Chapter 9

Finite Pointset Method: Stability and Convergence

9.1 Filtering Algorithms

In Chapter 7, we have shown that different weight functions have impact on FPM simulation in terms of better conditioned system as well as accuracy of the system. We have established that, among all other weight functions, which were used in literature so far, only Gaussian function has better accuracy but it gives very high condition number (see [Som04c]). So, the question is still remain unsolved about good conditioned system with better accuracy and sufficiently fast convergence. Therefore, in this section, we have made an attempt to answer the following questions:

1. Do we have efficeint algorithms to extract number of particles (to filter number of particles) around the central particle in such a way that it preserves accuracy and at the same time gives good conditioned system?
2. Does this algorithms are fast enough?

This idea for filtering the particles around the central particle is to enhance the speed of computation and getting better conditioned system. We have proposed seven kinds of constructions of particles, distributed regularly as well as irregularly around the central particle in some domain, where we want to solve our Poisson equation. We shall construct an algorithm on the basis of this distribution and we shall incorporate these algorithms in our weighted least square method

for solving PDE. At the end, we have given some numerical test example to check the efficiency of algorithms. We have considered the following cases for distribution of points (particles) around the central particles:

- **Case-1:** Select any one particle as a central particle in a given domain. Around that central particle, consider all regularly or randomly distributed particles of the domain, as neighboring particles (see Figure- 9.1 and 9.8).
- **Case-2:** Select any one particle as a central particle in a given domain. Around that central particle, consider all regularly or randomly distributed particles which are immediate adjacent to central particle, as neighboring particles (see Figure-9.2 and 9.9).
- **Case-3:** Select any one particle as a central particle in a given domain. Around that central particle, consider all regularly or randomly distributed particles on the left of central particle and some randomly distributed points on right which are immediate adjacent to central particle, as neighboring particles (see Figure -9.3 and 9.10).
- **Case-4:** Select any one particle as a central particle in a given domain. Around that central particle, consider all regularly or randomly distributed particles on the right of central particle and some randomly distributed points on left which are immediate adjacent to central particle, as neighboring particles (see Figure-9.4 and 9.11).
- **Case-5:** Select any one particle as a central particle in a given domain. Around that central particle, consider all regularly or randomly distributed particles which are far right from central particle and on left the randomly distributed particles which are immediate adjacent to central particle, as neighboring particles (see Figure -9.5 and 9.12).
- **Case-6:** Select any one particle as a central particle in a given domain. Around that central particle, consider all regularly or randomly distributed particles which are far left from central particle and on right the randomly distributed particles which are immediate adjacent to central particle, as neighboring particle (see Figure -9.6 and 9.13).
- **Case-7:** Select any one particle as a central particle in a given domain. Around that central particle, consider some regularly or randomly distributed particles on left as well as on right which are far from central particle, as a neighboring particle (see Figure -9.7 and 9.14).

9.1.1 New Filtering Algorithms for the Neighboring Particles

Algorithm-1

- Consider any random particle as a central particle in the computational domain. Around that central particle, consider six circles of small radius $r_i, (i = 1, 2, \cdots, 6)$. We should note that $r_1 < r_2 < r_3 < r_4 < r_5 < r_6$.
- Consider the circle C_1. The point at an angle $\theta_1 = 0°$ on the circle C_1 is denoted by $(r_1, 0°)$. We shall consider this point as first neighboring particle to the central particle.
- The point at an angle $\theta_{i+1} = \theta_i + 60° \quad (i = 1 : 5)$, on circle $C_i \quad (i = 2 : 6)$ is denoted by $(r_i, \theta_i)(i = 2 : 6)$. These five more points are the another neighboring particles of the central particle.
- For the point on boundaries as a central particle, we shall take immediate neighboring points of a central particles lies in the computational domain as neighboring particles.
- See Figure-9.15.

Algorithm-2

- Consider any random particle as a central particle in the computational domain. Around that central particle, consider six circles of small radius $r_i, (i = 1, 2, \cdots, 6)$. We should note that $r_1 < r_2 < r_3 < r_4 < r_5 < r_6$.
- Consider the circle C_1. The point at an angle $\theta_1 = 360°$ on the circle C_1 is denoted by $(r_1, 360°)$. We shall consider this point as first neighboring particle to the central particle.
- The point at an angle $\theta_{i+1} = \theta_i - 60° \quad (i = 1 : 5)$, on circle $C_i \quad (i = 2 : 6)$ is denoted by $(r_i, \theta_i)(i = 2 : 6)$. These five more points are the another neighboring particles of the central particle.
- For the point on boundaries as a central particle, we shall take immediate neighboring points of a central particles lies in the computational domain as neighboring particles.
- See Figure-9.16.

Algorithm-3

- Consider any random particle as a central particle in the computational domain. Around that central particle, consider two circles of small radius $r_i, (i = 1, 2)$. We should note that $r_1 < r_2$.
- Consider the first circle C_1. The point at an angle $\theta = 0°, 120°, 240°$ on the circle C_1 is denoted by $(r_1, 0°)$, $(r_1, 120°)$, and $(r_1, 240°)$. We shall consider these points as first three neighboring particles to the central particle.
- Similarly, consider the second circle C_2. The point at an angle $\theta = 60°, 180°, 360°$ on the circle C_2 is denoted by $(r_2, 60°)$, $(r_2, 180°)$, and $(r_2, 300°)$. We shall consider these points as another three neighboring particles to the central particle.
- So, we have total number of six neighboring particles around the central particle.
- For the point on boundaries as a central particle, we shall take immediate neighboring points of a central particles lies in the computational domain as neighboring particles.
- See Figure-9.17.

Algorithm-4 (Voronoi Diagram)

- The partitioning of a plane with n points in to n convex polygons such that each polygon contains exactly one point and every point in a given polygon is closer to its central point than to any other describes a *Voronoi diagram.* A Voronoi diagram is sometimes also known as *Dirichlet tessellation.* The cells are called Dirichlet regions, Thiessen polytopes or Voronoi polygons.
- Consider one particle as central particle and we shall consider neighboring particles as all points which lies in immediate adjacent cells of the central particle.
- See Figure-9.18, Figure-9.19, Figure-9.20.

We have considered the following test examples for implementing these algorithms.

9.2 Test Examples

9.2.1 Dirichlet Problem

We have considered the following Dirichlet problem in $1D$:

$$\frac{d^2y}{dx^2} + 4\frac{dy}{dx} + 4y = \exp x; \;\; y(0) = 1, \;\; y(1) = 0. \tag{9.1}$$

We have considered the following problem in $2D$. Let Ω be a domain.

$$\Omega = \{(x, y) | -1 < x < 1; \;\; and \;\; -1 < y < 1\}$$

$$u_{xx} + u_{yy} = 4; \;\; on \;\; \Omega \tag{9.2}$$

$$u = x^2 + y^2; \;\; on \;\; \partial\Omega \tag{9.3}$$

9.2.2 Neumann Problem

We have considered the following Neumann problem in $1D$:

$$\frac{d^2y}{dx^2} + 4\frac{dy}{dx} + 4y = x^2 + 1; \;\; \frac{dy}{dx}(0) = 0, \;\; \frac{dy}{dx}(1) = 0 \tag{9.4}$$

We have considered the following problem in $2D$. Let Ω be a domain.

$$\Omega = \{(x, y) | 0 < x < 1, \;\; and \;\; 0 < y < 1\}$$

$$u_{xx} + u_{yy} = -\cos(\pi x), \;\; on \;\; \Omega \tag{9.5}$$

$$\frac{\partial u}{\partial \mathbf{n}} = 0, \;\; on \;\; \partial\Omega \tag{9.6}$$

9.2.3 Mixed Problem

We have considered the following mixed problem in $1D$:

$$\frac{d^2y}{dx^2} + 4\frac{dy}{dx} + 4y = \exp x; \;\; y(0) = 1, \frac{dy}{dx}(1) = 0 \tag{9.7}$$

We have considered the following Helmholtz problem in $2D$. Let Ω be a domain.

$$\Omega = \{(x, y) | -1 < x < 1, \;\; and \;\; -1 < y < 1\}$$

$$
\begin{aligned}
u_{xx} + u_{yy} + u &= 4 + x^2 + y^2, \quad on \ \ \Omega \\
\frac{\partial u}{\partial \mathbf{n}} &= 2y, \quad on \ \ y = -1, \\
\frac{\partial u}{\partial \mathbf{n}} &= -2y, \quad on \ \ y = 1, \\
\frac{\partial u}{\partial \mathbf{n}} &= 2x, \quad on \ \ x = -1, \\
\frac{\partial u}{\partial \mathbf{n}} &= -2x, \quad on \ \ x = 1.
\end{aligned} \tag{9.8}
$$

9.3 Influence of Particle Distribution on Error and Condition Number of Matrix for $1D$ Problems

9.3.1 Conclusion for Dirichlet Problem in $1D$

We have made the following observations:

- We have a better condition number for **Case-1** and **Case-7** in compare to other cases.
- For **Case-2**, we have high condition number but error is $2,2517 \ \ e-13$.
- For **Case-3**, **Case-4**, **Case-5**, and **Case-6**, we have a high condition number but error is $2.5194 \ \ e-4$ but not good comparable to **Case-2**.
- For **Case-7**, condition number is higher slightly in compare to **Case-1** but error is good in compare to **Case-2**.
- *So, it is observed that when we have a far left and far right particles from central particle, we have a better error and better condition number.*
- See Table - 9.1.

9.3.2 Conclusion for Neumann and Mixed Problem in $1D$

We have made the following observations:

- We have a better condition number for **Case-1** and **Case-7** but not good and effective as in compare to Dirichlet problem.
- In other cases, including **Case-7**, condition number is too high.
- For **Case-2**, we have high condition number but more accurate.
- For **Case-3**, **Case-4**, **Case-5**, and **Case-6**, we have a high condition number but error is of order $1e-3$, so not good comparable to **Case-2**.
- For **Case-7** and **Case-2**, condition number is higher slightly but error approximation is good.
- *So, it is observed that when we have considered all points neighboring of central particle or far left and far right particles from central particle, then we have a better error approximation but high condition number. For **Case-1**, good condition number but error is not good.*
- See Table - 9.2 and Table - 9.3.

9.4 Influence of Particle Distribution on Error and Condition Number of Matrix for $2D$ Problem

9.4.1 Conclusion for Dirichlet Problem in $2D$

We have made the following observations:

- When we consider only particles which are immediate surrounding to central particle then condition number of big matrix in equation (7.15) as well as small matrix in equation () is high in compare to all the particles which are in neighbors of central particle.
- Accuracy of results is better in case of particles which are immediate surrounding to the central particle.
- **Case-1** and **Case-2** worked for all weight functions.
- See Table - 9.4 and Table - 9.5.

9.4.2 Conclusion for Neumann and Mixed problem in $2D$

We have made the following observations:

- When we consider only particles which are immediate surrounding to central particle then condition number of big matrix in equation 7.15 and condition number of small matrix in equation () is lower in compare to the case of all the particles which are in neighbors of central particle.
- Accuracy of results are better.
- **Case-1** and **Case-2** worked for only Gaussian type weight function.
- Other weight functions does not work.
- In case of considering all particles around the central particle, other weight functions do not work.
- See Table - 9.6 and Table - 9.7.

9.4.3 Conclusion for Helmholtz problem in $2D$

We have made the following observations:

- When we consider only particles which are immediate surrounding to central particle then condition number of big matrix in equation (7.15) is same but the condition number of small matrix in equation () is slightly higher in compare to the case taking all particles as neighboring particles to the central particle.
- Accuracy of result is better.
- **Case-1** and **Case-2** worked for only Gaussian type weight function.
- Other weight functions do not work.
- See Table - 9.8 and Table - 9.9.

9.5 Conclusion for Algorithms

9.5.1 Algorithm-1

- BICG and GMRES methods takes only 2 and 9 iterations respectively, but CGS and QMR takes 11 iterations to converge to the solution at desired tolerance.
- Error is less in CGS as well as QMR in compare to BICG and GMRES.
- LSQR takes number of iterations more and error is also very high.
- See Table - 9.10.

9.5.2 Algorithm-2

- GMRES is stagnated.
- BICG does not converge to its desired tolerance.
- CGS, QMR and LSQR converges to its desired tolerance. Less number of iterations in CGS in compare to QMR and LSQR.
- Error is same in all three cases.
- LSQR takes more iterations and error is also high.
- See Table - 9.11.

9.5.3 Algorithm-3

- GMRES is stagnated.
- BICG does not converge to its desired tolerance.
- CGS and QMR takes same numbers of iterations.
- LSQR takes more iterations in compare to CGS and QMR.
- Error is same in all three cases.
- See Table - 9.12.

In short, Algorithm-1 is better in compare to other algorithms in terms of iterations. Algorithm-1 takes nearly three times less iterations in compare to Algorithms-2 and Algorithm-3. Error in Algorithm-1 is less in compare to Algorithms-2 and Algorithm-3. Better conditioned system is for Algorithm-1. For Algorithm-2 and Algorithm-3, condition number is too high. CGS gives more accurate results.

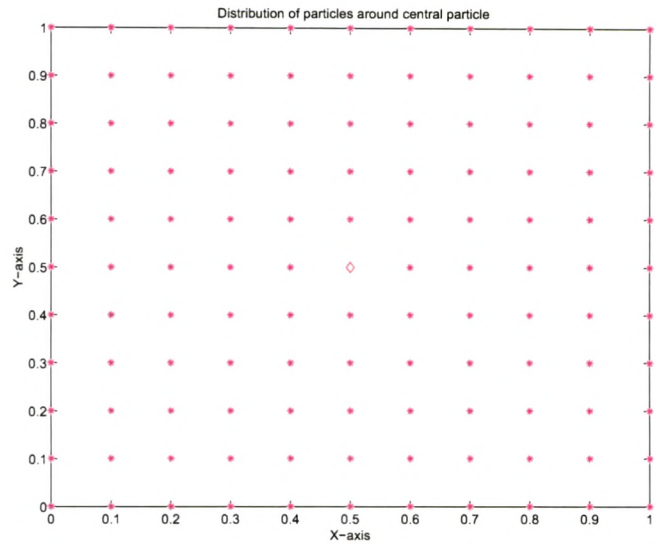

Figure 9.1: **Regular Distribution: All particles around the central particle.**

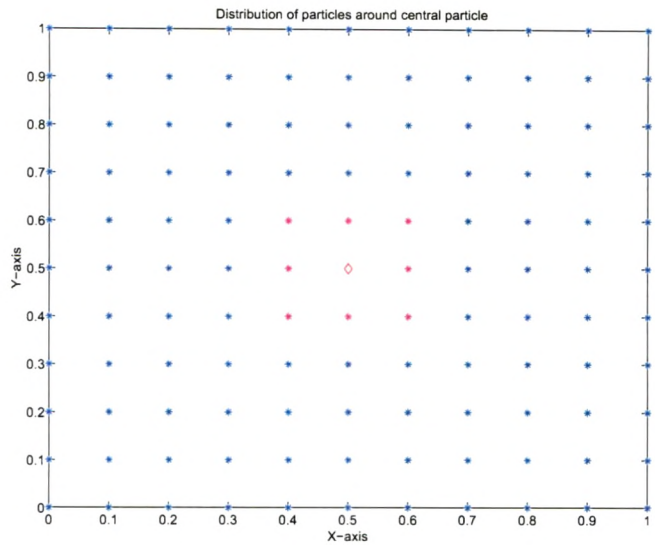

Figure 9.2: **Regular Distribution: Only particles which are immediate surrounding to central particle.**

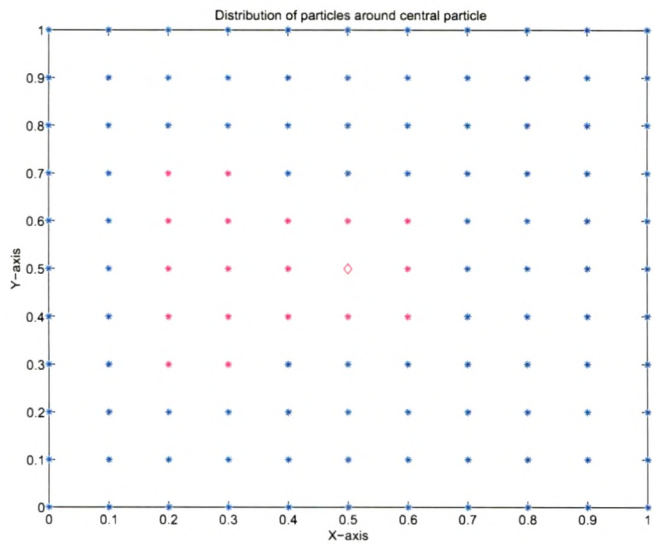

Figure 9.3: **Regular Distribution: More particles on left of central particle and some particle on right of central particle.**

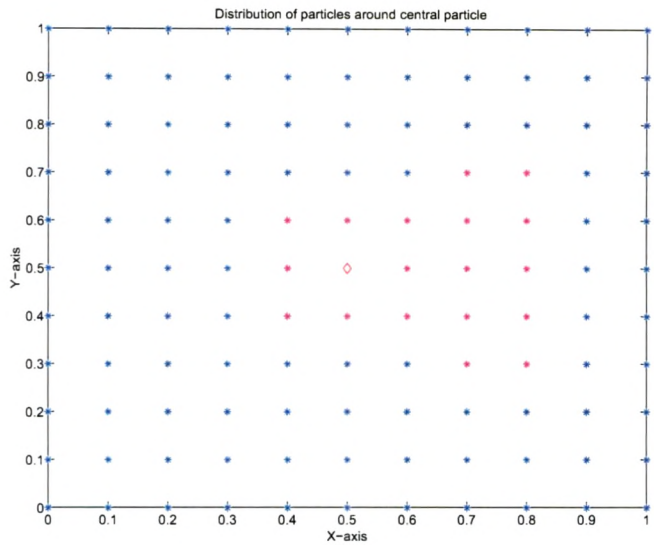

Figure 9.4: **Regular Distribution: More particles on right of central particle and some on left of central particle.**

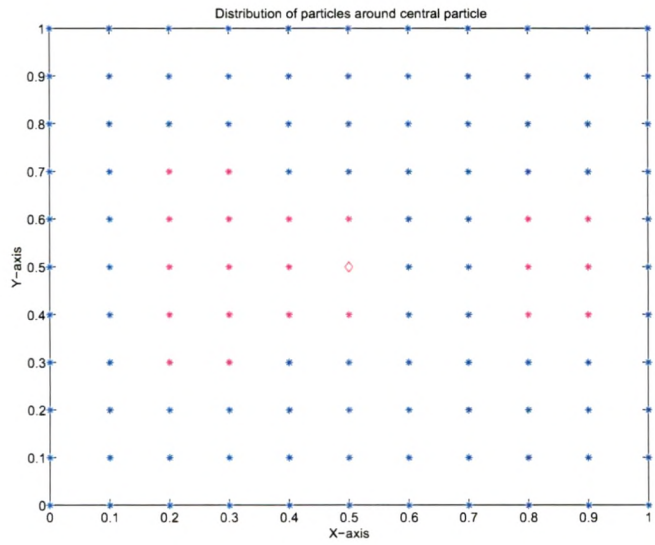

Figure 9.5: **Regular Distribution: Particles which are far right from central particle and some on left which are adjacent to central particle.**

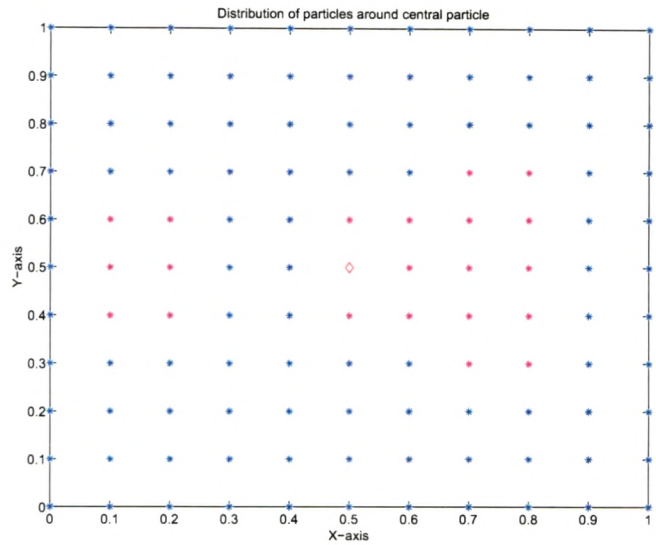

Figure 9.6: **Regular Distribution: Particles which are far left from central particle and some on right which are adjacent to particles.**

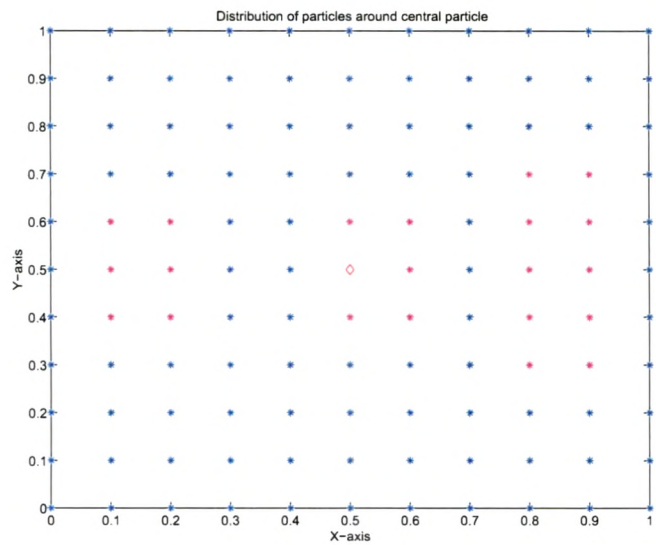

Figure 9.7: **Regular Distribution: Particles which are far right and far left from central particle.**

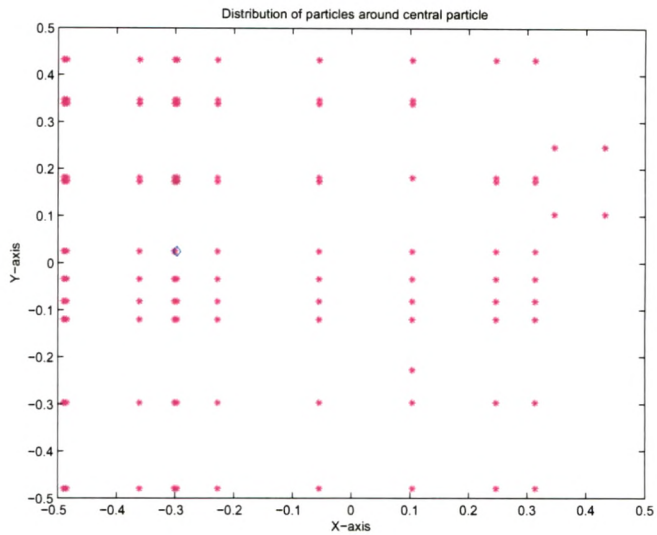

Figure 9.8: **Irregular Distribution: All particles around the central particle.**

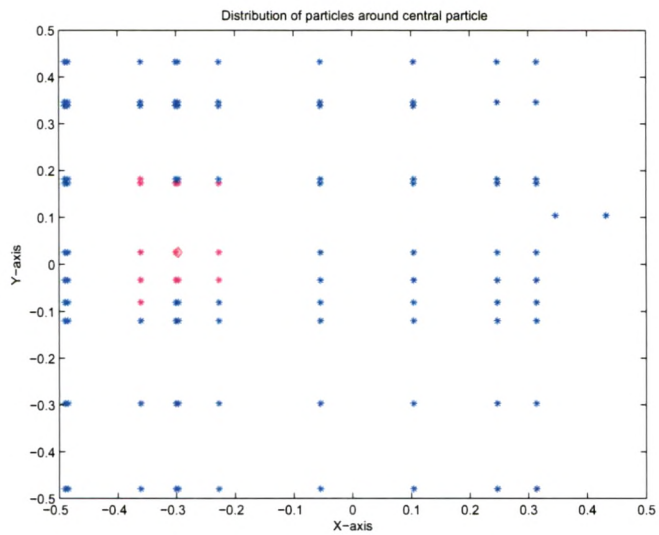

Figure 9.9: **Irregular Distribution: Only particles which are immediate surrounding to central particle.**

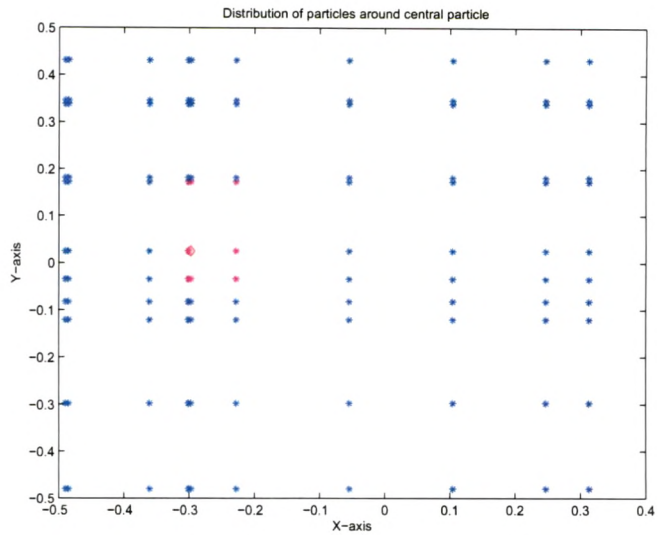

Figure 9.10: **Irregular Distribution: More particles on left of central particle and some particle on right of central particle.**

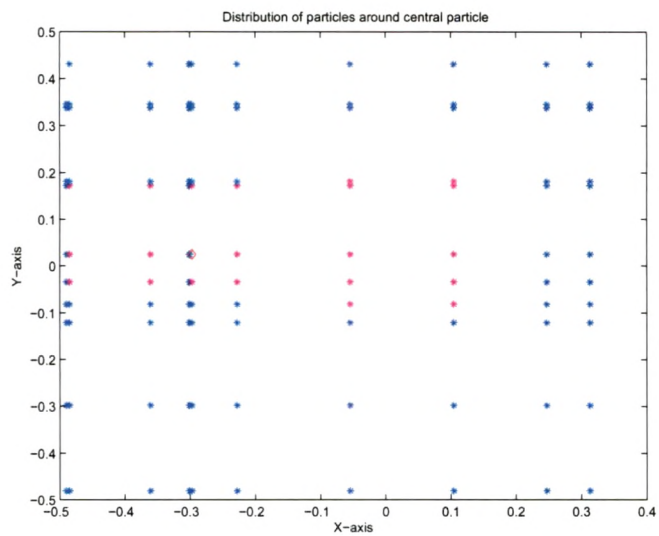

Figure 9.11: **Irregular Distribution: More particles on right of central particle and some particle on left of central particle.**

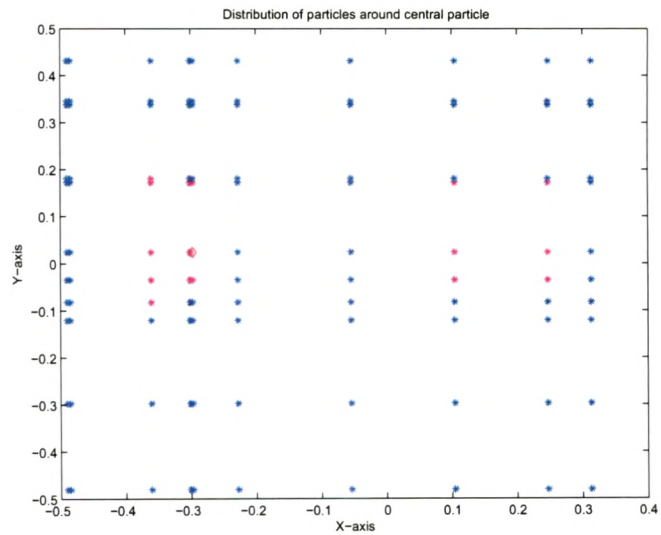

Figure 9.12: **Irregular Distribution: Particles which are far right from central particles and some on left which are adjacent to central particle.**

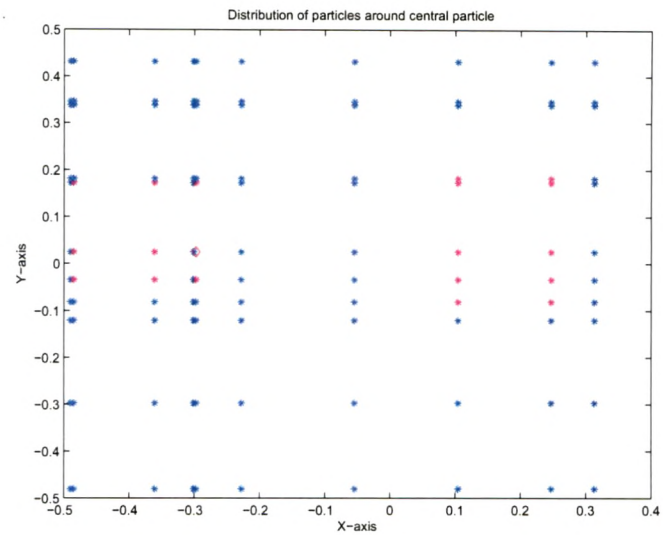

Figure 9.13: **Irregular Distribution: Particles which are far left from central particles and some on right which are adjacent to the central particles.**

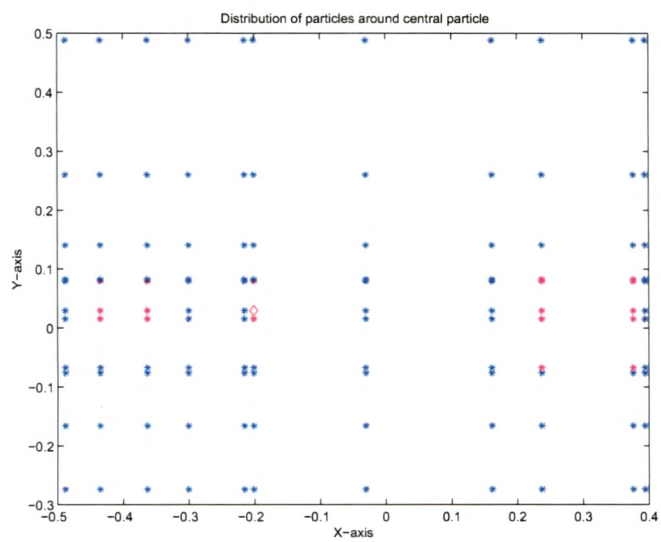

Figure 9.14: **Random Distribution: Particles which are far left and far right from the central particle.**

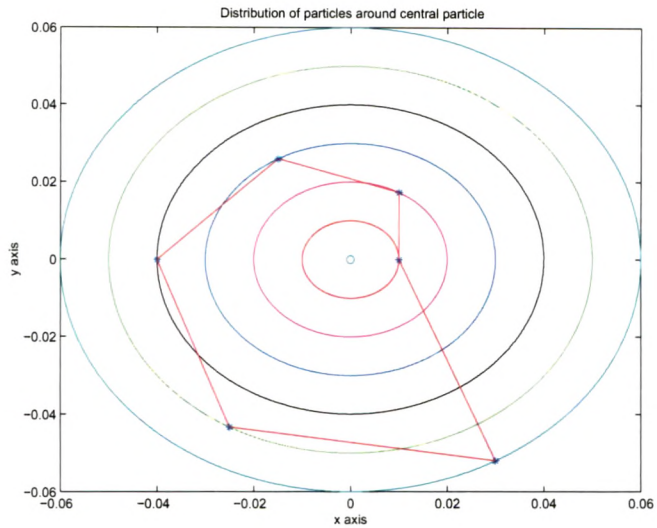

Figure 9.15: **Distribution of particles for Neighboring Algorithm of Type - III (Our Algorithm - 1).**

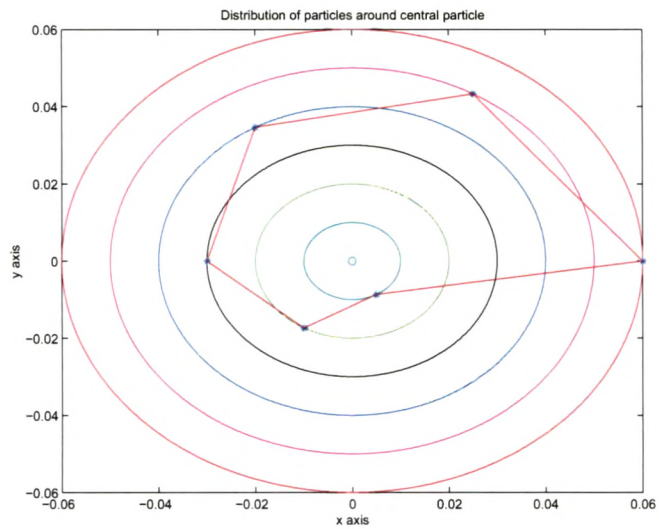

Figure 9.16: **Distribution of particles for Neighboring Algorithm of Type - III (Our Algorithm - 2).**

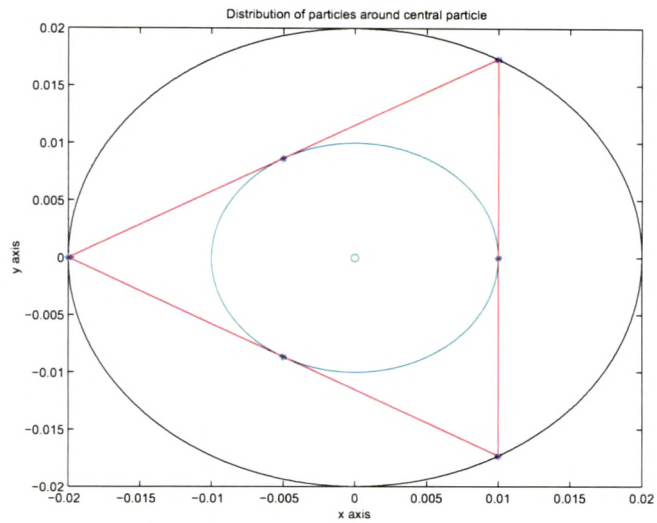

Figure 9.17: **Distribution of particles for Neighboring Algorithm of Type - III (Our Algorithm - 3).**

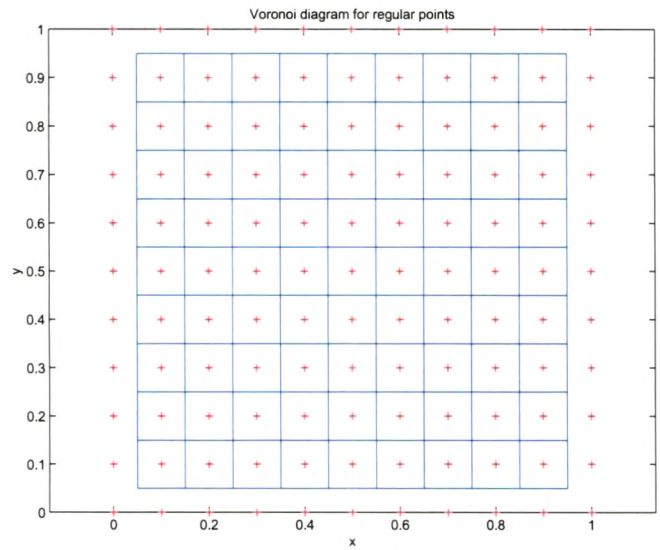

Figure 9.18: **Distribution of particles for Neighboring Algorithm of Type - III (Our Algorithm - 4(a)).**

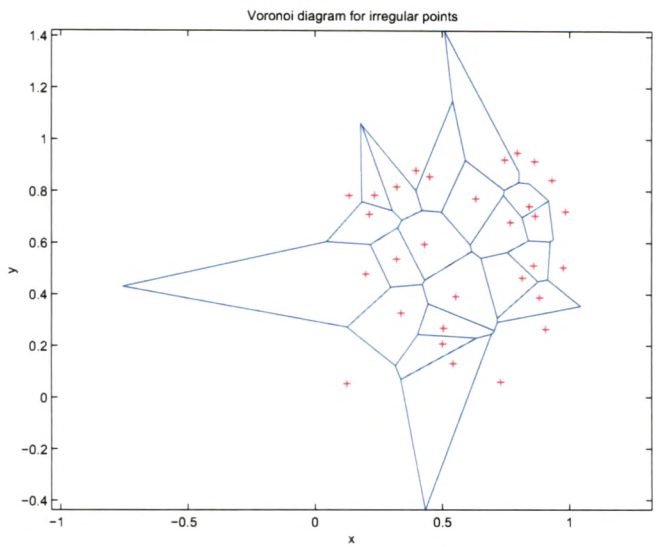

Figure 9.19: **Distribution of particles for Neighboring Algorithm of Type - III (Our Algorithm - 4(b)).**

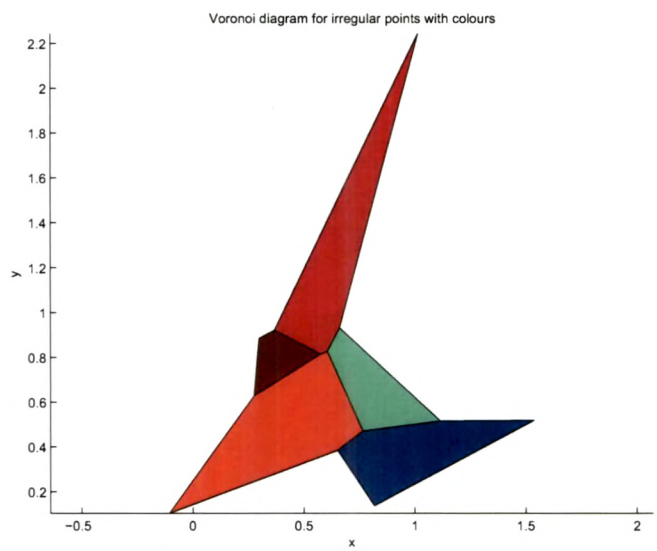

Figure 9.20: **Distribution of particles for Neighboring Algorithm of Type - III (Our Algorithm - 4(c)).**

Distribution of particles	Support of Weight function	Condition number of **A**	Error	Condition number of **IM1**	CPU time
Case 1	0.5	28.4371	0.0037	128.8511	7.7903
Case 2	0.5	3.9716 e3	2.2517 e-13	1.4714 e4	5.5658
Case 3	0.5	2.3963 e3	2.5195 e-4	7.8566 e3	6.2702
Case 4	0.5	2.3936 e3	2.5195 e-4	7.8566 e3	7.2626
Case 5	0.5	1.7945 e3	4.2076 e-4	1.4714 e4	7.6666
Case 6	0.5	1.7945 e3	4.2076 e-4	6.3178 e3	7.8060
Case 7	0.5	982.7925	9.3814 e-14	1.4714 e4	6.8917

Table 9.1: **Dirichlet BVP in** $1D$

Distribution of particles	Support of Weight function	Condition number of **A**	Error	Condition number of **IM1**	CPU time
Case 1	0.5	85.2393	7.3426 e-4	1.4913 e5	4.8322
Case 2	0.5	1.7524 e4	7.9040 e-5	2.9927 e6	5.6740
Case 3	0.5	1.0216 e4	3.8367 e-4	1.2404 e6	6.3419
Case 4	0.5	1.0837 e4	3.4912 e-4	1.2404 e6	6.5488
Case 5	0.5	8.0587 e3	6.2284 e-4	1.3238 e6	6.4106
Case 6	0.5	7.8689 e3	6.4353 e-4	2.9927 e6	6.3906
Case 7	0.5	7.3320 e3	9.1039 e-5	2.9927 e6	5.3046

Table 9.2: **Neumann BVP in** $1D$

Distribution of particles	Support of Weight function	Condition number of **A**	Error	Condition number of **IM1**	CPU time
Case 1	0.5	117.5890	0.0014	1.4913 e5	221.9651
Case 2	0.5	1.4416 e4	9.5683 e-5	2.9927 e6	7.6964
Case 3	0.5	8.3901 e3	0.0023	1.2404 e6	7.7744
Case 4	0.5	8.9417 e3	0.0024	1.2404 e6	5.9780
Case 5	0.5	6.6317 e3	0.0041	2.9927 e6	8.4033
Case 6	0.5	6.7063 e3	0.0041	1.3238 e6	7.2687
Case 7	0.5	5.8001 e3	1.2671 e-4	2.9927 e6	6.7666

Table 9.3: **Mixed BVP in** $1D$

Weight Function	Condition number of **A**	Condition number of **IM1**	CPU time	Error
1	3.48972275	425054.15335403	15.488322100	0.000000000000000036134290005
2	3.58676808	316639.04574120	5.57909700	0.000000000000000012056328158
3	3.56963114	318824.02652784	4.95036400	0.000000000000000015551795962
4	3.58416054	316947.18652286	6.34539300	0.000000000000000025682581062
5	3.58631725	316798.62943908	5.33122900	0.000000000000000031580640880
6	3.58416054	316947.18652286	5.70532700	0.000000000000000025682581062
7	3.57197980	318598.12440483	5.02208700	0.000000000000000007832276494
8	3.58129094	317297.26658455	4.48670100	0.000000000000000052726920052
9	3.55874645	321056.91897760	5.24096100	0.000000000000000029767854848
10	3.56425191	319796.55646024	4.27009700	0.000000000000000074610456702
11	3.56580991	319474.47496300	4.96899400	0.000000000000000021770779624
12	3.56969780	318969.32327599	4.97791500	0.000000000000000025769317236

Table 9.4: **Dirichlet BVP in** $2D$ **with all points as neighbor**

Weight Function	Condition number of **A**	Condition number of **IM1**	CPU time	Error
1	16.44907755	73100149.68885227	2.85441800	0.00000000000000000052041704
2	16.44447491	73805181.49060351	2.61437800	0.00000000000000000052041704
3	16.44617011	73688822.76493615	3.04564300	0.00000000000000000034694470
4	16.44451194	73799099.12972730	3.12241900	0.00000000000000000069388939
5	16.44444640	73809550.56094065	2.94602600	0.00000000000000000043368087
6	16.44451194	73799099.12972730	4.18206000	0.00000000000000000069388939
7	16.44473362	73762909.19764405	3.32911300	0.00000000000000000039031278
8	16.44455501	73792036.50716364	3.86452700	0.00000000000000000073725748
9	16.44791726	73585578.11070329	3.62543900	0.00000000000000000043368087
10	16.44486785	73741563.44173692	4.54642600	0.00000000000000000052041704
11	16.44486245	73742637.63867536	3.22718700	0.00000000000000000088904578
12	16.44473453	73762691.04516909	2.82934000	0.00000000000000000095409791

Table 9.5: **Dirichlet BVP in $2D$ with only surrounding points as neighbor**

Weight Function	Condition number of **A**	Condition number of **IM1**	CPU time	Error
1	14229997787508984.0	3400.28016366	5.072536	0.012309698220416456782 0
2	11595833427435300.0	752.53410099	5.297743	0.138185897387389455825 9
3	18103654320350228.0	950.48046050	5.579455	0.039245540817305892000 7
4	50839067389589288.0	914.23815890	5.203202	0.064184447650023304654 5
5	1991531716173526016.0	1308.01209024	5.707144	0.025009523092875643457 4
6	50839067389589288.0	914.23815890	5.264706	0.064184447650023304654 5
7	26173415440907664.0	894.96949313	5.435310	0.143505983787468094359 2
8	21662542129326736.0	957.92204810	7.786391	0.047977719908149800420 7
9	7787563318858249.0	773.18774495	6.708736	0.053634336827296139049 7

Table 9.6: **Neumann BVP in $2D$ with all points as neighbor**

Weight Function	Condition number of **A**	Condition number of **IM1**	CPU time	Error
1	75.27867417	113731.06493480	3.306129	0.08007565254952014877076 7
2	86.50816935	117861.37174652	3.709107	0.08011185809846926852983 9
3	81.57021890	117025.62751937	4.893839	0.08010601755364704645856 4
4	13777359204508062.0	117611.02805440	3.577892	0.00287045727187443255878 9
5	85.82097444	117762.77461266	3.600470	0.08011408946490153337727 2
6	85.49372046	117611.02805440	3.773402	0.08010970512896520290269 1
7	82.58371069	116391.03100269	3.482546	0.08011730870064256992968 4
8	84.44685277	117336.58926021	3.327916	0.08010776317965562220990 7
9	78.85005936	116752.68320127	3.968903	0.08010317726195895216001 4

Table 9.7: **Neumann BVP in $2D$ with only surrounding points as neighbor**

Weight Function	Condition number of **A**	Condition number of **IM1**	CPU time	Error
1	138.43244883	551.02545203	3.262596	0.00000000000009337031148 37
2	21.95919098	86.42688419	4.107031	0.00000000000000355271367 88
3	334569614411849.0	383.49961920	2.670136	0.79788185526057286978129 92
4	576567944851310.0	329.89796258	2.552536	1.15395512862514548757530 97
5	1877844056983399.75	311.44726095	5.316212	0.46447286560549022604860 17
6	576567944851310.0	329.89796258	2.713129	1.15395512862514548757530 97
7	2791144685591203.5	297.50915993	2.385044	0.51112090693081857484258 99
8	231721886793789.312	338.21671399	2.843871	0.24217558788325402319152 60
9	27.20252167	154.13431804	3.065251	0.00000000000000555111512 31

Table 9.8: **Helmholtz BVP in $2D$ with all points as neighbor**

Weight Function	Condition number of **A**	Condition number of **IM1**	CPU time	Error
1	137.39167030	598.47827253	2.178279	0.006135748874007029840 5
2	35.05215468	339.45836745	2.605127	0.16931048616770827397 45
3	33456914411849.0	383.49961920	1.436225	0.79788185526057286978 12
4	576567944851310.0	329.89796258	2.473082	1.153955128625145487575 3
5	1813721827505988.25	311.44726095	1.469909	0.568862449687998691828 7
6	576567944851310.0	329.89796258	2.824943	1.153955128625145487575 3
7	2791144685591203.5	297.50915993	2.214847	0.511120906930818574842 5
8	231721886793789.3125	338.21671399	2.823380	0.242175587883254023191 5
9	31.07932420	154.13431804	2.865604	0.110352417540876324153 4

Table 9.9: **Helmholtz BVP in $2D$ with only surrounding points as neighbor**

No. of Points	Iterative Methods	Condition number of **A**	CPU time	Flag	Iteration	Error
10	1	16.44907755	5.197217	0	11	0.000000000000000048572257 3
10	2	16.44907755	4.317186	0	9	0.000000000060832987518882 9
10	3	16.44907755	4.076477	0	2	0.000000000015714825274404 4
10	4	16.44907755	6.103348	0	11	0.000000000000000028622937 4
10	5	16.44907755	17.081977	0	14	0.000000000067404272701548 9
20	1	73.08328573	11.355607	0	26	0.000000000136740795769752 0
20	2	73.08328573	9.650954	0	19	0.000000000195862522381351 5
20	3	73.08328573	9.295604	0	4	0.000000000946664508447766 4
20	4	73.08328573	16.725614	0	26	0.000000000161110288304788 8
20	5	73.08328573	20.653403	0	41	0.000000000069332971221886 0

Table 9.10: **Poisson Dirichlet Problem in $2D$ with neighboring points around the central particle according to Algorithm-1**

No. of Points	Iterative Methods	Condition number of **A**	CPU time	Flag	Iteration	Erro
10	1	27306.82695862	3.954169	0	38	0.000000001286860380408927
10	2	27306.82695862	6.100721	1	49	0.000007986084065056756053
10	3	27306.82695862	5.828619	3	16	0.013544197098597294398603
10	4	27306.82695862	5.314301	0	40	0.000000002993150229405050
10	5	27306.82695863	17.312218	0	91	0.000000006429741370404806
20	1	97434441.80001177	36.447269	0	308	0.000000023222938291128647
20	2	97434441.80001177	80.439117	1	2	0.022611757322765359268235
20	3	97434441.80001177	18.853217	3	27	0.020088411285363143299331
20	4	97434441.80001177	17.451144	0	335	0.000000032719853184509650
20	5	97434442.47474562	26.315135	0	1060	0.012413406103623391210644

Table 9.11: **Poisson Dirichlet Problem in $2D$ with neighboring points around the central particle according to Algorithm-2**

No. of Points	Iterative Methods	Condition number of **A**	CPU time	Flag	Iteration	Erro
10	1	27306.82695862	4.112913	0	38	0.000000000679704754286136
10	2	27306.82695862	5.127650	1	45	0.000009553691372348946386
10	3	27306.82695862	7.280873	3	18	0.013544197098991796537892
10	4	27306.82695862	4.632069	0	38	0.000000001553805608484060
10	5	27306.82695870	16.716050	0	88	0.000000003126958091764775
20	1	97434441.95551975	18.802345	0	360	0.000000434240064999674713
20	2	97434441.95551975	43.941537	1	2	0.022611757322765227429251
20	3	97434441.95551975	28.715431	3	34	0.020088411285366217229331
20	4	97434441.95551975	16.851560	0	304	0.000000011331637381900572
20	5	97434442.14432764	27.289348	0	1046	0.012410451965677710833846

Table 9.12: **Poisson Dirichlet Problem in $2D$ with neighboring points around the central particle according to Algorithm-3**