UM–SJTU Joint Institute

19th IPP

Final Report

Acoustic Temperature Field Reconstruction Algorithms

SHEN Yueyang 517370910226

XU Xueming 517370910258

ZHOU Jian 516370910103

Instructor: Sung-Liang Chen

Contents

1 Introduction

Visualizing the distribution of temperature field has various important applications in the field of industry. Specifically, in a industrial boiler, if we can obtain essential information of its temperature distribution, we will be able to control and monitor certain statistics: the highest temperature, the average temperature to get a rough idea on the thermal characteristics of the system thus ensures the safety of equipment. Hence, it is important to find out a proper method to reconstruct temperature field through proper measurement.

In industrial application field, contact temperature measurement method (where we place thermocouple at certain positions) is not practical since the high temperature will quickly damage the apparatus. We introduced an alternative method by utilizing the relationship between acoustic velocity and temperature, $c = Z\sqrt{T}$[1], where c is the velocity of the acoustic velocity, Z is a constant, T is absolute temperature (K). We apply acoustical method to avoid being affected by dust, as dust will affect optical measurement significantly. This non-contact-required method reconstructs temperature field via the information acquired by transmitters on the sides. In this report, we are focusing on using least square method(LSQM) and algebraic reconstruction technique(ART) to reconstruct temperature field. Some other current methods are also introduced theoretically.

In general, there are many advanced numerical techniques in terms of reconstructing the temperature field. Their method varies as they propose certain assumption on the heat source, the objective function they are trying to maximize, and the approach they are using.

2 Methods

2.1 Measurement Model

The schematic of the problem that we are trying to tackle throughout the project is the following:

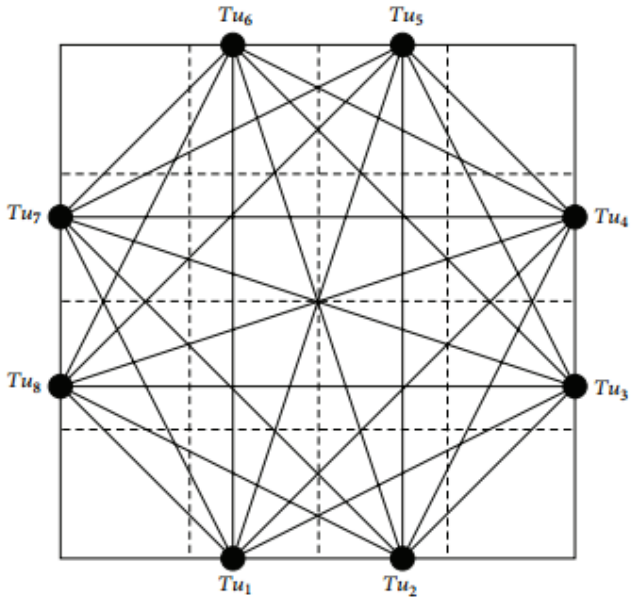


Figure 1: Schematic view of paths array and blocks.[2]

On the edges of rectangle, there are some ultrasonic transmitters and receivers($Tu_1, Tu_2, ...Tu_8$) installed. Every pair of transmitter and receiver measures the time of flight(TOF) for one path. The paths in measure area can be combined to compute the temperature field distribution with proper algorithms. [2] We divide the measure area into $n * n$ small blocks. After obtaining all the TOF of 24 paths, we can calculate the temperature of small blocks.

In total, we have 24 ray paths and $n * n = N$ blocks. We record the value of TOF as $t_1, t_2, t_3, ..., t_{24}$. Due to the relationship between temperature and acoustic velocity, we can assume the velocity are $v_1, v_2, ..., v_N$ and their reciprocals are $a_1, a_2, ..., a_N$. And the distance i_{th} ray path go through every blocks can be determined by calculation, recorded as $s_{i,1}, s_{i,2}, ..., s_{i,N}$. For a single path, we have:

$$s_{i,1} \times a_1 + s_{i,2} \times a_2 + s_{i,3} \times a_3 + ... + s_{i,N} \times a_N = t_i$$

Thus, combine all 24 paths, we have:

$$\begin{cases} s_{1,1} \times a_1 + s_{1,2} \times a_2 + s_{1,3} \times a_3 + ... + s_{1,N} \times a_N = t_1 \\ s_{2,1} \times a_1 + s_{2,2} \times a_2 + s_{2,3} \times a_3 + ... + s_{2,N} \times a_N = t_2 \\ . \\ . \\ . \\ s_{24,1} \times a_1 + s_{24,2} \times a_2 + s_{24,3} \times a_3 + ... + s_{24,N} \times a_N = t_{24} \end{cases} \tag{1}$$

In matrix form, we can write the above equations as

$$\boldsymbol{S} \times \boldsymbol{A} = \boldsymbol{t}$$

where $\boldsymbol{S} = \begin{bmatrix} s_{1,1} & \cdots & s_{1,N} \\ \vdots & \ddots & \vdots \\ s_{24,1} & \cdots & s_{24,N} \end{bmatrix}$, $\boldsymbol{A} = [a_1, a_2, a_3, \cdots, a_N]^T$ and $\boldsymbol{t} = [t_1, t_2, t_3, \cdots, t_{24}]^T$. After solving the above matrix equations, we can determine the temperature field distribution by

$$\boldsymbol{T} = \frac{1}{\boldsymbol{A}^2 Z^2}$$

, where Z is a constant and is proper to be 19.8 in this problem.

2.2 Least Square Method

We can use least square method when N is smaller than the number of ray paths 24. It can simply solve the overdetermined equation problems. Assume we divide the measure area into $4 \times 4 = 16$ blocks. The time of flight of acoustic wave propagates along an arbitrary ray path is:

$$TOF_k = \sum_{i=1}^{16} s_{k,i} a_i$$

With the measured time of flight t_k, the error can be calculated as:

$$\varepsilon_k = t_k - TOF_k = t_k - \sum_{i=1}^{16} s_{k,i} a_i$$

Thus, the error of total 24 paths is:

$$\sum_{k=1}^{24} (t_k - \sum_{i=1}^{16} s_{k,i} a_i)$$

Apply the least square method, we have:

$$\frac{\partial}{\partial \boldsymbol{A}} [(\boldsymbol{SA} - \boldsymbol{t})^T (\boldsymbol{SA} - \boldsymbol{t})] = 0$$

The $\boldsymbol{A}$ matrix can be solved as:

$$\boldsymbol{A} = (\boldsymbol{S}^T \cdot \boldsymbol{S})^{-1} \cdot \boldsymbol{S}^T \cdot \boldsymbol{t}$$

Hence, the average temperature of every small blocks can be calculated as:

$$\boldsymbol{T} = \frac{1}{\boldsymbol{A}^2 Z^2}$$

2.3 Algebraic Reconstruction Technique

Least square method has a very important deficiency, which is that it can only solve overdetermined equations but not underdetermined equations. To obtain more precise temperature field distribution, we need to explore a method to solve underdetermined equations, where N is larger than the number of ray paths 24. Here we apply algebraic reconstruction technique[4].

For the measure area divided into N blocks, we assume a group of initial value of temperature filed as $a_1^{(0)}, a_2^{(0)}, \cdots, a_N^{(0)}$, in N dimension space we can represent it by a initial vector: $\boldsymbol{a}^{(0)} = [a_1^{(0)}, a_2^{(0)}, \cdots, a_N^{(0)}]^T$. Then we project $\boldsymbol{a}^{(0)}$ onto the first ray path and get $\boldsymbol{a}^{(1)}$:

$$\boldsymbol{a}^{(1)} = \boldsymbol{a}^{(0)} + \frac{\boldsymbol{t_1} - \boldsymbol{a}^{(0)}\boldsymbol{S}_1}{\boldsymbol{S}_1 \cdot \boldsymbol{S}_1^T}\boldsymbol{S}_1^T$$

where t_1 represents the first measured value of TOF, and S_1 represent the first row of coefficient matrix S, which is the length matrix that the first ray path go through blocks. Similarly, we project $\boldsymbol{a}^{(0)}$ onto the second ray path and continue projection. When we get $a^{(24)}$, all the temperature of blocks get corrected and the first iteration is complete. For the second iteration, we set $a^{(24)}$ as initial value and repeat. Combine with $\boldsymbol{SA} = \boldsymbol{t}$, we get:

$$\boldsymbol{a}^{(n)} = \boldsymbol{a}^{(n-1)} - \frac{\boldsymbol{t_i} - \boldsymbol{a}^{(n-1)}\boldsymbol{S}_i}{\boldsymbol{S}_i \cdot \boldsymbol{S}_i^T}\boldsymbol{S}_i^T$$

Based on the equation above, there is an improved ART algorithm, damping ART algorithm[5]. Before correction, we add a correction efficient:

$$\boldsymbol{a}^{(n)} = \boldsymbol{a}^{(n-1)} - \lambda\frac{\boldsymbol{t_i} - \boldsymbol{a}^{(n-1)}\boldsymbol{S}_i}{\boldsymbol{S}_i \cdot \boldsymbol{S}_i^T}\boldsymbol{S}_i^T$$

where λ is the damping factor, in common condition $\lambda \in (0, 2)$. A proper λ value can accelerate ART algorithm by reducing iterations. In Power Plant reconstruction, we find $\lambda = 1.2$ is proper.

When solving the underdetermined equations, we may have multiple converge results. To solve this problem, we need to add constraint range for the temperature of every block. We can set by LSQM results for 4×4 blocks. We set a max value for a block as a_{max}, min value as a_{min}. During iterations, when $a > a_{max}$, let $a = a_{max}$. When $a < a_{min}$, let $a = a_{min}$.

To determine when to stop converge, we choose a condition:

$$\boldsymbol{W}^{(k)} = (\boldsymbol{S}_i\boldsymbol{a}^{(k)} - \boldsymbol{t}_i)^2$$

when $\sqrt{\frac{\boldsymbol{W}^{(k)}}{n^2}} < \varepsilon$($\varepsilon$ is the allowable error), we terminate the iterations. And we calculate the temperature matrix as:

$$\boldsymbol{T} = \frac{1}{\boldsymbol{A}^2 Z^2}$$

2.4 TSVD

The idea behind Truncated Singular value decomposition (TSVD) is basically to truncate the small singular values on the diagonals of the matrix to reduce computation and make the reconstruction more stable. The formalism for Singular Value Decomposition (SVD) is the following:

$$A = U\Sigma V^T$$

where A is m×n, U is r×r and V^T is r×m. It is also important to note the fact that U, V are column orthonormal, and the singular values are positive. That is,

$$UU^T = I, \quad V^T V = I, \quad \sigma_1 \geq \sigma_2 \geq ... \geq 0$$

For a general m×n matrix A, we embed the information contained by m (number of ray paths), and n (the number of voxels or segmented square) into a latent space T, where the dimension of the latent space T is r. Intuitively, V^T shrinks down/expands the information in a m dimensional space to r dimensional latent space. Then the Σ matrix scales such element in the latent space and then the U matrix expand/shrinks down to the n dimensional space. It is also important to note if A is square matrix, that is the number of ray paths is equal to the number of squares, SVD would degenerate into eigenvalue decomposition. In block matrix formalism, SVD can be written as

$$A = U\Sigma V^T = \underbrace{\begin{matrix}[u_1 & u_2 & \dots & u_r]\end{matrix}}_{\begin{matrix}r & n\times 1 & vectors\end{matrix}} \left.\begin{bmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 \\ \dots & & & \\ 0 & 0 & \dots & \sigma_r \end{bmatrix} \begin{bmatrix} v_1^T \\ v_2^T \\ \dots \\ v_r^T \end{bmatrix}\right\} \begin{matrix} r \\ \\ \\ \\ \end{matrix} \quad \begin{matrix} 1\times \text{m} \\ \\ \\ \\ \end{matrix} \quad \begin{matrix} \text{vectors} \\ \\ \\ \\ \end{matrix}$$

More succinctly,

$$A = \sum_{i=1}^{r} u_i \sigma_i v_i^T \approx \sum_{i=1}^{k} u_i \sigma_i v_i^T$$

where k≤r, and is referred to as the truncated value.

2.5 Solving linear system : Ax=b for large A

Apart from the above methods which we have already implemented the related code in Matlab, we have also studied the theory of solving the linear system Ax=b theoretically, where computing inverses on A is practically impossible. In theory, if we have a lot of transmitters on the boundaries, we will obtain a fairly large matrix, whereas in our simulation and discussion we are mainly restricting ourselves to the scenario where there is only

two transmitters on each side of the rectangle. Computing a solution from a huge matrix via Gaussian elimination (or LSQR) or via Truncated singular value decomposition is in general computationally expensive. The bottleneck in computation can be solved by methods in numerical linear algebra. Specifically, we could **Krylov Solvers** to iteratively approximate such solutions. To be more specific, we explore the space[6]

$$K_k(A, c) = \{c, Ac, ..., A^{k-1}c\}$$

where a common practice for c is c=b. The core idea behind it is to utilize power iterations on matrices and as the time complexity for matrix multiplication far lesser than taking the inverse, we have bypass the bottleneck that is limiting our computing speed. Different solvers include GMRES, LGMRES, Cg and BicGSTAB. A more advanced model to utilize is the **Algebraic Multigrid Method**[7], where the convergence of ART model is improved, but due to the restriction on time, we are unable to fully implement such method.

2.6 Summary of Current Methods

In general, current methods on temperature field reconstruction can be divided into three main categories: non-iterative algorithm, iterative algorithm, and intelligent algorithms[8]. Non-iterative algorithms are in general easier to implement in application and some of them are faster to compute, but in general have lower accuracy. This include Least-Square Method, Singular Decomposition and Regularization family, where we trade a certain bias for more stability in reconstruction, hence less variance(Tikohonov Regularization method, Fourier Regularization Method, Multi-objective Function Regularization Method, Truncated SVD). Iterative Algorithms in general have higher accuracy but requires longer to compute. Current articles surveyed methods including Algebraic Reconstruction technique(ART), Simultaneous Reconstruction technique(SIRT), Krylov solvers. In practice ART converges notably slow when the computation is heavy. In terms of intelligent algorithm, articles have applied genetic algorithm, RBF neural network, and some articles manifest novel methods in deep learning like Generative Adversial Network (GAN) promising.

3 Results

3.1 Temperature Field Model

To test the algorithm, we apply four ideal temperature field models:

- Unimodal symmetric: $T(x, y) = 800 + 1000sin(\frac{x\pi}{15}) \cdot sin(\frac{y\pi}{15})$
- Unimodal deflection: $T(x, y) = 800 + 1000e^{(-((x-10)^2+(y-7.5)^2)/100)}$
- Bimodal symmetric: $T(x, y) = 800 + 35(2 - cos(\frac{1.92\pi x}{15}))(8 - cos(\frac{4\pi y}{15}))$

- Bimodal deflection: $T(x,y) = 800 + 35(2 - cos(\frac{1.9\pi x}{14}))(8 - cos(\frac{2.8\pi y}{12}))$

The theoretical temperature figures are displayed as follows:

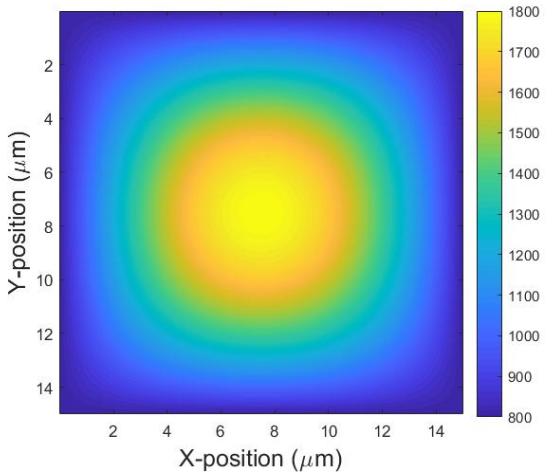


Figure 2: Theoretical Temperature Field for $T(x,y) = 800 + 1000sin(\frac{x\pi}{15}) \cdot sin(\frac{y\pi}{15})$

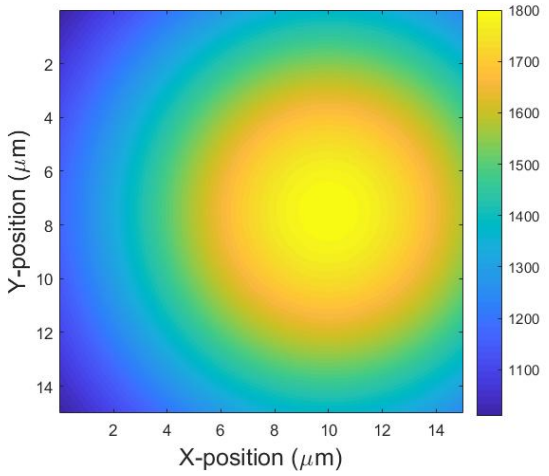


Figure 3: Theoretical Temperature Field for $T(x,y) = 800 + 1000e^{(-((x-10)^2+(y-7.5)^2)/100)}$

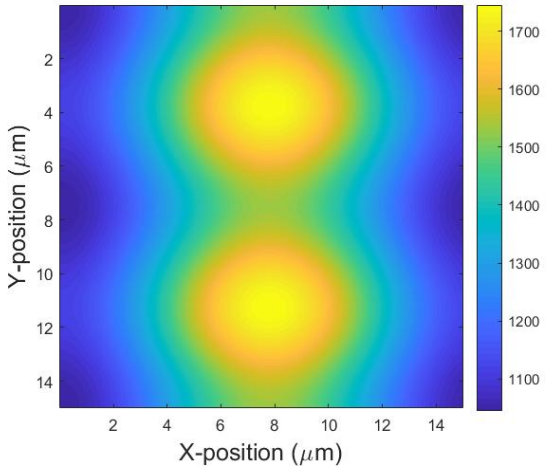


Figure 4: Theoretical Temperature Field for $T(x, y) = 800 + 35(2 - cos(\frac{1.92\pi x}{15}))(8 - cos(\frac{4\pi y}{15}))$

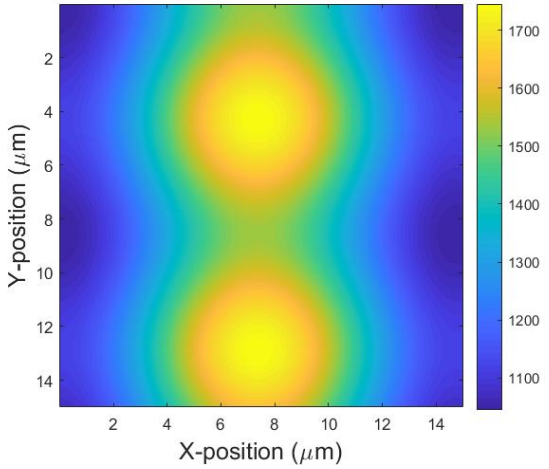


Figure 5: Theoretical Temperature Field for $T(x, y) = 800 + 1000sin(\frac{x\pi}{15}) \cdot sin(\frac{y\pi}{15})$

3.2 Error Definition

To test the effectiveness of reconstruction, we set T_m as the average temperature of the model field and give some definition of errors:

- Relative maximum error:
$$max\{\frac{|T(i,j) - T'(i,j)|}{T_m} \times 100\%\}$$
- Relative mean error:
$$\frac{\frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N} T(i,j) - T'(i,j)}{T_m} \times 100\%$$

- Root Mean Square Error:

$$\frac{\sqrt{\frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}T(i,j)-T'(i,j)^2}}{T_m}\times 100\%$$

- MSE: Mean Square Error.

- SSIM: The Structural Similarity Index. The larger value of SSIM, the better reconstruction effect it is.

where MN represent the number of blocks.

3.3 LSQM Results

We apply least square method with 4×4 blocks structure. And the results are displayed as follows:

3.3.1 Unimodal Symmetric

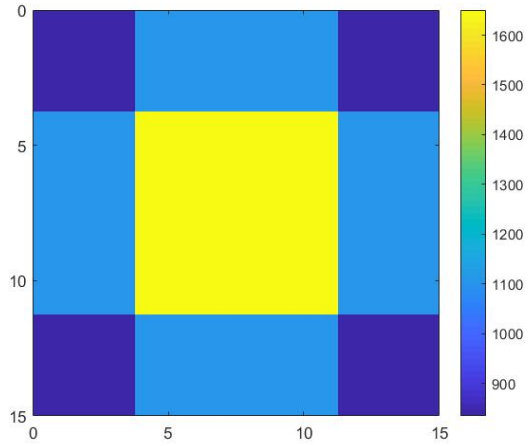


Figure 6: LSQM results

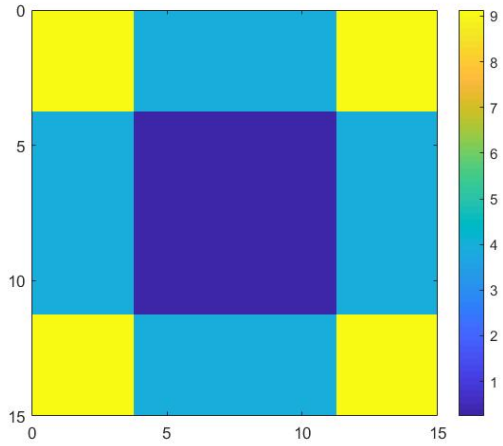


Figure 7: Difference between results and model

- Relative maximum error: 9.1349

- Relative mean error: 4.3139

- Root Mean Square Error: 9.7058

3.3.2 Unimodal Deflection

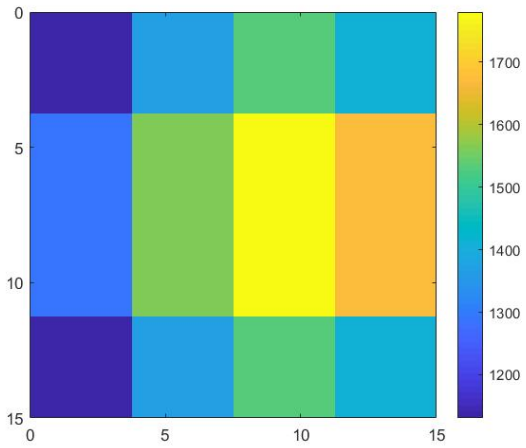


Figure 8: LSQM results

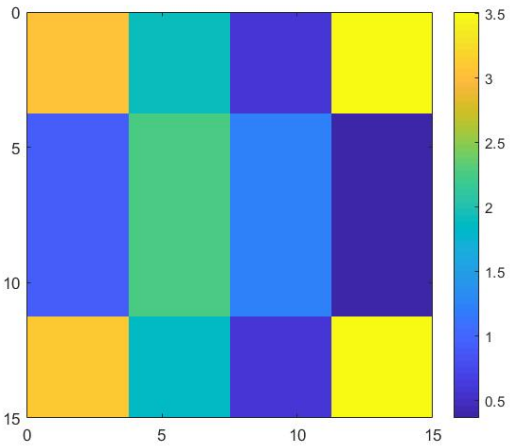


Figure 9: Difference between results and model

- Relative maximum error: 3.5110
- Relative mean error: 1.7227
- Root Mean Square Error: 3.5167

3.3.3 Bimodal Symmetric

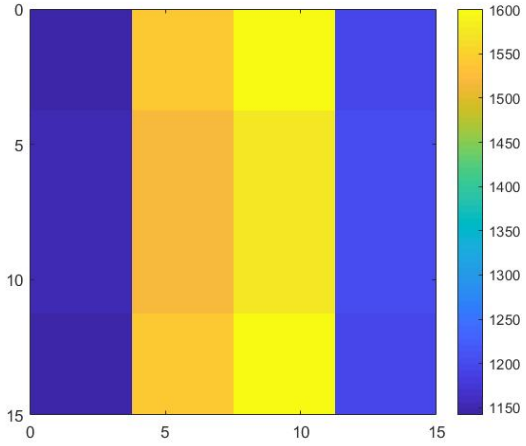


Figure 10: LSQM results

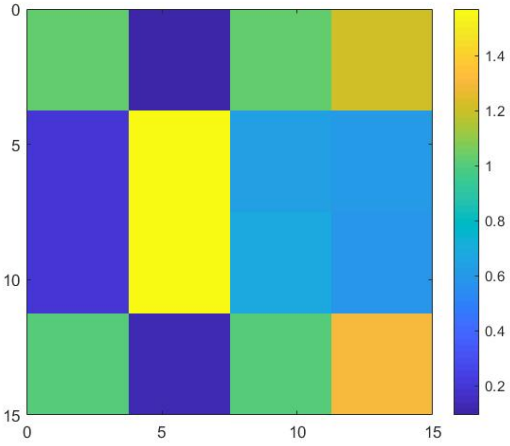


Figure 11: Difference between results and model

- Relative maximum error: 1.5693

- Relative mean error: 0.8022
- Root Mean Square Error: 1.6030

3.3.4 Bimodal Deflection

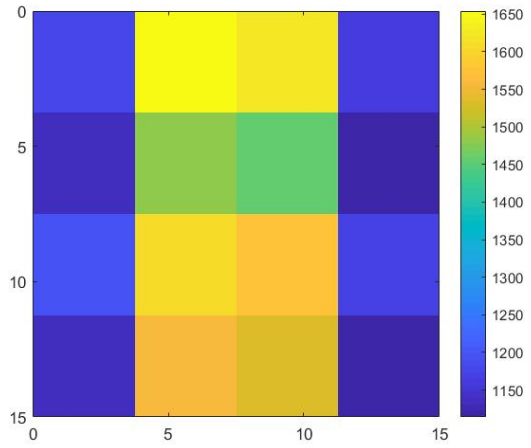


Figure 12: LSQM results

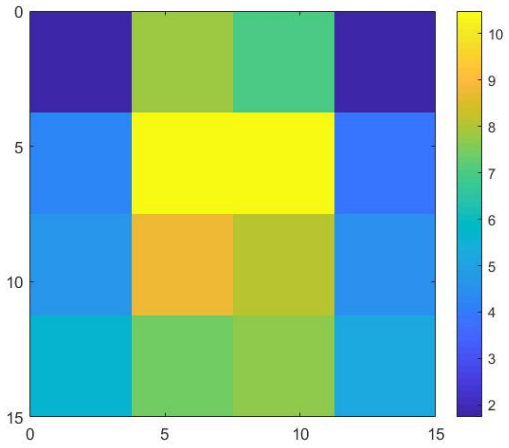


Figure 13: Difference between results and model

- Relative maximum error: 10.4856
- Relative mean error: 6.2244
- Root Mean Square Error: 12.6880

3.4 ART Results

We apply algebraic reconstruction techniques with 10×10 blocks structure. And the results are displayed as follows:

3.4.1 Unimodal Symmetric

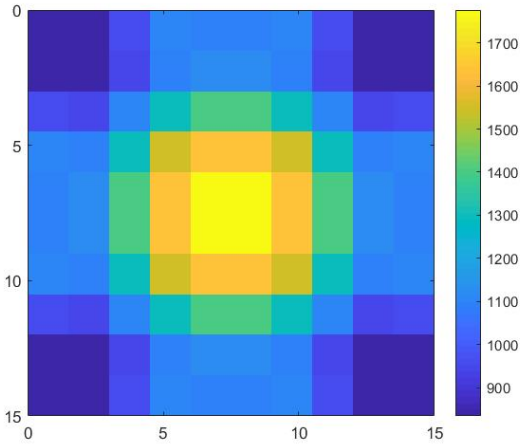


Figure 14: ART results

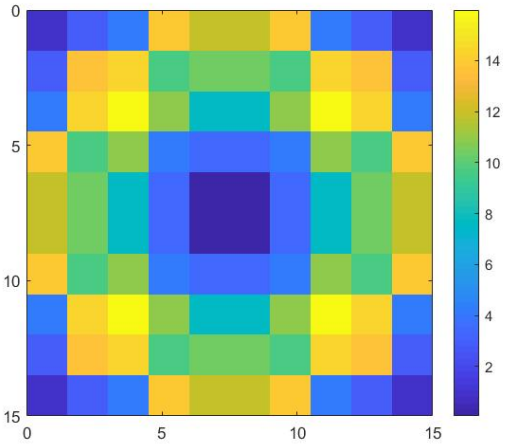


Figure 15: Difference between results and model

- Relative maximum error: 15.9852
- Relative mean error: 8.5320
- Root Mean Square Error: 27.4829

3.4.2 Unimodal Deflection

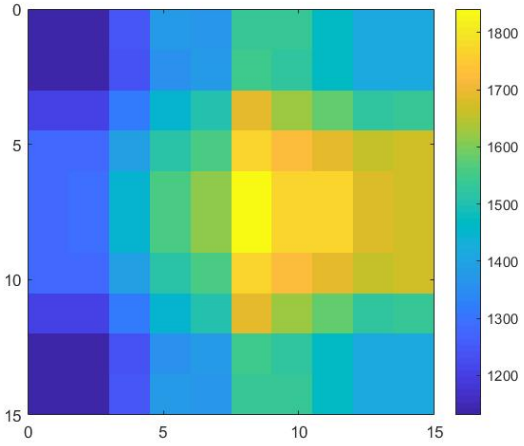


Figure 16: ART results

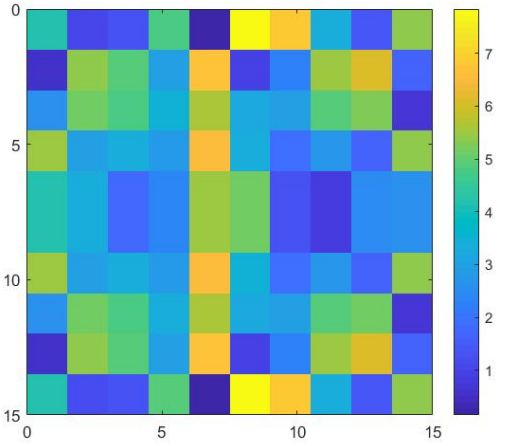


Figure 17: Difference between results and model

- Relative maximum error: 7.8394

- Relative mean error: 3.5573
- Root Mean Square Error: 11.1852

3.4.3 Bimodal Symmetric

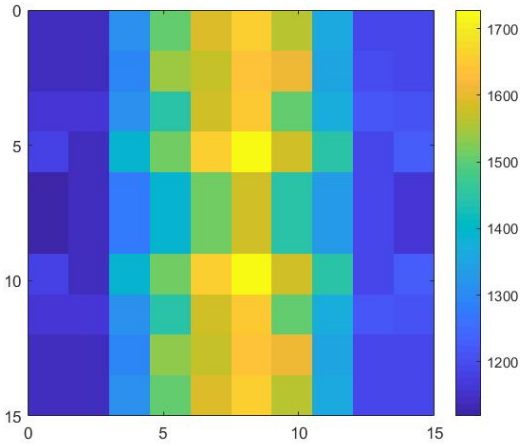


Figure 18: ART results

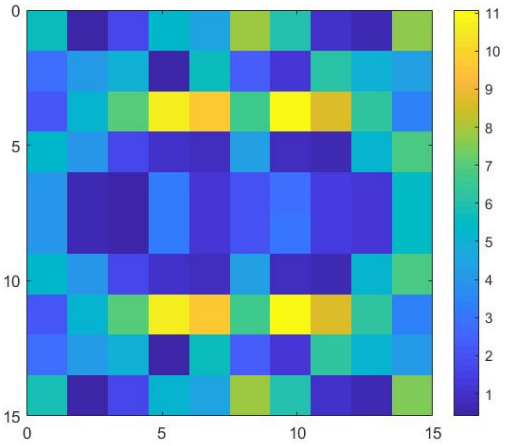


Figure 19: Difference between results and model

- Relative maximum error: 11.0767
- Relative mean error: 4.0277
- Root Mean Square Error: 12.5153

3.4.4 Bimodal Deflection

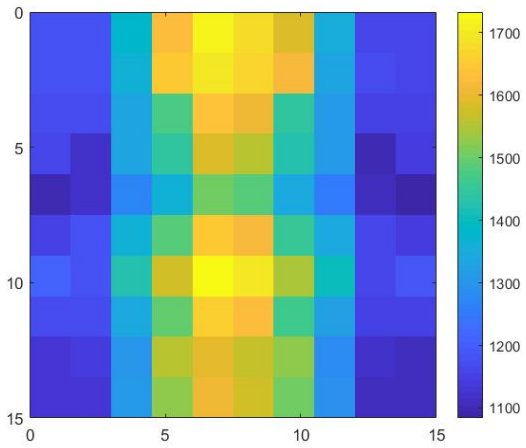


Figure 20: ART results

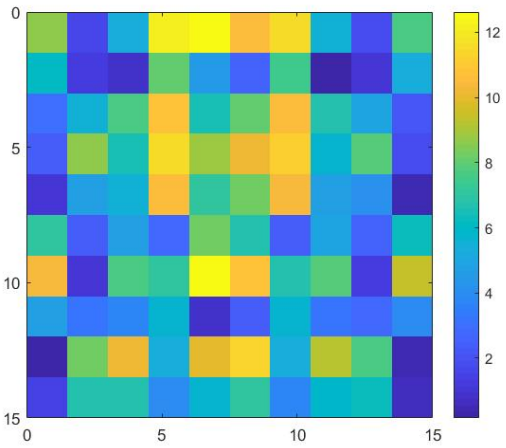


Figure 21: Difference between results and model

- Relative maximum error: 12.6258
- Relative mean error: 5.8828
- Root Mean Square Error: 18.7500

3.5 Test For Real Data

We obtain the path temperature and the zone temperature data from a company. And the real structure is showed below.

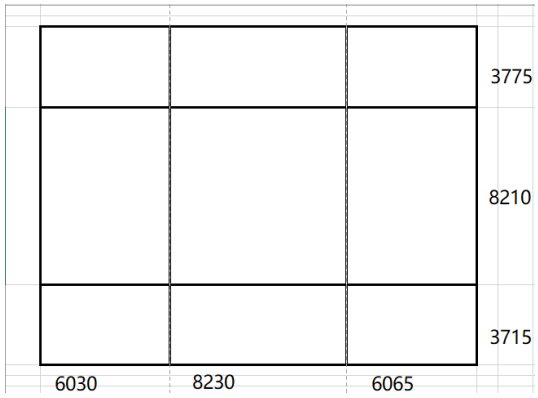
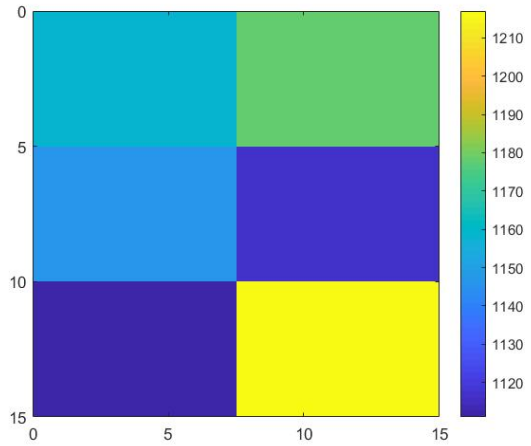


Figure 22: The real structure for the measure area

And the figures about measured temperature field are:



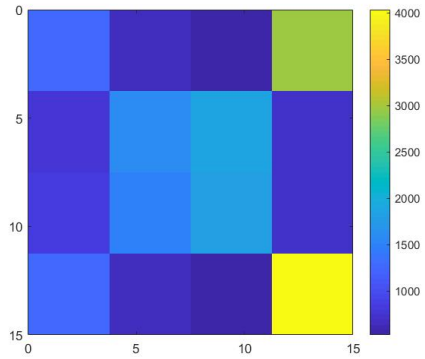
1158	1178
1147	1116
1111	1217

Table 1: Measured Data

Figure 23: Measured Temperature Field

3.5.1 LSQM

Due to the limitation of LSQM, we just calculate the value of 4×4 blocks structure.



1285.23	684.34	525.62	2950.85
783.41	1578.99	1878.14	735.34
834.59	1486.51	1826.07	734.73
1276.42	688.42	550.72	4036.71

Table 2: LSQM Data

Figure 24: LSQM Figure

3.5.2 ART

To get a similar figure as measured data, we first run ART in 6×6 blocks and then convert every 2×3 blocks into one blocks by taking average value.

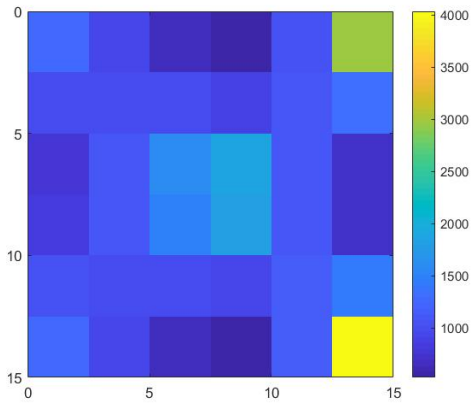


Figure 25: ART Figure

1285.23	914.94	684.34	525.62	1039.69	2950.85
988.22	991.79	995.38	899.31	1075.36	1308.67
783.41	1078.74	1578.99	1878.14	1112.90	735.34
834.59	1090.96	1486.51	1826.07	1100.32	734.73
1028.44	1005.02	982.39	924.60	1145.88	1457.26
1276.42	915.41	688.42	550.72	1174.77	4036.71

Table 3: ART Data

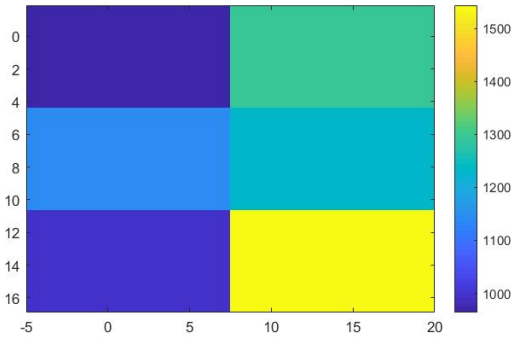


Figure 26: ART Converted Figure

963.84	1293.97
1140.16	1233.35
997.62	1543.51

Table 4: ART Converted Data

4 Structure Improvement

Consider the limitation, there is an assumption that may be able to improve the accuracy of reconstruction effect. Adding transmitters and receivers at corners can increase the number of effective paths passing through the edge sub region to optimize the reconstruction result.

Hence, in this section, we will add transmitters and receivers at four corners and compare the results of original structure and new one.

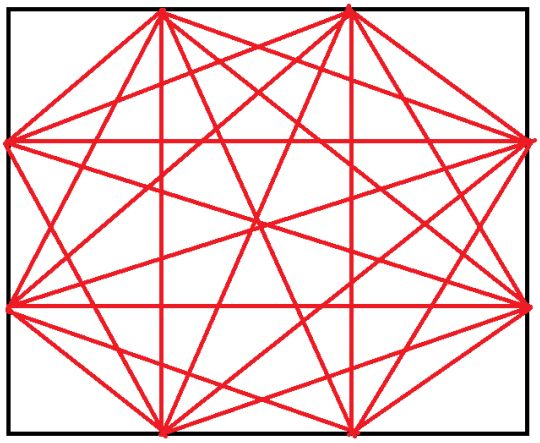


Figure 27: Original Structure

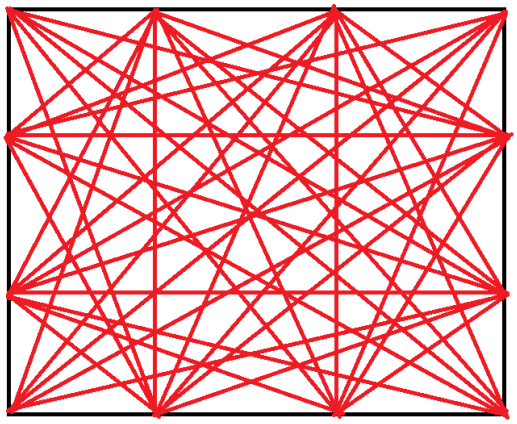


Figure 28: New Structure

From the comparison, we can see that the path density increases. To test whether the new structure can improve the accuracy of reconstruction effect, we will apply TSVD method.

4.1 Model 1: One Peaks

Model applied: $T(x, y) = 800 * exp((-70 * (x - 10).^2 - 30 * (y - 10).^2)/1000) + 800$

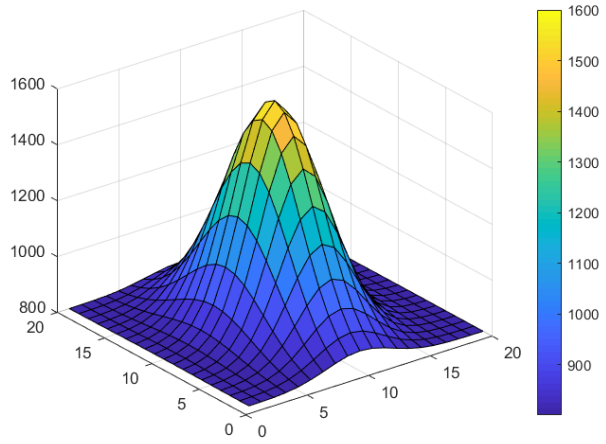


Figure 29: 3-D Plot

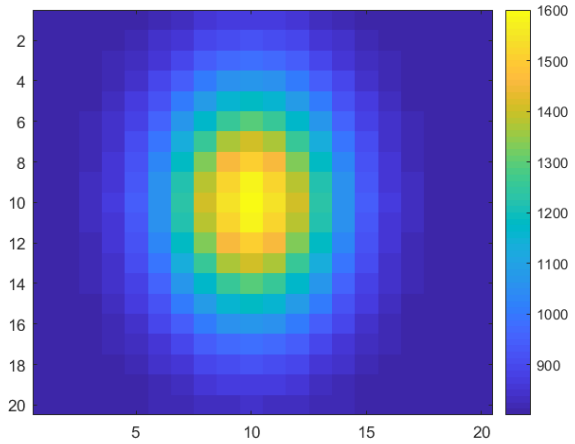


Figure 30: 2-D Plot

Error	Origin	At Corners
Relative maximum error	0.1009	0.1202
Relative mean error	0.0447	0.0346
Root Mean Square Error	0.0502	0.0494
MSE	0.0388	0.0364
SSIM	0.9602	0.9653

Table 5: Error Comparison

4.2 Model 2: Two Peaks

Model applied: $T(x, y) = 800 * exp((-70 * (x - 10).^2 - 30 * (y - 10).^2)/1000) + 800 * exp((-70 * (x - 16.6).^2 - 30 * (y - 16.6).^2)/300) + 800$

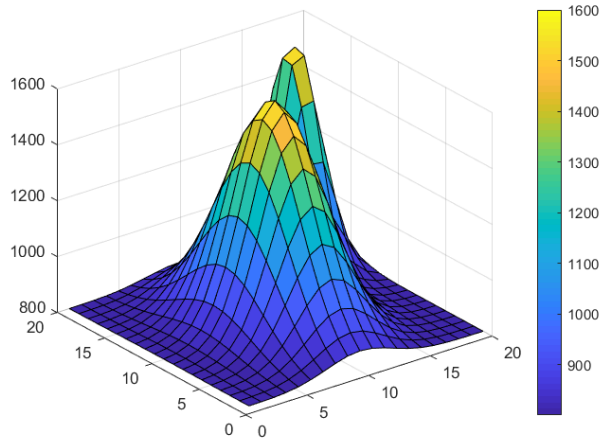


Figure 31: 3-D Plot

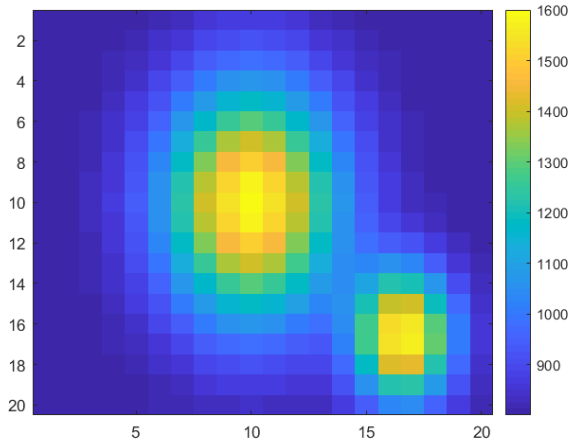


Figure 32: 2-D Plot

Error	Origin	At Corners
Relative maximum error	0.2986	0.1147
Relative mean error	0.066	0.0365
Root Mean Square Error	0.0939	0.0467
MSE	0.082	0.0348
SSIM	0.8444	0.9604

Table 6: Error Comparison

4.3 Model 3: Four Peaks

Model applied: $T(x,y) = 800 * exp((-70 * (x-10).^2 - 30 * (y-10).^2)/1000) + 800 * exp((-70 * (x-13.6).^2 - 30 * (y-13.6).^2)/300) + 800 * exp((-70 * (x-6.3).^2 - 30 * (y-13.6).^2)/300) + 800 * exp((-70 * (x-13.6).^2 - 30 * (y-6.3).^2)/300) + 800 * exp((-70 * (x-6.3).^2 - 30 * (y-6.3).^2)/300) + 800$

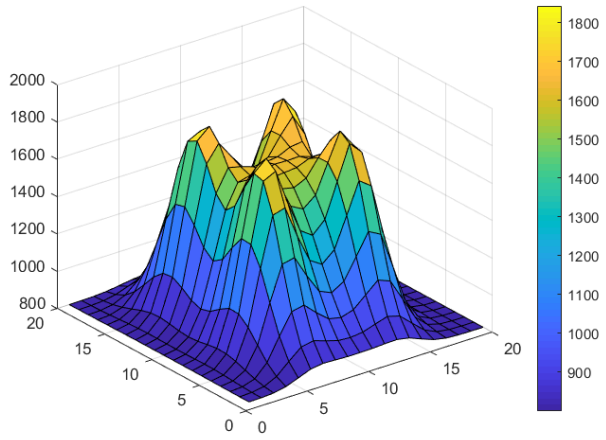


Figure 33: 3-D Plot

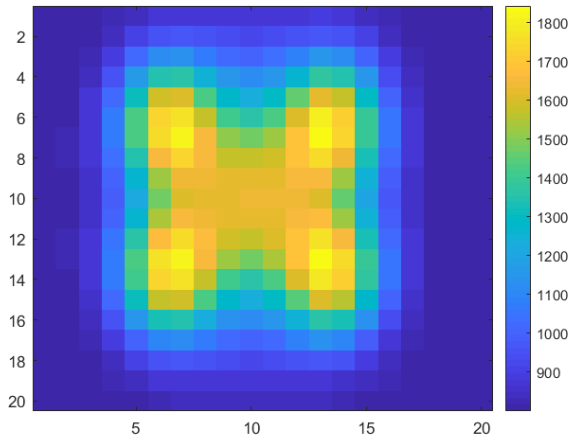


Figure 34: 2-D Plot

Error	Origin	At Corners
Relative maximum error	0.2635	0.2799
Relative mean error	0.1042	0.0833
Root Mean Square Error	0.1228	0.1142
MSE	0.0799	0.0687
SSIM	0.9232	0.9425

Table 7: Error Comparison

4.4 Model 4: Two Peaks

Model applied: $T(x, y) = 800 * exp((-70 * (x - 6).^2 - 30 * (y - 6).^2)/1000) + 800 * exp((-70 * (x - 13.6).^2 - 30 * (y - 13.6).^2)/300) + 800$

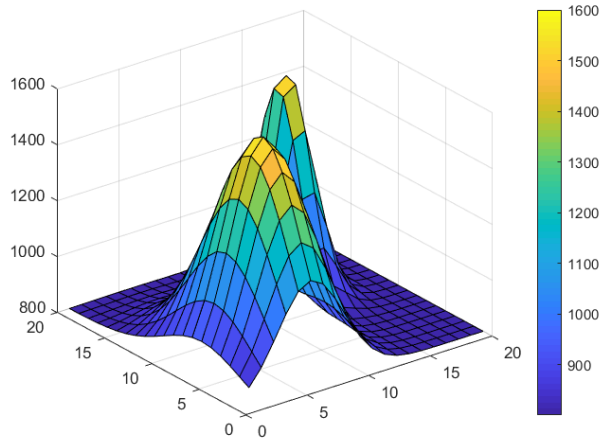


Figure 35: 3-D Plot

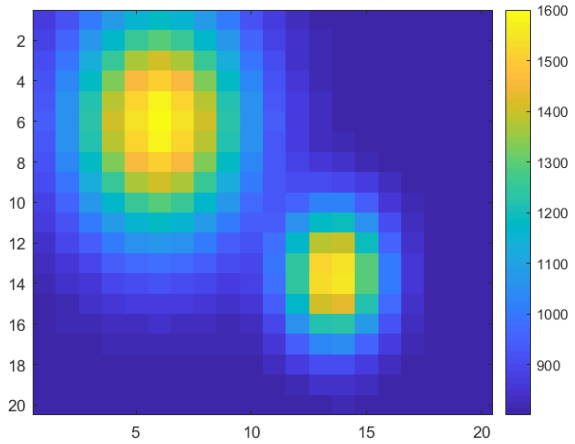


Figure 36: 2-D Plot

Error	Origin	At Corners
Relative maximum error	0.4997	0.5481
Relative mean error	0.1189	0.0731
Root Mean Square Error	0.1827	0.1573
MSE	0.1133	0.0967
SSIM	0.5700	0.7141

Table 8: Error Comparison

5 Conclusion

Least square method has simple principle and fast reconstruction speed. But the limitation is that it can only applied when the number of blocks is smaller than the number of ray paths. Its precision is not high enough.

Algebraic reconstruction technique can solve the underdetermined equations, thus its precision is high. We can set higher dimension value to get more detailed temperature field distribution. But it need many iterations, which makes the algorithm slow.

Compare LSQM and ART performance, we can see that LSQM has faster running speed and lower error value but it can only calculate the limited number of blocks. As for ART, it can reconstruct a more precise temperature field distribution appearance.

In conclusion, we can choose LSQM when we need fast reaction with real time observation and choose ART when we want to obtain a more precise temperature field distribution figure.

5.1 Structure Improvement

From the result of section 4, we can find that for a field that can be well reconstructed, adding T/R sensors at corners can improve the reconstruction accuracy (Model 1,2,3). One problem is that sometimes the relative maximum error of new structure is larger than original one. But other criterion indicates that the new structure can reconstruct the temperature field better. And we checked the whole temperature field, we find that the max reconstructed error happens in the edge area, where the temperature is lower than average.

For model 4, since TSVD cannot well reconstruct the temperature field, the comparison is not meaningful.

So, we can draw a conclusion that the new structure can help us better reconstruct the temperature field.

6 Discussion

In the process that test data, we have the following problems and corresponding potential reasons:

1. We find the relative magnitude of calculated value and measure data has some difference. The difference may come from the process of transform the path temperature into time of flight. I apply the relationship between acoustic velocity and temperature $c = Z\sqrt{T}$, and use the formula $t = \frac{D}{\sqrt{ZT}}$ to transform the given path temperature into TOF, where D is the ray path length. The constant Z is determined by the gas in the measure area. So the value of Z may be not the most proper, thus the relative magnitude differs.

2. When running ART, we find the allowable error ε cannot be as small as model testing. And the result is that we can only run few iterations, or we cannot quit the iterations when we reduce the allowable error ε. Lack of iterations causes the inaccuracy of results. The deeper reasons are discussed in (3) and (4).

3. For $A^{(0)}$, we use LSQM to initialize and then take average value, which may be a problem. Simply adding A matrix is not the most proper way. So we cannot converge to a correct temperature matrix. We need to find a better way to initialize $A^{(0)}$.

4. We cannot determine the range of a_{max} and a_{min} well, which relates to the converge value. Our current method is: $A_{range} = [A_{initial} - range_{const} \quad A_{initial} + range_{const}]$;
   Give A matrix a constant range. With smaller constant range, the error to quit the iteration is smaller. But the error is still large. We need to find a better way to determine the A range.

7 Packaging in Matlab

Currently, we have packaged all of our code and created a interface to better demonstrate our result in Matlab. Currently, out interface is able to share within a locally trusted network via webapp urls. For users outside of a

shared network, we packaged the code so that it is still able to run without having to download Matlab. A sample of our current interface is attached at the end.

8 Future Work

1. We can use squeeze method to find the most proper value of constant Z for specific model. First we should ensure the relative magnitude of temperature among every blocks accord with the measure data. Then we can change the value of Z to make the relative magnitude of temperature field approach the measure data. Finally, when we get the smallest root mean square error, the value of Z is the most suitable for the model.

2. Find a better way to initialize $A^{(0)}$ and determine the converge range.

3. The value of damping coefficient λ is a factor to determine the run time. After solving the iterations too small or endless problem, we can test different values of λ and its corresponding run time to obtain the most effective λ value.

4. We can develop new algorithms based on LSQM and ART to get better performance. Such as SART(Simultaneous Algebraic Reconstruction Technique) can project all the ray paths at the same time. Or we can combine ART with Gaussian Process Regression.

9 Reference

1. R. Jia, Q. Xiong, K. Wang, L. Wang, G. Xu, and S. Liang, The study of three-dimensional temperature field distribution reconstruction using ultrasonic thermometry, AIP Advances, vol. 6, no. 7, Article ID 075007, 2016.

2. Xuehua Shen, Qingyu Xiong, Weiren Shi, Shan Liang, Xin Shi, and Kai Wang, A New Algorithm for Reconstructing Two-Dimensional Temperature Distribution by Ultrasonic Thermometry, Mathematical Problems in Engineering, vol. 2015, Article ID 916741, 10 pages, 2015. https://doi.org/10.1155/2015/916741.

3. Chuyoung Kim, "Algorithms for Tomographic Reconstruction of Rectangular Temperature Distributions using Orthogonal Acoustic Rays", 2016.

4. Zhipeng Chen, "Research on Reconstruction Algorithm of Temperature Field by Acoustic Pyrometry of Coal Boiler in Power Plant", 2014.

5. Zhihai Zhang, Jianhui Yang, "An Iteration Method of Solving Linear Equations–An Improvement in ART", CT Theory and Applications, vol.3, No.4, 1994.

6. Ipsen, Ilse Meyer, Carl. (1997). The Idea Behind Krylov Methods. The American Mathematical Monthly. 105. 10.2307/2589281.

7. Kstler, Harald Popa, Constantin Prmmer, M Rde, Ulrich. (2006). Towards an algebraic multigrid method for tomographic image reconstruction-improving convergence of ART.

8. Li, Y.; Liu, S.; Inaki, S.H. Dynamic Reconstruction Algorithm of Three-Dimensional Temperature Field Measurement by Acoustic Tomography. Sensors 2017, 17, 2084.

10 Appendix

10.1 Code for LSQM

```
Z=19.08;%const T= 1/(A^2Z^2)
dim=4;
length=15;
L_grid=length/dim;
S = section_divide(dim,length,3);
%generate S matrix, input:dimension, length of the edge, division of the
positions of the transmitter
t=tof_generator(1000,length,3);
%based on the simulation T distribution generate simulation TOF
A=pinv(S'*S)*S'*t;
T=1./(A.^2*Z^2);
```

10.2 Code for Initializing $A^{(0)}$

```
function initial_A = ART_initialize(dim,t,S_four) %dim is something like 5*5
    %objective: calculate initial A based on 4*4 A.
    %idea: generate pixels on the region and assgin each pixel a value of
    %corresponding A in the 4*4, then for each grid in dim*dim average the
    %pixel values lying in the grid

    count=400;%# pixel dimension of blocks
    %t=tof_generator(1000,15,3);
    pixel_num = count*dim;
    %S_four = section_divide(4,15,3);
    %S_four=S;
```

```
    %A_four = (S_four'*S_four)\S_four'.*t;
    A_four=pinv(S_four'*S_four)*S_four'*t;
    Pixel = [];
    for i=1:4
        tempp=[];
        for j=1:4
            temp = ones(pixel_num/4,pixel_num/4)*A_four(j+4*(i-1));
            tempp = [tempp;temp];
        end
        Pixel = [Pixel tempp];
    end
    temp_A=zeros(dim,dim);
    for i=1:dim
        for j=1:dim
            calculator = Pixel(1+count*(j-1):count*j,1+count*(i-1):count*i);
            temp_A(i,j) = sum(sum(calculator))/(count*count);
        end
    end
   final = temp_A';
   initial_A = final(:);
end
```

10.3 Code for ART

```
Z=19.08;%const T= 1/(A^2Z^2)
error = 0.0000000000001; %terminate condition
range_const=0.0002;%half the range width, need to be modified based on A_initial
par = 1.2; %control the convergence speed
dim=10; %grid division 10*10
length=15;
L_grid=length/dim;
S = section_divide(dim,length,3); %generate S matrix, input:dimension, length
of the edge, division of the positions of the transmitter
t=tof_generator(1000,length,3);%based on the simulation T distribution generate
simulation tof
A_initial = ART_initialize(dim);%initialize ART, based on 4*4 case
A_range = [A_initial-range_const A_initial+range_const];
i=0;
iteration=0;%number of iteration
A= A_initial;
```

```
while(1)
    iteration=iteration+1;
    i=i+1;
    if(i>24)
        i=1;
    end
    A = A + par*(t(i) - S(i,:)*A)/(S(i,:)*S(i,:)')*S(i,:)';
    %keep A in the range to avoid local optimization
    bol_temp1 = (A>A_range(:,1));
    A(bol_temp1==0) = A_range(bol_temp1==0,1);
    bol_temp2 = (A<A_range(:,2));
    A(bol_temp2==0) = A_range(bol_temp2==0,2);
    w = (S(i,:)*A-t(i))^2;
    if(sqrt(w/dim^2)<error)
        break;
    end
end
T=1./(A.^2*Z^2);
```

10.4 Interface

Input for temperature:

```
@(x,y)800*exp((-70*(x-10).^2-30*(y-10).^2)/1000)+800*exp((-70*(x-13.3).^2-30*(y-13.3).^2)/300)+800
```

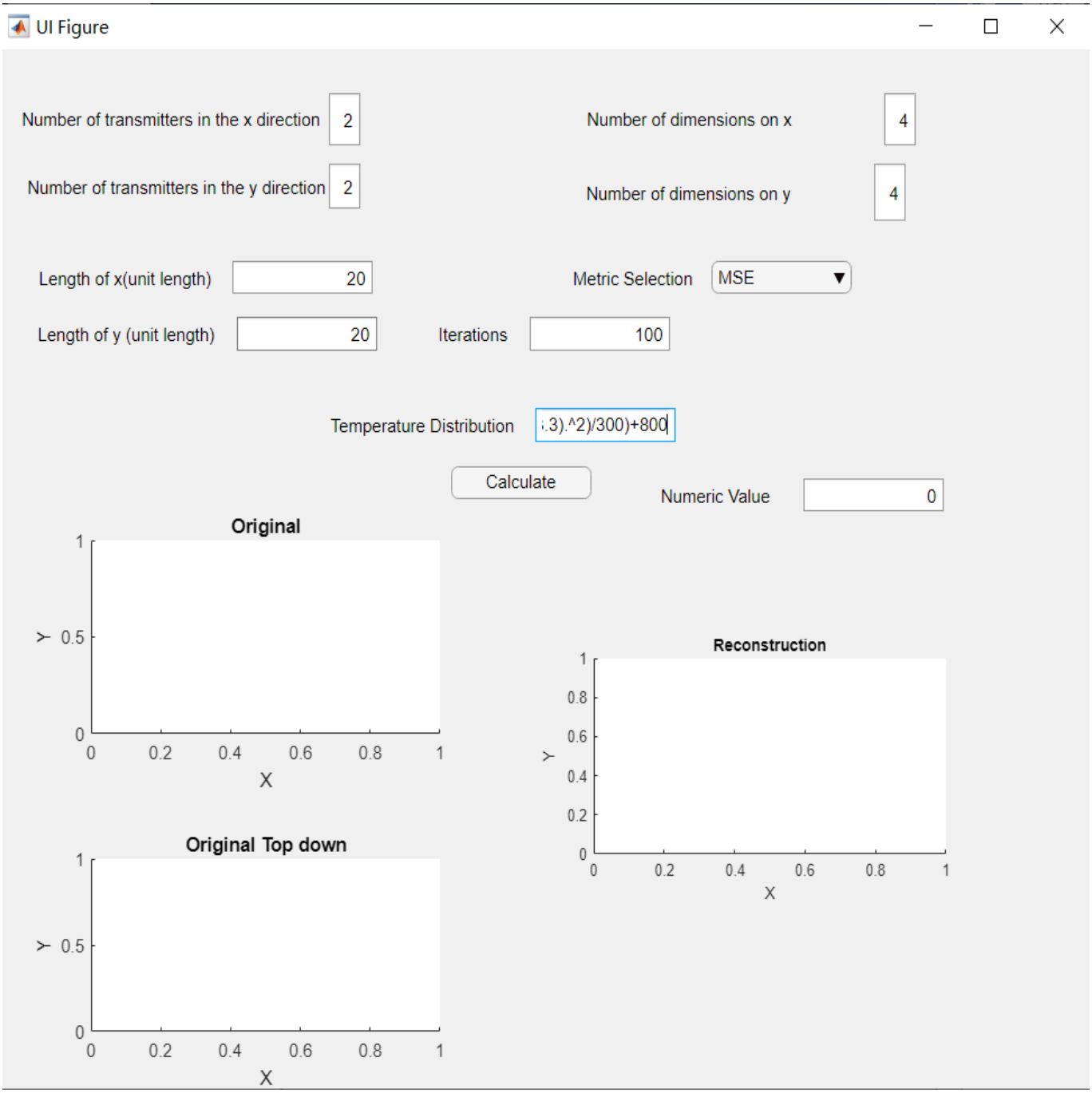


Figure 37: Interface input

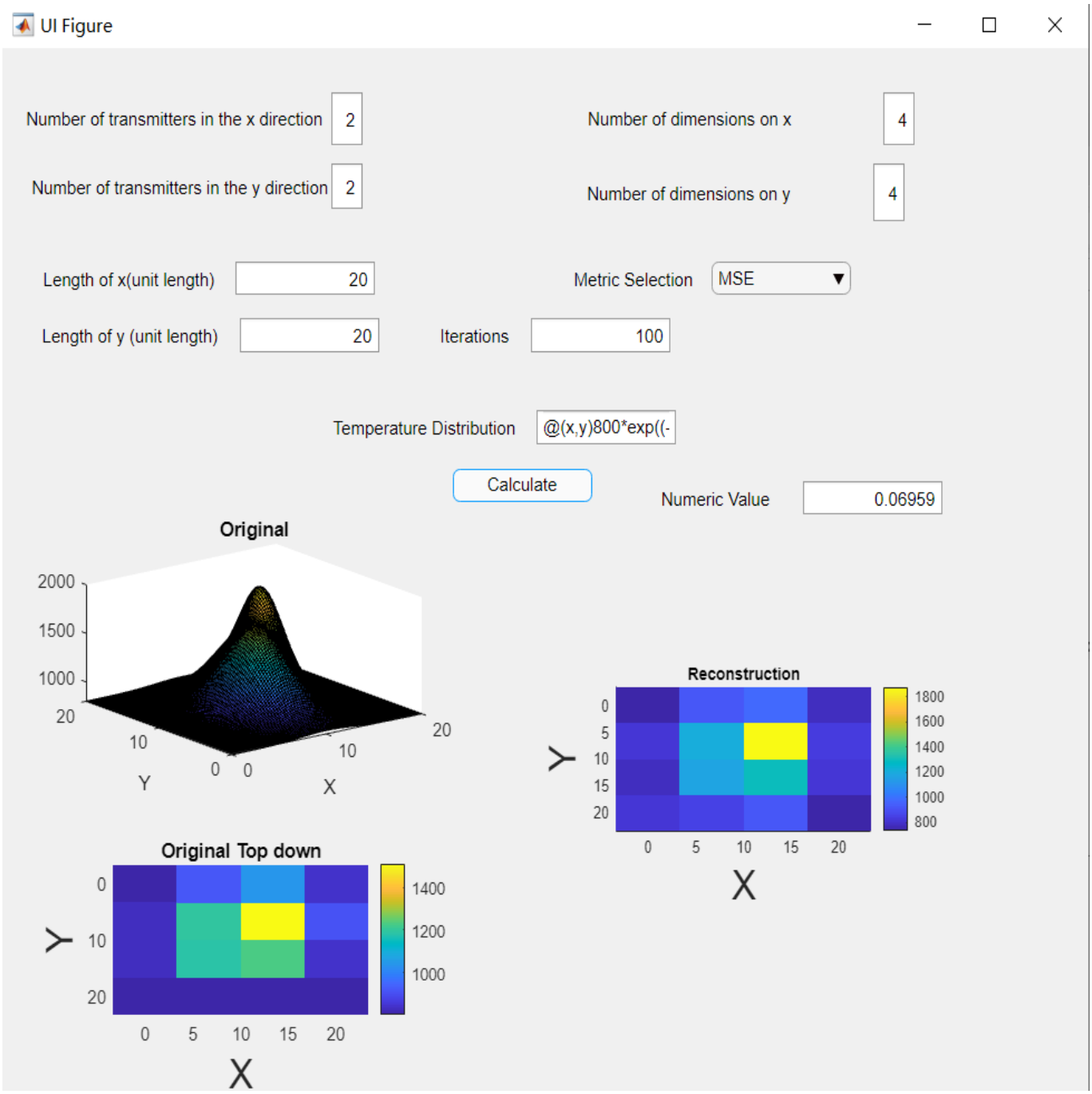


Figure 38: Interface output