

MASTER THESIS

POISSON EQUATION SOLVER IMPLEMENTATION FOR USE IN GLOBAL PLACEMENT ALGORITHMS

Student:

[Athanasia Karanika](#)
atkaranika@uth.gr

Supervisor:

[Christos Sotiriou](#)
chsotiriou@e-ce.uth.gr

Committee:

[Georgios Stamoulis](#)
georges@e-ce.uth.gr
[Fotios Plessas](#)
fplessas@e-ce.uth.gr

*A thesis submitted in fulfillment of the requirements
for the degree of Master
in the*

**Circuits & Systems Laboratory (CASlab)
Department of Electrical and Computer Engineering**

October 25, 2022

Abstract

Many physical problems in engineering can be modeled mathematically. In general, these problems are governed by partial differential equations. The goal of this thesis is to provide a solution for Poisson partial differential equation with small execution time and with high precision and examine the conditions that have to be satisfied in order for Poisson equation to have a solution which is unique. Poisson equation has broad utility in numerous areas. For example it is used for gravitational potential, for electrostatics where Poisson equation relates the charge density to the second derivatives of potential and for pressure field in incompressible fluid flow. The interest of this thesis is the use of Poisson equation in Electronic Design Automation(EDA) tools and specifically in global placement stage. Placement is a very important step of EDA tools and it highly affects the quality of circuit, since it determines the positions of the gates, taking into account some constraints. For the rest of this thesis the gates are called cells. This work is integrated to our custom EDA tool called ASP, in which there are already some solutions. To date, the solution that gives the most desirable results uses the Math Kernel Library(MKL) to solve the Poisson equation. The initial goal of this thesis was to have the same or better results than MKL so as to have a Poisson solver without external library. We have developed 2 approaches, the first is based on Finite Differences and the second is based on Spectral methods. Advantages and disadvantages of both methods are examined in comparison to existing methods in our tool.

Περίληψη

Πολλά φυσικά προβλήματα στην μηχανική μπορούν να μοντελοποιηθούν μαθηματικά. Γενικά, αυτά τα προβλήματα διέπονται από μερικές διαφορικές εξισώσεις. Ο στόχος αυτής της διπλωματικής εργασίας είναι να παρέχουμε μια λύση για την μερική διαφορική εξίσωση Poisson με χαμηλό χρόνο εκτέλεσης και υψηλή ακρίβεια και να εξετάσουμε τις συνθήκες που πρέπει να ικανοποιούνται ώστε η Poisson εξίσωση να έχει λύση και αυτή η λύση να είναι μοναδική. Η εξίσωση Poisson έχει ευρεία χρησιμότητα σε πολλούς τομείς. Για παράδειγμα χρησιμοποιείται για την εύρεση του βαρυτικού δυναμικού, στην ηλεκτροστατική όπου η εξίσωση Poisson συσχετίζει την πυκνότητα φορτίου με τις δεύτερες παραγώγους του δυναμικού και για το πεδίο πίεσης στην ροή ασυμπίεστου υγρού. Το ενδιαφέρον της παρούσας διπλωματικής είναι η χρήση της εξίσωσης Poisson στα Electronic Design Automation(EDA) εργαλεία και συγκεκριμένα στο στάδιο της καθολικής τοποθέτησης. Η τοποθέτηση είναι ένα πολύ σημαντικό βήμα των εργαλείων EDA και επηρεάζει πολύ την ποιότητα του κυκλώματος, αφού καθορίζει τις θέσεις των πυλών, λαμβάνοντας υπόψη ορισμένους περιορισμούς. Για το υπόλοιπο αυτής της εργασίας οι πύλες ονομάζονται κελιά. Αυτή η εργασία είναι ενσωματωμένη στο δικό μας προσαρμοσμένο εργαλείο EDA που ονομάζεται ASP, στο οποίο υπάρχουν ήδη κάποιες λύσεις. Μέχρι σήμερα, η λύση που δίνει τα επιθυμητά αποτελέσματα χρησιμοποιεί την Math Kernel Library(MKL) για να λύσει την εξίσωση Poisson. Ο αρχικός στόχος αυτής της εργασίας ήταν να έχει τα ίδια ή καλύτερα αποτελέσματα από την MKL, ώστε να έχουμε έναν Poisson solver χωρίς εξωτερική βιβλιοθήκη. Εχουμε αναπτύξει 2 προσεγγίσεις, η πρώτη βασίζεται σε πεπερασμένες διαφορές και η δεύτερη σε φασματικές μεθόδους. Τα πλεονεκτήματα και τα μειονεκτήματα τις κάθε μεθόδου εξετάζονται σε σχέση με τις υπάρχουσες μεθόδους στο εργαλείο μας

Acknowledgements

I would like to thank my thesis supervisor Prof. Christos Sotiriou who was always willing to help me whenever i had a question or a problem regarding my research.

I would like also to thank the members of the CASlab, who were always friendly and willing to provide assistance.

Finally, I would like to express my acknowledgements to my parents and my friends supporting me throughout my years of study at the University of Thessaly

Athanasia Karanika
Volos, 2022

Poisson Equation Solver Implementation for use in Global Placement Algorithms

Athanasia Karanika
atkaranika@uth.gr

Copyright © Athanasia Karanika 2022

Με επιφύλαξη παντός δικαιώματος. All rights reserved.

Υλοποίηση Επίλυσης Εξίσωσης Poisson για χρήση σε Αλγόριθμους Καθολικής Τοποθέτησης

Αθανασία Καρανίκα
atkaranika@uth.gr

Copyright © Αθανασία Καρανίκα 2022

Με επιφύλαξη παντός δικαιώματος. All rights reserved.

Contents

1	Introduction	5
2	Theoretical Background	6
2.1	Introduction to EDA	6
2.2	Placement	7
2.2.1	Global Placement	7
2.2.2	Legalization	9
2.2.3	Detailed Placement	9
2.3	Analytic Problem Formulation	10
2.3.1	Net Modeling	12
2.4	Force-Directed Placement	14
2.4.1	Kraftwerk2	14
2.5	Poisson Equation	20
2.5.1	Finite Differences	22
2.5.2	Spectral Methods	25
3	Existing Works	28
3.1	ADI Method	28
3.2	4 Points Modified Explicit Group Method	30
3.3	Cyclic Reduction Method	33
3.4	MathWorks MATLAB	35
3.5	Existing work in ASP	37
4	Poisson Solution - Contribution of this Work	39
4.1	Finite Differences Approach	39
4.2	Cosine Transform Approach	45
5	Experimental Results	47
5.1	MATLAB	47
5.2	Our Custom Tool	52
5.2.1	Flow	52
5.2.2	Matrix Approach Results	52
5.2.3	FFT Approach Results	55
6	Conclusion and Future work	59
A	Acronyms	60
B	Designs Layouts	61

B.1	AES	61
B.2	ca53dpu	63
B.3	apdtoaes128	64
B.4	PID	66
B.5	LDPC	67

List of Figures

2.1 Moore's Law from 1970 till 2020	6
2.3 Most Common techniques for global placement	8
2.4 common placers	9
2.5 Global Placement Example	11
2.6 Star Model	13
2.7 Clique Model	13
2.8 HPWL Net Model	14
2.9 Placement example	15
2.10 density demand supply at y and x axis	16
2.11 density demand supply for both y and x axis	16
2.12 Illustration of a placement iteration. The number of arrows represents the order of steps. The hold force(b) and the Demand Supply system(d) are identified by initial placement(a). Potential Φ (e) is identified by (b) and (d). In (d),(e) black and white areas represent high and low density respectively. The move force(f) results from Φ	19
2.13 Poisson general domain	21
2.14 rectangular discretized domain	22
2.15 finite differences domain	23
2.16 Dirichlet boundary conditions on 4x4 grid	25
3.1 Solution domain for the four points EG and EDG methods	31
3.2 Stencil of each method	32
3.3 Solution domain for the four points MEDG methods	33
4.2 5-points finite differences stencil	41
4.4 ghost points	42
4.5 weights modification for corner points	43
4.6 weights modification for non-corner point	44
5.1 All the cells are in upper left corner of density map	48
5.2 All the cells are in upper right corner of density map	48
5.3 All the cells are in lower right corner of density map	49
5.4 All the cells are in lower left corner of density map	49
5.5 All the cells are in the middle of density map	50
5.6 cells are opposite horizontally	50
5.7 cells are opposite vertically	51
5.8 Experiments Flow	52
B.1 AES: Global Placement	61

B.2	AES: Global Placement with number of bins equal to the previous power of 2 from default	62
B.3	AES: Global Placement with number of bins equal to the next power of 2 from default	62
B.4	ca53dpu: Global Placement	63
B.5	ca53dpu: Global Placement with number of bins equal to the previous power of 2 from default	63
B.6	ca53dpu: Global Placement with number of bins equal to the next power of 2 from default	64
B.7	apdtoaes: Global Placement	64
B.8	apdtoaes: Global Placement with number of bins equal to the previous power of 2 from default	65
B.9	apdtoaes: Global Placement with number of bins equal to the next power of 2 from default	65
B.10	PID: Global Placement	66
B.11	PID: Global Placement with number of bins equal to the previous power of 2 from default	66
B.12	PID: Global Placement with number of bins equal to the next power of 2 from default	67
B.13	LDPC: Global Placement	67
B.14	LDPC: Global Placement with number of bins equal to the previous power of 2 from default	68
B.15	LDPC: Global Placement with number of bins equal to the next power of 2 from default	68

Chapter 1

Introduction

Poisson equation

Poisson equation is an elliptic partial differential equation [1] with extensive utility in areas of physics. There are numerous physical phenomena expressed by Poisson Equation. Most common are [2] [3] [4]:

- Poisson Equation for gravity:

$$\nabla^2 \phi = -Q$$

Where Φ is the gravitational potential and Q depends on gravity and is known.

- Poisson Equation for Electrostatics:

$$\nabla^2 V = -4\pi\rho$$

where V is the potential and ρ is the charge density distribution.

- Poisson Equation for pressure in incompressible fluid flow:

$$\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} = -\frac{\partial}{\partial x} \left(u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} \right) - \frac{\partial}{\partial y} \left(u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} \right) + \frac{1}{\Delta t} \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right)$$

where u, v are the velocity components in x and y direction respectively and p is the pressure divided with the density

- Poisson equation is used in ASP EDA tool in global placement stage

First an introduction to EDA tools is given, then global placement algorithms are presented briefly and the kraftwerk2, which is used in ASP, is analysed in order to make it clear where Poisson equation finds application. Next we will see the solution provided from literature. After that, it is discussed how desired boundary conditions can be satisfied by applying some modifications to traditional finite differences methods. Poisson solution provided by spectral methods exists in literature and is implemented and evaluated firstly in Mathworks MATLAB and then is integrated to our custom tool ASP.

The results of our solvers are tested and compared with existing solvers in ASP. Now a brief explanation of the necessity of EDA tools is given.

Chapter 2

Theoretical Background

2.1 Introduction to EDA

Electronic design Automation(EDA) is a type of specialised software tools developed in order to design electronic systems such as integrated circuits(ICs). ICs have provided a breakthrough in the semiconductor technology, by playing a major role in a variety of fields. For instance, they are used in several devices such as radars, televisions, calculators and audio amplifiers. Such was the revolution of ICs performance that in accordance with Moore's law, as depicted in Fig 2.1 the number of transistors per integrated circuit has doubled every two years. This sharp development, coupled with satisfyingly small size of transistors have resulted in the construction of admirable, smart and useful devices comprised numerous transistors. So, it is quite evident that the more performance of EDA tools improves the more design process of very large-scale integrated (VLSI) circuits is facilitated.

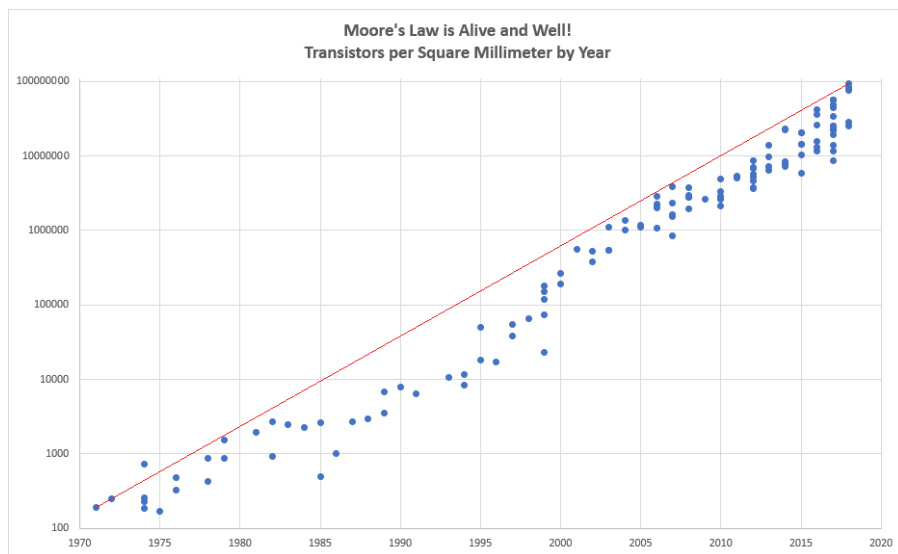
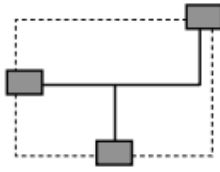


Figure 2.1: Moore's Law from 1970 till 2020

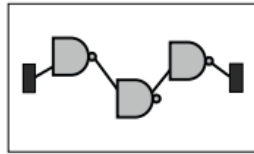
Fabrication of contemporary ICs is a complicated procedure and for this reason EDA tools have contributed significantly in its every stage. EDA tools carry out required functionalities such as simulation, placement and routing, as well as verification of the circuit. The goal is to have an automated complete design flow from high level design to fabrication, in conjunction with optimal results in some metrics such as power, performance and area. The high complexity of modern ICs established EDA almost in all stages of ICs' design flow. EDA tools contain methodologies for physical design, simulation, verification, synthesis, and testing [5]. The work of this thesis is integrated in Placement step which is explained briefly.

2.2 Placement

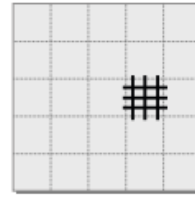
Placement constitutes a vital and complicated step of Physical design flow. The objective of placement is to determine the locations and orientations of all circuit elements within a layout, while addressing optimization metrics such as total wirelength, power, congestion etc. These objectives can be satisfied by some cost functions that are measured during the placement procedure. Some of the most known metrics are [6]:



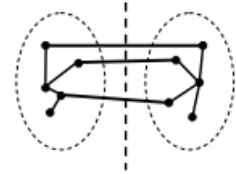
(a) *Total wirelength:* Most of the contemporary placers set as goal to minimize the sum of the length of all the wires in the design. Not only does this metric affect the chip size but also the power and wire delays as they depend on the wirelength



(b) *delay:* The clock cycle of a chip is determined by the delay of its longest path, usually is referred to as critical path. Given a performance specification, a placer must ensure that no path exists with delay exceeding the maximum specified delay



(c) *congestion:* It is necessary to meet the routing resources within various local regions of the core area. A congested region might lead to excessive routing detours, or make it impossible to complete all routes



(d) *Number of cut nets:* The area is divided by cutlines, after finding the cutline with the most crossings, it is attempted to minimize the crossings in order to reduce the local congestion

Most of the modern placers are able to handle these requirements. Placement procedure includes global placement, legalization and detailed placement which are explained briefly below.

2.2.1 Global Placement

Global Placement is the first step of placement. It attempts to give an initial location to cells, without taking into account their dimension, shape and size, thus considering them as points. This stage of placement tolerates overlap up to a point, which is specified by each global placement algorithm. Overlap is eliminated in the

legalization step. In the event that global placer does not meet some metric optimizations, wirelength may be excessive and manufacturing of chip may be unfeasible. There are several techniques for circuit placement summarized as follows [6].

In partition based algorithms the system is decomposed into two or more smaller parts. That is the netlist of design as well as the layout are divided into smaller sub-netlists and sub-regions respectively. Consecutively, the sub-netlists and sub-regions are divided again into smaller ones and this procedure is continued until each sub-region contains a sufficient small number of cells. A typical example of placement that uses partition based algorithms is the Min Cut placement which divides the regions into sub regions with the goal of unique netlist for each sub-region. It tries to eliminate as much as possible the connection between the regions, thus reducing their interdependence.

In analytic algorithms an objective function is used, which can be minimized or maximized via mathematical analysis. An objective can be quadratic or non-convex. A common aim of this kind of algorithms is to minimize the total wirelength. Quadratic Placement uses as cost function the squared Euclidean distance of the centers of connected components and attempts to minimize it. With this metric the placement problem becomes a convex quadratic optimization problem, that is it has a unique global minimum that can be found by equalizing the partial derivatives of cost function with zero. Force-directed placement is a special case of quadratic placement which models the cells and wires as a mass-spring system. A cell exercises attraction force to another cell which is proportional to their distance. In this case the total potential energy of the springs, which is a quadratic function of their length, is minimized to produce a placement solution. This is equivalent to a force equilibrium state in the spring system

In stochastic algorithms, randomized moves are used to optimize the cost function. An example of this approach is the simulated annealing, which at each step generates a new placement by perturbing the current placement. Acceptance of new placement depends on the difference between the wire length of the current and of the potential placement. If the new placement enhances the wire length, then is accepted otherwise is probabilistically accepted. In this algorithm, this possibility is expressed by temperature. Algorithm initially sets the temperature to a high value and in each iteration decreases it. High value is necessary due to annealing procedure. These algorithms often find the global minimum but suffer from long run times

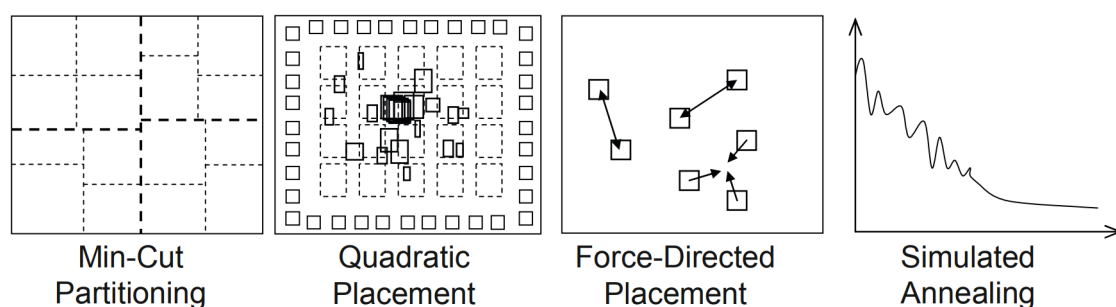


Figure 2.3: Most Common techniques for global placement

Fig 2.4 shows the categorization of each placement according to which of the three techniques it uses. ASP is a full flow EDA tool with many functionalities such as global placement, legalization, detailed placement, static timing analysis etc. For the experiments of this thesis, global placement and legalization stages are used. [7].

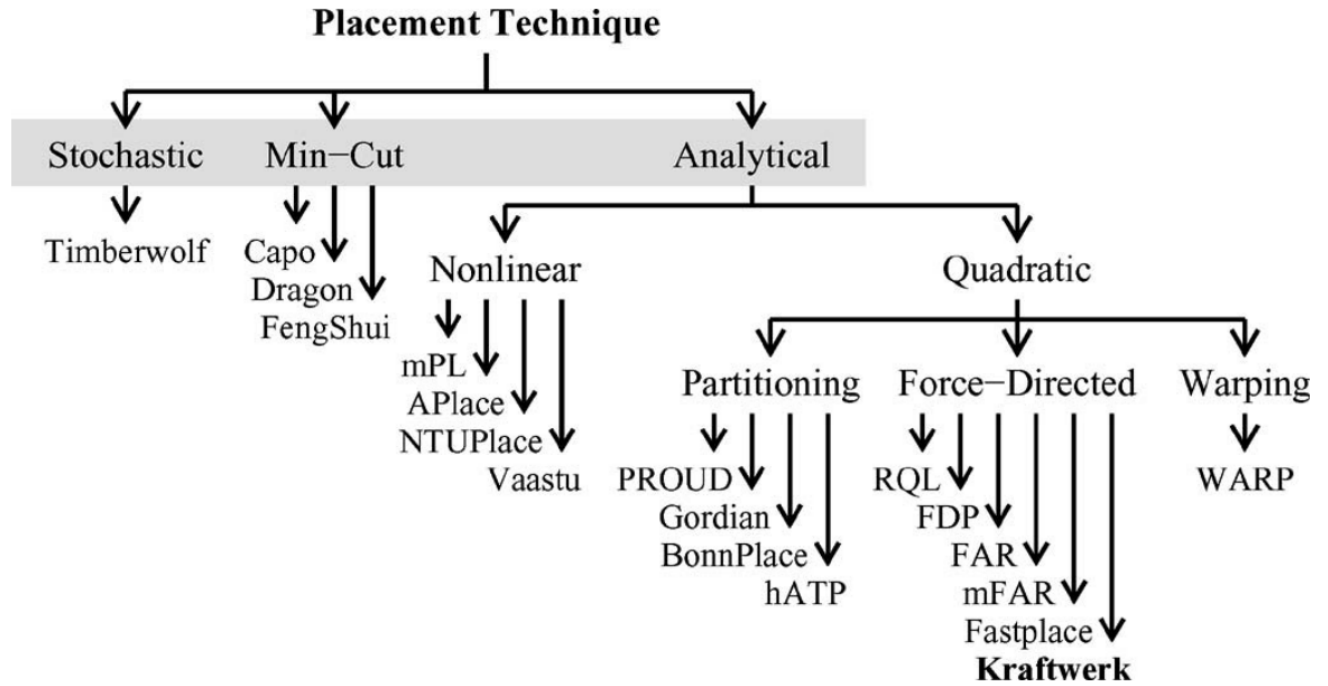


Figure 2.4: common placers

In ASP tool, Kraftwerk2 force directed placer is implemented in which our work is integrated.

2.2.2 Legalization

Global placement considers cells dimensionless, not taking into account their dimension, shape and size. As a result it places the cells with overlap. Legalization is performed after Global Placement in order to find legal, non-overlapping placements for all placeable modules. It tries to align the location of standard cells with power rails, while attempting to make the less possible changes to placement, thus minimizing the impact to total wirelength and delay. Not only is legalization necessary after global placement but also after incremental changes such as cell resizing and buffer insertion.

2.2.3 Detailed Placement

A common problem after legalization algorithms is that due to the fact that some cells may travel a long distance, the wirelength and hence, delay may be increased. Detail placement optimizes placement by reordering cells that are not adjacent, as

well as by utilizing unused space between cells placed in a row. The objective is specified by each algorithm of detailed placement

2.3 Analytic Problem Formulation

Analytic placement minimizes a given objective such as wirelength or circuit delay, using mathematical techniques such as numerical analysis or linear programming. The objective function represents the cost function. Analytical methods often require cost function to be differentiable because minimization procedure is deduced to equalization of derivative with zero. This is the reason why quadratic function is preferred which is the squared euclidean distance [6]

$$\Gamma(x, y) = \frac{1}{2} \sum_{p=1}^n \sum_{q=1}^n c(p, q) \left((x_p - x_q)^2 + (y_p - y_q)^2 \right) \quad (2.1)$$

This formulation replaces multipin nets with two-pin nets. n is the total number of cells, x and y are the coordinates of each component in x and y axis respectively, $(x_p - x_q)^2$ and $(y_p - y_q)^2$ represent the horizontal and vertical squared distance respectively. This model focuses on reducing the long connections considering that they are responsible for the high cost. $c(p, q)$ is the connection cost between p and q . If two cells are not connected, $c(p, q)$ is 0. These costs introduce the connectivity matrix $\mathbf{C} = [c_{p,q}]$. Equation (2.1) should be expressed in matrix-vector notation in order to be passed to a linear equation solver. Each dimension can be analysed independently. We focus on x -direction and y -direction is obtained similarly: [6]

$$\Gamma(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{L} \mathbf{x} \quad (2.2)$$

x , y are n -vectors that specify the coordinates of n cells, $\mathbf{L}$ Laplacian matrix is the modified connectivity matrix with $\mathbf{L}(\mathbf{p}, \mathbf{q}) = -\mathbf{C}(\mathbf{p}, \mathbf{q})$ when $p \neq q$, and $\mathbf{L}(\mathbf{p}, \mathbf{p}) =$ the sum of incident connection weights of cell p

In chip area there are input/output(I/O) cells that are not movable [7]. These cells have not been taken into consideration so far. The cost function for these cells is the same with the one from equation (2.1) but with x_f which is the x -coordinate of the fixed cell to be known constant. Hence the cost function in x -axis is:

$$c_{pf}(x_p - x_f)^2 = c_{pf}x_p^2 - 2c_{pf}x_px_f + c_{pf}x_f^2 \quad (2.3)$$

The goal is to integrate these connections in the existing system of equation (2.2). Given that the existing Laplacian matrix has already information about connections of p movable cell, the first term of equation (2.3) contributes to the system of equation (2.2) by just adding c_{pf} to $\mathbf{L}$ diagonal at p row. As for the second term is considered, x_f is known so it can be written with vector notation as $\mathbf{b}^T \mathbf{x}$, where $\mathbf{b}$ vector contains all the costs of unmovable cells connections c_{pf} . The last term is a constant. So cost function in x -axis after taking into account connections of unmovable cells becomes:

$$\Gamma(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T \mathbf{L} \mathbf{x} + \mathbf{b}^T \mathbf{x} + \text{const} \quad (2.4)$$

Generally for every connection, cost function changes as follows

- If both p, q are movable cells then $c_{p,q}$ is added to the diagonal of Laplacian matrix $\mathbf{L}$ and is subtracted from $L_{p,q}$ and $L_{q,p}$ off diagonal entries
- If one of the cells e.g p is fixed then $c_{q,p}$ is added to L_{qq} and $c_{p,q}x_p$ is subtracted from vector entry b_q
- If both cells are fixed, the Laplacian matrix $\mathbf{L}$ and the vector $\mathbf{b}$ are not affected

An example follows to illustrate the theory. Let it be a circuit with 2 fixed cells, f1, f2, whose coordinates are (100,175) and (200,225) respectively, 3 cells to be placed as depicted in figure 2.5. and the following nets.

$$\begin{aligned} N1(f_1, a) \\ N2(a, b) \\ N3(b, c) \\ N4(c, f_2) \end{aligned}$$

There are 3 cells to be placed so the connectivity matrix is 3x3. The goal is to find the coordinates of a, b, c. This can be performed in x-axis and y-axis separately.

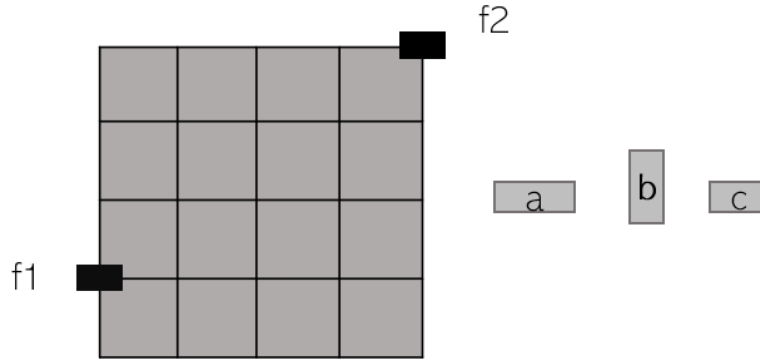


Figure 2.5: Global Placement Example

For x-axis the connectivity matrix $\mathbf{C}$ is:

$$\begin{bmatrix} 0 & c_{a,b} & 0 \\ c_{a,b} & 0 & c_{b,c} \\ 0 & c_{b,c} & 0 \end{bmatrix} \quad (2.5)$$

Hence, $\mathbf{L}$ matrix is:

$$\begin{bmatrix} c_{a,b} + c_{a,f_1} & -c_{a,b} & 0 \\ -c_{a,b} & c_{a,b} + c_{b,c} & -c_{b,c} \\ 0 & -c_{b,c} & c_{b,c} + c_{c,f_2} \end{bmatrix} \quad (2.6)$$

In accordance with equation (2.3), b_p is defined by $\sum_{i=1}^{|f|} c_{pfi} x_{fi}$

$$\mathbf{b} = \begin{bmatrix} \sum_{i=1}^{|f|} c_{af_i} x_{f_i} \\ \sum_{i=1}^{|f|} c_{bf_i} x_{f_i} \\ \sum_{i=1}^{|f|} c_{cf_i} x_{f_i} \end{bmatrix} = \begin{bmatrix} c_{af_1} f_1 \\ 0 \\ c_{cf_2} f_2 \end{bmatrix} \quad (2.7)$$

In order to minimize the cost function shown in equation (2.4), it is sufficient to find the partial derivatives with respect to x and y and equalize both of them with zero.

$$\frac{d\Gamma_x(\mathbf{x})}{d\mathbf{x}} = \mathbf{L}\mathbf{x} + \mathbf{b} \quad (2.8)$$

Global minimum can be found by solving the following equation

$$\mathbf{L}\mathbf{x} = -\mathbf{b} \quad (2.9)$$

Solving the system of linear equations with respect to x , the x -positions of the cells with minimal wirelength are found. The same procedure is performed to find the optimal y -coordinate of cells. In our custom tool the system is solved either by direct solvers or by iterative solvers for better execution time

2.3.1 Net Modeling

Quadratic placers are based on net models to represent nets by two-pin connections. The sum of quadratic length of all pin connections identifies the cost function. Integrated circuits can be represented as a set of standard cells and a set of nets, M and N respectively. Every net $n \in N$ is represented by a set of edges $e_i \in E$. Each edge e_i is a connection between two pins which is used for calculation of cost function. Thus equation (2.1) can be written as:

$$\Gamma(x, y) = \frac{1}{2} \sum_{n \in N} \sum_{e=(p,q) \in E_n} w(p, q) \left((x_p - x_q)^2 + (y_p - y_q)^2 \right) \quad (2.10)$$

where p, q are two different cells in net n , $w(p, q)$ is the weight of their edge and x_p, y_p, x_q, y_q are their coordinates in x and y axis respectively. Conventionally the clique, star or Bound2Bound models are used in global placement. Clique model considers that all cells in a net are connected, thus bringing them close finally. Figure 2.7 shows the clique model in which the dotted lines represent connections added due to model. Clique model creates dense matrices that are difficult to be solved by linear solvers. Star model introduces a cell in the middle of the net and connects it with all other cells. Figure 2.6 shows the star model. Star model simulates each net as the sum of connections between every cell and the star node.

Bound2Bound model relies on the idea of removing all inner pin connections and keeping only the connections of boundary cells. Bound2Bound model uses the same quadratic cost function as clique with different weight:

$$w_x(p, q) = \begin{cases} 0 & p, q \text{ inner cells} \\ \frac{2}{P-1} \frac{1}{|x_p - x_q|} & \text{otherwise} \end{cases} \quad (2.11)$$

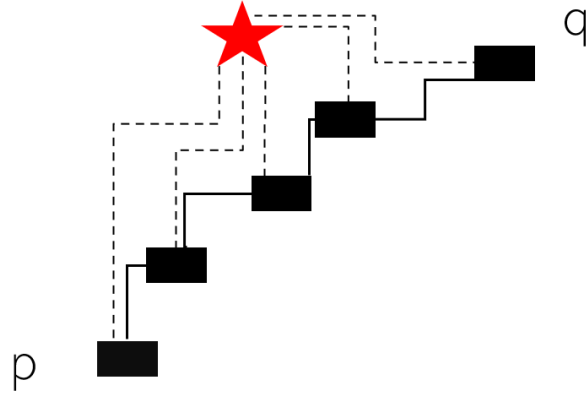


Figure 2.6: Star Model

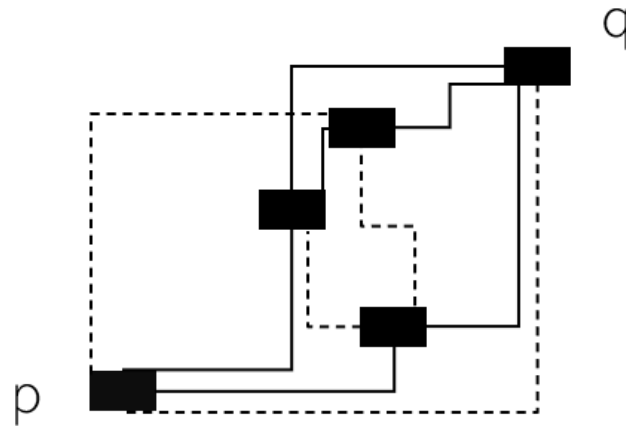


Figure 2.7: Clique Model

As previously mentioned, the main purpose of global placement is to minimize the total wirelength. This metric becomes known only after routing which is completed after global placement. Hence it is necessary to find a method of lower cost to estimate the total wirelength. A sufficient way to achieve so is the Half Perimeter Wire Length (HPWL). HPWL is the half perimeter of the smallest rectangle that contains all pins of a net. Bound2Bound net model represents HPWL approach accurately, thus estimating the wirelength satisfactorily. Figure 2.8 shows HPWL model where dotted lines represent the connections between cells. Hence net wirelength can be estimated as: $HPWL = (x_{max} - x_{min}) + (y_{max} - y_{min})$ that is the half perimeter of the shown rectangle.

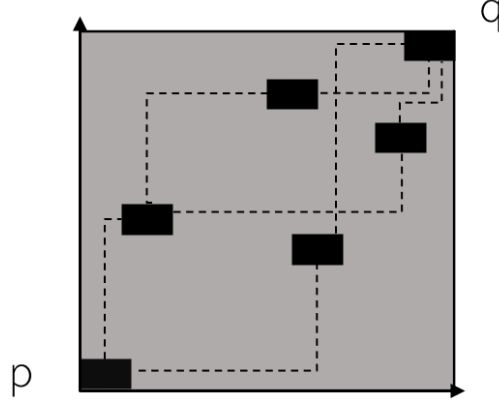


Figure 2.8: HPWL Net Model

For nets with two and three fanout, Bound2Bound net model has the same number of two-pin connections as the clique net model. [6]

2.4 Force-Directed Placement

Force directed algorithms for placement are known for their high efficiency as they produce results of high quality in short time. In these algorithms each two-pin connection is viewed as an elastic spring, and the total cost function is considered as the total energy of the spring system. As previously, the derivative of cost function is set to zero. In this case cost function is energy, so its derivative is the force called net force . [7]

$$\mathbf{F}_x^{\text{net}} = \mathbf{L}_x \mathbf{x} + \mathbf{b}_x \quad (2.12)$$

Force directed placers use Bound2Bound net model depicted in figure 2.8 due to the fact that it offers lower run time and better net length

As previously mentioned, global placers use the connections between cells and attempt to bring the connected cells close in order to minimize the total wirelength. This is not enough because in this way, overlap is generated. So, not only is "gathering" of cells needed but "spreading" too. In this thesis Kraftwerk2 algorithm is analyzed briefly, which uses a force to keep cells close and an another force to spread them since it falls into force-directed placers

2.4.1 Kraftwerk2

Kraftwerk2 is an iterative force-directed quadratic placer [7]. At the start of each iteration pins are located to x' , After minimization of cost function a new position x is identified, So at each iteration Δx is the change of x coordinates. The change of y coordinates can be obtained similarly

$$\Delta x = x - x' \quad (2.13)$$

Kraftwerk2 separates the additional force into hold force and move force aiming at energy equilibrium

- Move force: Move force minimizes the overlap by spreading the cells over the chip. To minimize overlap it is necessary to model the distribution of cells so as to move the cells from regions of high density to those of low density. This model can be determined by a demand supply system

$$D(x, y) = D^{dem}(x, y) - D^{sup}(x, y) \quad (2.14)$$

where D^{dem} represents the cells, D^{sup} represents the placement area given by chip and $D(x,y)$ is the determined distribution. Essentially, this system relates the provided area of a bin with the area of cells in it. Hence, if a bin is empty, D from equation (2.14) is -1. Similarly if a bin is filled, D is 0 and if there is overlap, D is bigger than 0,

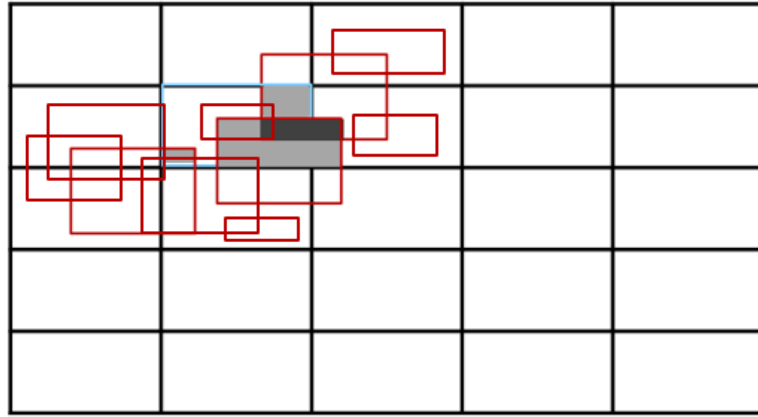


Figure 2.9: Placement example

In order to determine the demand supply system, we divide the core area into bins and we define the density into each bin. For instance, if we have the placement of figure 2.9, where the red boxes represent the cells, the demand supply system in x and y axis respectively is illustrated in figure 2.10

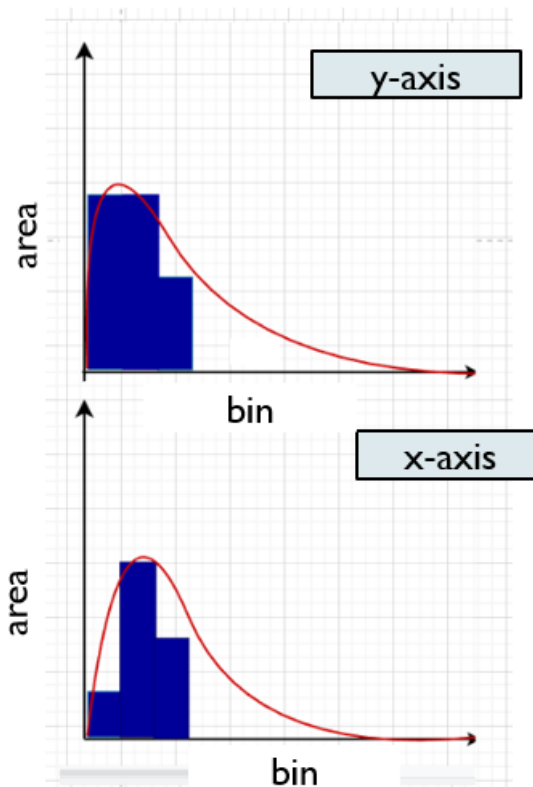


Figure 2.10: density demand supply at y and x axis

where the blue columns represent the density in each bin. The demand supply system in 2 dimensions is illustrated in figure 2.11

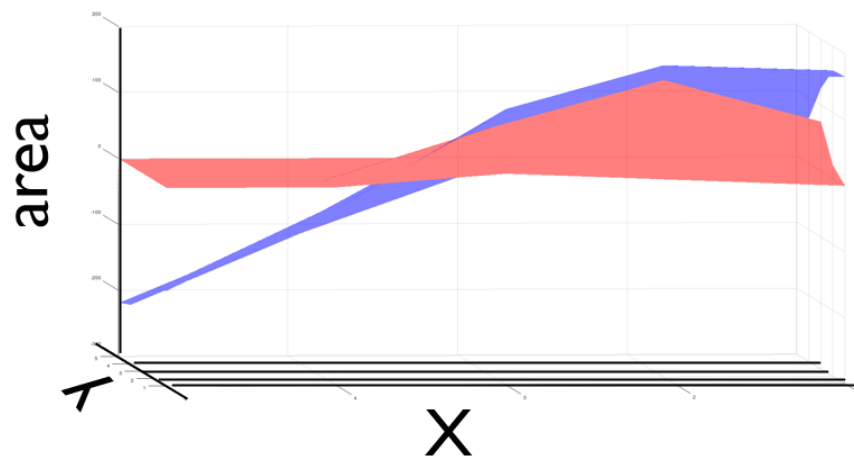


Figure 2.11: density demand supply for both y and x axis

The goal is to move the cells from high density regions to low density regions. This can be achieved by moving the cells to the direction of density gradient. In order for a function to have gradient, it is necessary to be differentiable. As a result, density function needs to be smoothed and be like the red line in figure 2.10 and blue function in figure 2.11

To sum up, from the placement we deduce a demand supply system which needs to be smoothed in order to take its derivative and move the cells according to it. This thesis works on this smoothing. That is, in this work we try to smooth the density function and produce the red line from blue columns in figure 2.10. So we have explained the use of the demand supply system in our work and now we will see how this system is obtained mathematically.

In order to determine the value of D^{dem} at a point (x,y) , a rectangle function R is used. R function is 1 if point (x,y) is inside the rectangle, otherwise for points outside the rectangle is 0.

$$R(x, y, x_{ll}, y_{ll}, w, h) = \begin{cases} 1 & 0 \leq x - x_{ll} \leq w \text{ and } 0 \leq y - y_{ll} \leq h \\ 0 & \text{otherwise} \end{cases}$$

where (x_{ll}, y_{ll}) is the lower left corner of the rectangle and w and h is the width and height of the rectangle respectively. The demand of a cell is defined as below:

$$D_{cell_i}^{dem}(x, y) = d_{cell_i} R(x, y, x'_i - \frac{w_i}{2}, y'_i - \frac{h_i}{2}, w_i, h_i) \quad (2.15)$$

where (x'_i, y'_i) is the current location of $cell_i$, and w_i, h_i its width and height respectively. In most cases individual cell density d_{cell_i} is set to 1. Hence the R function returns 1 if the cell covers the point (x,y)

Thus the demand for both movable M and fixed F points is equal to the number of cells that cover the (x, y) point :

$$D^{dem}(x, y) = \sum_{i=1}^{M+F} D_{cell_i}^{dem}(x, y) \quad (2.16)$$

Additionally to cell demand, a cell supply D^{sup} is also required for the demand supply system, D^{sup} is determined as follows:

$$D_{cell_i}^{sup}(x, y) = d_{cell_i}^{sup} R(x, y, x_{chip}, y_{chip}, w_{chip}, h_{chip}) \quad (2.17)$$

where w_{chip}, h_{chip} is the chip width and height respectively and x_{chip}, y_{chip} is the lower left corner of chip.

In general the demand-supply system has to be balanced, that is the area over the demand has to be equal with the area over the supply [7]

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} D^{dem}(x, y) dx dy = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} D^{sup}(x, y) dx dy \quad (2.18)$$

This is necessary to guarantee the convergence to a placement where the demand is completely adapted to supply.

According to 2.15 and in order to satisfy 2.18 the individual cell density supply is

$$d_{cell_i}^{sup} = \sum_{i=1}^{M+F} (d_{cell_i}) A_{cell_i} / A_{chip} \quad (2.19)$$

In most cases d_{cell_i} is 1, thus the $d_{cell_i}^{sup}$ is the total area covered by cells divided with the total area of chip.

After defining the D^{dem} and D^{sup} , the total Demand Supply system of equation (2.14) can be calculated. Global placer of ASP tool uses the gradient of smoothed density demand-supply(DS) in global placement. Smoothing of density function is performed either by solving the Poisson equation or by applying Gaussian blurring. Both of methods are explained briefly later.

The distribution D of the demand-supply system can be viewed as a charge distribution, which creates an electrostatic potential ϕ generated by Poisson's equation

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) \phi(x, y) = -D(x, y) \quad (2.20)$$

where the expression on the left denotes the second partial derivatives of ϕ with respect of x and y. The purpose of this thesis is to solve this partial differential equation with low execution time and high accuracy.

In the electrostatic formulation in equation (2.20), the potential ϕ is high in high density regions and low in low density regions. Hence gradient of potential can be used to move the cells away from high density areas to low density areas thus reducing the overlap. Specifically in Kraftwerk2, the gradients of potential, smoothed density function, are directly used to determine the location $x_{i_{target}}$ of the target point of each cell i

$$x_{i_{target}} = x'_i - \frac{\partial \phi}{\partial x}(x, y)|_{(x'_i, y'_i)} \quad (2.21)$$

The spring connection between the cell and its target point determines the move force.

$$F_{cell_i}^{move} = w_{i_{target}}(x_i - x_{i_{target}}) \quad (2.22)$$

where x_i represents the new location of the $cell_i$, $w_{i_{target}}$ represents the move force strength and affects the distance that the cell moves during each iteration of the algorithm. In each iteration of global placement, cells are moved away from high-density regions, that is regions with a lot of demand, to low-density regions, that is regions with remaining supply. Each cell can be moved up to $x_{i_{target}}$, that is the movement is limited.

For ease, equation (2.22) is transformed to matrix-vector notation.

$$\mathbf{F}_x^{move} = \mathbf{C}_{x_{target}}(\mathbf{x} - \mathbf{x}_{target}) \quad (2.23)$$

where $\mathbf{C}_{x_{target}}$ matrix collects the weights in its diagonal, that is $\mathbf{C}_{x_{target}} = \text{diag}(w_{i_{target}})$, and $\mathbf{x}_{target}$ is identified by 2.21 in matrix form

$$\mathbf{x}_{target} = \mathbf{x}' - \Phi_x \quad (2.24)$$

Φ_x is a vector which collects the gradients of the potential.

$$\Phi_x = \begin{bmatrix} \frac{\partial}{\partial x} \phi|_{(x'_1, y'_1)} \\ \frac{\partial}{\partial x} \phi|_{(x'_2, y'_2)} \\ \frac{\partial}{\partial x} \phi|_{(x'_3, y'_3)} \\ \vdots \\ \frac{\partial}{\partial x} \phi|_{(x'_M, y'_M)} \end{bmatrix} \quad (2.25)$$

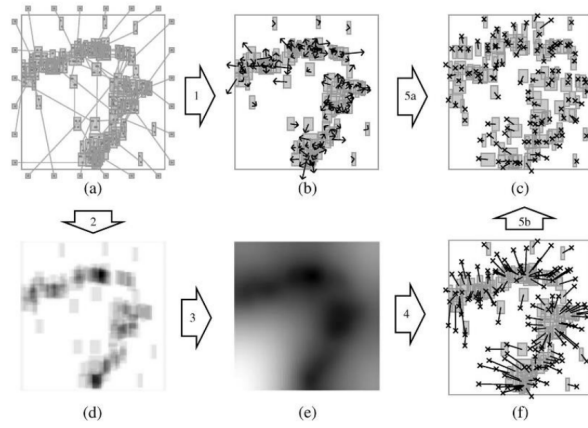


Figure 2.12: Illustration of a placement iteration. The number of arrows represents the order of steps. The hold force(b) and the Demand Supply system(d) are identified by initial placement(a). Potential Φ (e) is identified by (b) and (d). In (d),(e) black and white areas represent high and low density respectively. The move force(f) results from Φ

- hold force

Move force is responsible for spreading the cells across the core area, nevertheless there is the net force defined by equation (2.12) which tries to minimize wirelength thus bringing the cells close. For this reason it is possible for these two forces to neutralize one another thus having a negligible result to placement. Hence, it is required to prevent cells from collapsing back to their previous position. This compensation of the net force is done by hold force, Thus the hold force $\mathbf{F}_x^{\text{hold}}$ is equal to the negative net force

$$\mathbf{F}_x^{\text{hold}} = -(\mathbf{L}_x \mathbf{x}' + \mathbf{b}_x) \quad (2.26)$$

where $\mathbf{x}'$ is the previous position of cells

Figure 2.12 shows the use of these forces during placement. Firstly in 2.12a, net force is applied to cells so they get closer. To inhibit this a hold force with direction of arrows as depicted in 2.12b is applied. Next figure 2.12c shows the created demand supply system D , based on the placed cells. Figure 2.12e shows the electrostatic

potential Φ which can be thought as the smoothed density D . The gradients of the potential determine the target points, that is the crosses in figure 2.12e. Move forces are considered as spring connections between the cells and the target points. Global placer tries to achieve force equilibrium, that is:

$$\begin{aligned}\mathbf{F}_{\text{net}} + \mathbf{F}_{\text{move}} + \mathbf{F}_{\text{hold}} &= \mathbf{0} \\ \mathbf{L}_x \mathbf{x} + \mathbf{b}_x + (-\mathbf{L}_x \mathbf{x}' + \mathbf{b}_x) + \mathbf{C}_x \Phi &= \mathbf{0} \\ (\mathbf{L}_x + \mathbf{C}_x) \Delta \mathbf{x} &= -\mathbf{C}_x \Phi\end{aligned}\tag{2.27}$$

where $\mathbf{L}_x$ is the Laplacian matrix, $\mathbf{C}_x$ matrix is the matrix which contains the weights of move force to its diagonal and Φ is the potential.

2.5 Poisson Equation

As previously mentioned, Density function is $D(x,y)$ from equation (2.16) and it constitutes the density distribution function of cells in bins after each iteration of global placement and its potential is used for the next iteration of global placement described from equation (2.27). To find its potential, it is necessary to solve the Poisson equation

Poisson equation is a partial differential equation(PDE). Generally, a PDE can be written as [8]

$$a\phi_{xx} + 2b\phi_{xy} + c\phi_{yy} = f(x, y, \phi, \phi_x, \phi_y)\tag{2.28}$$

where ϕ is the unknown function and ϕ_{xx} represents the second derivative of ϕ with respect to x . There are 3 categories of PDEs:

- elliptic if $ac - b^2 > 0$
- parabolic if $ac - b^2 = 0$
- hyperbolic if $ac - b^2 < 0$

Poisson is an elliptic PDE and is simply the non-homogeneous Laplace equation

$$\phi_{xx} + \phi_{yy} = -D(x, y)\tag{2.29}$$

where ϕ_{xx} , ϕ_{yy} represent the second partial derivative of ϕ with respect to x and y respectively and the right hand side is corresponding to f from equation (2.28). Poisson equation can be also written as:

$$\nabla^2 \phi(x, y) = -D(x, y)\tag{2.30}$$

where ∇ is the gradient operator

Applications involving partial elliptic equations usually lead to boundary value problems in a region R . Poisson equation is of board utility in physics and engineering. It constitutes a generalization of Laplace equation. The main issue of elliptic partial differential equations is their solvability and the uniqueness of the solution. The uniqueness is subject to certain boundary conditions. Depending on the problem, there are several boundary conditions that can be used, the most common are [9]:

- Dirichlet boundary conditions

$$\phi(x, y) = g(x, y) \text{ on } \Gamma \quad (2.31)$$

- Neumann boundary conditions:

$$\frac{\Theta \phi}{\Theta n}(x, y) = g(x, y) \text{ on } \Gamma \quad (2.32)$$

- periodic boundary conditions:

$$\begin{aligned} \phi(0, y) &= \phi(b_x, y) \\ \phi(x, 0) &= \phi(x, b_y) \end{aligned} \quad (2.33)$$

where $g(x, y)$ is a known function and Γ is the boundary of domain as shown in figure 2.13.

In Dirichlet boundary conditions the values of potential ϕ are specified on the domain boundary Γ while in Neumann boundary conditions, the values of normal derivative of potential ϕ are specified on the domain boundary Γ . The derivative with respect to n means "in direction normal to the boundary". For the case of Dirichlet boundary conditions the solution of Poisson equation always exists and is unique. The case of Neumann boundary conditions demands special attention because a solution may or may not exist and it depends on whether the compatibility conditions are satisfied [10].

$$\int \int_R \phi = - \int \int_R D \quad (2.34)$$

This thesis focuses on a rectangular discretized domain, which is a grid, as illustrated in figure 2.14, where b_x is the rightmost boundary domain coordinate in x-axis and b_y is the upmost boundary domain coordinate in y-axis

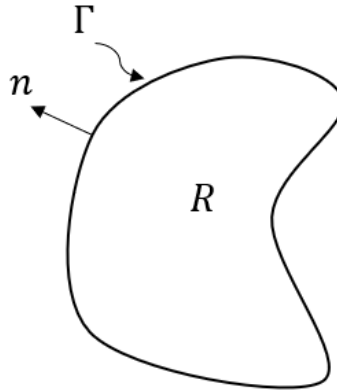


Figure 2.13: Poisson general domain

The grid where Poisson equation is applied is uniform:

$$\begin{aligned} x_j &= j \frac{b_x}{bins_x}, j = 0, \dots, bins_x \\ y_j &= j \frac{b_y}{bins_y}, j = 0, \dots, bins_y \end{aligned}$$

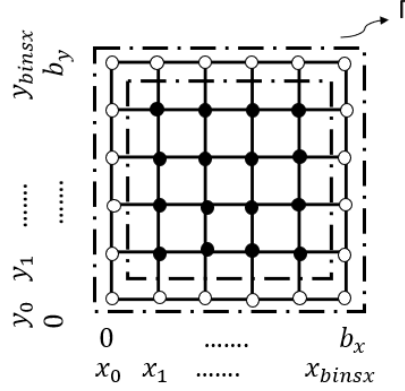


Figure 2.14: rectangular discretized domain

The solution of Poisson equation is the potential field ϕ , caused by a given electric charge distribution ψ . This solution is based either on finite differences or on spectral methods. For the rest of this thesis we will use the letter ψ to refer to the demand supply system and vice versa

2.5.1 Finite Differences

The finite differences method is a numerical procedure which solves a partial differential equation by discretizing the continuous physical domain into a discrete finite difference grid, approximating the individual exact partial derivatives in the PDE by algebraic finite difference approximations(FDA), substituting the FDAs into the PDE to obtain an algebraic finite difference equation and solving the system of equations. To apply finite differences, the solution domain must be a grid of lines, whose intersections are the grid points at which the finite difference solution is to be obtained. Such a grid is illustrated in figure 2.15. After specifying the grid finite difference approximations of the individual, exact partial derivatives appearing in the partial differential equation must be developed. This can be accomplished by writing the Taylor series for the dependent variable in neighboring grid points, having as the base point (x,y) , and combining these to approximate the derivative. In our case of Poisson equation, the derivatives of equation (2.40) can be approximated by writing the Taylor series of $\phi(x-1, y)$ and $\phi(x+1, y)$. [11].

$$\phi(x+1, y) = \phi(x, y) + \frac{\partial}{\partial x}\phi(x, y)\Delta x + \frac{1}{2}\frac{\partial^2}{\partial x^2}\phi(x, y)\Delta x^2 + \frac{1}{6}\frac{\partial^3}{\partial x^3}\phi(x, y)\Delta x^3 + \dots \quad (2.35)$$

$$\phi(x-1, y) = \phi(x, y) - \frac{\partial}{\partial x}\phi(x, y)\Delta x + \frac{1}{2}\frac{\partial^2}{\partial x^2}\phi(x, y)\Delta x^2 - \frac{1}{6}\frac{\partial^3}{\partial x^3}\phi(x, y)\Delta x^3 + \dots \quad (2.36)$$

where Δx is the distance between the x and $x-1$ point in x -axis. The infinite Taylor series are truncated after n -th derivative to obtain $\phi(x+1, y)$ and $\phi(x-1, y)$, and the remainder sum is referred to as truncation error.

Hence by adding the two above equations and dividing by Δx^2 we take:

$$\frac{\partial^2}{\partial x^2}\phi(x, y) = \frac{\phi(x-1, y) - 2\phi(x, y) + \phi(x+1, y)}{\Delta x^2} \quad (2.37)$$

Similarly for $\frac{\partial^2}{\partial y^2} \phi(x, y)$. According to the above equation , second derivatives of Poisson equation (2.29) can be written as:

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) \phi(x, y) = \frac{\phi(x + \Delta x, y) - 2\phi(x, y) + \phi(x - \Delta x, y)}{\Delta x^2} + \frac{\phi(x, y + \Delta y) - 2\phi(x, y) + \phi(x, y - \Delta y)}{\Delta y^2} \quad (2.38)$$

In our case Poisson equation is applied to a discrete domain, that is $\phi(x + \Delta x)$ in continuous domain corresponds to $\phi(x + 1)$ in discrete domain. Subsequently the above equation becomes:

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) \phi(x, y) = \frac{\phi(x + 1, y) - 2\phi(x, y) + \phi(x - 1, y)}{\Delta x^2} + \frac{\phi(x, y + 1) - 2\phi(x, y) + \phi(x, y - 1)}{\Delta y^2} \quad (2.39)$$

In our problem, Poisson equation is applied to a rectangular domain $R = [a_x, b_x] \times [a_y, b_y]$ which is divided into bins as illustrated in Figure 2.15

First some notation:

- h_y : bin height respective to Δy from Taylor series
- h_x : bin width respective to Δx from Taylor series
- M: number of grid points in x-axis
- N: number of grid points in y-axis
- $bins_x$: number of bins in x-axis, equal to M - 1
- $bins_y$: number of bins in y-axis, equal to N - 1

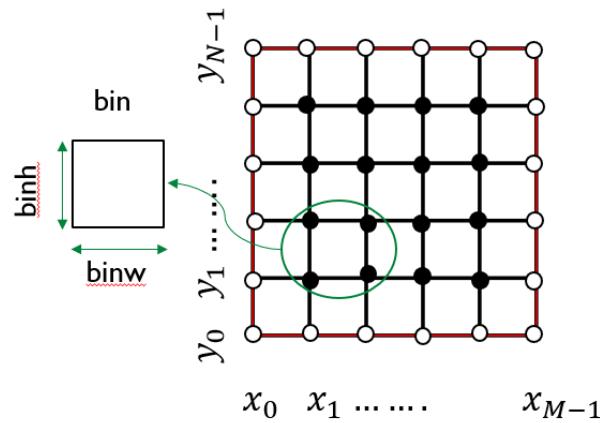


Figure 2.15: finite differences domain

Using formula from equation (2.39) for every (x,y) point of grid and considering a grid with $M = 3$, and $N = 3$. the Poisson problem is deduced to the following system of linear equations:

$$\mathbf{A} \times \mathbf{\Phi} = \mathbf{\Psi} \quad (2.40)$$

where $\mathbf{A} =$

$$\begin{bmatrix} \frac{2}{(h_x)^2} + \frac{2}{(h_y)^2} & -\frac{1}{(h_x)^2} & 0 & -\frac{1}{(h_y)^2} & 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{1}{(h_x)^2} & \frac{2}{(h_x)^2} + \frac{2}{(h_y)^2} & -\frac{1}{(h_x)^2} & 0 & -\frac{1}{(h_y)^2} & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{1}{(h_x)^2} & \frac{2}{(h_x)^2} + \frac{2}{(h_y)^2} & 0 & 0 & -\frac{1}{(h_y)^2} & 0 & 0 & 0 & 0 \\ -\frac{1}{(h_y)^2} & 0 & 0 & \frac{2}{(h_x)^2} + \frac{2}{(h_y)^2} & -\frac{1}{(h_x)^2} & 0 & -\frac{1}{(h_y)^2} & 0 & 0 & 0 \\ 0 & -\frac{1}{(h_y)^2} & 0 & -\frac{1}{(h_x)^2} & \frac{2}{(h_x)^2} + \frac{2}{(h_y)^2} & -\frac{1}{(h_x)^2} & 0 & -\frac{1}{(h_y)^2} & 0 & 0 \\ 0 & 0 & -\frac{1}{(h_y)^2} & 0 & -\frac{1}{(h_x)^2} & \frac{2}{(h_x)^2} + \frac{2}{(h_y)^2} & 0 & 0 & -\frac{1}{(h_x)^2} & -\frac{1}{(h_y)^2} \\ 0 & 0 & 0 & -\frac{1}{(h_y)^2} & 0 & 0 & \frac{2}{(h_x)^2} + \frac{2}{(h_y)^2} & -\frac{1}{(h_x)^2} & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{(h_y)^2} & 0 & -\frac{1}{(h_x)^2} & \frac{2}{(h_x)^2} + \frac{2}{(h_y)^2} & -\frac{1}{(h_x)^2} & -\frac{1}{(h_y)^2} \\ 0 & 0 & 0 & 0 & 0 & -\frac{1}{(h_y)^2} & 0 & -\frac{1}{(h_x)^2} & \frac{2}{(h_x)^2} + \frac{2}{(h_y)^2} & -\frac{1}{(h_x)^2} \\ 0 & 0 & 0 & 0 & 0 & -\frac{1}{(h_y)^2} & 0 & -\frac{1}{(h_x)^2} & \frac{2}{(h_x)^2} + \frac{2}{(h_y)^2} & -\frac{1}{(h_x)^2} \end{bmatrix}$$

$$\Phi = \begin{bmatrix} \phi(0,0) \\ \phi(1,0) \\ \phi(2,0) \\ \phi(0,1) \\ \phi(1,1) \\ \phi(2,1) \\ \phi(0,2) \\ \phi(1,2) \\ \phi(2,2) \end{bmatrix}, \Psi = \begin{bmatrix} \psi(0,0) \\ \psi(1,0) \\ \psi(2,0) \\ \psi(0,1) \\ \psi(1,1) \\ \psi(2,1) \\ \psi(0,2) \\ \psi(1,2) \\ \psi(2,2) \end{bmatrix}$$

The above equation is based on the following individual equations:

$$\begin{aligned} \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) \phi(0,0) &= \frac{\phi(1,0) - \phi(0,0)}{\Delta x^2} + \frac{\phi(0,1) - \phi(0,0)}{\Delta y^2} \\ \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) \phi(1,0) &= \frac{\phi(2,0) - \phi(1,0) + \phi(0,0)}{\Delta x^2} + \frac{\phi(1,1) - \phi(1,0)}{\Delta y^2} \\ \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) \phi(2,0) &= \frac{-\phi(2,0) + \phi(1,0)}{\Delta x^2} + \frac{\phi(2,1) - \phi(2,0)}{\Delta y^2} \\ \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) \phi(0,1) &= \frac{\phi(1,1) - \phi(0,1)}{\Delta x^2} + \frac{\phi(0,2) - \phi(0,1) + \phi(0,0)}{\Delta y^2} \\ \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) \phi(1,1) &= \frac{\phi(2,1) - \phi(1,1) + \phi(0,1)}{\Delta x^2} + \frac{\phi(1,2) - \phi(1,1) + \phi(1,0)}{\Delta y^2} \\ \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) \phi(2,1) &= \frac{-\phi(2,1) + \phi(1,1)}{\Delta x^2} + \frac{\phi(2,2) - \phi(2,1) + \phi(2,0)}{\Delta y^2} \\ \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) \phi(0,2) &= \frac{\phi(1,2) - \phi(0,2)}{\Delta x^2} + \frac{-\phi(0,2) + \phi(0,1)}{\Delta y^2} \\ \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) \phi(1,2) &= \frac{\phi(2,2) - \phi(1,2) + \phi(0,2)}{\Delta x^2} + \frac{-\phi(1,2) + \phi(1,1)}{\Delta y^2} \\ \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) \phi(2,2) &= \frac{\phi(1,2) - \phi(2,2) + \phi(0,2)}{\Delta x^2} + \frac{-\phi(2,2) + \phi(2,1)}{\Delta y^2} \end{aligned}$$

Given that the grid is $N \times M$, Φ will be a $N \times M$ vertical vector. For this reason A matrix need to be $[N \times M] \times [N \times M]$ matrix

Given that A is block tridiagonal matrix, there are several efficient methods that solve the system of equation (2.40). Among the solutions are a generalized Thomas algorithm, cyclic reduction, successive overrelaxation, Jacobi with complexity $O(n^2)$ and multigrid methods which are optimal with complexity $O(n)$.

Dirichlet Boundary Conditions

For the satisfaction of Dirichlet boundary conditions given by equation (2.31), the right hand side of Poisson need to be modified as the solution of boundary points is known, that is for a $N \times M$ grid the A matrix will be $[(N-2) \times (M-2)] \times [(N-2) \times (M-2)]$ instead of $(N \times M) \times (N \times M)$. An example to illustrate this is the figure 2.16 in which the solution at empty circles is provided by Dirichlet boundary conditions and the filled circles are the unknown points. Hence the resultant matrix of system is 4×4 instead of 16×16 .

The Poisson system is:

$$\begin{bmatrix} \frac{2}{(h_x)^2} + \frac{2}{(h_y)^2} & -\frac{1}{(h_x)^2} & -\frac{1}{(h_y)^2} & 0 \\ -\frac{1}{(h_x)^2} & \frac{2}{(h_x)^2} + \frac{2}{(h_y)^2} & 0 & -\frac{1}{(h_y)^2} \\ -\frac{1}{(h_y)^2} & 0 & \frac{2}{(h_x)^2} + \frac{2}{(h_y)^2} & -\frac{1}{(h_x)^2} \\ 0 & -\frac{1}{(h_y)^2} & -\frac{1}{(h_x)^2} & \frac{2}{(h_x)^2} + \frac{2}{(h_y)^2} \end{bmatrix} \begin{bmatrix} \phi(1, 1) \\ \phi(2, 1) \\ \phi(1, 2) \\ \phi(2, 2) \end{bmatrix} = \begin{bmatrix} \psi(1, 1) + \frac{1}{h_x^2} g(0, 1) + \frac{1}{h_y^2} g(1, 0) \\ \psi(2, 1) + \frac{1}{h_x^2} g(3, 1) + \frac{1}{h_y^2} g(2, 0) \\ \psi(1, 2) + \frac{1}{h_x^2} g(0, 2) + \frac{1}{h_y^2} g(1, 3) \\ \psi(2, 2) + \frac{1}{h_x^2} g(3, 2) + \frac{1}{h_y^2} g(2, 3) \end{bmatrix} \quad (2.41)$$

where $\phi(i, j) = g(i, j)$ if point (i, j) is a boundary point.

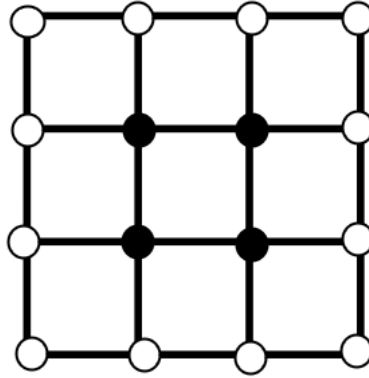


Figure 2.16: Dirichlet boundary conditions on 4x4 grid

2.5.2 Spectral Methods

Spectral methods are similar with Finite differences but in this case a solution at a point depends on the solution at every point of grid. Spectral methods are a class of techniques used in applied mathematics and scientific computing to numerically solve certain differential equations. The idea is to write the solution of the partial differential equation as a sum of basis functions such as sigmoids or Fourier series [12]. This thesis focuses on Fourier series as basis functions.

Fourier Series

In Fourier series theory, it is investigated to what extent a function $f(x)$ can be

represented in an infinite series of the form [13]:

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} \left(a_n \cos \frac{n\pi x}{L} + b_n \sin \frac{n\pi x}{L} \right) \quad (2.42)$$

where a_n, b_n are constants. The Right hand side of equation 2.42 is called Fourier Series and a_n, b_n Fourier Coefficients.

Fast Fourier Transform

Fourier series can be applied to periodic functions, while Fourier Transform can be applied to any function. In this method, Fourier Transform is used to project the equation to Fourier space and the inverse transform to return to physical space. The type of discrete Fourier Transform is [14]:

$$\hat{f}_k = \sum_{n=0}^{N-1} f_n e^{-\frac{2\pi i}{N} kn} \quad (2.43)$$

$$= \sum_{n=0}^{N-1} f_n \left[\cos \left(\frac{2\pi}{N} kn \right) + i \sin \left(\frac{2\pi}{N} kn \right) \right] \quad (2.44)$$

and the inverse Fourier transformation is:

$$f_n = \sum_{k=0}^{N-1} \hat{f}_k e^{\frac{2\pi i}{N} kn} \quad (2.45)$$

$$= \sum_{k=0}^{N-1} \left[\cos \left(\frac{2\pi}{N} kn \right) - i \sin \left(\frac{2\pi}{N} kn \right) \right] \quad (2.46)$$

Poisson Equation Solution

In our case in which we try to solve Poisson equation in 2 dimensions, we need the Fourier transformation in 2 dimensions. ψ from Poisson equation 2.29 is known and so is its Fourier Transform:

$$\hat{\psi}_{u,v} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \psi_{mn} e^{(\frac{2\pi u}{M} + \frac{2\pi v}{N})j} \quad (2.47)$$

$$(2.48)$$

we seek to find DFT of ϕ . Generally it is known that:

$$\nabla^2 \hat{\phi} = -\pi^2 (m^2 + n^2) \hat{\phi} \quad (2.49)$$

$$(2.50)$$

Thus, together with 2.30 we have:

$$\hat{\phi} = -\frac{\hat{\psi}}{\pi^2 (m^2 + n^2)} \quad (2.51)$$

Now if we apply the inverse Fourier transform to $\hat{\phi}$ we obtain the Poisson solution. According to boundary condition, different type of transform should be applied

[14]. Dirichlet boundary conditions require discrete sine transform and Neumann boundary conditions require discrete cosine transform and Periodic require discrete Fourier transform. In order for periodic boundary conditions to have unique solution, it is necessary to specify $\hat{\phi}(0, 0)$ which constitutes the mean value of solution

Chapter 3

Existing Works

In recent years, there has been interest in algorithms that combine high order and fast solvers. The basic algorithms that have been developed in literature are finite differences, spectral methods and differenced corrections. Furthermore there are algorithms which combine these methods to achieve better results.

Fourier spectral methods for accurately solving multidimensional Poisson equation on rectangular domain have been discussed extensively. There have been implemented algorithms which introduce the discrete Fourier transform in finite-difference approximations. These methods can be applied to other elliptic partial differential equations as well.

There are approaches which incorporate Fourier transform into Finite Differences in non-staggered and staggered grid, where the boundary conditions are applied in the middle of 2 grid points. Such a work is described in [14] by Schumann and Sweet. In this work the transform deduced from various boundary conditions are subject to pre and post process in order to be expressed in terms of real periodic transform. In this way they can exploit existing, efficient algorithms for fast Fourier transform to solve the most kinds of boundary conditions. Fast Fourier transform is developed in FORTRAN routines by the two authors in three dimensions

Many fast Poisson solvers have been developed by exploiting some properties of Poisson matrix deduced from finite differences 2.5.1. One of these algorithms is the alternating direction implicit(ADI) [15]. One key initial requirement of this method is that the grid must be equispaced that is, in 2.5.1, $h_x = h_y = h$. ADI method is explained briefly with homogeneous Dirichlet boundary conditions 2.31, that is $g = 0$ at every point(x,y)

3.1 ADI Method

The finite differences discretization of 2.20 with 5-point stencil on a $(n - 1) \times (n - 1)$ equispaced grid can be written as the following Sylvester matrix equa-

tion:

$$\mathbf{K}\Phi + \Phi\mathbf{K}^T = \Psi, \quad (3.1)$$

where

$$\mathbf{K} = -\frac{1}{h^2} \begin{bmatrix} 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 \end{bmatrix} \in (n-1) \times (n-1) \quad (3.2)$$

and Φ is the matrix with the values of the unknown function

$$\begin{bmatrix} \phi(0,0) & \phi(0,1) & \phi(0,2) & \cdots & \phi(0,n-2) \\ \phi(1,0) & \phi(1,1) & \phi(1,2) & \cdots & \phi(1,n-2) \\ \phi(2,0) & \phi(2,1) & \phi(2,2) & \cdots & \phi(2,n-2) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \phi(n-2,0) & \phi(n-2,1) & \phi(n-2,2) & \cdots & \phi(n-2,n-2) \end{bmatrix}$$

ADI method exploits the eigendecomposition to achieve $O(n^2 \log(n))$ complexity. The eigendecomposition of a matrix is:

$$\mathbf{K} = \mathbf{S}\mathbf{\Lambda}\mathbf{S}^{-1} \quad (3.3)$$

where $\mathbf{S}$ is a square $(n-1) \times (n-1)$ matrix whose i -th column is the eigenvector $\mathbf{q}_i$ and $\mathbf{\Lambda}$ is the diagonal matrix whose diagonal elements are the corresponding eigenvalues $\Lambda_{ii} = \hat{\lambda}_i$, that is

$$\mathbf{S} = \begin{bmatrix} q_0 & q_1 & \cdots & q_{n-2} \end{bmatrix}, \mathbf{\Lambda} = \begin{bmatrix} \hat{\lambda}_0 & & & \\ & \hat{\lambda}_1 & & \\ & & \ddots & \\ & & & \hat{\lambda}_{n-2} \end{bmatrix}$$

In case of Poisson equation with Dirichlet boundary conditions, $\mathbf{S}$ is the normalized discrete sine transform of $\mathbf{K}$ and $\hat{\lambda}_k = -\frac{4}{h^2 \sin^2(\frac{\pi k}{2n})}$. After identifying $\mathbf{S}$, $\mathbf{\Lambda}$ and substituting 3.3 into 3.1, Φ can be identified directly:

$$\Phi = \mathbf{S}(\mathbf{C} \circ (\mathbf{S}^{-1}\Psi\mathbf{S}^{-T}))\mathbf{S}^T \quad (3.4)$$

where 'o' is the Hadamard matrix product that is :

$$\mathbf{A} \circ \mathbf{b} = (a_{ij} \cdot b_i) = \begin{bmatrix} a_{11} \cdot b_1 & \cdots & a_{1n} \cdot b_1 \\ \vdots & \ddots & \vdots \\ a_{m1} \cdot b_m & \cdots & a_{mn} \cdot b_m \end{bmatrix} \quad (3.5)$$

For the computation of Φ of equation 3.4, it is sufficient to compute only $\mathbf{S}$ and $\mathbf{C}$ since $\mathbf{S} = \mathbf{S}^{-1} = \mathbf{S}^T$ [15]. By using fast Fourier algorithms , equation 3.4 can be solved in $O(n^2 \log n)$. Consecutively, this work proceeds in generalizing the method of solution of any Sylvester equation:

$$\mathbf{A}\Phi + \Phi\mathbf{B} = \Psi \quad (3.6)$$

and it analyzes the convergence criteria.

3.2 4 Points Modified Explicit Group Method

This work introduces a four point modified explicit group method for solving a 2 dimensional Poisson equation with Dirichlet boundary conditions. It compares their method with the existing four points explicit group(EG) and explicit decoupled group(EDG). All 3 methods are explained briefly. Each method expresses in different way the approximation of second derivative but all have a local truncation error of order of $O(h^2)$. One key requirement of all 3 methods is that the bin width is equal to the bin height. [16]

Four points explicit group method

This method approximates Poisson equation $\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)\phi(i,j) = -f(i,j)$ with the most common four point approximation:

$$\phi_{i+1,j} + \phi_{i-1,j} + \phi_{i,j+1} + \phi_{i,j-1} - 4\phi_{i,j} = -h^2 f_{i,j} \quad (3.7)$$

where h is the bin width and bin height. Method constructs a system for every group of 4 points as it is shown in Figure 3.1

$$\begin{bmatrix} 4 & -1 & 0 & -1 \\ -1 & 4 & -1 & 0 \\ 4 & -1 & 4 & -1 \\ -1 & 0 & -1 & 4 \end{bmatrix} \begin{bmatrix} \phi_{i,j} \\ \phi_{i+1,j} \\ \phi_{i+1,j+1} \\ \phi_{i,j+1} \end{bmatrix} = \begin{bmatrix} \phi_{i-1,j} + \phi_{i,j-1} - h^2 f_{i,j} \\ \phi_{i+2,j} + \phi_{i+1,j-1} - h^2 f_{i+1,j} \\ \phi_{i+2,j+1} + \phi_{i+1,j+2} - h^2 f_{i+1,j+1} \\ \phi_{i-1,j+1} + \phi_{i,j+1} - h^2 f_{i,j+1} \end{bmatrix}$$

The four point EG deduced equation is [16]

$$\begin{bmatrix} \phi_{i,j} \\ \phi_{i+1,j} \\ \phi_{i+1,j+1} \\ \phi_{i,j+1} \end{bmatrix} = \frac{1}{24} \begin{bmatrix} 7 & 2 & 1 & 2 \\ 2 & 7 & 2 & 1 \\ 1 & 2 & 7 & 2 \\ 2 & 1 & 2 & 7 \end{bmatrix} \begin{bmatrix} \phi_{i-1,j} + \phi_{i,j-1} - h^2 f_{i,j} \\ \phi_{i+2,j} + \phi_{i+1,j-1} - h^2 f_{i+1,j} \\ \phi_{i+2,j+1} + \phi_{i+1,j+2} - h^2 f_{i+1,j+1} \\ \phi_{i-1,j+1} + \phi_{i,j+1} - h^2 f_{i,j+1} \end{bmatrix} \quad (3.8)$$

Equation 3.8 is applied to every point in grid and 3.7 is applied to boundary points. Method follows the below SOR iterative procedure to identify the final solution:

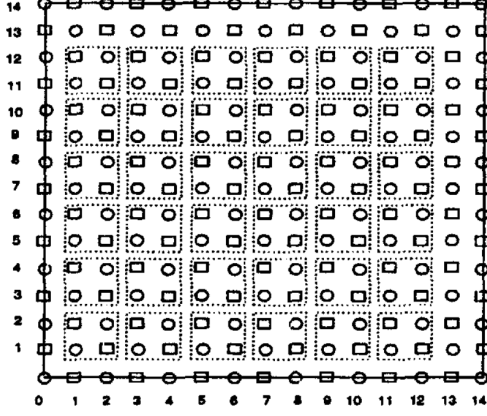


Figure 3.1: Solution domain for the four points EG and EDG methods

- Group all points into 4 points groups as shown in figure 3.1
- perform 1 iteration of SOR iterative method for all points

Four points explicit decoupled group method

Four points EDG method uses another type of approximation derived from the rotated five point finite difference approximation. This approximation can be obtained by rotating the x-y axis clockwise by 45° as it is shown in Figure 3.2c. The rotated finite difference approximation of equation 3.7 is:

$$\phi_{i+1,j+1} + \phi_{i-1,j-1} + \phi_{i+1,j-1} + \phi_{i-1,j+1} - 4\phi_{i,j} = -2h^2 f_{i,j} \quad (3.9)$$

This method takes any group of four points in the solution domain and by applying equation 3.9 results in the below system:

$$\begin{bmatrix} 4 & -1 & 0 & 0 \\ -1 & 4 & 0 & 0 \\ 0 & 0 & 4 & -1 \\ 0 & 0 & -1 & 4 \end{bmatrix} \begin{bmatrix} \phi_{i,j} \\ \phi_{i+1,j+1} \\ \phi_{i+1,j} \\ \phi_{i,j+1} \end{bmatrix} = \begin{bmatrix} \phi_{i-1,j-1} + \phi_{i+1,j-1} + \phi_{i-1,j+1} - 2h^2 f_{i,j} \\ \phi_{i,j+2} + \phi_{i+2,j} + \phi_{i+2,j+2} - 2h^2 f_{i+1,j+1} \\ \phi_{i,j-1} + \phi_{i+2,j-1} + \phi_{i+2,j+1} - 2h^2 f_{i+1,j} \\ \phi_{i-1,j+2} + \phi_{i-1,j} + \phi_{i+1,j+2} - 2h^2 f_{i,j+1} \end{bmatrix}$$

These two equations can be decoupled as follows:

$$\begin{bmatrix} \phi_{i,j} \\ \phi_{i+1,j+1} \end{bmatrix} = \frac{1}{15} \begin{bmatrix} 4 & 1 \\ 1 & 4 \end{bmatrix} \begin{bmatrix} \phi_{i-1,j-1} + \phi_{i+1,j-1} + \phi_{i-1,j+1} - 2h^2 f_{i,j} \\ \phi_{i,j+2} + \phi_{i+2,j} + \phi_{i+2,j+2} - 2h^2 f_{i+1,j+1} \end{bmatrix} \quad (3.10)$$

$$\begin{bmatrix} \phi_{i+1,j} \\ \phi_{i,j+1} \end{bmatrix} = \frac{1}{15} \begin{bmatrix} 4 & 1 \\ 1 & 4 \end{bmatrix} \begin{bmatrix} \phi_{i,j-1} + \phi_{i+2,j-1} + \phi_{i+2,j+1} - 2h^2 f_{i+1,j} \\ \phi_{i-1,j+2} + \phi_{i-1,j} + \phi_{i+1,j+2} - 2h^2 f_{i,j+1} \end{bmatrix} \quad (3.11)$$

In this way it is clear that evaluation of 3.10 and 3.11 can be performed concurrently thus reducing the execution time approximately to half.

Four points modified explicit decoupled group method

This method separates the grid points into 3 categories as depicted in figure 3.3:

- rectangle points
- circle points
- circle filled points

The goal is to obtain a solution for each category without the need of finding the solution on points of a different category. To achieve this, the method uses equation 3.7 with grid spacing $2h$ as it is shown in figure 3.2c:

$$\phi_{i+2,j} + \phi_{i-2,j} + \phi_{i,j+2} + \phi_{i,j-2} - 4\phi_{i,j} = -4h^2 f_{i,j} \quad (3.12)$$

Method applies equation 3.12 to every four point group and results in the following system.

$$\begin{bmatrix} 4 & -1 & 0 & -1 \\ -1 & 4 & -1 & 0 \\ 0 & -1 & 4 & -1 \\ -1 & 0 & -1 & 4 \end{bmatrix} \begin{bmatrix} \phi_{i,j} \\ \phi_{i+2,j} \\ \phi_{i+2,j+2} \\ \phi_{i,j+2} \end{bmatrix} = \begin{bmatrix} \phi_{i-2,j} + \phi_{i,j-2} - 4h^2 f_{i,j} \\ \phi_{i+4,j} + \phi_{i+2,j-2} - 4h^2 f_{i+2,j} \\ \phi_{i+4,j+2} + \phi_{i+2,j+4} - 4h^2 f_{i+2,j+2} \\ \phi_{i-2,j+2} + \phi_{i,j+4} - 4h^2 f_{i,j+2} \end{bmatrix}$$

The inverted system is:

$$\begin{bmatrix} \phi_{i,j} \\ \phi_{i+2,j} \\ \phi_{i+2,j+2} \\ \phi_{i,j+2} \end{bmatrix} = \frac{1}{24} \begin{bmatrix} 7 & 2 & 1 & 2 \\ 2 & 7 & 2 & 1 \\ 1 & 2 & 7 & 2 \\ 2 & 1 & 2 & 7 \end{bmatrix} \begin{bmatrix} \phi_{i-2,j} + \phi_{i,j-2} - 4h^2 f_{i,j} \\ \phi_{i+4,j} + \phi_{i+2,j-2} - 4h^2 f_{i+2,j} \\ \phi_{i+4,j+2} + \phi_{i+2,j+4} - 4h^2 f_{i+2,j+2} \\ \phi_{i-2,j+2} + \phi_{i,j+4} - 4h^2 f_{i,j+2} \end{bmatrix} \quad (3.13)$$

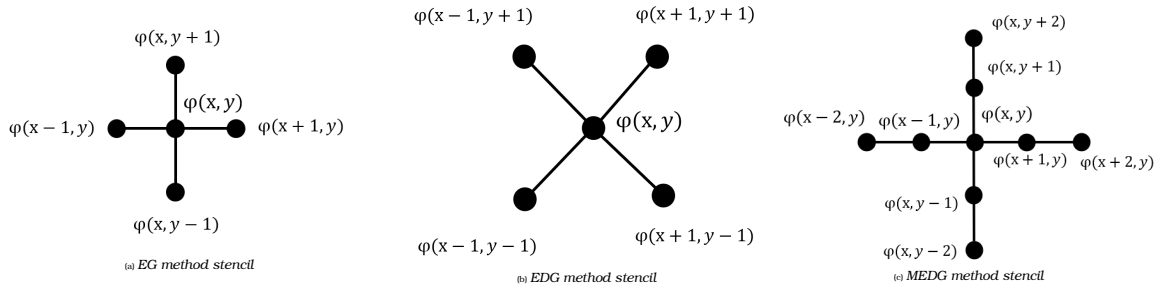


Figure 3.2: Stencil of each method

In this way the execution time can be reduced to a quarter of that of the initial EG method. Experiments of this method show that it is superior compared to EG and EDG

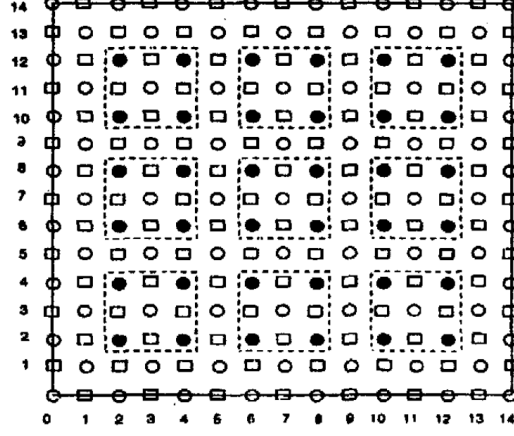


Figure 3.3: Solution domain for the four points MEDG methods

3.3 Cyclic Reduction Method

The method of cyclic reduction was used by Hockney [17]. In this work it is purported that cyclic reduction is the optimal method for cyclic-boundary conditions 2.33. Consider a $[N \times M]$ grid. The deduced system of finite differences for this boundary problem is written in block-matrix form: [17]:

$$\begin{bmatrix} \mathbf{A} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I} \\ \mathbf{I} & \mathbf{A} & \mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{A} & \mathbf{I} & \cdots & \mathbf{0} \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{A} & \mathbf{I} \\ \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{A} \end{bmatrix} \begin{bmatrix} \Phi_0 \\ \Phi_1 \\ \vdots \\ \Phi_{N-1} \end{bmatrix} = \begin{bmatrix} \Psi_0 \\ \Psi_1 \\ \vdots \\ \Psi_{N-1} \end{bmatrix} \quad (3.14)$$

where

$$\Phi_j = \begin{bmatrix} \phi_{0,j} \\ \phi_{1,j} \\ \phi_{2,j} \\ \vdots \\ \phi_{M-1,j} \end{bmatrix}, \quad \Psi_j = \begin{bmatrix} \psi_{0,j} \\ \psi_{1,j} \\ \psi_{2,j} \\ \vdots \\ \psi_{M-1,j} \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} -4 & 1 & 0 & \cdots & 0 & 1 \\ 1 & -4 & 1 & & & 0 \\ 0 & & \ddots & & & \vdots \\ 0 & & & 1 & -4 & 1 \\ 1 & 0 & & 0 & 1 & -4 \end{bmatrix} \quad (3.15)$$

Φ_j is a vertical vector with M elements and it constitutes the j -th row of density map, $\mathbf{A}$ and $\mathbf{I}$ are $[N \times M]$ square matrices. The first step of the algorithm is to halve the number of equations. For example, if $\mathbf{A}$ is 48x48, it is reduced to a 24x24 matrix. By taking 3 successive equations from equation 3.14 we obtain:

$$\begin{aligned} \Phi_{j-2} + \mathbf{A}\Phi_{j-1} + \Phi_j &= \Psi_{j-1} \\ \Phi_{j-1} + \mathbf{A}\Phi_j + \Phi_{j+1} &= \Psi_j \\ \Phi_j + \mathbf{A}\Phi_{j+1} + \Phi_{j+2} &= \Psi_{j+1} \end{aligned} \quad (3.16)$$

By multiplying the second equation by $-\mathbf{A}$ on the left and by adding we obtain:

$$\Phi_{j-2} + (2\mathbf{I} - \mathbf{A}^2)\Phi_j + \Phi_{j+2} = \Psi_{j-1} - \mathbf{A}\Psi_j + \Psi_{j+1} \quad (3.17)$$

Hence, the resultant system contains only the even terms, that is if the initial system was 48x48, after this step it will be 24x24. There are several methods to solve the system 3.17. In this work, Fourier analysis is selected.

- A modified form of right hand side of 3.17 is introduced $\Psi_j^* = \Psi_{j-1} - \mathbf{A}\Psi_j + \Psi_{j+1}$. Hence, the modified 3.17 is

$$\Phi_{j-2} + (2\mathbf{I} - \mathbf{A}^2)\Phi_j + \Phi_{j+2} = \Psi_j^* \quad (3.18)$$

- Fourier transform of 3.18 is identified:

$$\hat{\Phi}_{j-2} + \hat{\lambda}\hat{\Phi}_j + \hat{\Phi}_{j+2} = \hat{\Psi}_j^* \quad (3.19)$$

where $\hat{\lambda}$ is the eigenvalue of matrix $2\mathbf{I} - \mathbf{A}^2$ [17]

The process of reducing the number of equations by half as described in 3.16, may now be similarly applied:

$$\Phi_{j-2^2} + \hat{\lambda}^2\Phi_j + \Phi_{j+2^2} = \Psi_{j^2} \quad (3.20)$$

where the Fourier transform is omitted for simplicity.

To make it clear for a 48 x 48 system, in the first iteration, the 0, 2nd, 4th, ... lines of system remain, in the second iteration, 0, 4th, 8th... lines of the initial system remain. The process of reduction may therefore be carried out recursively until a small number of equations are obtained which are solved directly. Let t be the depth of the recursion, the recursive formula becomes

$$\Phi_{j-2^t} + \hat{\lambda}^{(t)}\Phi_j + \Phi_{j+2^t} = \Psi_{j^t} \quad (3.21)$$

Finally the system has 3 equations, Φ_0 , Φ_{16} , Φ_{32} , which are solved directly. In order to find the other values of Φ , intermediate values are determined recursively. Firstly Φ_8 , Φ_{24} , Φ_{40} are determined and then Φ_4 , Φ_{12} , Φ_{20} , Φ_{28} , Φ_{36} , Φ_{44} etc. from the relation

$$\Phi_j = \frac{1}{\hat{\lambda}^{(t)}}\Psi_j - \Phi_{j-2^t} - \Phi_{j+2^t} \quad (3.22)$$

In this way the solution of even lines is obtained. Solution of odd lines can be found from equation 3.16:

$$\mathbf{A}\Phi_j = \Psi_j - \Phi_{j-1} - \Phi_{j+1} \quad (3.23)$$

where the right hand side is known from the even case

3.4 MathWorks MATLAB

Matlab uses `poisson2Dneumann.m` to solve Poisson equation with Neumann boundary conditions. The solution is based on Fourier analysis, that is the Fourier Transform is applied to finite differences in order for the Poisson equation to be solved fast.

The finite difference equation is:

$$\phi_{i,j-1} + \phi_{i-1,j} - 4\phi_{i,j} + \phi_{i,j+1} + \phi_{i+1,j} = \psi_{i,j} \quad (3.24)$$

In order to solve the Poisson equation, it is required for ϕ and ψ to be represented by the The Inverse cosine transform.

$$\psi_{i,j} = \frac{4}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \mathcal{F}_c \phi_{m,n} \cos\left(\frac{mj\pi}{M}\right) \cos\left(\frac{nj\pi}{N}\right) \quad (3.25)$$

where $\mathcal{F}_c$ symbolises the Discrete cosine transform.
Initially some notations for simplicity:

$$s_b^a = \sin\left(\frac{a\pi b}{A}\right) \quad (3.26)$$

$$c_b^a = \cos\left(\frac{a\pi b}{A}\right) \quad (3.27)$$

Substituting 3.25 into 3.24 and removing the summations:

$$\mathcal{F}_c \phi_{m,n} (c_m^{j+1} c_n^i + c_m^{j-1} c_n^i + c_m^j c_n^{i+1} + c_m^j c_n^{i-1} - 4) = \mathcal{F}_c \psi_{m,n} \quad (3.28)$$

According to [18]:

$$\cos(a+x) = \cos(a)\cos(x) - \sin(a)\sin(x) \quad (3.29)$$

that is

$$\cos_b^{a+x} = c_b^a c_b^x - s_b^a s_b^x \quad (3.30)$$

$\mathcal{F}_c \psi_{m,n}$ is known, the above equation is solved in terms of $\mathcal{F}_c \phi_{m,n}$

Equation 3.28 is simplified:

$$\begin{aligned} c_m^{j+1} c_n^i + c_m^{j-1} c_n^i &= (c_m^j c_m^1 - s_m^j s_m^1) c_n^i + (c_m^j c_m^{-1} - s_m^j s_m^{-1}) c_n^i \Rightarrow \\ c_m^{j+1} c_n^i + c_m^{j-1} c_n^i &= (c_m^j c_m^1 - s_m^j s_m^1) c_n^i + (c_m^j c_m^1 + s_m^j s_m^1) c_n^i \Rightarrow \\ c_m^{j+1} c_n^i + c_m^{j-1} c_n^i &= 2(c_m^j c_m^1) c_n^i \end{aligned} \quad (3.31)$$

and similarly

$$c_m^j c_n^{i+1} + c_m^j c_n^{i-1} = 2(c_n^i c_n^1) c_m^j \quad (3.32)$$

Hence, 3.28 becomes:

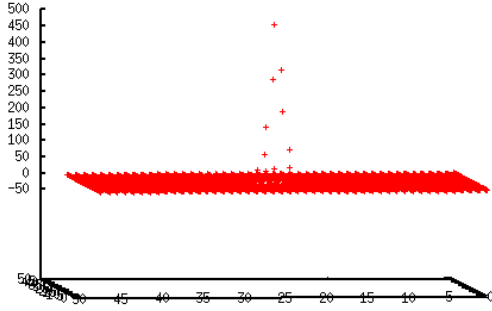
$$\begin{aligned} 2\mathcal{F}_c \phi_{m,n} c_m^j c_n^i (c_m^i + c_n^j - 2) &= \mathcal{F} \psi_{m,n} c_m^i c_n^j \Rightarrow \\ \mathcal{F}_c \phi_{m,n} &= \frac{\mathcal{F}_c \psi_{m,n}}{2(c_m^i + c_n^j - 2)} \end{aligned} \quad (3.33)$$

MATLAB uses 3.33 to solve the Poisson equation with Newmann boundary conditions

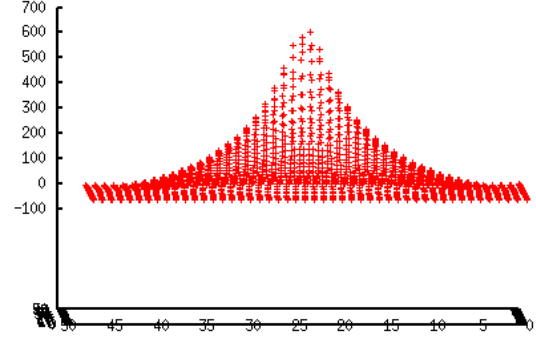
3.5 Existing work in ASP

The goal of ASP is to smooth the density distribution D from equation 2.14. To achieve so, there are 2 existing ways:

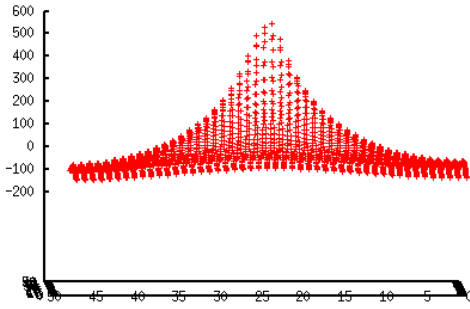
- Poisson solver provided by Math Kernel Library(MKL) to compute the solution of Poisson equation
- Fast Gaussian blur from web reference [19]



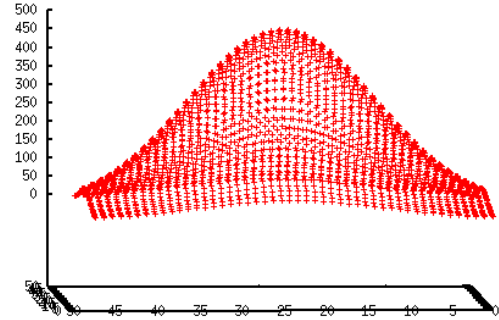
(a) Initial Demand-Supply system



(b) MKL with Dirichlet boundary conditions



(a) MKL with Neumann boundary conditions



(b) Gaussian Blur

As we can see, different solutions give different density maps. The different density maps lead to totally different placements due to the fact that the gradient of the density map influences significantly the move force. For example, if we have a very steep curvature, then the cells will move far away and the placer will have difficulty in returning them back. On the other hand if the curvature is very small, then the placer will perform a lot of iterations to reduce the overlap because it will do very small steps thus having an adverse effect on execution time. Also as we observe, the solution provided by MKL with Neumann boundary conditions compared to that with Dirichlet boundary conditions have some remarkable differences. As for derivatives at boundaries, in Dirichlet boundary conditions shown in figure 3.4b are 0 in large areas instead of Neumann boundary conditions shown in figure 3.5a where the derivative is 0 almost nowhere. This means that in case of Neumann boundary conditions, almost all cells are moved in each iteration. An additional noticeable

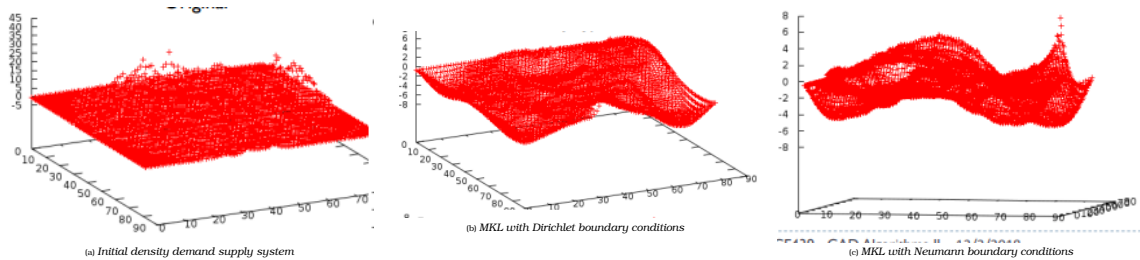
difference is the curvature, which in case of Dirichlet is much more steep than the case of Neumann. Finally, one desirable result provided by Neumann conditions is that the area is balanced. This can be observed if we look at the graphs of each solution. In figure 3.5a the graph starts from a negative value while in 3.4b the graph begins from 0.

MKL accepts as input the grid domain, the initial density demand supply and the type of boundary conditions and returns the smoothed density. So far, the density function is smoothed by MKL with Neumann boundary conditions by default, due to the fact that the experimental results proved that MKL gives better HPWL than Gaussian blur smoothing.

Chapter 4

Poisson Solution - Contribution of this Work

As it has been mentioned in previous chapters, there are two types of boundary conditions provided by MKL, Newmann and Dirichlet. In this thesis we try to satisfy Newmann boundary conditions due to the fact that they have more desirable experimental results. Furthermore solution provided by Newmann boundary conditions leads to a balanced area, that is, there is the same area above and below x-axis



By experiments it is observed that by using Neumann boundary conditions, cells are spread more uniformly. Given that Neumann figure has sharp gradient throughout the area even at the edge of the grid, it is certain that all cells are to move. In the other case, Dirichlet boundary conditions almost render the cells close to edges immobile. This can be observed from the figure in which gradient decreases sharply and remains close to zero. So our first goal is to have same result as this of MKL with Newmann boundary conditions, by an utterly our method. We have approached the problem in 2 ways. The first method is based on Finite differences and the second method is based on Spectral methods. We have developed both of methods in MATLAB for the proof of concept and then in C. Furthermore we have parrallelized the procedure of construction of Poisson matrix Also we have implemented in MATLAB an algorithm that calculates the discrete cosine transform of a function in 2 Dimensions

4.1 Finite Differences Approach

As we have already mentioned, our objective is to solve:

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) \phi(x, y) = -\psi(x, y) \quad (4.1a)$$

$$\frac{\partial \phi}{\partial n} = 0, \text{ on } \Gamma \quad (4.1b)$$

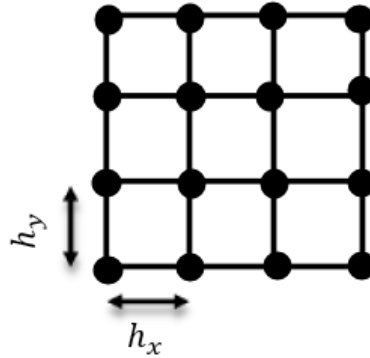
where ϕ is the unknown smoothed density function and ψ is the given density function, Γ is the boundary of core area and the derivative with respect to n means "normal to boundary"

According to 5-point Finite differences methods in which this method is based on

$$\frac{\partial^2 \phi(x, y)}{\partial x^2} = \frac{\phi(x - h_x, y) - 2 \times \phi(x, y) + \phi(x + h_x, y)}{h_x^2} \quad (4.2a)$$

$$\frac{\partial^2 \phi(x, y)}{\partial y^2} = \frac{\phi(x, y - h_y) - 2 \times \phi(x, y) + \phi(x, y + h_y)}{h_y^2} \quad (4.2b)$$

Where h_x and h_y correspond to bin width and bin height respectively, (x, y) corresponds to a grid point.



As explained in 2.5.1, the Poisson problem ends up a system of linear equations 2.40. This thesis works on the modification of 2.40 to satisfy pure homogeneous Neumann boundary conditions. As pinpointed in Chapter 2.5.1, replacing $\frac{\partial^2 \phi}{\partial x^2}$ and $\frac{\partial^2 \phi}{\partial y^2}$ by the second-order approximations at grid point (i, j) equation 4.1b yields:

$$\frac{\phi(x - 1, 1) - 2 \times \phi(x, y) + \phi(x + 1, 1)}{h_x^2} + \frac{\phi(x, y - 1) - 2 \times \phi(x, y) + \phi(x, y + 1)}{h_y^2} = -\psi(x, y) \quad (4.3)$$

The implicit nature of the finite difference equation is obvious in Equation 4.3. The solution at every point depends on the solutions of the four neighboring grid points, which are not known until the entire solution is known, as illustrated in Figure 4.2

In order to apply finite differences in our equation and satisfy the boundary conditions we follow the procedure below:

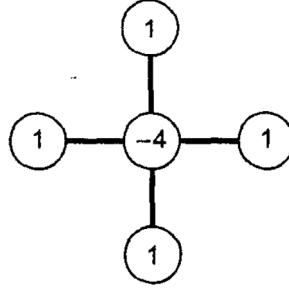
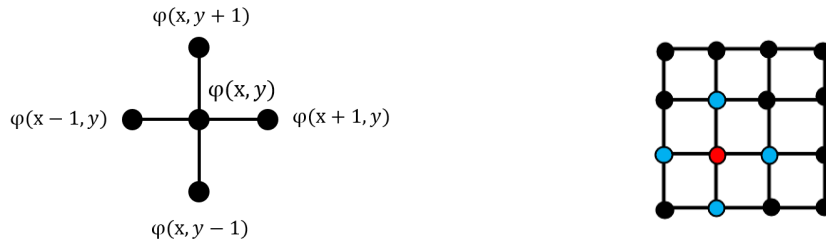


Figure 4.2: 5-points finite differences stencil



- Neumann boundary conditions:

The Neumann boundary conditions involve a derivative and we need to represent it using Finite Differences. We should use an order of finite difference which is the same as the one that we used inside the domain. The five point finite differences is of second order. We thus need to express Neumann boundary conditions using a second order scheme, we have 3 main options that are shown for partial derivatives in x-axis and derivatives in y-axis are obtained similarly:

- central differences

$$\frac{\partial \phi(x, y)}{\partial x} = \frac{\phi(x + 1, y) - \phi(x - 1, y)}{h_x}$$

- forward differences

$$\frac{\partial \phi(x, y)}{\partial x} = \frac{\phi(x + 1, y) - \phi(x, y)}{h_x}$$

- backward differences

$$\frac{\partial \phi(x, y)}{\partial x} = \frac{\phi(x, y) - \phi(x - 1, y)}{h_x}$$

We decided to apply central differences due to the fact that we observed from experimental results that MKL did so. Hence, the first x and y derivatives of boundary points are approximated as:

$$\frac{\partial \phi(x, y)}{\partial x} = \frac{\phi(x+1, y) - \phi(x-1, y)}{h_x}, \quad \frac{\partial \phi(x, y)}{\partial y} = \frac{\phi(x, y+1) - \phi(x, y-1)}{h_y} \Rightarrow$$

$$\phi(x+1, y) = \phi(x-1, y), \quad \phi(x, y-1) = \phi(x, y+1)$$

(4.4a)

Subsequently it is attempted to integrate this information in the existing array.

Boundary nodes, in which Neumann boundary conditions are applied, have not all the neighboring nodes of cross. They are either corner points, which miss 2 neighbors, or non-corner points which miss 1 neighbor. e.g for 4x4 grid:

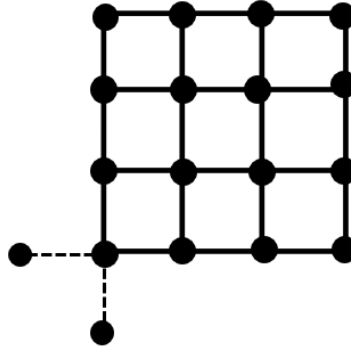


Figure 4.4: ghost points

points $(0, -1)$ and $(-1, 0)$ do not exist in grid but are considered equal to $(0, 1)$ and $(1, 0)$ respectively. In 2.40, equations for all boundary points are to be modified accordingly. Below, it is explained in which way each equation is modified according to the kind of node, that is, corner or non-corner

- Corners: Corners do not have 2 points of their corresponding cross, e.g for point, $(0,0)$ points $(0, -1)$ and $(-1,0)$ do not exist in the grid. Thus putting together 4.3 and 4.4a above, the resultant equation for the $\phi(0, 0)$ corner point becomes:

$$\frac{-2 \times \phi(0, 0) + 2 \times \phi(1, 0)}{h_x^2} + \frac{-2 \times \phi(0, 0) + 2 \times \phi(0, 1)}{h_y^2} = -\psi(0, 0) \quad (4.5)$$

This modification is illustrated in figure 4.5

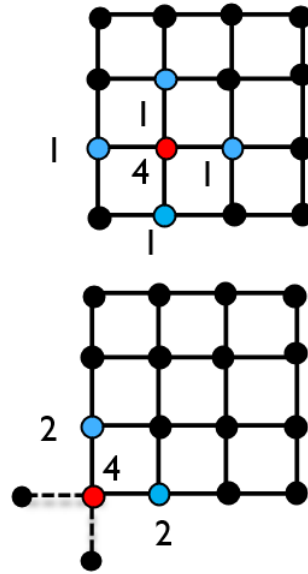


Figure 4.5: weights modification for corner points

- Non-Corner Boundary points: Similarly to changes above, now only 1 grid point takes a 2 x weight, the single point just across the non-existing points. e.g all x-axis points $\phi(0,j)$ have point $\phi(-1,j)$ off grid, thus $\phi(1,j)$ assumes a 2 X weight. E.g equation for point $\phi(0, 1)$

$$\frac{-2 \times \phi(0, 1) + 2 \times \phi(1, 1)}{h_x^2} + \frac{\phi(0, 0) - 2 \times \phi(0, 1) + \phi(0, 2)}{h_y^2} = -\psi(0, 1) \quad (4.6)$$

the above equations yield the following linear system of equations:

$$A \times \phi = \psi \quad (4.7)$$

This modification is illustrated in figure 4.6

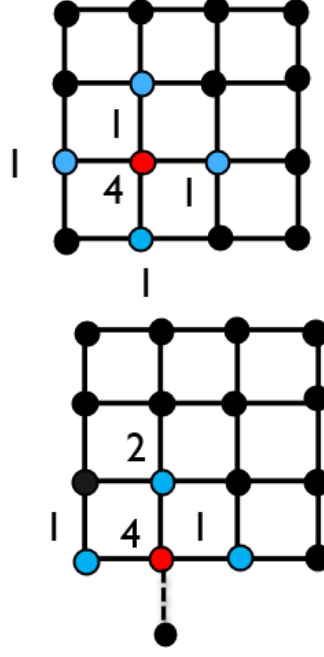


Figure 4.6: weights modification for non-corner point

A =

$$\begin{bmatrix}
 \frac{2}{(h_x)^2} + \frac{2}{(h_y)^2} & -\frac{2}{(h_x)^2} & 0 & -\frac{2}{(h_y)^2} & 0 & 0 & 0 & 0 & 0 & 0 \\
 -\frac{1}{(h_x)^2} & \frac{2}{(h_x)^2} + \frac{2}{(h_y)^2} & -\frac{1}{(h_x)^2} & 0 & -\frac{2}{(h_y)^2} & 0 & 0 & 0 & 0 & 0 \\
 0 & -\frac{2}{(h_x)^2} & \frac{2}{(h_x)^2} + \frac{2}{(h_y)^2} & 0 & 0 & -\frac{2}{(h_y)^2} & 0 & 0 & 0 & 0 \\
 -\frac{1}{(h_y)^2} & 0 & 0 & \frac{2}{(h_x)^2} + \frac{2}{(h_y)^2} & -\frac{2}{(h_x)^2} & 0 & -\frac{1}{(h_y)^2} & 0 & 0 & 0 \\
 0 & -\frac{1}{(h_y)^2} & 0 & -\frac{1}{(h_x)^2} & \frac{2}{(h_x)^2} + \frac{2}{(h_y)^2} & -\frac{1}{(h_x)^2} & 0 & -\frac{1}{(h_y)^2} & 0 & 0 \\
 0 & 0 & -\frac{1}{(h_y)^2} & 0 & -\frac{2}{(h_x)^2} & \frac{2}{(h_x)^2} + \frac{2}{(h_y)^2} & 0 & 0 & -\frac{1}{(h_x)^2} & 0 \\
 0 & 0 & 0 & -\frac{2}{(h_y)^2} & 0 & 0 & \frac{2}{(h_x)^2} + \frac{2}{(h_y)^2} & -\frac{2}{(h_y)^2} & 0 & 0 \\
 0 & 0 & 0 & 0 & -\frac{2}{(h_y)^2} & 0 & -\frac{2}{(h_x)^2} & \frac{2}{(h_x)^2} + \frac{2}{(h_y)^2} & -\frac{1}{(h_x)^2} & 0 \\
 0 & 0 & 0 & 0 & 0 & -\frac{2}{(h_y)^2} & 0 & -\frac{1}{(h_x)^2} & \frac{2}{(h_x)^2} + \frac{2}{(h_y)^2} & -\frac{1}{(h_y)^2} \\
 0 & 0 & 0 & 0 & 0 & 0 & -\frac{2}{(h_x)^2} & 0 & -\frac{1}{(h_x)^2} & \frac{2}{(h_x)^2} + \frac{2}{(h_y)^2}
 \end{bmatrix} \quad (4.8)$$

where ψ is the given non-smoothed density vector and ϕ is the unknown smoothed vector. ϕ is a $M \times N$ vector, considering that the core area is $M \times N$. Hence A is $(M \times N) \times (M \times N)$ array, where M and N are the number of grid points in x and y-axis respectively.

The above matrix is singular, there are infinitely many numerical solutions, any solution plus a constant is still a solution. This happens because unlike Dirichlet, Neumann boundary conditions do not give an initial value to some points of grid, thus making the partial differential equation unable to have a unique solution.

The clue that helps us to deal with the singularity is that we want to have equal area above and below the x-axis, that is, the total area of final density is required to be equal with zero.

$$\iint_R \phi \, dx \, dy = 0 \quad (4.9)$$

where R is the domain $[0, M - 1] \times [0, N - 1]$. In order to achieve this we add one row and one column to the existing matrix A. The new row consists of 1's, that is, new row = $[1, 1, \dots, 1]^T$ and the new column consists of all 0's except for the last $M - 1$ rows, whose equations have to do with the upper boundary points of grid with $y = N - 1$, where X is entered that is, new column = $[0, 0, 0 \mid 0, 0, 0 \mid X, X, X]^T$,

$$X = \frac{\sum_{y=0}^{bins_y} \sum_{x=0}^{bins_x} \psi(x, y)}{2 \times (M - 1)} \quad (4.10)$$

where $bins_x$ and $bins_y$ are the number of bins in x-axis and y-axis respectively

X Value is identified by performing reverse engineering of Math Kernel Library(MKL) provided by Intel corporation. After running experiments and observing the output of each input that we gave to MKL Poisson solver, X value is determined and the formulation of non-singular matrix becomes

A =

$$\begin{bmatrix} \frac{2}{(h_x)^2} + \frac{2}{(h_y)^2} & -\frac{2}{(h_x)^2} & 0 & -\frac{2}{(h_y)^2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{1}{(h_x)^2} & \frac{2}{(h_x)^2} + \frac{2}{(h_y)^2} & -\frac{1}{(h_x)^2} & 0 & -\frac{2}{(h_y)^2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{2}{(h_x)^2} & \frac{2}{(h_x)^2} + \frac{2}{(h_y)^2} & 0 & 0 & -\frac{1}{(h_y)^2} & 0 & 0 & 0 & 0 & 0 \\ -\frac{1}{(h_y)^2} & 0 & 0 & \frac{2}{(h_x)^2} + \frac{2}{(h_y)^2} & -\frac{2}{(h_x)^2} & 0 & -\frac{1}{(h_y)^2} & 0 & 0 & 0 & 0 \\ 0 & -\frac{1}{(h_y)^2} & 0 & -\frac{1}{(h_x)^2} & \frac{2}{(h_x)^2} + \frac{2}{(h_y)^2} & -\frac{1}{(h_x)^2} & 0 & -\frac{1}{(h_y)^2} & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{(h_y)^2} & 0 & -\frac{2}{(h_x)^2} & \frac{2}{(h_x)^2} + \frac{2}{(h_y)^2} & 0 & 0 & -\frac{1}{(h_y)^2} & 0 & 0 \\ 0 & 0 & 0 & -\frac{2}{(h_y)^2} & 0 & 0 & \frac{2}{(h_x)^2} + \frac{2}{(h_y)^2} & -\frac{2}{(h_x)^2} & 0 & X & \\ 0 & 0 & 0 & 0 & -\frac{2}{(h_y)^2} & 0 & -\frac{2}{(h_x)^2} & \frac{2}{(h_x)^2} + \frac{2}{(h_y)^2} & -\frac{1}{(h_x)^2} & X & \\ 0 & 0 & 0 & 0 & 0 & -\frac{2}{(h_y)^2} & 0 & -\frac{1}{(h_x)^2} & \frac{2}{(h_x)^2} + \frac{2}{(h_y)^2} & X & \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix} \quad (4.11)$$

In fact, after performing reverse engineering of MKL we realized that the system that MKL solves is $A\phi = \psi$, where A is the matrix given by 4.11, with equations of corner points divided by 4 and equations of non-corner boundary points divided by 2 and the whole system multiplied by $binh^2$. In our implementation, System $A\phi = \psi$ where A is the matrix given by 4.11, is applicable to direct solver UMFPACK [20]. UMFPACK is a set of routines for solving unymmetric sparse linear systems. In this thesis 4.7 is subject to LU factorization. Specifically, the used routine performs the numerical factorization $PRAQ = LU$, where P and Q are permutation matrices, R is the row scaling, L is unit-lower triangular and U is upper triangular. Finally, A matrix is represented as a csparse matrix and in order to have better execution time we make its construction parallel in each iteration of global placement

4.2 Cosine Transform Approach

Cosine and sine transforms are special cases of Fourier transform. What is more, they are used to solve initial boundary value problems associated with second order partial differential equations. Sine transform is applied to problems with Dirichlet boundary conditions. Similarly cosine transform should be used when boundary conditions are of Neumann type [13].

Due to the fact that the boundary conditions of our problem are Neumann, we apply the cosine transform to the PDE. Our strategy is to take the Cosine transform with respect to x and y and then solve the resultant partial differential equation. Let $\mathcal{F}_c\phi(x, y)$ denote the cosine transform of ϕ

$$\mathcal{F}_c\phi(x, y) = \frac{1}{\sqrt{M}} \frac{1}{\sqrt{N}} \phi(0, 0) + \frac{2}{\sqrt{M}} \frac{2}{\sqrt{N}} \sum_{n=1}^N \sum_{m=1}^M \phi(m, n) \cos\left(\frac{\pi n(2x+1)}{2N}\right) \cos\left(\frac{\pi m(2y+1)}{2M}\right) \quad (4.12)$$

for the sake of simplicity phases of cosines are represented as $w_n = \frac{\pi n(2x+1)}{2N}$, $w_m = \frac{\pi m(2y+1)}{2M}$

Thus by applying the DCT to Poisson equation, we obtain:

$$\mathcal{F}_c\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)\phi = -\mathcal{F}_c\psi \quad (4.13)$$

from reference [8] we know that:

$$\mathcal{F}_c\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)\phi = (w_n^2 + w_m^2)\mathcal{F}_c\phi \quad (4.14)$$

Thus by taking together 4.13 and 4.14

$$\begin{aligned} (w_n^2 + w_m^2)\mathcal{F}_c\phi &= -\mathcal{F}_c\psi \Rightarrow \\ \mathcal{F}_c\phi &= -\frac{\mathcal{F}_c\psi}{(w_n^2 + w_m^2)} \end{aligned} \quad (4.15)$$

Now that we know the cosine transform of ϕ , the inversion formula of discrete cosine transform gives the solution. The inverse cosine transform is:

$$\phi(m, n) = \frac{1}{\sqrt{M}} \frac{1}{\sqrt{N}} \mathcal{F}_c\phi(0, 0) + \sum_{u=1}^N \sum_{v=1}^M \frac{2}{\sqrt{M}} \frac{2}{\sqrt{N}} \mathcal{F}_c\phi(u, v) \cos\left(\frac{\pi u(2m+1)}{2N}\right) \cos\left(\frac{\pi v(2n+1)}{2M}\right) \quad (4.16)$$

Hence, by combining 4.16 and 4.15 we have

$$\phi(m, n) = \frac{1}{\sqrt{M}} \frac{1}{\sqrt{N}} \mathcal{F}_c\phi(0, 0) + \sum_{u=1}^N \sum_{v=1}^M \frac{2}{\sqrt{M}} \frac{2}{\sqrt{N}} \frac{-\mathcal{F}_c\psi(u, v)}{(w_u^2 + w_v^2)} \cos\left(\frac{\pi u(2m+1)}{2N}\right) \cos\left(\frac{\pi v(2n+1)}{2M}\right) \quad (4.17)$$

Now we observe that $\mathcal{F}_c\phi(0, 0)$ cannot be identified from Equation 4.17 due to the fact that we have $u = 0$, $v = 0$ and as a result $w_u = 0$, $w_v = 0$. This means that we have 0 in the denominator. To solve this problem we use the requirement of balanced area. Hence we set $\mathcal{F}_c\phi(0, 0)$ to be equal with zero since it constitutes the sum of all the values of ϕ .

To sum up, if the cosine transform is implemented, the solution can be obtained easily from 4.17. In this thesis, package which provided by reference [21] is used to find the DCT in $N \log N$ operations. The main restrictive requirement of this package is that the number of grid points at x and y axis need to be a power of two.

Chapter 5

Experimental Results

Initially, our 2 methods are implemented in Mathworks MATLAB in order to prove the concept. For the discrete cosine transform we used the existing function `dct2` [22] provided by MATLAB. This function identifies the cosine transform of a function in 2 dimensions. Furthermore we implemented the discrete cosine transform in matlab but with by far higher execution time than the provided `dct2` function. What is more, our Poisson solver is implemented in C and is integrated into an existing, under development, EDA tool called ASP [23]. For the testing of our solution the ASP tool is used solely. Test cases include a total of 13 designs, which have diverse number of components and bins, in order to cover possible corner cases. Now, we will see the results of MATLAB, the experimental flow and the results from ASP tool follow

5.1 MATLAB

In MATLAB experiments we tested several corner cases in order to examine the resultant density map of each method. It is important that in `dct2` function it is not necessary for the number of grid points to be power of 2.

In each experiment we used a 50 x 50 grid and we show the plot of the initial density map, which has red color, and the plot of DCT and Custom method with green and blue colors respectively, in the same plot. For each plot we used `plot3` function of MATLAB which generates 3D plots. Also, we note the maximum value of each result in order to observe the scaling made by each method compared to the initial density map. Observing the resultant density maps we see that if we locate all the cells in any of the 4 corners we have similar behavior, that is, the first 4 experiments have similar results. Afterward we locate all cells in the grid center and we take different results. As we can see in Figure 5.5, if we put all the cells in the middle of density map, the resultant graph of DCT solution has the same gradient right and left of the max value. Unlike the solution of DCT, this of Custom has entirely different gradient right and left of the max value, this means that the cells will move much more to the one side, to this of higher gradient. As we have already mentioned the gradient of the solution is used in global placement algorithm. All these experiments are related to the first iteration of global placement where the total of cells is located in the center. After the passage of some iterations of global

placer, the cells will be more outspread and hence there will be less overlap. With the decline of overlap the gradient will be smaller and the forces will be weaker. So, this behavior is observed only in the first iterations of global placement

- All the cells are in upper left corner of density map

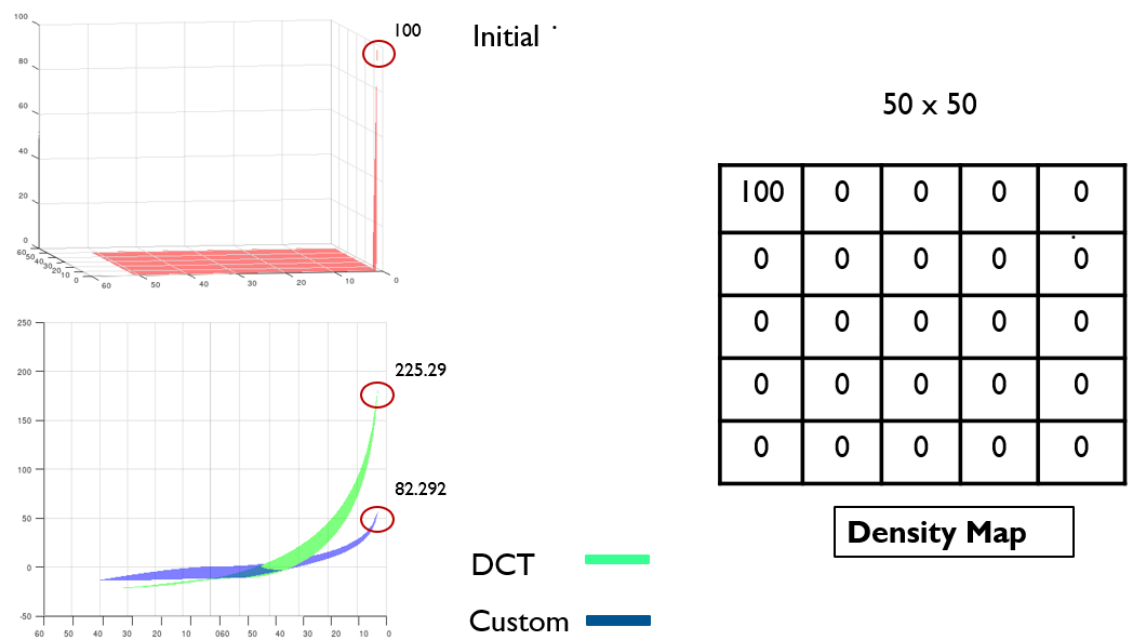


Figure 5.1: All the cells are in upper left corner of density map

- All the cells are in upper right corner of density map

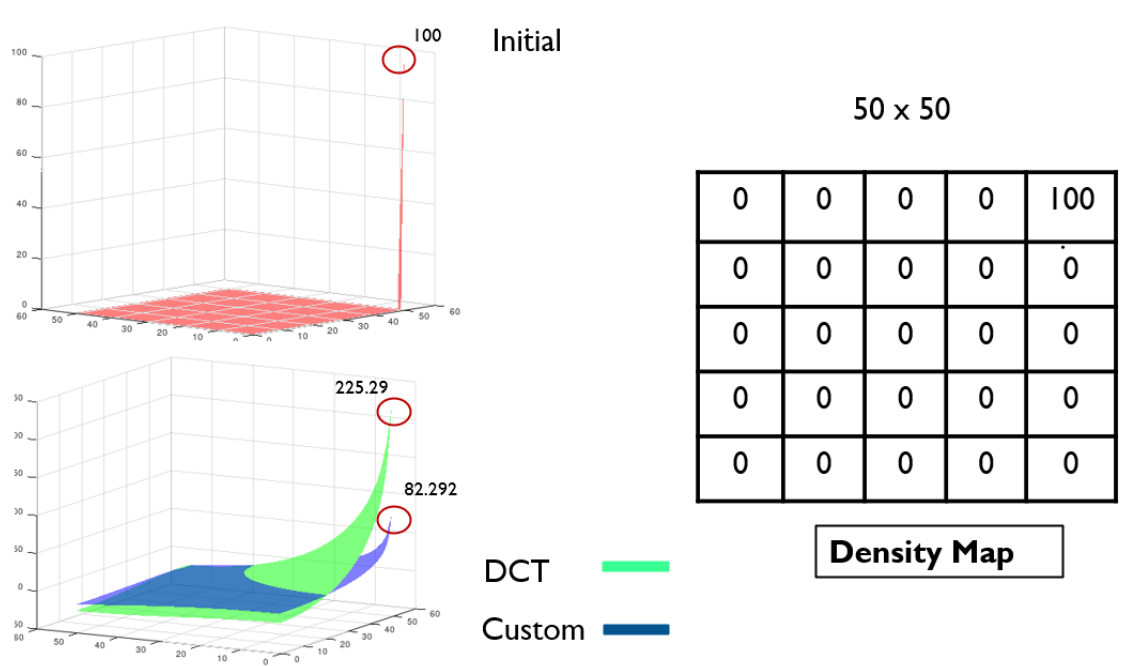


Figure 5.2: All the cells are in upper right corner of density map

- All the cells are in lower right corner of density map

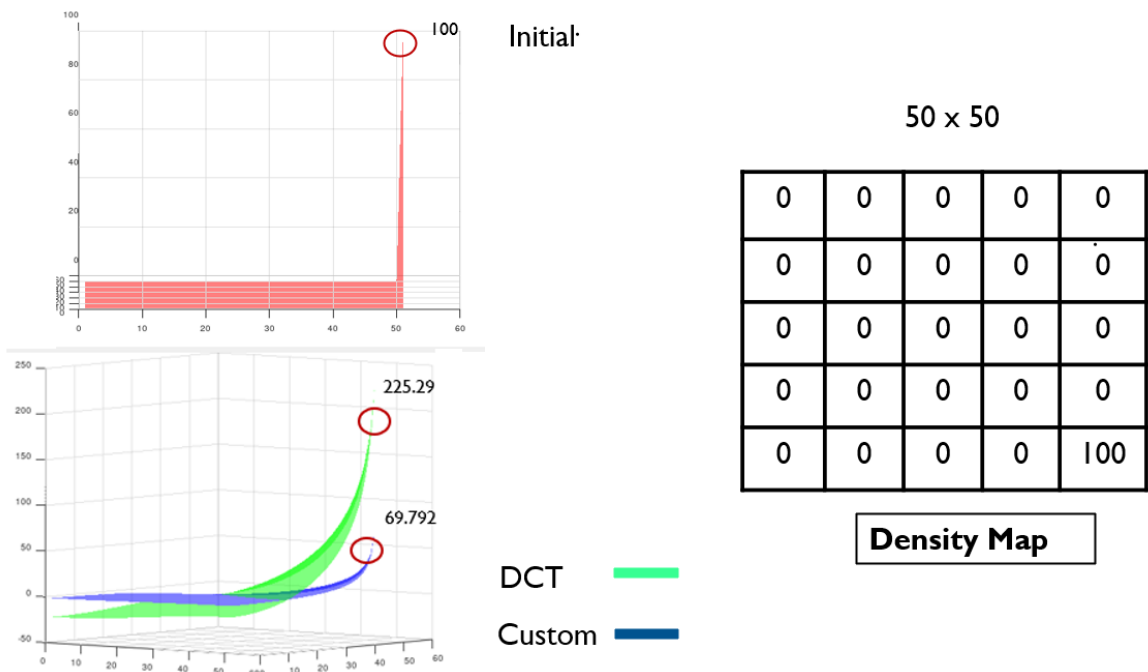


Figure 5.3: All the cells are in lower right corner of density map

- All the cells are in lower left corner of density map

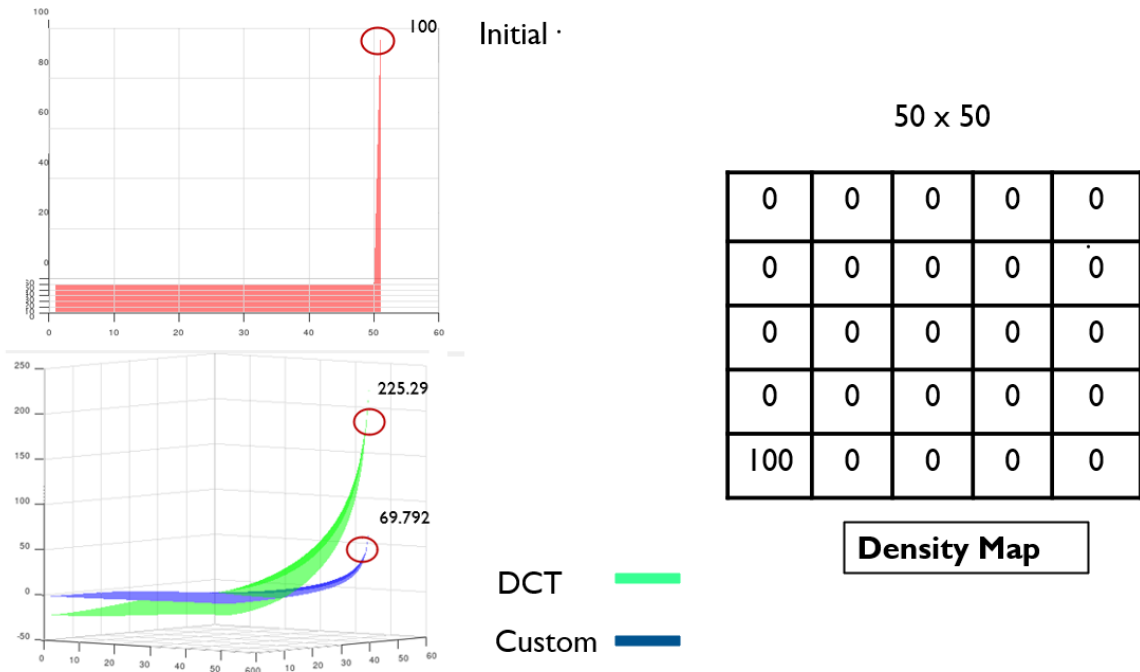


Figure 5.4: All the cells are in lower left corner of density map

- All the cells are in the middle of density map

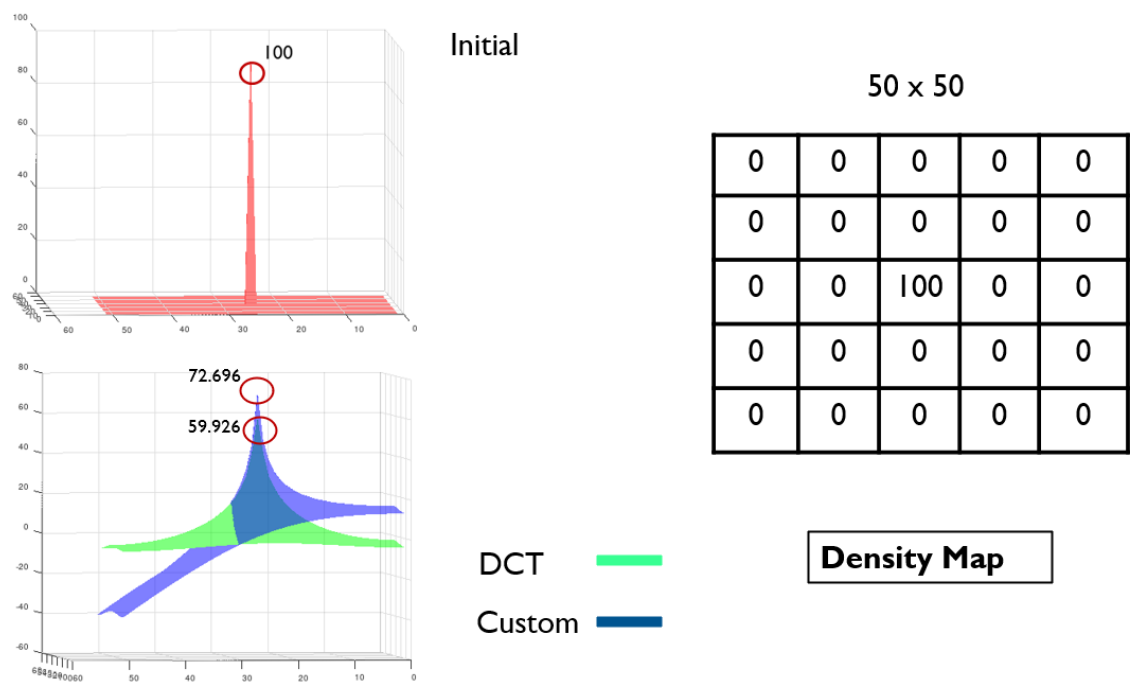


Figure 5.5: All the cells are in the middle of density map

- cells are opposite horizontally

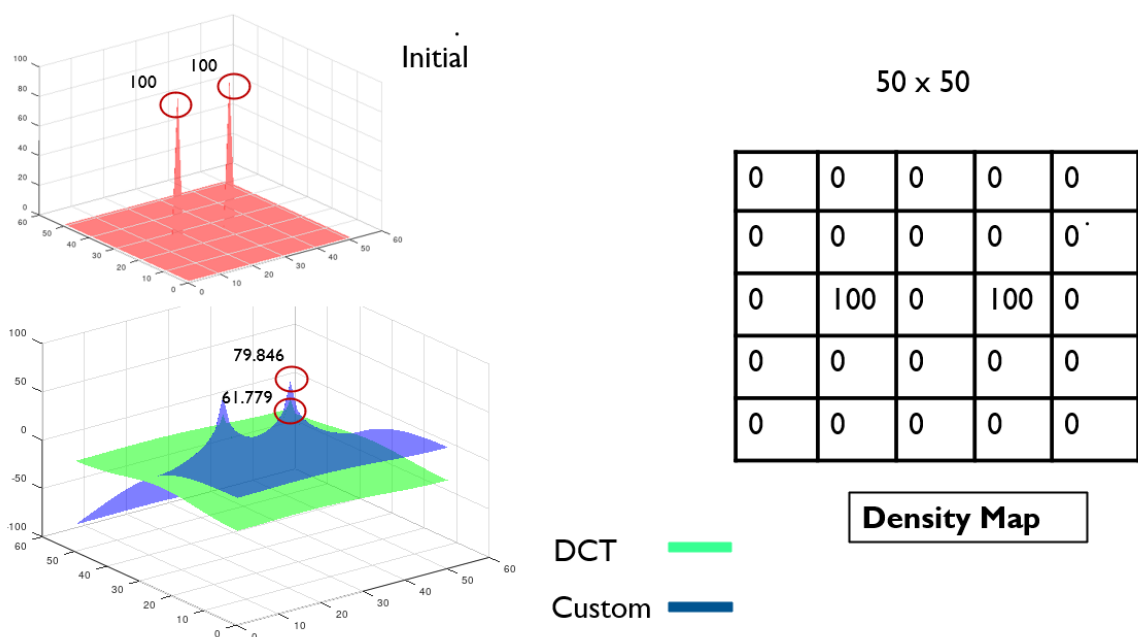


Figure 5.6: cells are opposite horizontally

- cells are opposite vertically

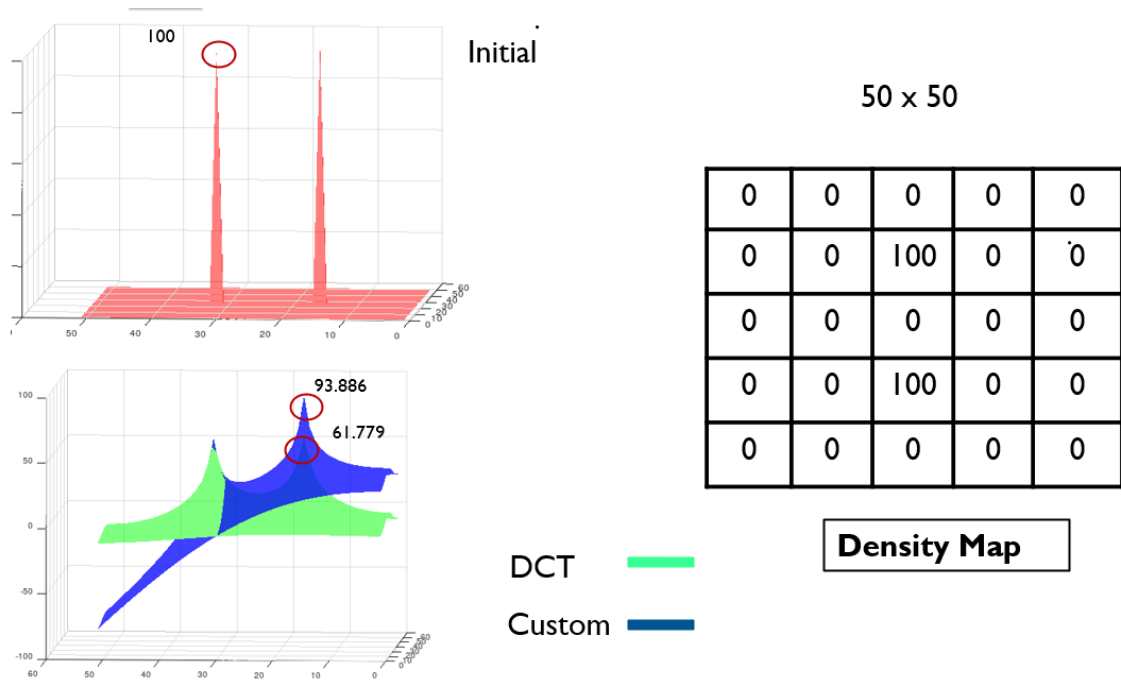


Figure 5.7: cells are opposite vertically

5.2 Our Custom Tool

5.2.1 Flow

For testing our methodology we perform the flow illustrated in Fig. 5.8. Initially a gate level netlist (.v) mapped to the standard cell library is passed to tool. In our case we use a 25nm standard-cell library (.lef) provided by a research institution called IHP. Standard Library encompasses design rules and abstract information about the cells used by placement. Such information is outline of the cell (size and shape), pin locations etc. The next step is the floorplaning, in which blocks are arranged and placed in the core area, taking into account the aspect ratio, target utilization and ensuring that blocks could not have overlap. Afterward, input and output pins ports of design are placed. Next, placement is performed for the purpose of giving the exact location to each instance. Finally we legalize the components to eliminate the overlap of cells. Thesis work affects the placement stage.

parameters for global placement:

- *solver*
- *gradient*

Parameter solver determines which solver is to solve the placement matrix. In the below testcases we test all Poisson solvers (MKL, custom and FFT) with Nesterov iterative solver of linear equations. By testing some designs we can identify whether the results of our custom solution are close to those of MKL which was our first goal. The design tests are summarized in table 5.1

5.2.2 Matrix Approach Results

Tables 5.2 and 5.1 show the HPWL of each design with MKL Poisson solver and with Custom Poisson solver with pure Neumann boundary conditions. The percentage difference is by what percentage Custom increased the HPWL, so a negative percentage means that Custom decreased it. The initial goal was to have the same result with MKL. Deviation results from rounding error between two methods due to the fact that in most cases percentage tends to be zero.

The default numbers of bins are:

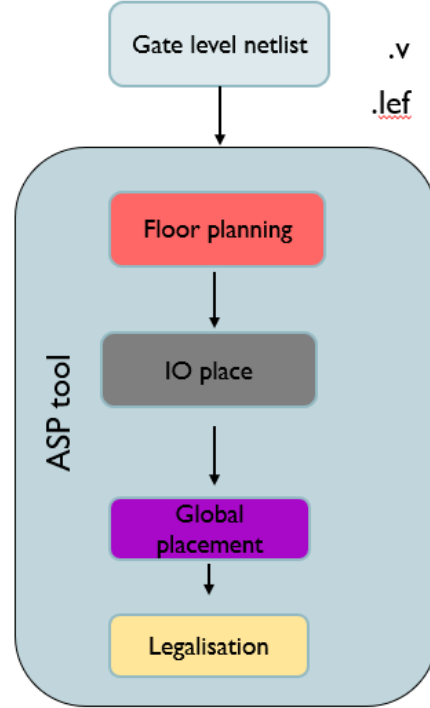


Figure 5.8: Experiments Flow

Design	bins
Design1	90
Design2	101
Design3	192
Design4	70
Design5	228
Design6	285
Design7	236
Design8	406
Design9	323
Design10	324
Design11	468
Design12	483
Design13	1035

Table 5.1: Designs information

Design	HPWL post GP		percentege difference HPWL
	MKL	Custom	
Design1	573069,55	573069,55	0,000%
Design2	503906,474	503906,474	0,000%
Design3	299567,169	299557,145	-0,003%
Design4	222472,706	222472,706	0,000%
Design5	1000937,086	999783,664	-0,115%
Design6	3970804,803	3954043,482	-0,424%
Design7	9253976,32	9254007,079	0,000%
Design8	4055561,609	4064792,309	0,227%
Design9	641754,903	642291,794	0,084%
Design10	635730,997	635401,328	-0,052%
Design11	11440105,13	11448847,94	0,076%
Design12	9465741,105	9437054,378	-0,304%
Design13	17027151,69	17025420,65	-0,010%

Table 5.2: MKL-Custom HPWL comparison post Global placement

The goal of having results close to MKL is achieved, however, execution time of global placement with Custom is by far higher than with MKL. This can be shown from table 5.4. This happens because UMFPACK direct solver is used to solve the deduced linear system 4.11.

design	HPWL post LG		percentege difference HPWL
	MKL	Custom	
Design1	578756,22	578756,22	0,000%
Design2	513909,48	513909,48	0,000%
Design3	303594,882	303543,966	-0,017%
Design4	227516,52	227516,52	0,000%
Design5	1012096,13	1011132,63	-0,095%
Design6	4215097,84	4270323,77	1,293%
Design7	9318834	9318498,42	-0,004%
Design8	4122396,07	4132882,83	0,254%
Design9	674082,204	674572,688	0,073%
Design10	680513,214	679602,136	-0,134%
Design11	11611131,42	11618920,32	0,067%
Design12	9677714,57	9742376,24	0,664%
Design13	17801444,94	17862575,83	0,342%

Table 5.3: MKL-Custom HPWL comparison post Legalisation

MKL Custom execution time

Design	execution time	
	MKL	Custom
Design1	0m15.893s	0m17.431s
Design2	0m19.893s	0m22.952s
Design3	1m0.526s	1m17.850s
Design4	0m9.565s	0m11.571s
Design5	2m15.540s	2m58.181s
Design6	16m24.601s	17m54.982s
Design7	4m45.742s	5m31.830s
Design8	52m17.832s	61m9.636s
Design9	10m28.914s	13m58.465s
Design10	9m44.165s	12m8.257s
Design11	33m50.392s	45m32.374s
Design12	40m6.280s	51m0.969s
Design13	185m14.826s	489m14.192s

Table 5.4: execution time MKL-Custom

5.2.3 FFT Approach Results

For FFT tests, the number of grid points is required to be power of two. In our custom tool we can define the number of bins which is the number of grid points minus 1. There are two types of experiments:

- experiments with number of points rounded up to the next power of 2

Tables 5.5 and 5.6 show the HPWL of each design with MKL Poisson and with Poisson solver based on fast Fourier Transform with pure Neumann boundary conditions. The percentage difference is by what percentage FFT increased the HPWL, hence a negative percentage means that custom decreased HPWL. The initial goal was to have similar results with MKL.

Design	bins
Design1	63
Design2	63
Design3	127
Design4	63
Design5	127
Design6	255
Design7	127
Design8	255
Design9	255
Design10	255
Design11	255
Design12	255
Design13	1023

Design	HPWL post GP		percentege difference HPWL
	MKL	FFT	
Design1	585595,194	582466,441	-0,537%
Design2	530471,66	574798,254	7,712%
Design3	359105,049	378488,592	5,121%
Design4	215049,187	217533,496	1,142%
Design5	1039361,93	1041123,697	0,169%
Design6	4451323,579	4144811,900	-7,395%
Design7	10738797,01	10848540,052	1,012%
Design8	4964117,398	5808312,931	14,534%
Design9	559388,813	581728,103	3,840%
Design10	665650,603	707585,544	5,926%
Design11	13771959,7	12749082,972	-8,023%
Design12	9108659,033	9299177,035	2,049%
Design13	17660673,8	17571836,137	-0,506%

Table 5.5: MKL-FFT HPWL comparison post Global Placement

Design	HPWL post LG		percentege difference HPWL
	MKL	FFT	
Design1	594858,6	591893,82	-0,501%
Design2	545397,72	597282	8,687%
Design3	365058,736	384425,697	5,038%
Design4	221207,7	224066,64	1,276%
Design5	1059474,68	1067693,86	0,770%
Design6	4556583,86	4610114,43	1,161%
Design7	10809805,86	10936012,08	1,154%
Design8	5069881,63	5920987,28	14,374%
Design9	599899,432	623586,95	3,799%
Design10	714983,436	753287,136	5,085%
Design11	14034661,62	13060440,96	-7,459%
Design12	9885508,77	10568296,07	6,461%
Design13	17660673,8	17571836,14	-0,506%

Table 5.6: MKL-FFT HPWL comparison post-legalization

- experiments with number of bins rounded to the previous power of 2

Design	bins
Design1	127
Design2	127
Design3	255
Design4	127
Design5	255
Design6	511
Design7	255
Design8	511
Design9	511
Design10	511
Design11	511
Design12	511
Design13	2047

Design	HPWL post GP		percentege difference HPWL
	MKL	FFT	
Design1	570752,362	571491,075	0,129%
Design2	543901,481	529073,547	-2,803%
Design3	319147,454	337937,327	5,560%
Design4	213635,474	215450,739	0,843%
Design5	1012577,596	1027051,645	1,409%
Design6	3867652,819	3936451,267	1,748%
Design7	9588233,464	9610659,861	0,233%
Design8	4921973,138	4787500,701	-2,809%
Design9	599248,416	589780,93	-1,605%
Design10	672027,98	666649,17	-0,807%
Design11	12006110,56	13520508,72	11,201%
Design12	9372762,578	9805042,825	4,409%
Design13	16680869,77	16312516,06	-2,258%

Design	HPWL post LG		percentege difference HPWL
	MKL	FFT	
Design1	575175,300	576854,88	0,291%
Design2	551074,860	539009,1	-2,239%
Design3	321859,398	340706,271	5,532%
Design4	217005,600	218,445	0,659%
Design5	1021931,720	1035025,51	1,265%
Design6	4226981,420	4176413,33	-1,211%
Design7	9642052,560	9663979,92	0,227%
Design8	4962599,450	4826736,25	-2,815%
Design9	620007,295	616805,909	-0,519%
Design10	709483,508	702513,182	-0,992%
Design11	12132771,000	13662031,74	11,194%
Design12	9854968,520	10292688,34	4,253%
Design13	16680869,773	17435095,49	4,326%

MKL FFT execution time

- Number of bins are rounded to the previous power of two

Design	Execution time	
	MKL	FFT
Design1	0m24.007s	0m18.935s
Design2	0m24.885s	0m24.889s
Design3	1m4.063s	1m4.591s
Design4	0m11.905s	0m18.806s
Design5	2m17.120s	2m16.181s
Design6	17m43.223s	16m14.899s
Design7	4m28.493s	4m16.074s
Design8	35m28.862s	47m5.687s
Design9	13m38.794s	11m38.664s
Design10	11m57.016s	10m44.701s
Design11	24m2.563s	26m49.931s
Design12	40m24.341s	39m37.699s
Design13	132m38.632s	223m32.283s

- Number of bins are rounded to the next power of two

Design	Execution time	
	MKL	FFT
Design1	0m13.825s	0m15.672s
Design2	0m19.380s	0m20.405s
Design3	1m0.124s	1m3.103s
Design4	0m10.033s	0m9.878s
Design5	2m4.183s	2m9.997s
Design6	15m45.982s	16m1.512s
Design7	3m45.306s	4m31.997s
Design8	34m59.594s	30m29.846s
Design9	11m2.630s	10m57.918s
Design10	10m14.736s	11m16.392s
Design11	22m2.867s	22m55.784s
Design12	41m36.809s	41m43.021s
Design13	115m36.179s	117m31.187s

Chapter 6

Conclusion and Future work

Our goal was to explore the solution of Poisson equation and more specifically, for the global placement stage and for algorithms of force directed solvers. Hence, we researched different ways to provide a solution of Poisson equation. Specifically, we examined and we extended the solution based on Finite Differences and the solution based on spectral methods. In results we saw the implementation of 2 methods and we compared them with some existing solutions like this of MKL and we observed that MKL and custom solution generate almost the same results and their difference can be justified by a rounding error. The main problem of custom method was the execution time which was caused by UMFPACK direct linear solver. As a result, a possible future work may encompass the implementation of an iterative solver as well as optimizations such as preconditioning and modifications to the Poisson matrix, which may reduce the execution time significantly. What is more, our custom approach may be extended in order to satisfy additional boundary conditions and even a combination of them. As for spectral method is considered, it reduced the execution time significantly and it had comparable HPWL with MKL. Nevertheless the restriction that this method sets is that it requires the number of grid points in x and y axis to be a power of 2. The solution based on DCT has $O(n \log n)$ complexity but there are additional open source packages that do not have any requirement about the number of bins and they provide as fast solutions as the used package. What is more, we implemented the discrete cosine transform in 2 dimensions in MATLAB and we observed by far worse execution time. Hence another idea of future work is the implementation of a faster DCT algorithm

Appendix A

Acronyms

EDA	Electronic Design Automation
MKL	Math Kernel Library
IC	Integrated Circuit
VLSI	Very Large Scale Integration
HPWL	Half Perimeter Wirelength
GP	Global Placement
LG	Legalisation
DS	Demand-Supply
PDE	Partial Differential Equation
ADI	Alternating Direction Implicit
FDA	Finite Difference Approximations
FDM	Finite Difference methods
FFT	Fast Fourier Transformation
DFT	Discrete Fourier Transformation
DCT	Discrete Cosine Transformation
EG	Explicit Group
EDG	Explicit Decoupled Group
MEDG	Modified Explicit Decoupled Group
SOR	successive over-relaxation

Appendix B

Designs Layouts

B.1 AES

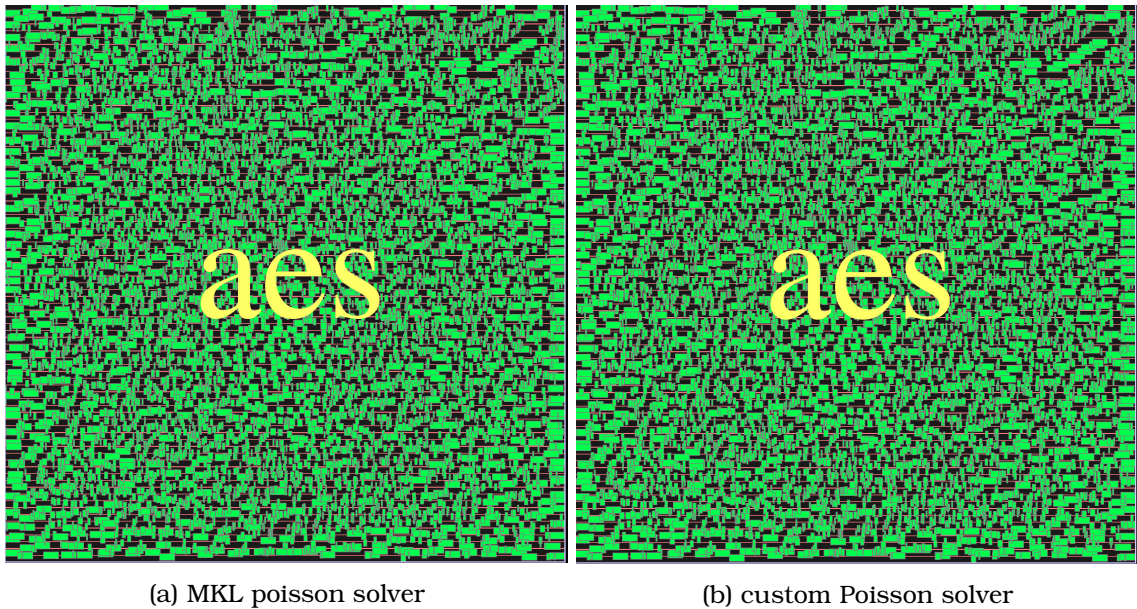
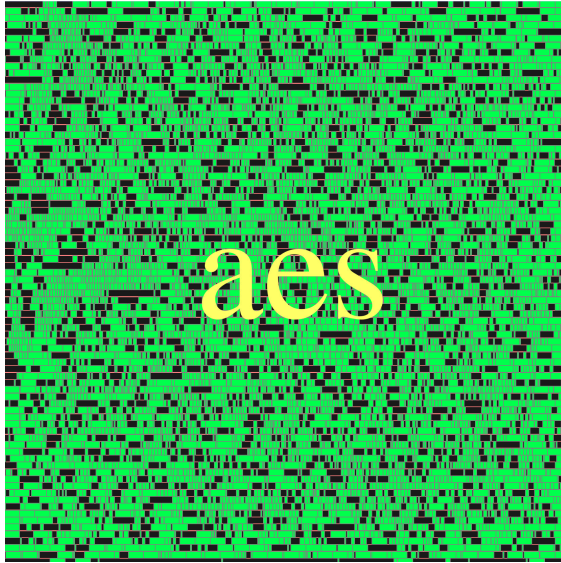
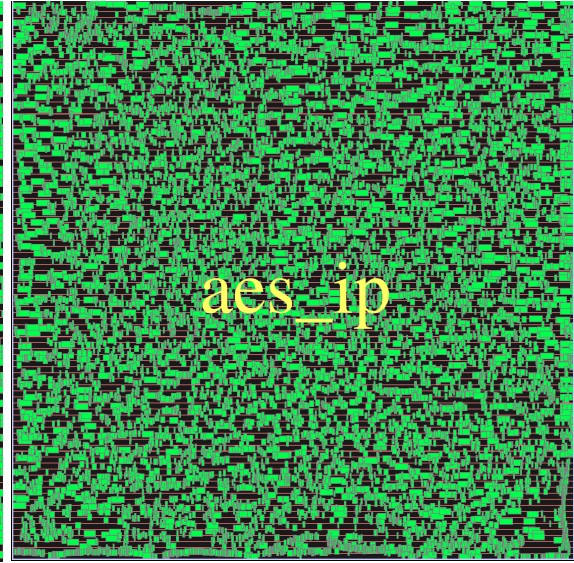


Figure B.1: AES: Global Placement



(a) MKL Poisson solver



(b) FFT Poisson solver

Figure B.2: AES: Global Placement with number of bins equal to the previous power of 2 from default



(a) MKL poisson solver



(b) FFT Poisson solver

Figure B.3: AES: Global Placement with number of bins equal to the next power of 2 from default

B.2 ca53dpu

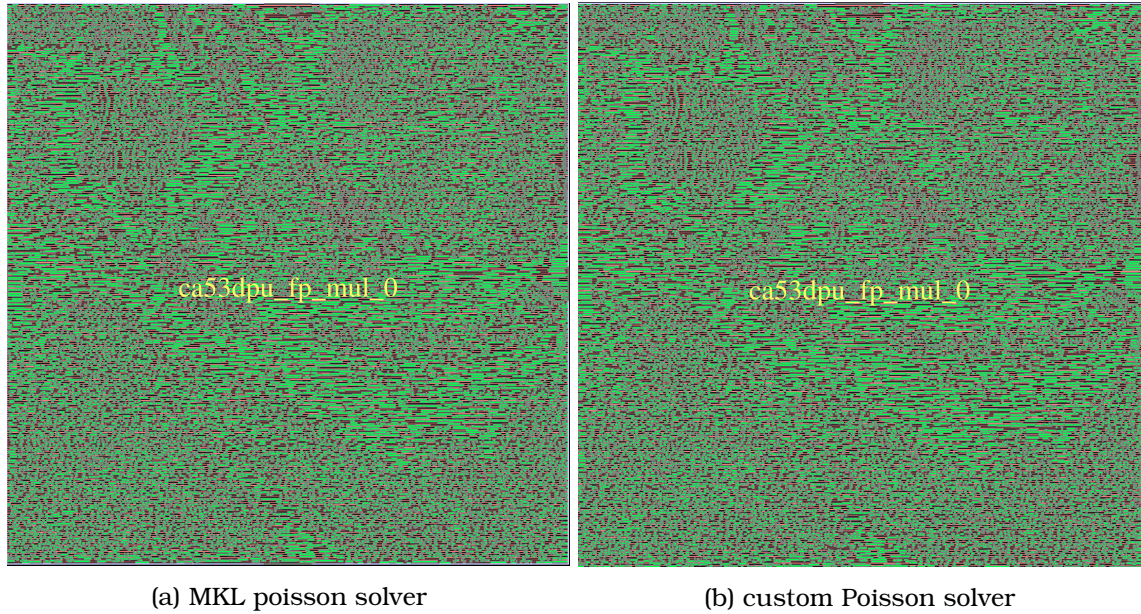


Figure B.4: ca53dpu: Global Placement

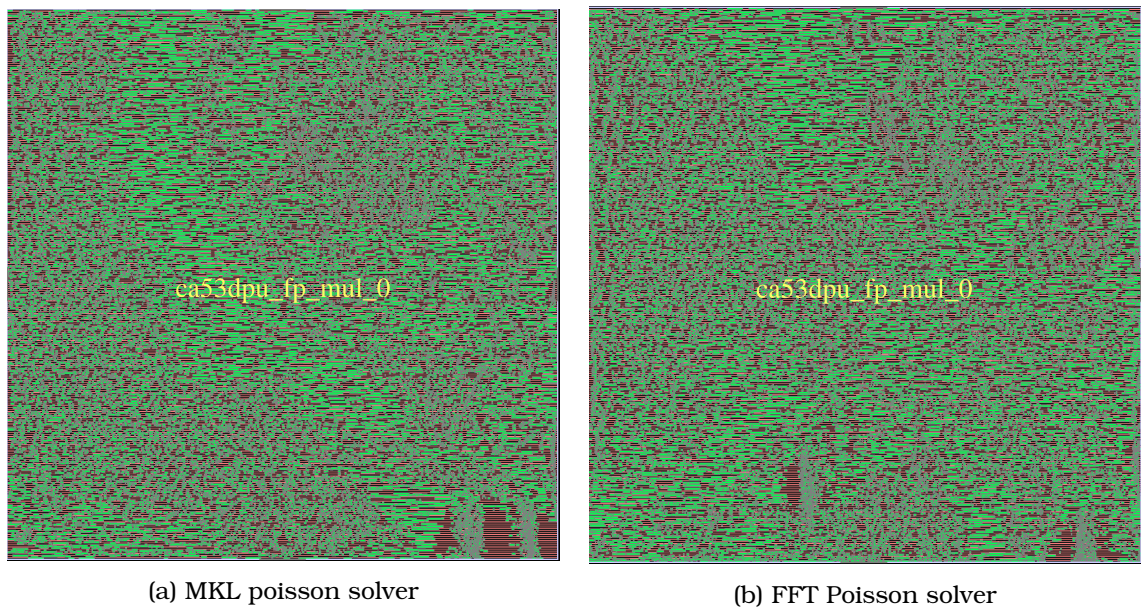
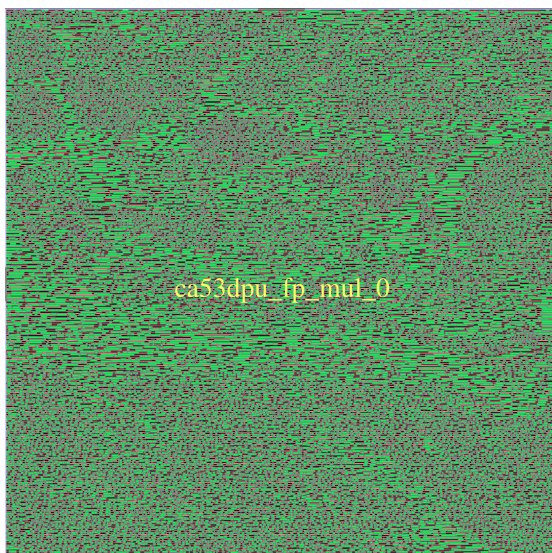
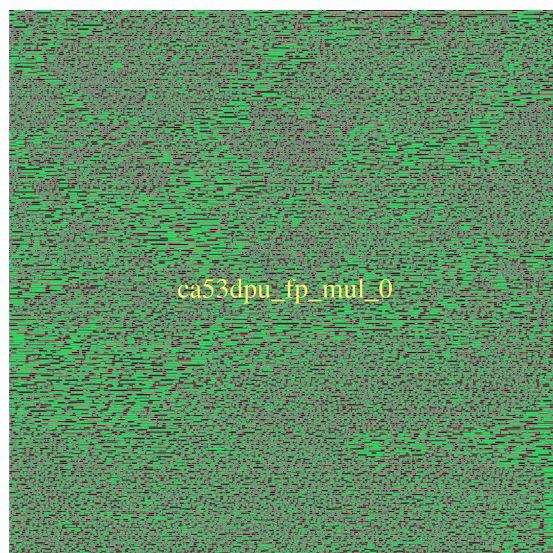


Figure B.5: ca53dpu: Global Placement with number of bins equal to the previous power of 2 from default



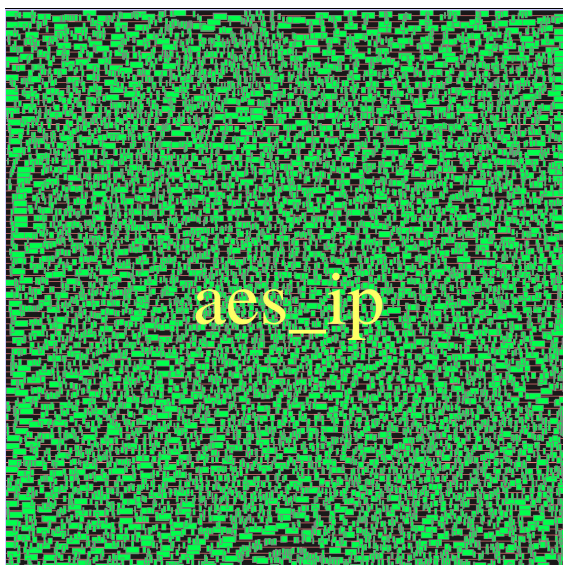
(a) MKL poisson solver



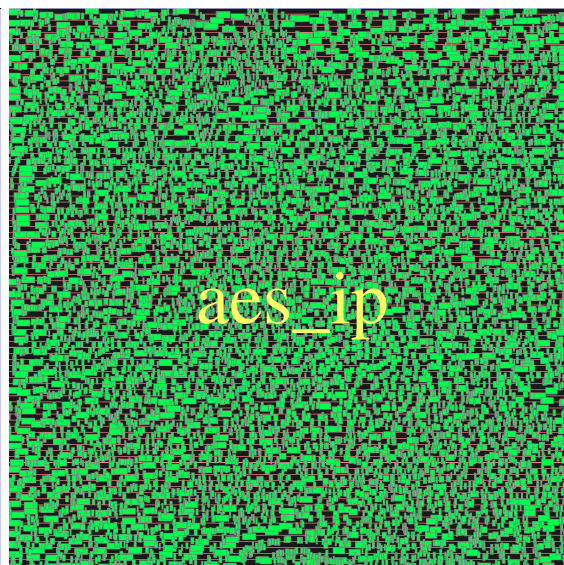
(b) FFT Poisson solver

Figure B.6: `ca53dpu`: Global Placement with number of bins equal to the next power of 2 from default

B.3 `apdtoaes128`

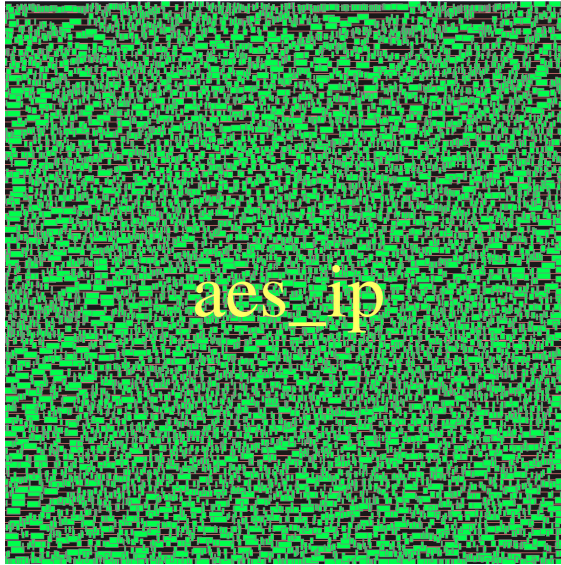


(a) MKL poisson solver

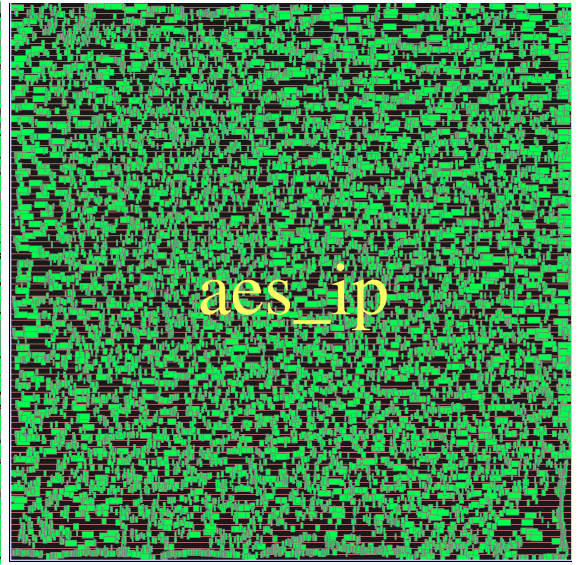


(b) FFT Poisson solver

Figure B.7: `apdtoaes`: Global Placement

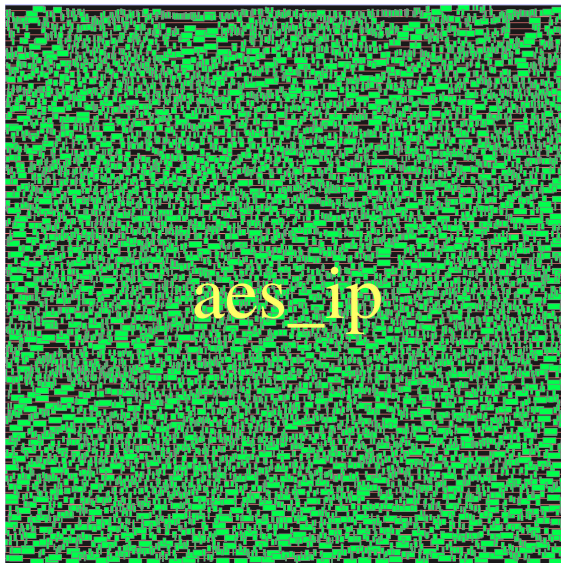


(a) MKL poisson solver

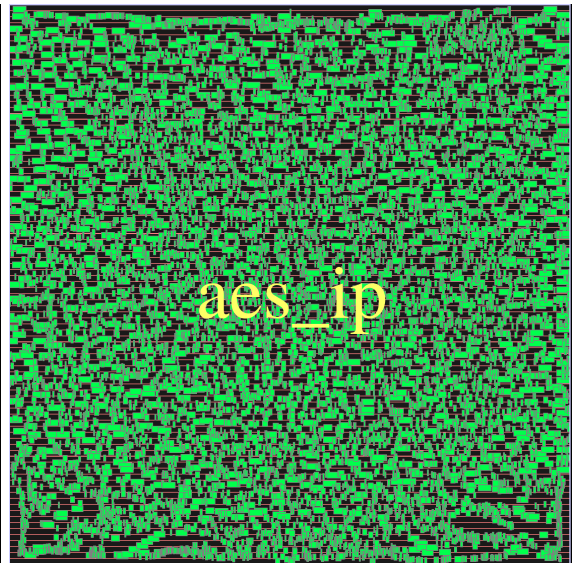


(b) FFT Poisson solver

Figure B.8: apdtoaes: Global Placement with number of bins equal to the previous power of 2 from default



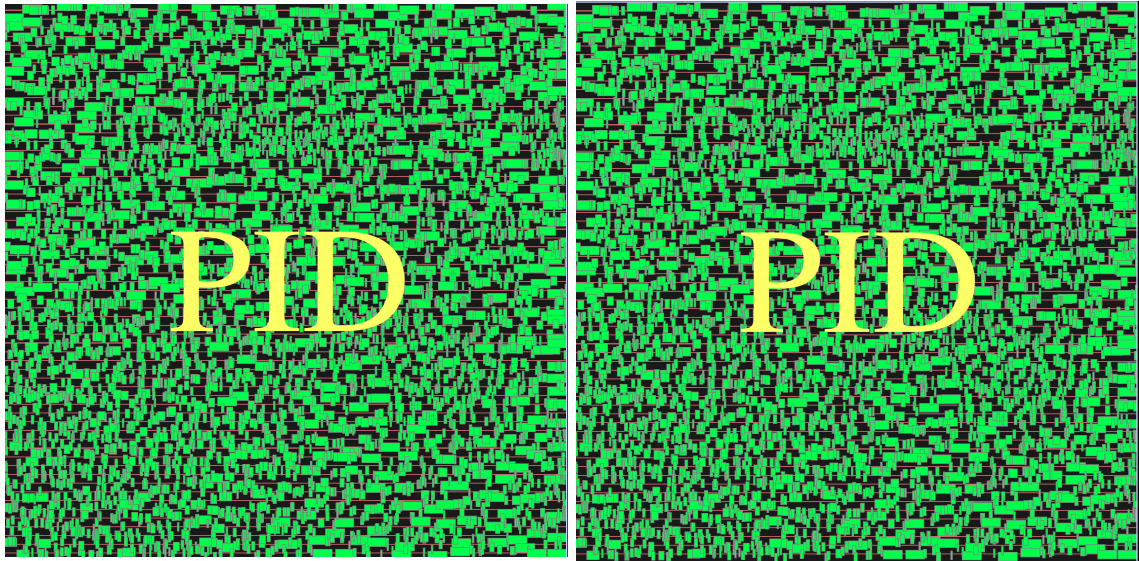
(a) MKL poisson solver



(b) FFT Poisson solver

Figure B.9: apdtoaes: Global Placement with number of bins equal to the next power of 2 from default

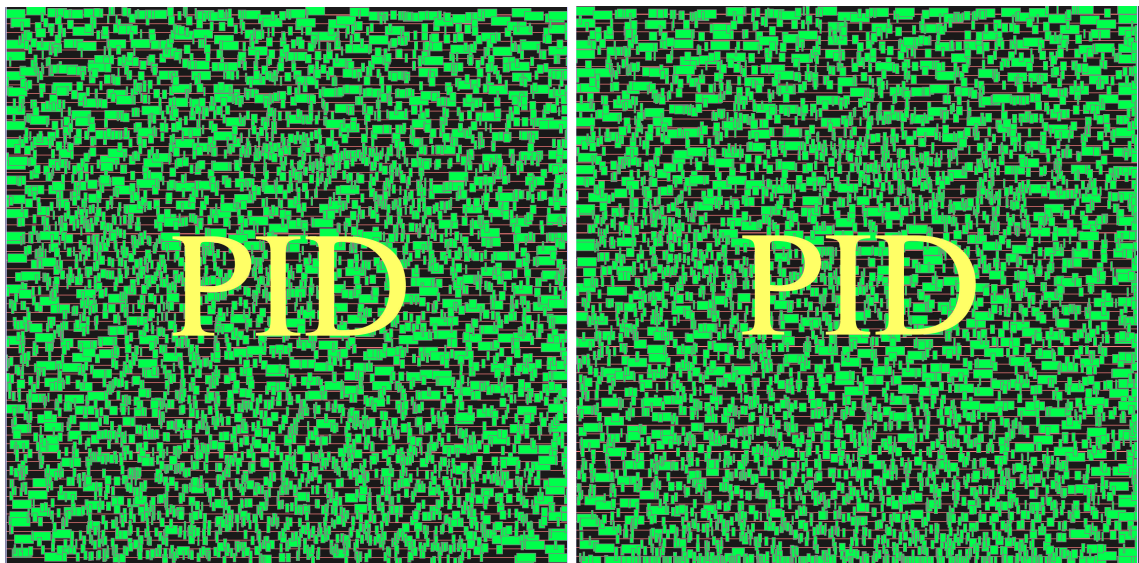
B.4 PID



(a) MKL poisson solver

(b) FFT Poisson solver

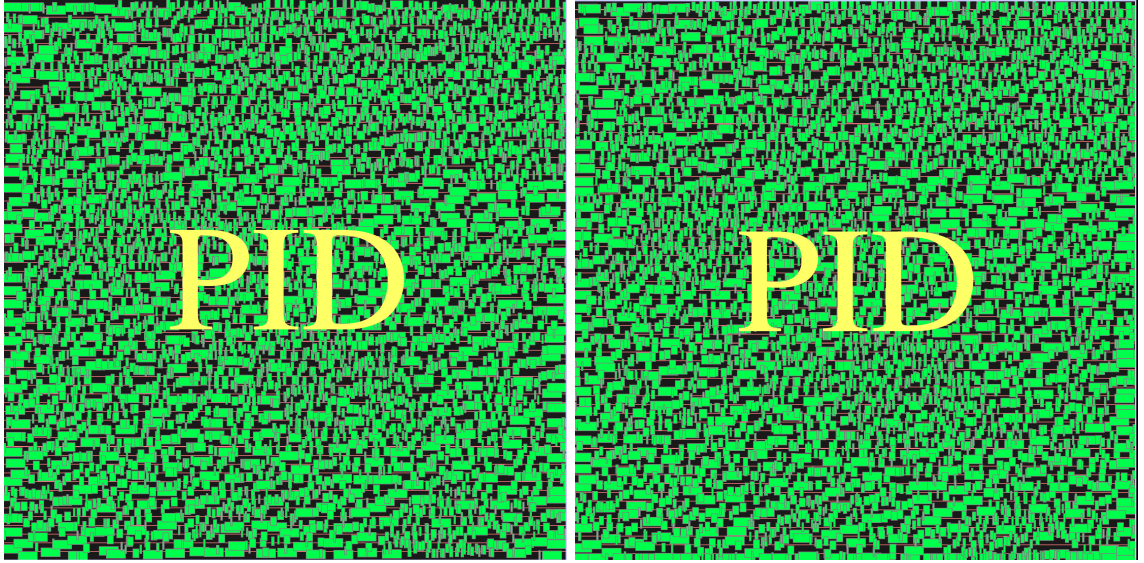
Figure B.10: PID: Global Placement



(a) MKL poisson solver

(b) FFT Poisson solver

Figure B.11: PID: Global Placement with number of bins equal to the previous power of 2 from default

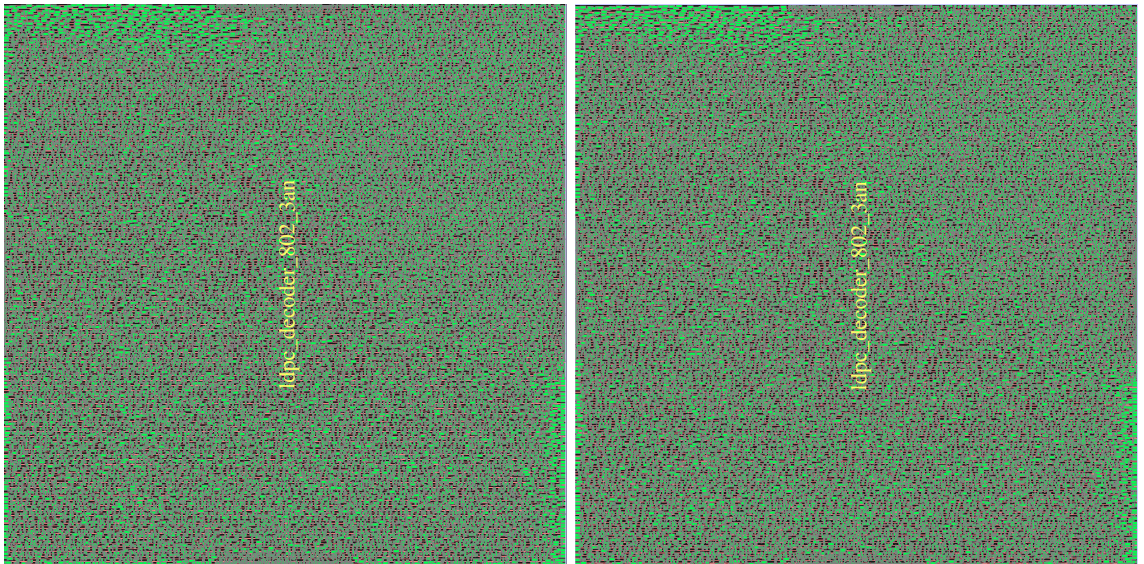


(a) MKL poisson solver

(b) FFT Poisson solver

Figure B.12: PID: Global Placement with number of bins equal to the next power of 2 from default

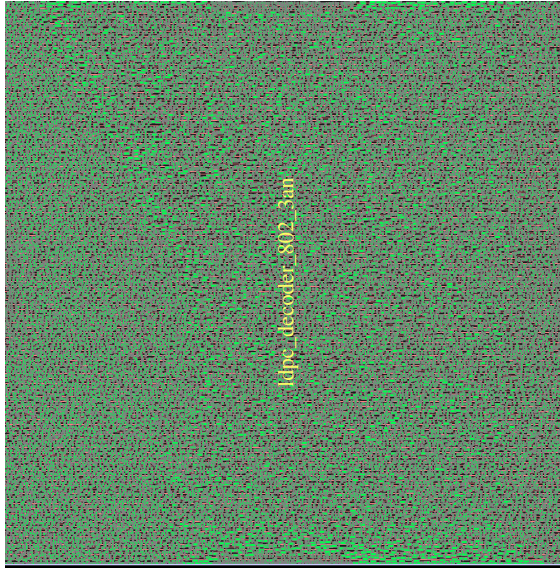
B.5 LDPC



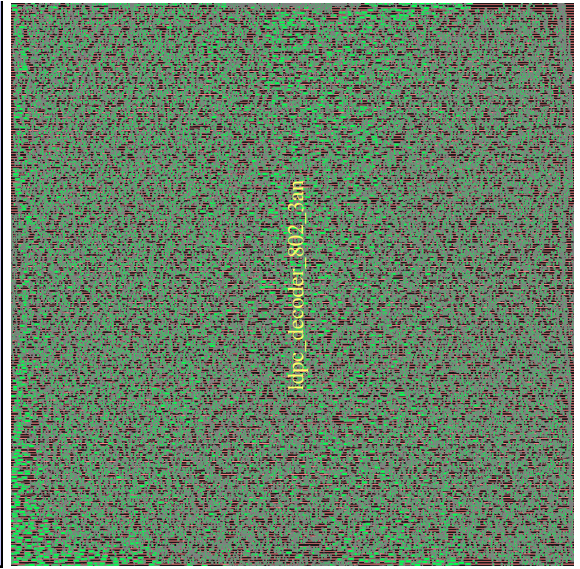
(a) MKL poisson solver

(b) FFT Poisson solver

Figure B.13: LDPC: Global Placementt

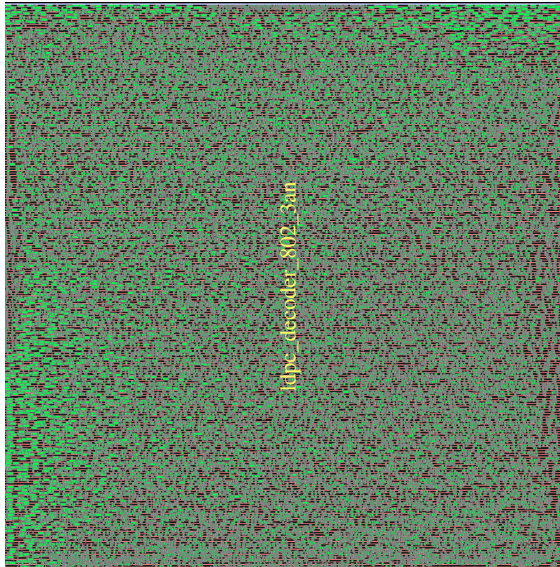


(a) MKL poisson solver

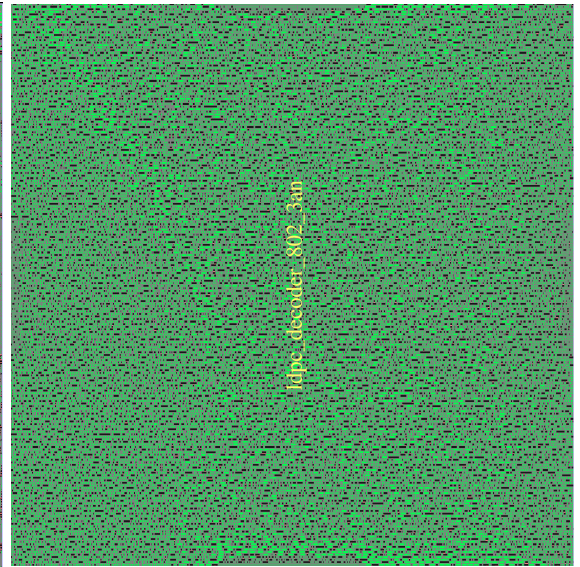


(b) FFT Poisson solver

Figure B.14: LDPC: Global Placement with number of bins equal to the previous power of 2 from default



(a) MKL poisson solver



(b) FFT Poisson solver

Figure B.15: LDPC: Global Placement with number of bins equal to the next power of 2 from default

Bibliography

- [1] A. D. Polyanin, W. E. Schiesser, and A. I. Zhurov, "Partial differential equation," *Scholarpedia*, vol. 3, no. 10, p. 4605, 2008, revision #121514.
- [2] D. M. Christodoulou and D. Kazanas, "Gauss's law and the source for Poisson's equation in modified gravity with Varying G," *Monthly Notices of the Royal Astronomical Society*, vol. 484, no. 1, pp. 1421–1425, 01 2019. [Online]. Available: <https://doi.org/10.1093/mnras/stz120>
- [3] J. C. Maxwell, *A treatise on electricity and magnetism*. Clarendon press, 1873, vol. 1.
- [4] Q. Luo, T. Ren, and D. Liang, "Discretized pressure poisson algorithm for the steady incompressible flow on a nonstaggered grid," *Numerical Heat Transfer, Part B: Fundamentals*, vol. 71, no. 6, pp. 549–559, 2017. [Online]. Available: <https://doi.org/10.1080/10407790.2017.1326773>
- [5] D. MacMillen, R. Camposano, D. Hill, and T. Williams, "An industrial view of electronic design automation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 12, pp. 1428–1448, 2000.
- [6] A. B. Kahng, J. Lienig, I. L. Markov, and J. Hu, *VLSI physical design: from graph partitioning to timing closure*. Springer Science & Business Media, 2011.
- [7] P. Spindler, U. Schlichtmann, and F. M. Johannes, "Kraftwerk2—a fast force-directed quadratic placement approach using an accurate net model," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 8, pp. 1398–1411, 2008.
- [8] D. Owen, "Advanced engineering mathematics, erwin kreyszig, wiley, london, 1972. no. of pages: 898. price:£ 6.25," 1973.
- [9] V. Fuka, "PoisFFT – a free parallel fast poisson solver," *Applied Mathematics and Computation*, vol. 267, pp. 356–364, sep 2015. [Online]. Available: <https://doi.org/10.10162Fj.amc.2015.03.011>
- [10] Z. Pan, J. Whitehead, S. Thomson, and T. Truscott, "Error propagation dynamics of PIV-based pressure field calculations: How well does the pressure poisson solver perform inherently?" *Measurement Science and Technology*, vol. 27, no. 8, p. 084012, jul 2016. [Online]. Available: <https://doi.org/10.1088/0957-0233/27/8/084012>
- [11] J. D. Hoffman and S. Frankel, *Numerical methods for engineers and scientists*. CRC press, 2018.

- [12] C. Bernardi and Y. Maday, "Spectral methods," in *Techniques of Scientific Computing (Part 2)*, ser. Handbook of Numerical Analysis. Elsevier, 1997, vol. 5, pp. 209–485. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1570865997800038>
- [13] D. W. Trim, *Applied partial differential equations*. PWS-Kent Publishing Company Boston, MA, 1990.
- [14] U. Schumann and R. A. Sweet, "Fast fourier transforms for direct solution of poisson's equation with staggered boundary conditions," *Journal of Computational Physics*, vol. 75, no. 1, pp. 123–137, 1988. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/0021999188901027>
- [15] D. Fortunato and A. Townsend, "Fast Poisson solvers for spectral methods," *IMA Journal of Numerical Analysis*, vol. 40, no. 3, pp. 1994–2018, 11 2019. [Online]. Available: <https://doi.org/10.1093/imanum/drz034>
- [16] M. Othman and A. Abdullah, "An efficient four points modified explicit group poisson solver," *International Journal of Computer Mathematics*, vol. 76, no. 2, pp. 203–217, 2000.
- [17] R. W. Hockney, "A fast direct solution of poisson's equation using fourier analysis," *J. ACM*, vol. 12, no. 1, p. 95–113, jan 1965. [Online]. Available: <https://doi.org/10.1145/321250.321259>
- [18] S. Habre and M. Abboud, "Students' conceptual understanding of a function and its derivative in an experimental calculus course," *The Journal of Mathematical Behavior*, vol. 25, no. 1, pp. 57–72, 2006.
- [19] <http://blog.ivank.net/fastest-gaussian-blur.html>.
- [20] T. Davis, "Umfpack version 4.3 user guide," 11 2005.
- [21] T. Ooura. (2006). [Online]. Available: <https://www.kurims.kyoto-u.ac.jp/~ooura/fft.html>
- [22] <https://www.mathworks.com/help/images/ref/dct2.html>.
- [23] "ASP (Automated Structured Placement) Tool."