approximation size S_{app} of the exponential expansion used and the number of iterations required, while the latter is independent of these two factors.

Computational time $O(S_{app})$

The computational time as a function of the number of panels N and the number of iterations is shown in Table 4.10. Here, the approximation size of the exponential expansion used is $S_{app} = 31$. The computational time for each iteration, mainly from matrix-vector multiplication, is found to increase linearly with N . This is illustrated in Fig. 4.3. With preconditioning, the number of iterations increases much more slowly in comparison with that of panels. In this case, the charge density problem can be solved in linear time $O(S_{app}N)$.

Panel	Iteration	Time (s)
160	10	4
996	11	9
2,656	14	30
8,148	16	115
28,132	16	454
102,772	21	2627
393,368	33	19081

Table 4.10: Computational time as a function of number of panels and iterations.

Panel	Memory (Mb)
160	6
996	7
2,656	9
8,148	15
28,132	45
102,772	167
393,368	640

Table 4.11: Computational memory storage as a function of number of panels.

Memory storage independent of accuracy

The memory storage used in the simulation as a function of the number of panels N is shown in Table 4.11. It is linearly proportional to N and independent of the desired degree of accuracy (or equivalently on S_{app}) and of the number of iterations.

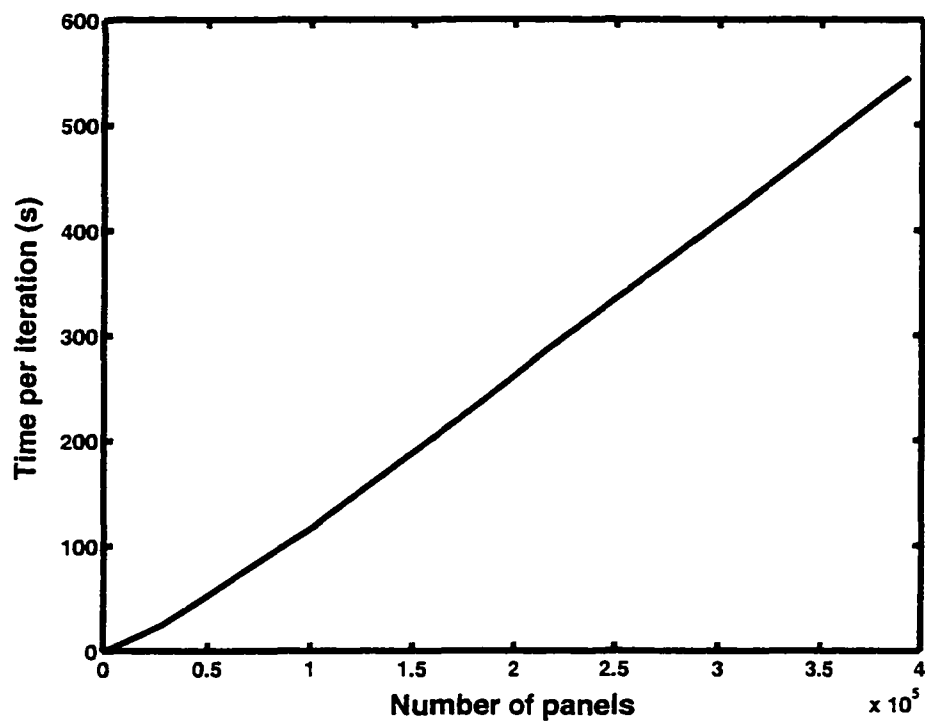


Figure 4.3: Computational time per iteration as a function of number of panels. Approximation size $S_{app} = 31$.

Chapter 5

Integral Equations of Second Kind for Charge Density Problem

The existing integral-equation approach for describing the charge density (see eqs. (1.16)–(1.17)) uses a single-layer potential description that yields either a Fredholm integral equation of the first kind or a combination of the first and second kinds [4, 8]. However, as the mesh is refined, the resultant matrix can become ill-conditioned, leading to a potentially high cost of obtaining the solution. This situation is exacerbated in the methods that provide only approximation of matrix-vector multiplication such as the MP and EE methods. Furthermore, the numerical solution of the integral equation of the first kind is not as well understood as that of the second kind, for which there are efficient numerical algorithms [45].

Integral equations of the second kind have been formulated for a homogeneous dielectric medium but with the use of double-layer potential description. Here, one can directly retrieve the capacitance since it is expressed as unknowns of the inte-

gral equation [56–58]. However, the charge density, which is necessary for analysis of electrostatic interaction, is not readily available; additionally, the extension of the double-layer potential description for multiple dielectric media does not appear to be straightforward.

In this chapter we describe a new formulation for the charge density that employs a single-layer potential description and yet yields Fredholm integral equations of the second kind [28,59]. The formulation is applicable to both homogeneous and multiple dielectric media. Here, we consider not only the potential but also the electric flux, offering a direct means of controlling the overall computational accuracy and efficiency. In §5.1, we describe the existing integral-equation model equations for the charge density in terms of integral operator form and identify difficulties associated with these model equations. We present our proposed formulation in §5.2 and discuss the subtleties in choice of parameter values that preserve the character of integral equations of the second kind without compromising the desired degree of accuracy. Numerical results are shown in §5.3 along with comparisons between the two approaches in terms of accuracy and convergence rate.

5.1 Existing approach

5.1.1 Integral equations

The model equations for the charge density of a multi-conductor system embedded in a homogeneous medium (eq. (1.16)) or multiple dielectric media (eqs. (1.16)–(1.17)) can be expressed in a more compact form using integral operator forms. We assume free space for the former and multi-layered dielectric for the latter.

Free space

For the homogeneous, the model equation (1.16) is a Fredholm integral equation of the first kind:

$$K_c \sigma = V, \quad (5.1)$$

where

$$K_c \sigma(X) = \int_S G(X, Y) \sigma(Y) dS_Y. \quad (5.2)$$

$$V(X) = \sum_{i=1}^n V_i \chi_{S_i}(X), \quad \chi_{S_i}(X) = \begin{cases} 1 & \text{if } X \in S_i \\ 0 & \text{if } X \notin S_i. \end{cases} \quad (5.3)$$

The surface S here represents conductor surfaces: $S = S_c$.

Multiple dielectric media

For the multiple dielectric media, the equation (1.14) for the discontinuity of the electric field across the interface between two dielectric layers is a Fredholm integral equation of the second kind:

$$(I + 2\lambda K_d) \sigma = 0. \quad (5.4)$$

Here, I is the identity operator, i.e., $I\sigma(X) \equiv \sigma(X)$ and K_d is defined similarly to K_c :

$$K_d \sigma(X) = \int_S \frac{\partial G(X, Y)}{\partial n_X} \sigma(Y) dS_Y, \quad (5.5)$$

where $\frac{\partial G(X,Y)}{\partial n_x}$ is the normal derivative of $G(X,Y)$ at a point X on the interface S_d . The surface S now consists of the conductor surfaces and the interfaces: $S = S_c \cup S_d$.

The model equations (1.16)–(1.17) are, therefore, a combination of Fredholm integral equations of the first and second kinds:

$$\begin{cases} K_c \sigma & = V \\ (I + 2\lambda K_d) \sigma & = 0. \end{cases} \quad (5.6)$$

Discretization of the above equations using the panel method yields a system of linear equations (4.3). We shall now discuss some difficulties associated with the above formulation.

5.1.2 Difficulties

Ill-conditioning of discretization matrix A

With a coarse mesh, the numerical solution may not yield reliable values of the charge density due to discretization error. To improve the accuracy, we need mesh refinement, which requires a larger number N of panels. An important observation is that as N increases the matrix A becomes more ill-conditioned. Therefore, the result of matrix-vector multiplication Ax is very sensitive to small changes in the vector x . As a consequence, convergence rate decreases, resulting in a large number of iterations. Furthermore, when the MP or EE method is used to approximate matrix-vector multiplication, the reliability of solutions becomes even more questionable. The error due to the approximation in matrix-vector multiplication is amplified by the ill-conditioning of matrix A . Consequently, a higher order of ex-

pansion is needed for a more accurate matrix-vector multiplication so as to reduce the effect of the ill-conditioning on the overall accuracy. To deal with this problem, one may also resort to the use of preconditioning as described in the previous chapter: for large N , however, the memory requirement for effective preconditioning can be significant.

Mismatch in mixed integral equation system

We examine the coefficients of matrix A , which can be decomposed into two submatrices A_c and A_d . These submatrices originate from the rows of A corresponding to panels on the conductors and on interfaces, respectively. If we use a scaling transformation: $(X, Y, Z) \rightarrow (sX, sY, sZ)$ with a scaling factor $s > 0$, we observe that A_d remains unchanged while A_c is scaled by s . The resultant matrix $s(A)$ is:

$$s(A) = \begin{bmatrix} sA_c \\ A_d \end{bmatrix} \quad (5.7)$$

We consider the two extreme cases: $s \rightarrow 0$ and $s \rightarrow \infty$. When s approaches zero (for example, when the geometry of the system is scaled down), the matrix $s(A)$ has the tendency of singular behavior due to one of its submatrices sA_c tending to zero; the other submatrix A_d remains unchanged. As a consequence, the convergence rate is poor and the sensitivity to errors in matrix-vector multiplication is high. When s approaches infinity, the coefficients of the submatrix sA_c becomes increasingly large, while those of A_d remain the same. This also results in ill-conditioning of A . To gain insight into the effect of scaling on the solution of eq. (4.3), we describe

the following form for the scaling:

$$sA_c\sigma = V \quad (5.8)$$

$$A_d\sigma = 0 \quad (5.9)$$

By replacing σ with $\sigma' = s\sigma$, equations (5.8) and (5.9) can be rewritten as:

$$A_c\sigma' = V \quad (5.10)$$

$$\frac{1}{s}(A_d\sigma') = 0 \quad (5.11)$$

In comparison with the original system (4.3), the first equation (i.e., eq. (5.10)) is unchanged, while the second equation (i.e., eq. (5.11)) is scaled by $\frac{1}{s}$. For a given tolerance ϵ on the residual, when s approaches zero (or equivalently when $\frac{1}{s}$ approaches infinity), eq. (5.11) requires more iterations to bring the residual below ϵ . When s approaches infinity (or equivalently, when $\frac{1}{s}$ approaches zero), it becomes easier to satisfy eq. (5.11); therefore, to ensure sufficient accuracy, one may need to choose much smaller ϵ . Note that with the above scaling, the factor $\frac{1}{s}$ in eq. (5.11) can easily be removed by scaling the original equation (5.4) with s . This suggests that one may need to modify the original equation (5.6) to deal with the mismatch between the equations, (5.1) and (5.4), which, in general, may not be as simple as the case considered above.

A similar situation arises when the mesh is refined and the areas of the panels are reduced; all the entries in the resultant matrix are reduced correspondingly except for the diagonal in A_d , which is unchanged and remains as unity.

In view of the above difficulties, we propose an alternative formulation which yields a Fredholm integral equation of the second kind, leading to less ill-conditioning of the matrix A .

5.2 Integral equations of the second kind

5.2.1 Integral equations

It is known that the potential within a conductor is constant and that the electric field vanishes in the interior. Therefore, the flux at the surface going into the interior of the conductor is zero.

$$\frac{d\Phi(X)}{dn_i} = \frac{d\Phi^-(X)}{dn} = 0, \quad X \in S_c. \quad (5.12)$$

With the jump relation (1.13), the above equation can be cast into a Fredholm integral equation of the second kind:

$$\sigma(X) + 2 \int_S \frac{\partial G(X, Y)}{\partial n_X} \sigma(y) dS_Y = 0, \quad X \in S_c. \quad (5.13)$$

This equation looks similar to eq. (1.14) with λ being set to 1, which can be viewed as $\epsilon^- = \infty$. In the integral operator form, eq. (5.13) can be rewritten as:

$$(I + 2K_f)\sigma = 0. \quad (5.14)$$

Here, K_f is identical to K_d . The domain of $K_f\sigma(X)$ is on S_c , while that of $K_d\sigma(X)$ is on S_d . The main observation is that a linear combination of eqs. (5.1) and (5.14) yields a Fredholm integral equation of the second kind, which at the same time can

be used to replace (5.1):

$$(\alpha(I + 2K_f) + \beta K_c)\sigma = \beta V. \quad (5.15)$$

Here, α and β are scalar (or possibly vector) parameters introduced for flexibility in choice of the final form of the equation. In system (5.6), by replacing the first equation (which originates from eq. (5.1)) with eq. (5.15), and by scaling the second equation with a parameter γ , we obtain an alternative system, which is purely a Fredholm integral equation of the second kind.

$$\begin{cases} (\alpha(I + 2K_f) + \beta K_c)\sigma = \beta V \\ \gamma(I + 2\lambda K_d)\sigma = 0. \end{cases} \quad (5.16)$$

The two eqs. (5.15) and (5.16), together with a proper choice of the parameters α , β , and γ , constitute the new formulation for the charge density of a conductor system in a homogeneous medium or multiple dielectric medium, respectively. Having established the equations, we now discuss choice of parameter values.

5.2.2 Choices of parameters

The efficient and accurate evaluation of the charge density lies in the proper choice of the parameters α , β , and γ . The formulation proposed here can be considered general.

- The choice of $\alpha = 0$, $\beta = 1$, and $\gamma = 1$ reduces the system to that of the existing approach.

- On the other hand, if we choose $\alpha = 1$, $\beta = 0$, and $\gamma = 1$, the result would likely be a zero charge density since $\sigma \equiv 0$ is one of the solutions and this solution gives the best residual (zero residual) that an iterative solver can hope for. Thus the choice of the above parameter set produces unreliable results.

The above choices are two extreme examples. The former gives $\frac{\alpha}{\beta} = 0$ and the latter yields $\frac{\alpha}{\beta} = +\infty$. If $\frac{\alpha}{\beta}$ is too large, this undermines the overall accuracy. Because of the discretization error, the left hand side of eq. (5.12) or (5.14) is not equal to zero, even at the centroid c_k of a panel Π_k . This non-zero value is an error. In eqs. (5.15) and (5.16), the error is amplified by α , which in turn affects the accuracy of the solution of the original potential equation (5.1). If $\frac{\alpha}{\beta}$ is too small, the character of (5.15) and (5.16) as integral equations of the second kind diminishes, reducing the system to eq. (5.6). Thus a non-zero α is needed in order to maintain the character of the integral equations as well as to prevent the deterioration of the conditioning of matrix A when the mesh is refined or when dimensional scaling is performed. This is the main reason why we introduce the flux equation (5.14) into the potential equation (5.1) to obtain (5.15) and (5.16). Note that for non-zero α , when the mesh is refined, the areas of the panels are reduced and the coefficients in the discretization matrix are reduced proportionally except for the diagonal entries. If α is large enough, mesh refinement has negligible effect on the portion of the diagonal associated with the conductor equations; the other portion is always maintained at γ . This is how α is used to control the conditioning of A .

From the above two extreme properties, it is essential that α be chosen in such a way that the error amplification is kept within some tolerance and at the same time, the character of the integral equations of the second kind be maintained in spite of the increase in the number of panels. The following observation suggests that the above requirement can be satisfied. Here, as the mesh is refined, the discretization error is improved and so is the error in the flux equation (5.12). This improvement makes the amplification of error by α less severe, especially when α is not too large.

- With $\beta = 1$, $\gamma = 1$, $\alpha \in (0, \infty)$ and independent of geometry, the main problem is that there is mismatch between the coefficients of the potential equation (5.1) and the corresponding coefficients of the flux equation (5.14). Again, we consider the scaling transformation: $(X, Y, Z) \rightarrow (sX, sY, sZ)$ on eq. (5.15). The resultant equation is:

$$(\alpha(I + 2K_f) + \beta s K_c)\sigma = \beta V. \quad (5.17)$$

The part originating from the flux equation, $(\alpha(I + 2K_f))\sigma$ remains unchanged, while that from the potential equation, $\beta s K_c \sigma$ is scaled by s . When s approaches zero, the above equation reduces to the following invalid form:

$$(\alpha(I + 2K_f))\sigma = \beta V. \quad (5.18)$$

for which seeking a solution is pointless. Thus we have seen another improper choice of the parameter α . From this, we conclude that the value of α should

be chosen relative to the coefficients from the potential equation (5.1). Since the coefficients on the diagonal are the largest, they can be used as the reference. We consider the largest diagonal value, denoted as $D_{c,max}$. This leads us to the following choice.

- $\beta = 1$ and $\alpha = \alpha_o D_{c,max}$. Here, α_o is a prefactor for weighting how large or how small α is with respect to $D_{c,max}$. In many cases, $\alpha_o = 1$ is a good compromise. With regard to γ , it is chosen in the range of α so that the scaling employed in both eqs. (5.15) and (5.16) are similar. One of the choices is $\gamma = (\alpha_o + 1) D_{c,max}$, yielding values for the diagonal entries in the discretization matrix between α and γ . It is clear from the above considerations that further work may be needed for selection of optimal values of these parameters.

5.3 Numerical experiments

In order to demonstrate the suitability of the new formulation for the charge density problem, we present two examples: Example 1: capacitance of a spherical conductor of unit radius ($1m$), which is known to be $C_1 = 4\pi\epsilon_o$; and Example 2: capacitance of a spherical conductor of unit radius embedded in a concentric dielectric sphere of radius 1.5 unit and dielectric constant 2, which is known to be $C_2 = 4.8\pi\epsilon_o$. These exact values can be calculated using the formulae described in [55]. In these examples, we compare the two approaches with respect to: (i) the number of iterations required for a given absolute error tolerance of the residual; (ii) the accuracy of the charge, the potential and its normal derivative (pointing into the conductor's interior, see (5.12)) at the centroids of the panels; (iii) the effect of mismatch between

the integral equations of the first and second kinds on the convergence rate. The simulation procedure is summarized as follows.

5.3.1 Simulation procedure

1. The surfaces are meshed into N panels. The charge density (to be computed) on each panel is assumed to be constant. The discretization yields a system of linear equations as given by eq. (4.3).
2. The system of linear equations is solved using the iterative method GMRES [46]. The matrix-vector multiplication is approximated using exponential expansion, with an average relative error 10^{-7} for the potential and 10^{-5} for its gradient. In most cases, we use no preconditioning so as to compare the conditioning number of the matrix A obtained from the existing formulation (eqs. (5.1) and (5.6)) with that from the new scheme (eqs. (5.15) and (5.16)). The conditioning of A is reflected in the convergence rate or equivalently, in the number of iterations required for the residual to fall below some given tolerance. Here, this is chosen as 10^{-6} .

5.3.2 Results

The results of the two simulation examples are summarized in Tables 5.1–5.7. The first column in each table lists the number of panels used for meshing the conductor surfaces and the interfaces. In Tables 5.1 and 5.3, the columns (2,3,4) and (5,6,7) list the number of iterations and the relative error of the capacitance obtained for the cases when $\alpha_o = 0, 1,$ and $10,$ respectively. In Tables 5.2 and 5.4, the third column shows the L_2 -norm error in the charge: $\|q - \tilde{q}\|$; the fourth and fifth

columns list the L_{\max} -norm error (absolute maximum error) in the potential and its normal derivative at the centroids of the panels on the conductors. In Table 5.5, the columns (2.3) and (4.5) list the number of iterations required with a scaling factor $s = 10^{-6}$ without an with preconditioning, respectively. Tables 5.6–5.7 shows the results for the case of $s = 10^6$. Here, the columns (2.3) list the number of iterations and the columns (4.5) the relative error of the capacitance computed.

5.3.3 Observation

- *Convergence rate:* as α_o (and hence α) increases, the number of iterations decreases. This is a result of eqs. (5.15) and (5.16) having increasingly more character of integral equations of the second kind.
- *Accuracy:* for a given mesh, as α gets larger, the difference in the results of the two approaches also grows. However, as the mesh is refined, this difference decreases. Although, the potential obtained in the case of $\alpha_o = 0$ is always superior, the overall L_2 norm error in the charge as well as the L_{\max} norm error in the normal derivative of potential are not as good as that obtained in the case of $\alpha_o \neq 0$. For the capacitance, the case of $\alpha_o = 0$ appears to yield more accurate values for Example 1, but not for Example 2. Note that the accuracy of the capacitance alone in some cases does not reflect the overall accuracy. In the case of $\alpha_o = 0$ in Example 2, the accuracy of the capacitance obtained from the coarse mesh (2560 panels) is higher than that obtained from a finer mesh. But this is most likely due to numerical cancellation. The results shown in tables 2 and 4 indicate that as the mesh is refined, the overall accuracy is improved and the difference in values from the two

Panel	Iteration			Capacitance error		
	0	1	10	$\alpha_o = 0$	$\alpha_o = 1$	$\alpha_o = 10$
2.048	17	11	6	-2.2×10^{-3}	-3.2×10^{-3}	-1.2×10^{-2}
8.192	22	12	7	-5.7×10^{-4}	-8.3×10^{-4}	-3.2×10^{-3}
32.768	27	14	9	-1.4×10^{-4}	-2.7×10^{-4}	-7.8×10^{-4}

Table 5.1: Convergence rate in terms of number of iterations and relative error in capacitance in Example 1.

approaches is insignificant: in fact, it is expected that this difference vanishes in the asymptotic limit.

- *Mismatch between the integral equations of first and second kinds:* the results in Tables 5.5–5.7 suggest that with the existing approach ($\alpha_o = 0$) the mismatch between the two kind of integral equations induced by scaling can cause a significant increase in the number of iterations. Furthermore, for the case of $s = 10^6$, if no preconditioning is used, the convergence rate is very slow; even with a residual in the range 10^{-6} , the error in the computed capacitance is only in the range 10^{-1} . This indicates that the matrix A is very much ill-conditioned. With the new approach ($\alpha_o \neq 0$), no such mismatch occurs and the number of iterations is much smaller. Note that preconditioning can significantly reduce the effect of the mismatch. But even with the use of preconditioning, with the scaling factor $s = 10^{-6}$, the number of iterations required in the case of $\alpha_o = 0$ is still larger than that in the case of $\alpha_o \neq 0$ without preconditioning (see Table 5.5).

Panel	α_o	L_2 -error	L_{\max} -error	
		q_i	$\phi(c_k)$	$\frac{d\phi(c_k)}{dn_i}$
2.048	0	1.7×10^{-3}	5.8×10^{-8}	3.3×10^{-3}
	1	1.6×10^{-3}	1.1×10^{-3}	3.2×10^{-3}
	10	3.4×10^{-3}	1.0×10^{-2}	2.9×10^{-3}
8.192	0	3.2×10^{-4}	2.5×10^{-8}	1.8×10^{-3}
	1	3.1×10^{-4}	3.0×10^{-4}	1.7×10^{-3}
	10	5.5×10^{-4}	2.7×10^{-3}	1.5×10^{-3}
32.768	0	6.0×10^{-5}	2.0×10^{-8}	9.1×10^{-4}
	1	6.0×10^{-5}	1.5×10^{-4}	8.8×10^{-4}
	10	8.9×10^{-5}	7.2×10^{-4}	8.3×10^{-4}

Table 5.2: Error in charge, potential and its normal derivative in Example 1.

Panel	Iteration			Capacitance error		
	0	1	10	$\alpha_o = 0$	$\alpha_o = 1$	$\alpha_o = 10$
2,560	13	9	6	-1.0×10^{-4}	-2.2×10^{-3}	-2.1×10^{-2}
10,240	17	10	6	1.9×10^{-3}	1.3×10^{-3}	-4.0×10^{-3}
40,960	21	12	8	1.4×10^{-3}	1.3×10^{-3}	-1.2×10^{-4}
163,840	27	14	10	8.3×10^{-4}	7.8×10^{-4}	4.3×10^{-4}

Table 5.3: Convergence rate in terms of number of iterations and relative error in capacitance in Example 2.

Panel	α_o	L_2 -error	L_{\max} -error	
		q	$\phi(c_k)$	$\frac{d\phi(c_k)}{dn_i}$
2,560	0	1.2×10^{-2}	4.6×10^{-8}	3.7×10^{-3}
	1	1.1×10^{-2}	2.3×10^{-3}	3.4×10^{-3}
	10	1.3×10^{-2}	2.2×10^{-3}	3.2×10^{-3}
10,240	0	2.5×10^{-3}	5.6×10^{-8}	2.0×10^{-3}
	1	2.2×10^{-3}	6.8×10^{-4}	1.9×10^{-3}
	10	2.3×10^{-3}	6.2×10^{-3}	1.7×10^{-3}
40,960	0	5.1×10^{-4}	3.1×10^{-8}	1.1×10^{-3}
	1	4.6×10^{-4}	1.8×10^{-4}	1.0×10^{-3}
	10	4.3×10^{-4}	1.6×10^{-3}	9.5×10^{-4}
163,840	0	1.1×10^{-4}	1.7×10^{-8}	5.7×10^{-4}
	1	1.0×10^{-4}	4.7×10^{-5}	5.4×10^{-4}
	10	9.2×10^{-5}	4.4×10^{-4}	5.0×10^{-4}

Table 5.4: Error in charge, potential and its normal derivative in Example 2.

Panel	No preconditioning		Preconditioning	
	$\alpha_o = 0$	$\alpha_o = 1$	$\alpha_o = 0$	$\alpha_o = 1$
2,560	48	8	11	6
10,240	84	10	14	8
40,960	> 100	14	17	12

Table 5.5: Effect of mismatch of integral equations of the first and second kinds due to scaling in Example 2. Scaling factor $s = 1.0 \times 10^{-6}$. Accuracy of potential approximation is 10^{-8} .

Panel	No preconditioning			
	iteration		cap. rel. err.	
	$\alpha_o = 0$	$\alpha_o = 1$	$\alpha_o = 0$	$\alpha_o = 1$
2.560	25	8	2.4×10^{-1}	-2.2×10^{-3}
10.240	43	10	5.1×10^{-1}	-1.3×10^{-3}
40.960	80	12	6.7×10^{-1}	-1.3×10^{-3}

Table 5.6: Without preconditioning. Effect of mismatch of the integral equations of the first and second kinds due to scaling in Example 2. Scaling factor $s = 1.0 \times 10^6$. Accuracy of potential approximation is 10^{-8} .

Panel	Preconditioning			
	iteration		cap. rel. err.	
	$\alpha_o = 0$	$\alpha_o = 1$	$\alpha_o = 0$	$\alpha_o = 1$
2.560	6	6	-4.7×10^{-5}	-2.2×10^{-3}
10.240	8	7	-1.8×10^{-3}	-1.3×10^{-3}
40.960	11	10	1.5×10^{-3}	-1.3×10^{-3}

Table 5.7: With preconditioning. Effect of mismatch of the integral equations of the first and second kinds due to scaling in Example 2. Scaling factor $s = 1.0 \times 10^6$. Accuracy of potential approximation is 10^{-8} .

Chapter 6

Numerical Extraction of Capacitance in a-Si TFTs and Imaging Arrays

The main objective of this work is to develop a computational procedure for numerically extracting the capacitance with a high degree of accuracy and in an efficient manner. Although the method presented here can be used for general device structures, our focus is on a-Si TFTs and interconnects in imaging arrays [60]. Following the approach and formulation presented in previous chapters, we now describe results of the EE-based numerical extraction of these parasitic coupling capacitances.

In many situations where direct measurement is not feasible or is too costly and the parallel-plate approximation is unreliable, numerical simulation is the preferred choice for extracting the capacitance. With its flexibility, numerical capacitance extraction can help to gain insight into device behaviour under different geometric

structures and operating conditions. Not only is this useful from the standpoint of establishing design rules for large-area systems, it also facilitates further development of equivalent circuits for effective SPICE-like simulations at the system level. The latter is necessary for design optimization and sensitivity analysis. Comparison of simulation results with those from measurement and the parallel-plate approximation are presented in §§ 6.1 and 6.2, respectively.

Numerical computation can also be used for extracting other geometric parameters. For example, the result from simulation and measurement can be used to extract the overlap length in a-Si TFTs. This is discussed in § 6.3.

As mentioned in Chapter 1, the extreme geometric structures in a-Si imaging electronics require a large number of panels to barely resolve the surfaces and high degrees of accuracy on the evaluation of the potential. This can be problematic and even out of reach for current numerical algorithms in view of the exorbitant memory requirements. These concerns, however, can be addressed effectively using the EE method. The simulation results presented in the sections that follow are obtained from the EE-based method.

6.1 Simulation vs. measurement

To verify the reliability of the simulation, we design test structures for a-Si TFTs and parallel-plate capacitors. The parallel-plate capacitors are used to calibrate the dielectric constant of SiN; its value is 7.15. Due to the limited resolution ($\approx 0.1\text{pF}$) of the available measurement equipment (Keithley, Model 82-DOS simultaneous C-V), these structures had to be oversized. Figure 6.1 shows the layout and a photograph of the fabricated test structures. For the TFTs (see Fig. 1.2 for the cross

section and symbols), the length is $L_{tft}=200\mu m$ and the width is $W_{tft}=400\mu m$; the length of the source and drain is $L_{sd}=60\mu m$; the channel length is $L_{ch}=50\mu m$; the overlap length assumes values: $L_{ov}=15\mu m$, $25\mu m$, and $35\mu m$; and the corresponding lengths of the gate are $80\mu m$, $100\mu m$, and $120\mu m$. The respective layer thicknesses and dielectric constants are listed in Table 6.1.

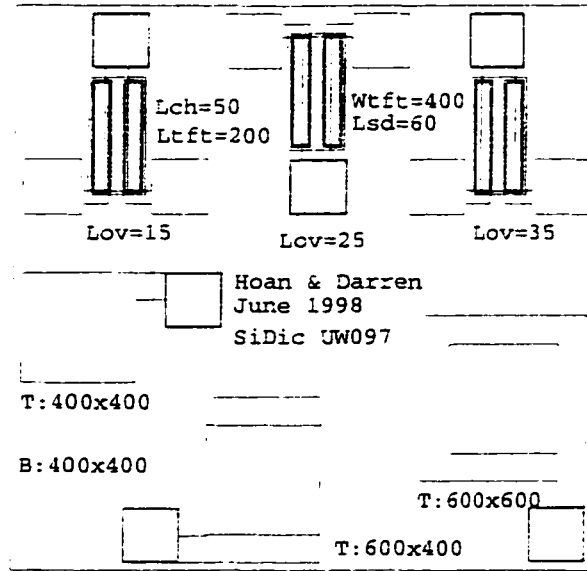
	SiN (gate)	a-Si	SiN (top)	Source & drain metal	Gate metal
Thickness (μm)	0.22	0.05	0.22	1.0	0.12
ϵ_r	7.15	11.9	7.15		

Table 6.1: Thickness of various layers (see Fig. 1.2) and their dielectric constants.

The capacitance of the test structures obtained from measurement, simulation, and parallel-plate approximation are shown in Table 6.2 for the overlap between gate and source/drain (C_{GS}) in the TFT. The results are in close agreement. Factors that may contribute to the discrepancy include (i) the uncertainty of the dielectric constant of SiN, which can range from 3.6 to 7.5 depending on the relative composi-

Overlap length (μm)	Capacitance				
	measurement (pF)	simulation		parallel-plate approximation	
		value (pF)	rel. error	value(pF)	rel. error
15	1.6	1.58	1.25%	1.52	5.00%
25	2.6	2.58	0.08%	2.53	2.62%
35	3.6	3.59	0.03%	3.54	1.54%

Table 6.2: Overlap capacitance in a-Si TFTs as a function of overlap length: values shown are obtained from measurement, simulation, and the parallel-plate approximation.



(a)



(b)

Figure 6.1: Fabricated test structures: (a) layout and (b) photograph.

tion of Si and N, (ii) the uncertainty of the overlap due to errors in patterning, and (iii) the presence of trapped charge as defects in the amorphous material, which has not been accounted for in the model equations.

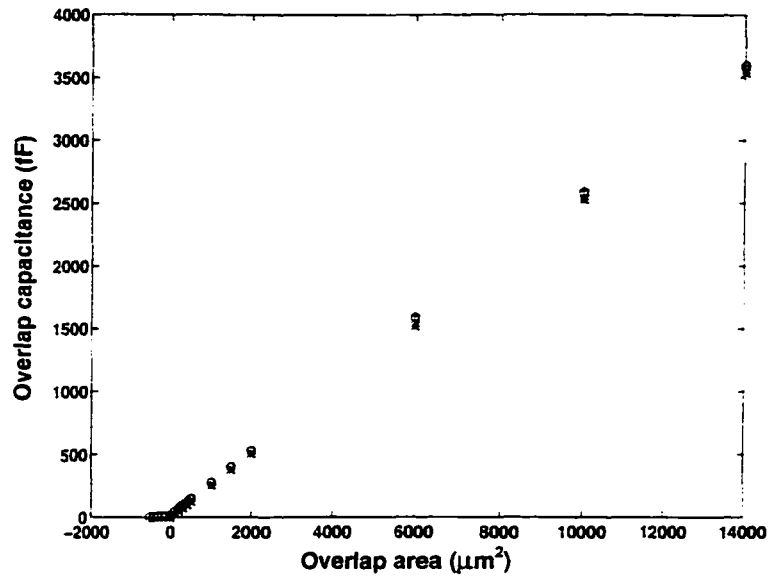
Figure 6.2(a) shows overlap capacitance as a function of overlap area. For large overlap areas, the results obtained from measurement, parallel-plate approximation, and simulation show good agreement. This confirms the reliability of the numerical simulations. For overlap areas in the range of a few hundred μm^2 , the overlap capacitance is too small to be measured. There is a large discrepancy, especially in cases where there is no overlap, between the parallel-plate approximation and simulation as illustrated in Fig. 6.2(b). This is further discussed in the next section.

6.2 Simulation vs. parallel-plate approximation

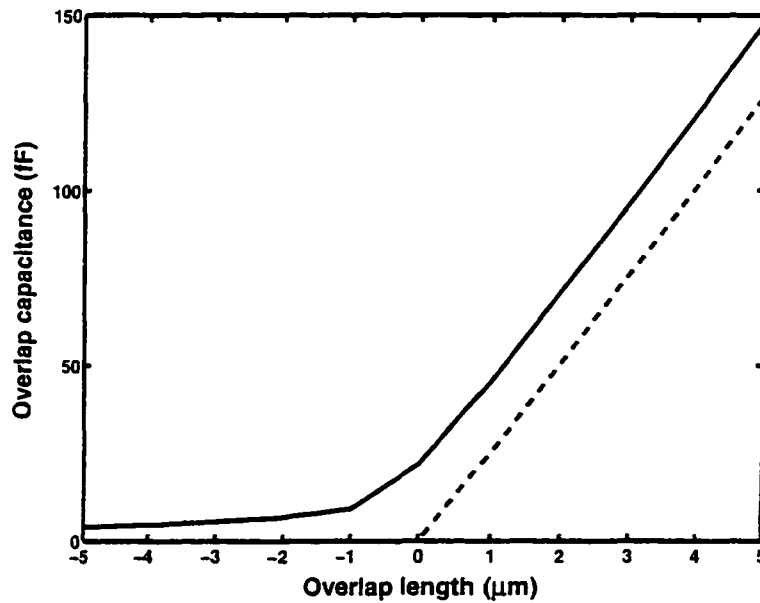
In this section, we compare the results of capacitance extraction obtained from the parallel-plate approximation and numerical simulation. Here, the capacitance computed is associated with gate and source/drain overlap in *a*-Si TFTs and the crossover of addressing lines in *a*-Si imaging arrays. When the overlap length between the gate and source/drain in the TFT is small or when the crossover area of addressing lines is small, the parallel-plate approximation is less reliable due to the dominance of the fringing field in these structures. Furthermore, the capacitance is found to increase with the permittivity of the substrate, which the parallel-plate approximation fails to predict.

6.2.1 Geometric overlap capacitance in *a*-Si TFTs

The number of panels used to discretize the conductor surfaces and dielectric interfaces is nearly 400,000. The length, width, and thickness of the structure considered



(a)



(b)

Figure 6.2: (a) Overlap capacitance (fF) as a function of overlap area (μm^2): measurement(+), parallel-plate approximation (*), and simulation (o). (b) Overlap capacitance (fF) as function of overlap length (μm): parallel-plate approximation (dashed) and simulation (solid); corresponding overlap areas are from $-500\mu\text{m}^2$ to $500\mu\text{m}^2$.

(see Fig. 1.2) in simulations is $100\mu\text{m} \times 100\mu\text{m} \times 11.3\mu\text{m}$. Here, only $10\mu\text{m}$ of the glass substrate ($\epsilon_r=5.84$) (or silicon substrate ($\epsilon_r=11.9$)) is considered so as to reduce panel count. The respective layer thicknesses and dielectric constants are listed in Table 6.1, except that SiN now has dielectric constant of 4.5 and thickness of $0.25\mu\text{m}$. The length of the source and drain regions is $L_{sd}=25\mu\text{m}$ while that of the channel is $L_{ch}=16\mu\text{m}$. The overlap length L_{ov} is varied from $-5\mu\text{m}$ to $20\mu\text{m}$, which corresponds to the gate length from $6\mu\text{m}$ to $56\mu\text{m}$. Here, the negative sign implies no overlap and the associated value denotes the horizontal separation between the gate and the source/drain electrodes.

The values of the geometric overlap capacitance between gate and source (C_{GS}) are listed in Table 6.3 for different L_{ov} . These values are obtained from parallel-plate formula (C_{approx} , column 2) and numerical simulation ($C_{\text{simulation}}$) for glass substrate (column 3) and silicon substrate (column 4). From the simulations, we observe that (i) the capacitance is a non-linear function of L_{ov} , and hence, the overlap area; (ii) even when $L_{ov} \leq 0$, the capacitance is non-trivial; and (iii) the capacitance increases with increasing permittivity of the substrate material.

The discrepancy between simulations and the parallel-plate formula, which increases with decreasing L_{ov} , can be expressed in terms of a fringing factor α :

$$\alpha = \frac{C_{\text{approx}}}{C_{\text{simulation}}} \quad (6.1)$$

The fact that $\alpha < 1$ signifies the presence of a fringing field component in the capacitance, which is not accounted for by the parallel-plate approximation. The fringing factor α is a highly non-linear function of L_{ov} (see Fig. 6.3), particularly

L_{ov} (μm)	C_{approx} (fF)	$C_{simulation}$ (fF)	
		glass($\epsilon_r = 5.84$)	Si($\epsilon_r = 11.9$)
-5	0	3.77	6.21
-4	0	4.82	7.05
-3	0	5.57	9.22
-2	0	6.13	9.68
-1	0	8.16	12.3
0	0	14.4	19.5
1	14.8	30.5	36.6
2	29.6	45.6	51.7
3	44.4	61.1	66.8
4	59.3	76.1	81.7
5	74.1	90.4	96.9
10	148	165	172
15	222	241	247
20	296	314	321

Table 6.3: Geometric overlap capacitances from parallel-plate formula (C_{approx}) and numerical simulation ($C_{simulation}$).

at low values of L_{ov} due to the dominance of the fringing field. This rules out the use of any constant prefactor α_o (i.e., replacing C_{approx} by $\alpha_o C_{approx}$) for better approximation of the true capacitance.

6.2.2 Interconnect crossover capacitance

The crossover capacitance of the gate and data lines (see Fig. 1.1b) is computed as a function of the crossover area. The dielectric layers are the same as in the previous simulation example of the TFT, but with the source and drain now replaced by data lines of $0.12\mu m$ thickness and $16\mu m$ of separation. The gate line, also of $0.12\mu m$ thickness, is on the glass substrate and runs perpendicular to the data lines. The widths of the both gate and data lines are the same and are varied from $1\mu m$ to

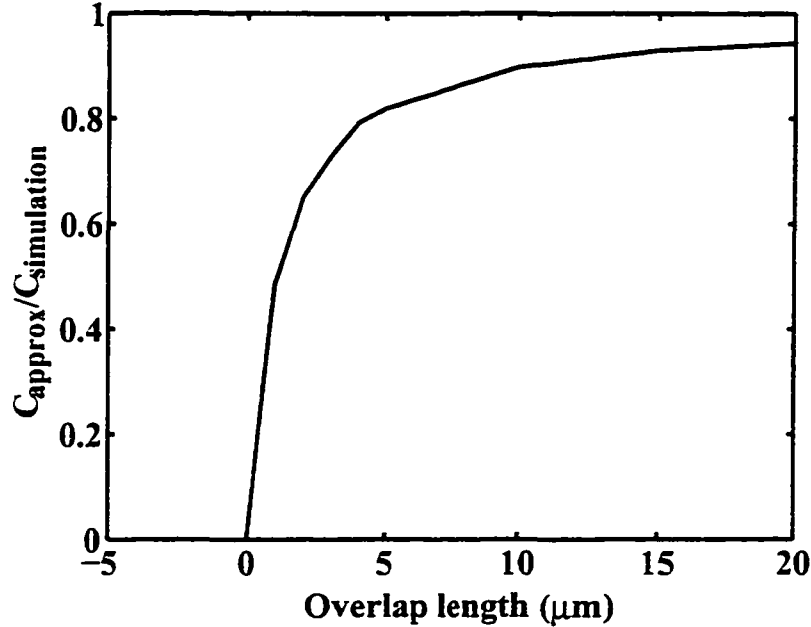


Figure 6.3: Fringing factor $\alpha = \frac{C_{\text{approx}}}{C_{\text{simulation}}}$ as a function of overlap length L_{ov} . The substrate is glass.

$25\mu\text{m}$: the corresponding crossover areas range from $1\mu\text{m}^2$ to $625\mu\text{m}^2$. The values of the crossover capacitance are listed in Table 6.4. They are obtained from numerical simulation (column 2) and from the parallel-plate formula (column 3). Again, as in the previous example, when the crossover area is small, we see the dominance of the fringing field, and hence, a large discrepancy in values (Fig. 6.4). In these structures, the fringing field has a truly two-dimensional character, which is strongly influenced by the close proximity of (intersection) corners. In contrast, for large crossover areas, the effect of the fringing field can be superposed, and following intuitive reasoning, we can construct an approximate edge factor whose value is

Crossover area (μm^2)	Capacitances (fF)	
	$C_{\text{simulation}}$	C_{approx}
1	1.7	0.27
25	9.5	6.70
100	31.6	26.8
225	66.8	60.2
400	115.4	107.0
625	177.1	167.2

Table 6.4: Crossover capacitances (fF) from parallel-plate approximation (C_{approx}) and numerical simulation ($C_{\text{simulation}}$).

the square of that of the previous example (see Fig. 6.3). Here, since $W_{\text{tft}} \gg L_{\text{ov}}$, the fringing field can be perceived to have a one-dimensional character. In this case, however, if the overlap area and crossover area (see Fig. 6.5) are the same, the parallel-plate formula appears to be better for the latter in view of the weaker effect of the fringing field.

6.3 Extraction of overlap length

In comparison with other parameter values such as width, channel length, and thickness, an accurate value of the overlap length (L_{ov}) is difficult to retrieve. Numerical simulation can facilitate the extraction of the gate-source/drain overlap length L_{ov} . For a given process, we calibrate simulations to account for variations in device structure (e.g., layer thicknesses) and physical parameters (e.g., permittivities) induced by the fabrication process. The simulations are then performed for a wide range of L_{ov} . Figure 6.6 shows the overlap capacitance in TFTs computed as a function of L_{ov} . By comparing the values obtained from measurement and the

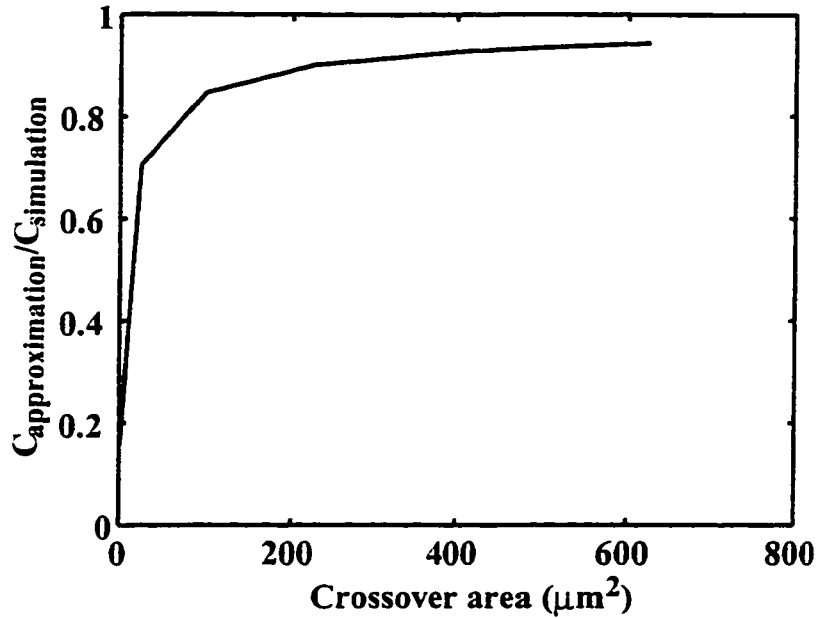


Figure 6.4: Fringing factor $\alpha = \frac{C_{\text{approx}}}{C_{\text{simulation}}}$ as a function of crossover area. The substrate is glass.

calibrated simulation, we can retrieve L_{ov} . Note that with the above procedure, it is possible to retrieve a negative L_{ov} since measurements will yield a non-trivial capacitance due to the fringing field. In this case, $|L_{ov}|$ denotes the horizontal separation between gate and source/drain electrodes.

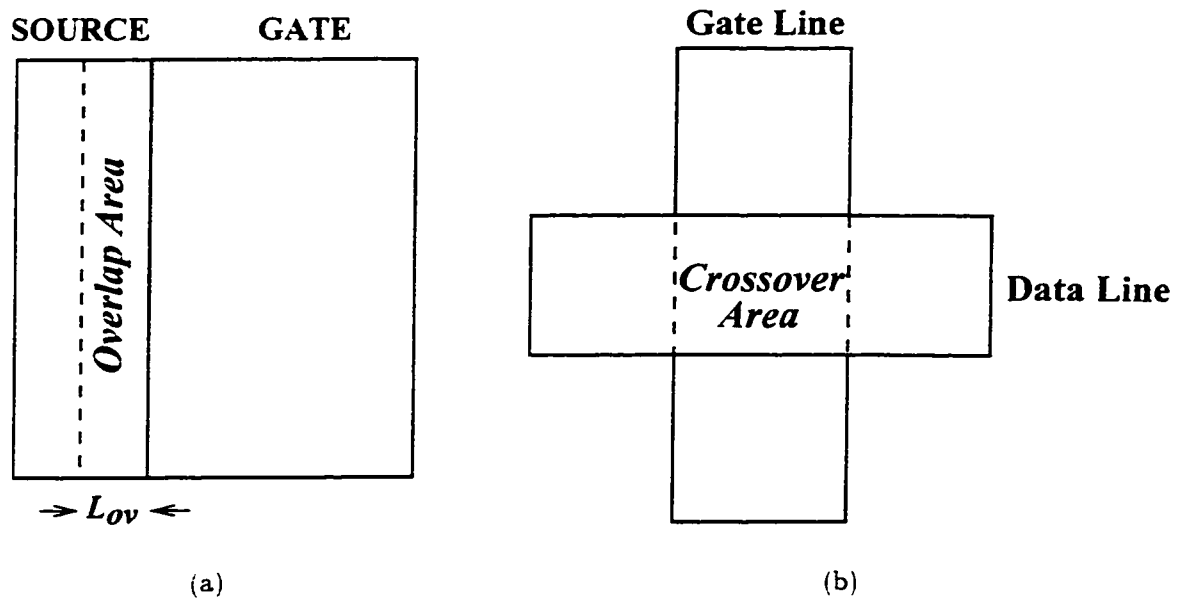


Figure 6.5: (a) Overlap area and (b) crossover area.

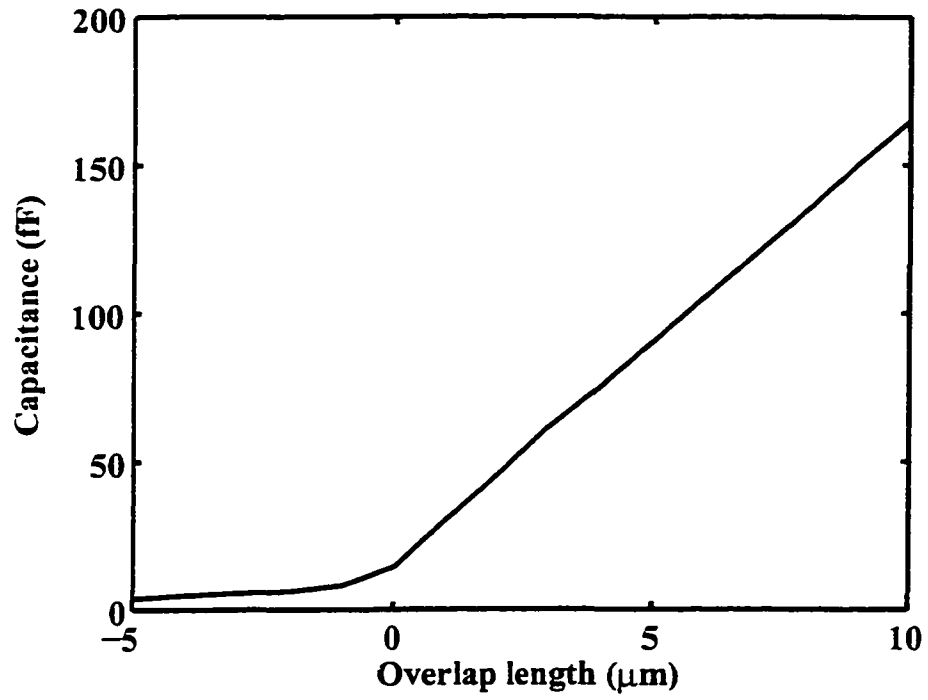


Figure 6.6: Capacitance (fF) as function of overlap length (μm).

Chapter 7

Conclusions

In this thesis, we describe the numerical extraction of parasitic coupling capacitance associated with a-Si TFTs and imaging arrays. The extreme geometries of device structures, the presence of the floating potential of the glass substrate, and the discontinuity of the electric field across the dielectric layers pose severe constraints on numerical methods that need exorbitant computer memory. This makes numerical capacitance extraction for large-area systems a challenging task. We address the above difficulties with a new method based on exponential expansion (EE) of the Green's function $\frac{1}{r}$ for efficient evaluation of the potential field, along with a new formulation for the charge density in a multilayered-dielectric multi-conductor system. With these constructs, we develop an EE-based simulator, which employs a multi-level multi-precision iterative scheme for solution of the charge density and extraction of capacitance.

With the EE method, the evaluation of the potential of an ensemble of N particles as well as its gradient can be performed in linear time ($O(S_{app}N)$) and with

a memory requirement $O(N)$, which is independent of the desired degree of accuracy. The latter feature makes the EE method distinctively unique among the algorithms reported for the well-known family of N-body problems. The EE method also opens new avenues for efficient distributed computing and parallel processing of large-scale N-body simulation of particle systems, ranging from the atomic to the cosmological scale.

As considerations for future work, the use of the EE method can be extended to physical systems in which the exponential expansion can be generated for the associated Green's function. The generation of the expansion can be realized using the randomized algorithm. Alternatively, translations on the expansion of $\frac{1}{r}$ can be manipulated to express such a function; it is a very attractive scheme for dealing with a silicon substrate of fixed potential or glass/polymer substrate of floating potential. However, the use of the randomized algorithm is more general in that once the exponential expansion is available, the EE method can be employed without any modification.

The parasitic coupling capacitance in a-Si systems is also dependent on bias and frequency. To include these dependences in the numerical extraction is an important consideration for future work. Here, we need models that describe the charge trapping and detrapping mechanisms in the a-Si and dielectric layers. These models need to be coupled with the charge transport (current continuity) equation. Hence, a combination of differential-equation formulation (for the volume) and integral-equation formulation (for surfaces) may be needed.

Bibliography

- [1] R. Street, X. Wu, R. Weisfield, S. Ready, R. Apte, M. Nguyen, and P. Nylen. "Two dimensional amorphous silicon image sensor arrays." in *Amorphous Silicon Technology* (M. Hack, E. Schiff, A. Madan, M. Powell, and A. Matsuda, eds.), vol. 377. (Pittsburgh, PA), pp. 757–765, Mater. Res. Soc. Proc., 1995.
- [2] R. S. I. Fujieda, R. Weisfield, S. Nelson, and P. Nylen, "Large area 2-dimensional a-Si:H imaging arrays." in *Amorphous Silicon Technology* (M. Thompson, Y. Hamakawa, P. LeComber, A. Madan, and E. Schiff, eds.), vol. 258, (Pittsburgh, PA), pp. 1145–1150, Mater. Res. Soc. Proc., 1992.
- [3] T. Tsukada. "Amorphous silicon thin-film transistors." *J. Non-Crystalline Solids*, vol. 164–166, pp. 721–726, 1993.
- [4] S. Rao, T. Sarkar, and R. Harrington, "The electrostatic field of conducting bodies in multiple dielectric media," *IEEE Trans. on Microwave Theory and Techniques*, vol. 32, pp. 1441–1448, 1984.
- [5] S. Marshall, R. DuBroff, and G. Skitek. *Electromagnetic Concepts and Applications*. Prentice Hall, 4th ed., 1996.

- [6] P. Dewilde and Z. Ning, *Models for large integrated circuits*. Kluwer Academic Publishers, 1990.
- [7] I. Rubinstein, *Partial differential equations in classical mathematical physics*. New York: Cambridge University Press, 1993.
- [8] R. Kress, *Linear integral equations*. Berlin: Springer-Verlag, 1989.
- [9] C. Brebbia, *Boundary elements : an introductory course*. New York: McGraw-Hill, 1992.
- [10] P. Zhou, *Numerical analysis of electromagnetic fields*. New York: Springer-Verlag, 1993.
- [11] R. Barrett et al., *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM Philadelphia PA, 1994.
- [12] J. Barnes and P. Hut, "A hierarchical $O(N \log N)$ force-calculation algorithm." *Nature*, vol. 324, pp. 446–449, 1986.
- [13] L. Greengard, *The Rapid Evaluation of Potential Fields in Particle Systems*. MIT Press, 1988.
- [14] L. Greengard and V. Rokhlin, "A new version of the fast multipole method for the Laplace equation in three dimension." Tech. Rep. YALEU/DCS/RR-1115. Yale University, September 1996.

- [15] L. Greengard and V. Rokhlin, "A new version of the fast multipole method for the Laplace equation in three dimension," in *Acta Numerica 1997*, pp. 229–269, Cambridge University Press, 1997.
- [16] E. W. Hobson, *The Theory of Spherical and Ellipsoidal Harmonics*. Cambridge University Press, 1955.
- [17] K. Nabors, S. Kim, and J. White, "Fast capacitance extraction of general three dimensional structures," *IEEE Trans. on Microwave Theory and Techniques*, vol. 40, no. 7, pp. 1496–1506, 1992.
- [18] Z. Wang, Y. Yuan, and Q. Wu, "A parallel multipole accelerated 3-d capacitance simulator based on an improved model," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 12, pp. 1441–1450, 1996.
- [19] M. Bächtold, J. Korvink, and H. Baltes, "Enhanced multipole acceleration techniques for the solution of large Poisson computations," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 12, pp. 1541–1546, 1996.
- [20] E. Steinborn and K. Ruedenberg, "Rotation and translation of regular and irregular solid spherical harmonics," in *Advances in quantum chemistry* (P. Löwdin, ed.), pp. 2–82, New York: Academic Press, 1973.

- [21] R. Coifman, V. Roklin, and S. Wandzura. "The fast multipole method for the wave equation: a pedestrian prescription," *IEEE Antennas and Propagation Magazine*, vol. 35, no. 3, pp. 7–12, 1993.
- [22] M. Epton and B. Dembart. "Multipole translation theory for the three-dimensional Laplace and Helmholtz equations." *SIAM Journal of Sci. Comput.*, vol. 16, no. 4, pp. 865–897, 1995.
- [23] H. Pham and A. Nathan. "A randomized scheme to generate Gauss quadratures for Green's function $\frac{1}{r}$ in exponential integral form." Tech. Rep. UW E&CE 97-06, Department of Electrical and Computer Engineering, University of Waterloo. August 1997.
- [24] H. Pham and A. Nathan. "A new approach for efficient and rapid evaluation of potential fields in three dimensions." Tech. Rep. UW E&CE 97-07, Department of Electrical and Computer Engineering, University of Waterloo. August 1997.
- [25] H. Pham and A. Nathan. "Rapid evaluation of gradient of potential in three dimensions," Tech. Rep. UW E&CE 97-09, Department of Electrical and Computer Engineering, University of Waterloo, Sept 1997.
- [26] H. Pham and A. Nathan. "Panel method for field computation using exponential expansion." Tech. Rep. UW E&CE 97-10, Department of Electrical and Computer Engineering, University of Waterloo, Sept 1997.

- [27] H. Pham and A. Nathan, "Solving the charge density problem using exponential expansion," Tech. Rep. UW E&CE 98-02, Department of Electrical and Computer Engineering, University of Waterloo. January 1998.
- [28] H. Pham and A. Nathan, "An integral equation of the second kind for the charge density problem in homogeneous and multiple dielectric media." Tech. Rep. UW E&CE 98-03, Department of Electrical and Computer Engineering, University of Waterloo. January 1998.
- [29] H. Pham and A. Nathan, "Rapid evaluation of the potential fields in three dimensions using exponential expansion." *Canadian Journal of Physics*, vol. 75, pp. 689–693. 1997.
- [30] H. Pham and A. Nathan, "A new approach for rapid evaluation of the potential field in three dimensions." *Proceedings of Royal Society London A*, to appear.
- [31] M. Abramowitz and I. Stegun, *Handbook of Mathematical Functions*. Applied Mathematics Series, Washington DC: National Bureau of Standard, 1970.
- [32] L. Debnath, *Integral Transforms and Their Application*. CRC Press, 1995.
- [33] G. Watson, *A Treatise of the Theory of Bessel functions*. Cambridge University Press, 2nd ed., 1966.
- [34] M. Cecchi and E. Pirozzi, "A recursive algorithm by the moments method to evaluate a class of numerical integrals over an infinite interval." *Numerical Algorithms*, vol. 10, pp. 155–165, 1995.

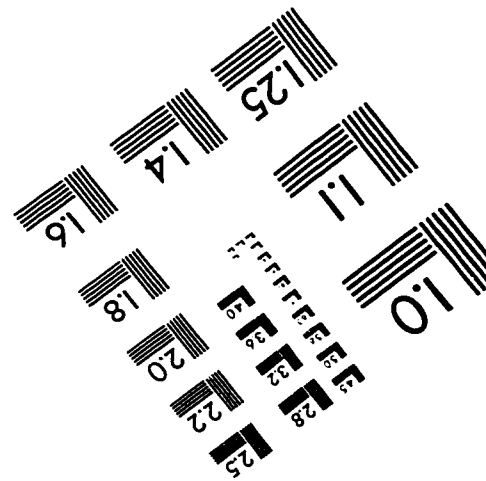
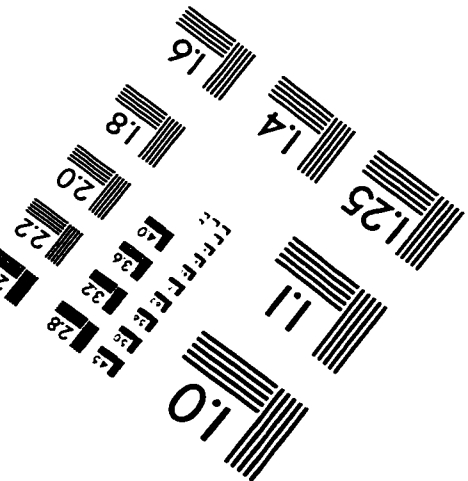
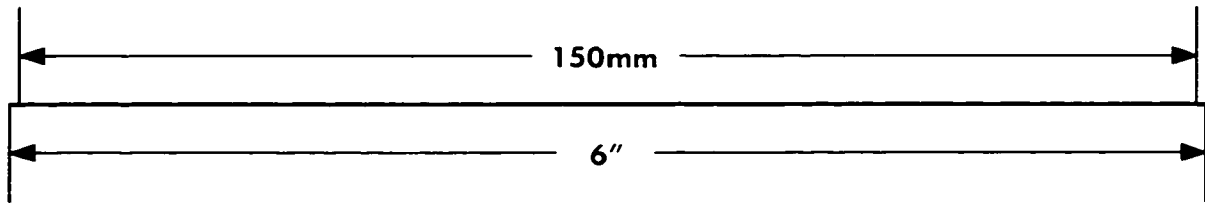
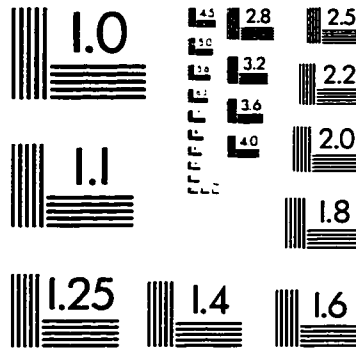
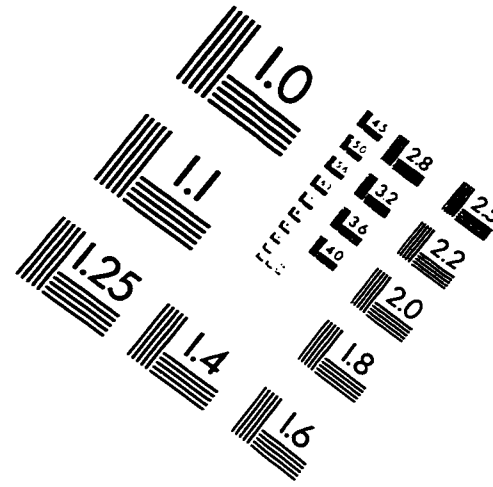
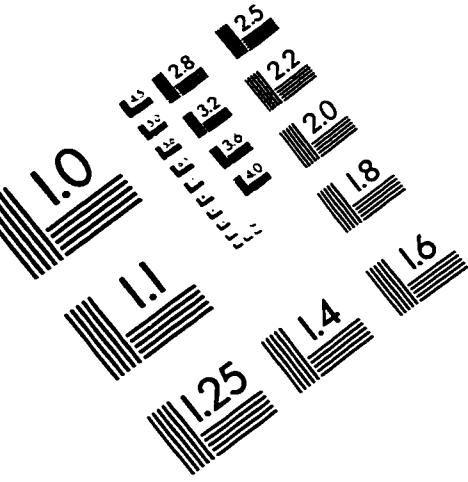
- [35] G. A. Evans, "Integrating oscillatory integrands of a non-trigonometric type over a finite range," *Intern. J. Computer Math.*, vol. 42, pp. 213–221, 1992.
- [36] S. Gustafson, "Quadrature rules derived from linear convergence acceleration schemes," in *Numerical Integration IV* (H. Brass and G. Hämmerlin, eds.), pp. 151–165, Birkhäuser, 1992.
- [37] A. Sidi, "Computation of infinite integrals involving bessel functions of arbitrary order by the \bar{D} -transformation," *Journal of Computational and Applied Mathematics*, vol. 78, pp. 125–130, 1997.
- [38] V. Rokhlin and N. Yarvin, "Generalized Gaussian quadratures and singular value decomposition," Tech. Rep. YALEU/DCS/RR-1109, Yale University, May 1996.
- [39] H. Pham and A. Nathan, "Generating Gauss quadratures for Green's function $\frac{1}{r}$: a randomized algorithm," in *IEEE 1998 Canadian Conference on Electrical and Computer Engineering*, (Waterloo, ON, Canada), pp. 657–660, May, 25–28 1998.
- [40] H. Pham and A. Nathan, "Computation of the potential field and its gradient using exponential expansion," in *IEEE 1998 Canadian Conference on Electrical and Computer Engineering*, (Waterloo, ON, Canada), pp. 750–753, May, 25–28 1998.
- [41] R. F. Harrington, *Field Computation by Moment Methods*. IEEE Press, New Jersey, 1993.

- [42] H. Petersen, E. Smith, and D. Soelvason, "Error estimates for the fast multipole method. II. the three-dimensional case." *Proc. R. Soc. Lond. A.* vol. 448, pp. 401–418, 1995.
- [43] W. Elliot. *Multipole algorithms for molecular dynamics simulation on high performance computers.* PhD thesis, Duke University, 1995.
- [44] H. Pham and A. Nathan. "Exponential expansion for rapid and accurate extraction of interconnect capacitance." in *1998 International Conference on Simulation of Semiconductor Processes and Devices* (in press). (Leuven, Belgium), September, 2–4 1998.
- [45] K. Atkinson. *The Numerical Solution of Integral Equations of the Second Kind.* Cambridge University Press, 1997.
- [46] Y. Saad and M. Schultz. "GMRES: a generalized minimum residual algorithm for solving nonsymmetric linear systems." *SIAM J. Sci. Stat. Comput.*, vol. 7, pp. 856–869, 1986.
- [47] J. Hess and A. Smith, "Calculation of potential flow about arbitrary bodies." *Progress in Aeronautical Sciences.* pp. 1–138, 1966.
- [48] J. Newman. "Distribution of sources and normal dipoles over a quadrilateral panel." *Journal of Engineering Mathematics*, vol. 20, pp. 113–126, 1986.
- [49] E. Goto, Y. Shi, and N. Yoshida, "Extrapolated surface charge method for capacity calculation of polygons and polyhedra." *Journal of Computational Physics*, vol. 100, pp. 105–115, 1992.

- [50] I. Gradshteyn and I. Ryzhik. *Tables of integrals, series, and products*. Academic Press. 5th ed., 1996.
- [51] S. Vavasis, "Preconditioning for boundary integral equations." *SIAM J. Matrix Analysis and Applications*, vol. 13, no. 3, pp. 905–925, 1992.
- [52] F. L. K. Nabors, F. Korsemeier and J. White. "Preconditioned, adaptive, multipole-accelerated iterative methods for three-dimensional potential integral equation of the first kind," *SIAM J. Sci. Stat. Comput.*, vol. 15, no. 3, pp. 713–735, 1994.
- [53] W. Briggs, *A Multigrid Tutorial*. Philadelphia: SIAM, 1987.
- [54] H. Pham and A. Nathan, "Rapid extraction of capacitance in a-Si imaging arrays." in *IEEE International Symposium on Circuits and Systems*. (Monterey, CA), June, 1–3 1998.
- [55] O. Kellogg, *Foundations of Potential Theory*. Springer-Verlag, 1929.
- [56] S. Mikhlin, *Integral Equations*. Pergamon Press, 1957.
- [57] G. Baker and J. Shelley, "Boundary integral techniques for multi-connected domains," *J. Computational Physics*, vol. 64, pp. 112–132, 1986.
- [58] L. G. A. Greenbaum and G. McFadden, "Laplace's equations and the Dirichlet-Neumann map in multiply connected domains," *J. Computational Physics*, vol. 105, pp. 267–278, 1993.

- [59] H. Pham and A. Nathan. "A new formulation for accurate numerical extraction of interconnect capacitance in ULSI." in *5th IEEE International Conference on Electronics and Systems* (in press), (Lisbon, Portugal), September, 7-10 1998.
- [60] H. Pham and A. Nathan. "Interconnect capacitance extraction in large-area a-Si imaging systems." in *Material Research Society Symposium Proceedings* (in press), vol. 507, (San Fransisco, CA), April, 12-17 Spring, 1998.

IMAGE EVALUATION TEST TARGET (QA-3)



APPLIED IMAGE, Inc
 1653 East Main Street
 Rochester, NY 14609 USA
 Phone: 716/482-0300
 Fax: 716/288-5989

© 1993, Applied Image, Inc., All Rights Reserved