Available online at www.qu.edu.iq/journalcm
JOURNAL OF AL-QADISIYAH FOR COMPUTER SCIENCE AND MATHEMATICS
ISSN:2521-3504(online) ISSN:2074-0204(print)

Decompress Image Using Finite Difference Method

Jumana H.S. ALKHALISSI [a] Ahmed Mohsin MAHDI [b]

[b]*Faculty of Computer Science and Information Technolog.University of Al-Qadisiyah. jumanaalsady@gmail.com*

[b]*Faculty of Computer Science and Information Technolog.University of Al-Qadisiyah. ahmed.m.mahdi@qu.edu.iq*

ARTICLE INFO

Article history:
Received: dd /mm/year
Rrevised form: dd /mm/year
Accepted : dd /mm/year
Available online: dd /mm/year

Keywords:

Image Processing,

Discretization of the PDEs Operators,

Finite Difference Method,

MSE,SNR,PSNR and SSIM

ABSTRACT

A lot of techniques are represented to have a useful for denoising and in-painting digital images. These techniques are based on a partial differ- ential equations (PDEs). Thus, we need to find the solution of PDEs. In this work, we consider a finite difference method to solve PDEs approxi- mately based on image compression. In addition, we compared the results in different criteria of the errors

MSC..

https://doi.org/ 10.29304/jqcm.2022.14.2.999

∗Corresponding author

Email addresses:

Communicated by ‘sub etitor’

1. Introduction

Image compression is getting more attention from researchers because of, the growth and development of the image and the vast size of data transferring through networks [1] and [2]. For this reason, required the need for an effective technique to store and conveyance with high quality for the images [3] and [4]. Compression techniques can be divided into two types which are lossy and lossless techniques. The difference between these types is that the compression ratio is higher in lossy technique compared with lossless technique. Some ofresearcher works published about these techniques can be seen in [5]- [9].

One of the computer science branches is an image compression, which is depending on the principle (reduced the size of an image with out lossing too much information). Several partial differential equations (PDEs) appeared their usefulness in called inpainting methods. To understand the basic idea of PDE based on image compression we need to do the following steps [11]:

- Select a subset of the pixels from the original image to be stored by using some algorithm.

- Compression algorithms i.e. Store these pixels in as condensed a way as possible.

- Decompression i.e. Using a PDE operator to Interpolate these stored pixels for restoring the image over the whole grid.

We need to find the solution of PDE, some times we can find some exact solutions for some PDEs. In the other hand, this is not possible generally. So, in this case we can look a way to find approximate numerical solutions for a given equation. Finite difference method is one of the main numerical techniques for PDEs. The report is organized as follows: In Section 2, we will show how certain PDEs used in image analysis can be discretized [10], what the principle of the finite difference method and how we can calculate it. Following, in section 3, we consider the basic information about the steps of the image compression. Section 4, we will consider some investigations about the finite difference method and compare the results with the linear interpolation by the summarizing them.

In order to recover our image, we set out to discretize the operators. Note that for now, we assume that we have already selected our subset of pixels by some methods, random or otherwise

2. Discretization of operators

In general, the PDEs form can be representing as:

$Lv = F, \quad (t, x, y) \in R+ \times \Omega$

$\partial v\ (t, x, y) = 0, \quad$ on $\quad R+ \quad \partial\Omega$

∂N

$v\ (0, x, y) = f\ (x, y),$

where Ω is a domain of the image which is bounded by the normal N , denoted by $\partial\Omega$. L is generally a second order differential operator such as:

$\partial v/\partial t(t,x,y)+H(x,y,v(t,x,y),\nabla v(t,x,y),\nabla\hat{}2\ v(t,x,y))=0$

We consider how we can use the finite difference method to discretize the PDE's operator. The reason behind choosing the finite difference method. The structure of a digital image, which is, distributed the pixels uniformly. Therefore we can relate an image with uniformly grids, as in Figure 1. As a simple way to explain the procedure of discretization in the difference method. Figure 2 can be used.

Now, we need to understand how finite differences can be evaluated and what their types are..

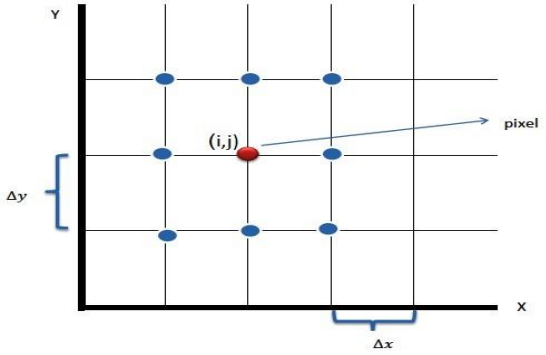


Figure 1: Grid on the space domain with 3 3 neighborhood of the vertex (i, j).

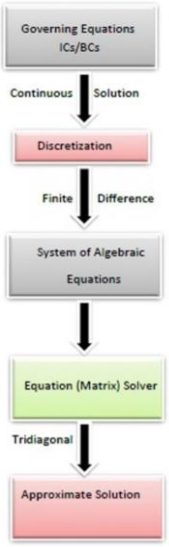


Figure 2: steps of the algorithms

2.1Finite Difference Method

The principle of the finite difference method based on an approximation of the derivatives in the PDE using linear combinations of function values at the grid points.

2.1.1. Approximation of first order derivative

The finite difference method has three types of differences as follows: (see Figure 3)

Backward Difference: which can be represented as $\left(\frac{\partial u}{\partial x}\right)_i \approx \frac{u_i - u_{i-1}}{\Delta x}$

Forward Difference: which can be represented as $\left(\frac{\partial u}{\partial x}\right)_i \approx \frac{u_{i+1} - u_i}{\Delta x}$

Central Difference: represented by the following rule

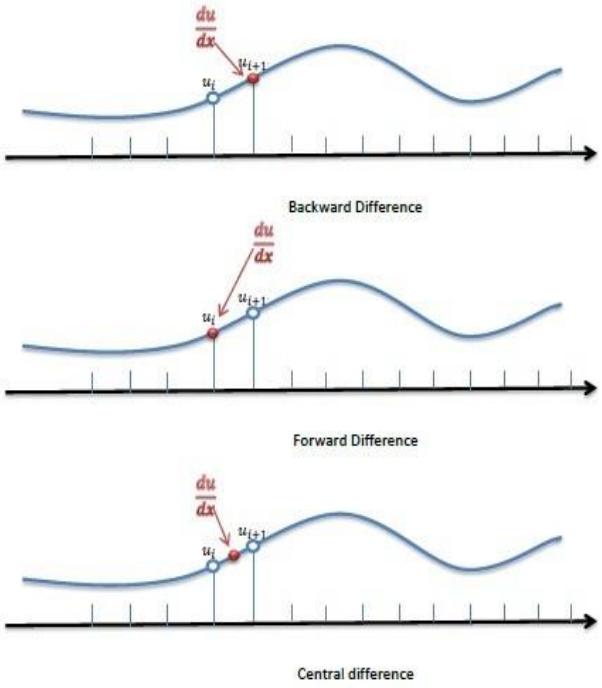


Figure 3: Geometric interpretation

We can use Taylor series expansions in Eq. (1) using one of the variables to see how well finite difference approximations work.

$$u(x) = \sum_{n=0}^{\infty} \frac{(x - x_i)^n}{n!} \left(\frac{\partial^n u}{\partial x^n}\right)_i, \qquad (1)$$

$$T_1: u_{i+1} = u_i + \Delta x \left(\frac{\partial u}{\partial x}\right)_i + \frac{(\Delta x)^2}{2}\left(\frac{\partial^2 u}{\partial x^2}\right)_i + \frac{(\Delta x)^3}{6}\left(\frac{\partial^3 u}{\partial x^3}\right)_i + \cdots \; T_2: u_{i-1}$$
$$= u_i - \Delta x \left(\frac{\partial u}{\partial x}\right)_i + \frac{(\Delta x)^2}{2}\left(\frac{\partial^2 u}{\partial x^2}\right)_i - \frac{(\Delta x)^3}{6}\left(\frac{\partial^3 u}{\partial x^3}\right)_i + \cdots$$

2.1.2 Accuracy of finite difference approximations

We can measure the accuracy of the approximation by truncation errors as follows:

$$T_1 \Longrightarrow \left(\frac{\partial u}{\partial x}\right)_i = \frac{u_{i+1} - u_i}{\Delta x} - \frac{(\Delta x)}{2}\left(\frac{\partial^2 u}{\partial x^2}\right)_i - \frac{(\Delta x)^2}{6}\left(\frac{\partial^3 u}{\partial x^3}\right)_i + \cdots$$

Forward difference truncation error O(Δx)

$$T_2 \Longrightarrow \left(\frac{\partial u}{\partial x}\right)_i = \frac{u_i - u_{i-1}}{\Delta x} - \frac{(\Delta x)}{2}\left(\frac{\partial^2 u}{\partial x^2}\right)_i - \frac{(\Delta x)^2}{6}\left(\frac{\partial^3 u}{\partial x^3}\right)_i + \cdots$$

Backward difference truncation error O(Δx)

$$T_1 - T_2 \Longrightarrow \left(\frac{\partial u}{\partial x}\right)_i = \frac{u_{i+1} - u_{i-1}}{2\,\Delta x} - \frac{(\Delta x)^2}{6}\left(\frac{\partial^3 u}{\partial x^3}\right)_i + \cdots$$

Central difference truncation error O(Δx)2.

This is more accurate becauseΔx2 → 0 is faster than Δx as Δx → 0.

2.1.3 Approximation of the second order derivative

Approximation of the second order derivative can be represented as the central difference scheme as follows:

$$T_1 + T_2 \Longrightarrow \left(\frac{\partial^2 u}{\partial x^2}\right)_i = \frac{u_{i+1} - 2u_i + u_{i-1}}{(\Delta x)^2} - O(\Delta x)^2$$

2.1.4 Approximation in two dimensions

Approximation in two dimensions can be represented as follows:

$$\left(\frac{\partial^2 u}{\partial x \partial y}\right)_{i,j} = \frac{\left(\frac{\partial u}{\partial y}\right)_{i+1,j} - \left(\frac{\partial u}{\partial y}\right)_{i-1,j}}{2\,\Delta x} + O(\Delta x)^2 . \qquad (2)$$

Therefore we need to calculate $\left(\frac{\partial u}{\partial y}\right)_{i+1,j}$ and $\left(\frac{\partial u}{\partial y}\right)_{i-1,j}$

$$\left(\frac{\partial u}{\partial y}\right)_{i+1,j} = \frac{u_{i+1,j+1} - u_{i+1,j-1}}{4\Delta y} + O(\Delta y)^2, \qquad (3)$$

$$\left(\frac{\partial u}{\partial y}\right)_{i-1,j} = \frac{u_{i-1,j+1} - u_{i-1,j-1}}{4\Delta y} + O(\Delta y)^2. \qquad (4)$$

Then we can substitute each of Eq. (3) and (4) in Eq. (2) we get: (see Figure 4)

$$\left(\frac{\partial^2 u}{\partial x \partial y}\right)_{i,j} = \frac{u_{i+1,j+1} - u_{i+1,j-1} - u_{i-1,j+1} + u_{i-1,j-1}}{4\Delta x \Delta y} + O[(\Delta x)^2, (\Delta y)^2] \quad (5)$$

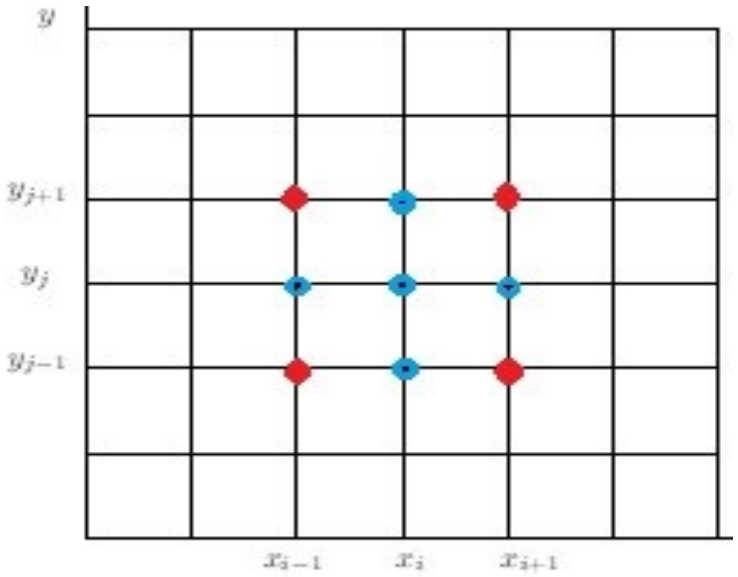


Figure 4: Approximation in two dimensions

We will consider an example about the simple finite difference method of the heat equation:

Example 1. [10]

$$\begin{cases} \frac{\partial v}{\partial t} = r\,\Delta v = r\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right), & (t,x,y) \in R^+ \times \Omega \\ = 0, & on \quad R^+ \times \partial\Omega \\ = f(x,y), & \end{cases} \quad \frac{\partial v}{\partial N}(t,x,y) \quad v(0,x,y) \quad (6)$$

where r is a positive constant. To discretized Eq. (1) we need Taylor expansions. In simple way, discretized each derivative separately in x and y. So we obtain:

$$\frac{\partial v}{\partial t} - r\,\Delta v|_{i,j}^n = \frac{v_{i,j}^{n+1} - v_{i,j}^n}{\Delta t} - \frac{v_{i+1}^n + v_{i-1}^n + v_{j+1}^n + v_{j-1}^n - 4v_{i,j}^n}{h^2} + O(\Delta t) + O(h^2),$$

where h = Δx = Δy. Then the difference scheme solution will propose as:

$$u_{i,j}^{n+1} = u_{i,j}^n + \frac{r\,\Delta t}{h^2}\left(u_{i+1,j}^n + u_{i-1,j}^n + u_{i,j+1}^n + u_{i,j-1}^n - 4u_{i,j}^n\right), \quad (7)$$

where $v_{i,j}^n$ is the exact solution for the PDE and $u_{i,j}^n$ is the discrete (approximate) solution at location (ih, jh) and time nΔt. For the boundary condition we can use the same procedure and the initial condition u0 = gi,j where g represents the discretization of f .

Now, we will consider the basic procedure of the compression to the PDE.

3. Image Compression

As we mentioned before there are three important steps to do compress for the image. The idea of image compression is to select some points from the original image as landmarks points and drop the reminder. But,

the question here how we can select these points to do the compression. For this part of the process, there are two types to select the pixels of image:

- Randomly.

- Uniformly, which is the best way to select where the color gradient of image is changing quickly.

Then we will store the pixels that we are selected in the previous step. General-purpose data compression can be used to compress the representation. Finally, we can interpolate the stored pixels using PDE operator.

4. Experimental Results

In this part of the work, we will implement the finite difference method on the images. Also, we will do comparison between the finite difference method and the linear interpolation.

We implemented different images by using a linear interpolation and the finite difference method. In addition we consider different PDE's (heat , Poisson and biharmonic equation) then apply the finite difference method of these equations on the images.

In this part, we implemented three different numbers of the iteration of the FDM of the heat equation. After, we consider another type of PDE, which is Poisson equation, and also we do the same procedure in Figure 5:

In the other hand, we applying the FDM for the biharmonic equation with the result as in figure 6: When we apply biharmonic equation and discretized each derivative we need 13 points to estimate a value. As a result, we can investigate the third equation less time than the others but it is not good when we repeat the iteration more than 2 because it will need a lot of points outside the domain.

Now, we evaluate the structural similarity index (SSIM)for each color. SSIM for the blue color of the yellow rose greater than the other colors because the lowest of intensity in this color(see Figure(15)). As we can see, from these results the different formula of the finite difference method having the same values approximately. While, at the higher iteration we got better result and approached to be the same value when we apply the linear interpolation approximately. Also, the method stencil of 13 points to find the missing pixel values by FDM interpolation when they increase in the order of iterations, the result was a bad comparison of the other methods because this method needs 13 points to estimate one value. So, when we repeat the same formula several times it will be out of the bounded of the image.

5. Conclusion

The partial differential equation represents a vital part in the image processing generally; we take these features to create the interpolation from their approximate solutions by using the finite difference method. We investigated the types of the equation affected on the quality of image and also the number of repeating the procedures.

References

[1] Chandler, Damon M, *Seven challenges in image quality assessment: past, present, and future research*, *International Scholarly Research No- tices*,Hindawi (2013).

[2] Vijayvargiya, Gaurav and Silakari, Sanjay and Pandey, Rajeev, *A survey: various techniques of image compression*, *arXiv preprint arXiv:1311.6877*, (2013).

[3] Taubman, David, *High performance scalable image compression with EBCOT*, *IEEE Transactions on image processing*, **9(7)**(2013),1158–1170.

[4] Masood, Saleha and Sharif, Muhammad and Yasmin, Mussarat and Raza, Mudassar and Mohsin, Sajjad, *Brain image compression: a brief sur- vey*, *Research Journal of Applied Sciences, Engineering and Technology*, **5(1)**(2013),49–59.

[5] Son, Chang-Hoon and Kim, Ji-Won and Song, Sung-Gun and Park, Seong- Mo and Kim, Young-Min, *Low complexity embedded compression algorithm for reduction of memory size and bandwidth requirements in the JPEG2000 encoder*, *IEEE Transactions on Consumer Electronics*, **56(4)**(2010),2421– 2429.

[6] Zhao, Dongyu and Zhu, Shiping and Wang, Fengchao, *Lossy hyperspec- tral image compression based on intra-band prediction and inter-band fractal encoding*, *Computers & Electrical Engineering*, **54**(2016),494–505.

[7] Engelson, Vadim and Fritzson, Dag and Fritzson, Peter, *Lossless compres- sion of high-volume numerical data from simulations*, *Proc. Data Compres- sion Conference*, (2000).

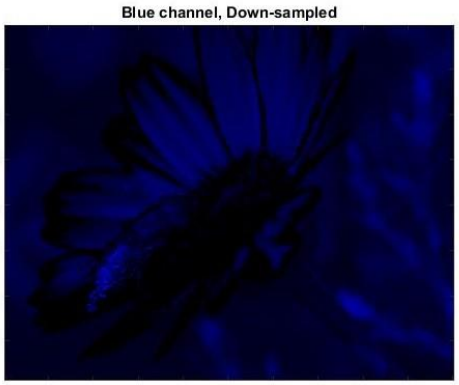
[8] Zhang, Yong and Adjeroh, Donald A, *Prediction by partial approximate matching for lossless image compression*, *IEEE transactions on image pro- cessing*, **17(6)**(2008),924–935.

[9] Andris, Sarah and Weickert, Joachim and Alt, Tobias and Peter, Pascal, *JPEG meets PDE-based Image Compressio*, *2021 Picture Coding Symposium (PCS)*, (2021),1–5.

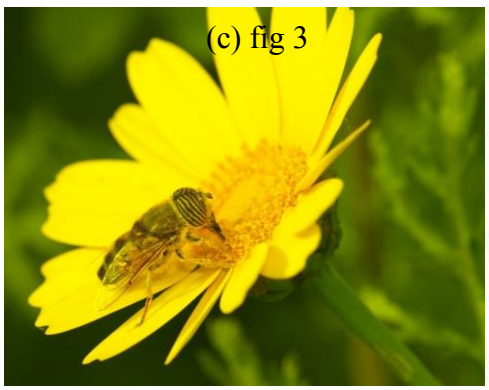
[10] Chang-Ock Lee, *Basic Numerical Methods for Image Process- ing*.Department Mathematical Sciences, KAIST, Daejeon, 305-701 Korea.

[11] Markus Peloquin and Leland Jeeris, *PDE-based Image Compression*. Dec. 18, 2009.

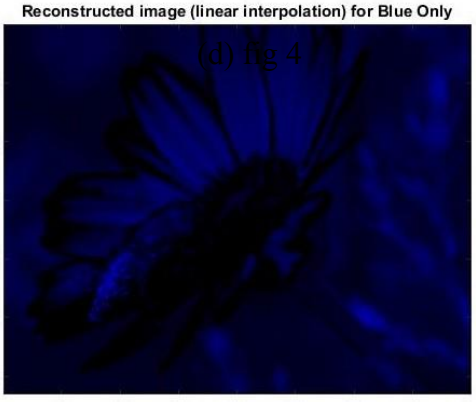
(a) fig 1



(b) fig 2



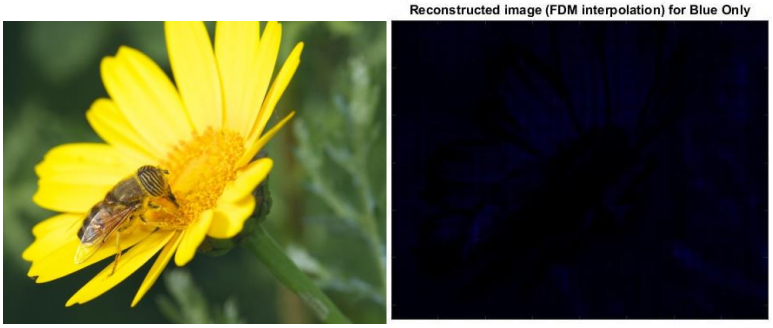
(c) fig 3



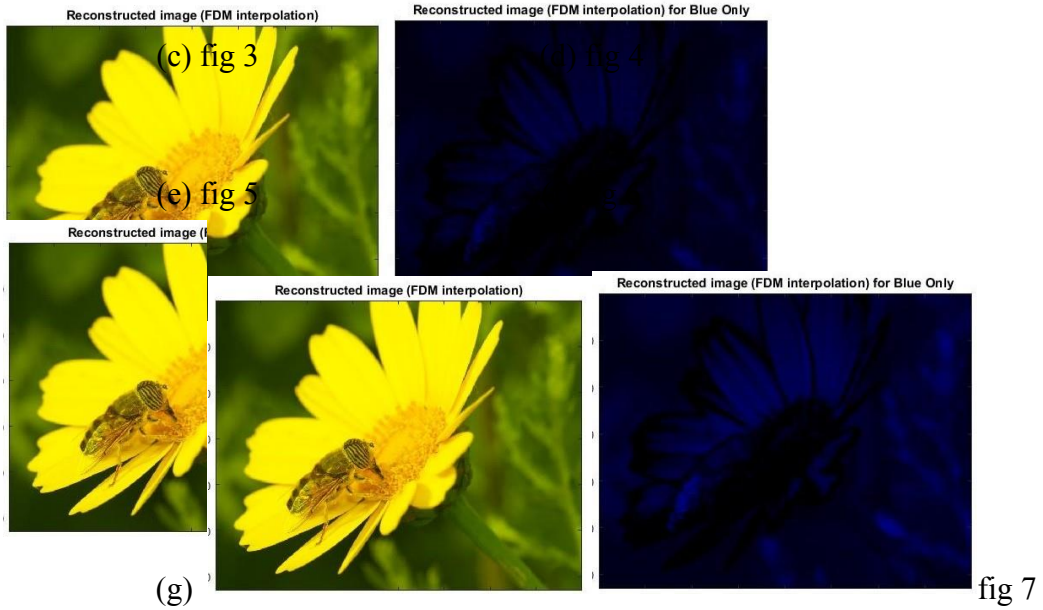
(d) fig 4

(e) fig 5

Figure 5: Yellow rose image with compress and decompress by linear interpolation



(a) fig 1 (b) fig 2



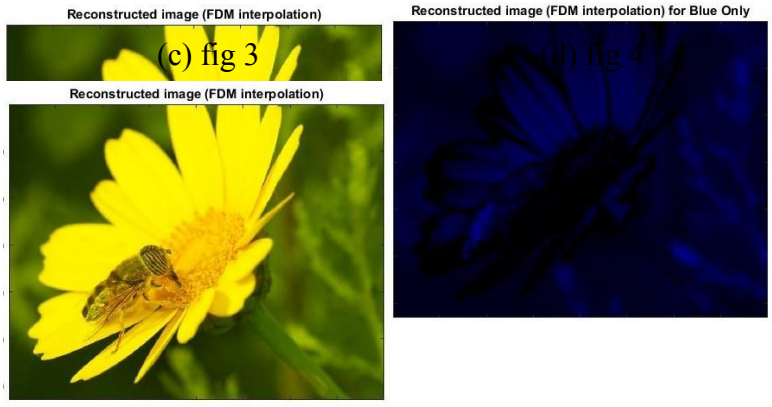
(c) fig 3 (d) fig 4

(e) fig 5

(g) fig 7

Figure 6: (a) Yellow rose image (b,c) FDM of heat equation with 2 iterations (d,e) FDM of heat equation with 5 iterations (f,g) FDM of heat equation with 10 iterations

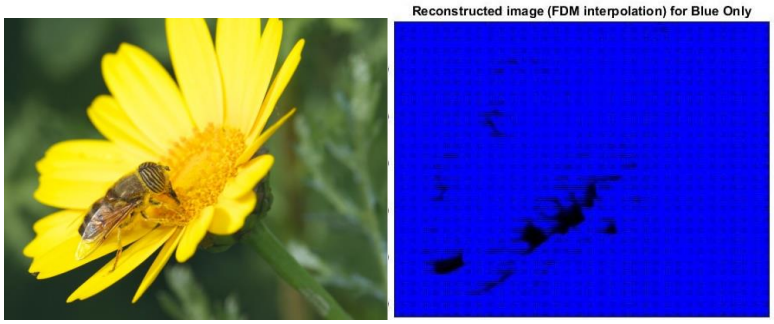
(a) fig 1 (b) fig 2



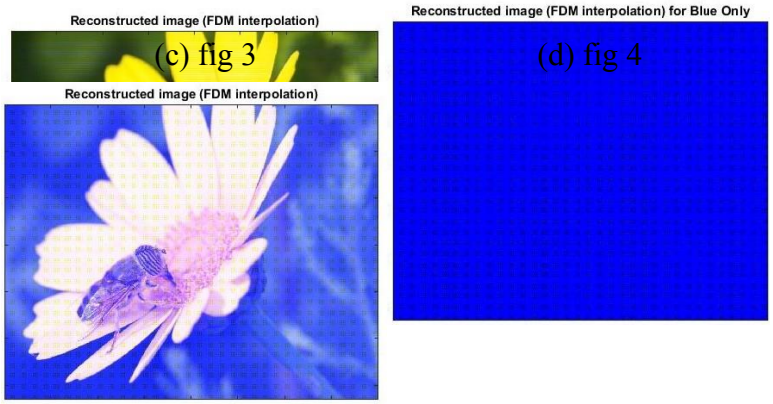
(c) fig 3 (d) fig 4

(e) fig 5

Figure 7: (a) Yellow rose image (b,c) FDM of Poisson equation with 2 iterations (d,e) FDM of Poisson equation with 10 iterations.



(a) fig 1 (b) fig 2



(c) fig 3 (d) fig 4

(e) fig 5

Figure 8: (a) Yellow rose image (b,c) FDM of biharmonic equation with 2 iterations (d,e) FDM of biharmonic equation with 10 iterations.

image name	Size of the image	Size after compresse	compression ratio
yellow rose	504000	31400	93.77%
lena	50625	3136	93.80%
animal (graysacle)	50325	3060	93.92%
frog (grayscale)	2496000	156000	93.75%

Figure 9: The compression ratio

method name	Number of iteration	MSE for red color	MSE for green color	MSE for blue color
linear	0	93.282	89.636	31.828
FDM(Heat eq.)	0	93.6	89.947	31.836
FDM(Heat eq.)	50	93.281	89.637	31.829
FDM(Stencils 9p.)	0	93.6	89.947	31.836
FDM(Stencils 9p.)	50	93.281	89.637	31.829
FDM(Stencils 13p.)	0	93.6	89.947	31.836
FDM(Stencils 13p.)	1	92.244	88.618	31.4
FDM(Stencils 13p.)	2	50.569	48.443	18.513

Figure 10: Mean square error for the yellow rose image

method name	Number of iteration	SNR for red color	SNR for green color	SNR for blue color
linear	0	1.8912	1.8909	1.5047
FDM(Heat eq.)	0	1.8947	1.8946	1.5047
FDM(Heat eq.)	50	1.8912	1.891	1.5047
FDM(Stencils 9p.)	0	1.8947	1.8946	1.5047
FDM(Stencils 9p.)	50	1.8912	1.891	1.5047
FDM(Stencils 13p.)	0	1.8947	1.8946	1.5047
FDM(Stencils 13p.)	1	1.825	1.8279	1.497
FDM(Stencils 13p.)	2	0.23545	0.20428	0.23592

Figure 11: Signal to noise ratio for the yellow rose image

method name	Number of iteration	PSNR for red color	PSNR for green color	PSNR for blue color
linear	0	4.3844	4.5576	9.0543
FDM(Heat eq.)	0	4.3696	4.5425	9.0532
FDM(Heat eq.)	50	4.3844	4.5575	9.0542
FDM(Stencils 9p.)	0	4.3696	4.5425	9.0532
FDM(Stencils 9p.)	50	4.3844	4.5575	9.0541
FDM(Stencils 13p.)	0	4.3696	4.5425	9.0532
FDM(Stencils 13p.)	1	4.433	4.6072	9.1131
FDM(Stencils 13p.)	2	7.0436	7.2301	11.408

Figure 12: Peak signal to noise ratio for the yellow rose image

method name	Number of iteration	MSE for red color	MSE for green color	MSE for blue color
linear	0	107.36	64.356	64.015
FDM(Heat eq.)	0	107.8	64.569	64.162
FDM(Heat eq.)	50	107.36	64.372	64.029
FDM(Stencils 9p.)	0	107.8	64.569	64.162
FDM(Stencils 9p.)	50	107.36	64.375	64.031
FDM(Stencils 13p.)	0	107.8	64.569	64.162
FDM(Stencils 13p.)	1	106.27	63.724	63.303

Figure 13: Mean square error for the Lena image

method name	Number of iteration	MSE for red color	MSE for green color	MSE for blue color
linear	0	107.36	64.356	64.015
FDM(Heat eq.)	0	107.8	64.569	64.162
FDM(Heat eq.)	50	107.36	64.372	64.029
FDM(Stencils 9p.)	0	107.8	64.569	64.162
FDM(Stencils 9p.)	50	107.36	64.375	64.031
FDM(Stencils 13p.)	0	107.8	64.569	64.162
FDM(Stencils 13p.)	1	106.27	63.724	63.303

Figure 14: Signal to noise ratio and peak signal noise ratio for the Lena image

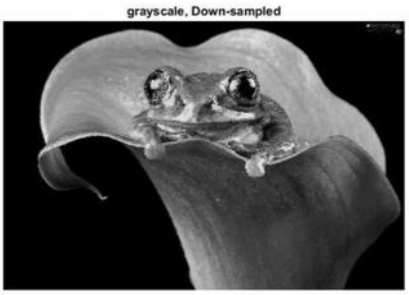
image name	SSIM for red color	SSIM for green color	SSIM for blue color
yellow rose	0.66805	0.668	0.71187
Lena	0.66853	0.66816	0.66732

Figure 15: Structural similarity index for the yellow rose image and the Lena image

image name	compresse the color	method name	Number of iteration	MSE	SNR	PSNR	SSIM
animal	grayscale	linear	0	0.14587	0.0050871	8.3772	0.67508
animal	grayscale	FDM(Heat eq.)	0	0.42153	12.164	3.7688	0.008781
animal	grayscale	FDM(Heat eq.)	50	0.12981	0.078966	8.884	0.64953
animal	grayscale	FDM(Stencils 9p.)	0	0.42153	12.164	3.7688	0.008781
animal	grayscale	FDM(Stencils 9p.)	50	0.13612	0.096659	8.6778	0.64177
animal	grayscale	FDM(Stencils 13p.)	0	0.42153	12.164	3.7688	0.008781
animal	grayscale	FDM(Stencils 13p.)	1	0.25017	0.59992	6.0347	0.090359
animal	grayscale	FDM(Stencils 13p.)	2	0.27197	0.98133	5.6718	0.063944
frog	grayscale	linear	0	0.072064	-0.047504	11.44	0.87164
frog	grayscale	FDM(Heat eq.)	0	0.34504	12.042	4.6383	0.40967
frog	grayscale	FDM(Heat eq.)	50	0.037241	-0.043737	14.307	0.88385
frog	grayscale	FDM(Stencils 9p.)	0	0.34504	12.042	4.6383	0.40967
frog	grayscale	FDM(Stencils 9p.)	50	0.037372	-0.047515	14.292	0.88254
frog	grayscale	FDM(Stencils 13p.)	0	0.34504	12.042	4.6383	0.40967
frog	grayscale	FDM(Stencils 13p.)	1	0.17822	0.34761	7.5075	0.4431
frog	grayscale	FDM(Stencils 13p.)	2	0.19443	0.6539	7.1294	0.42882

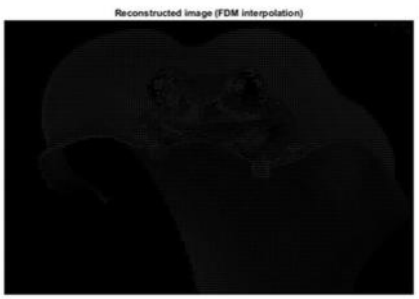
Figure 16: Grayscale Image

(a) Original image



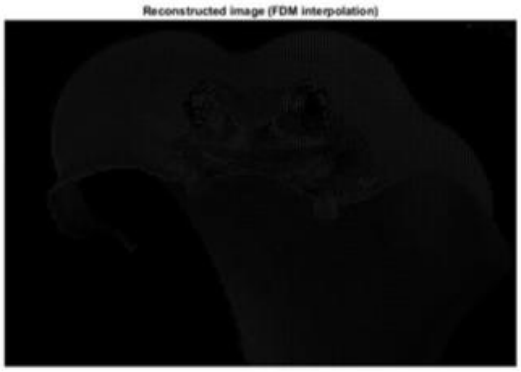
(b) compressed image by downsampling

(c) Decompress image by linear interpolation



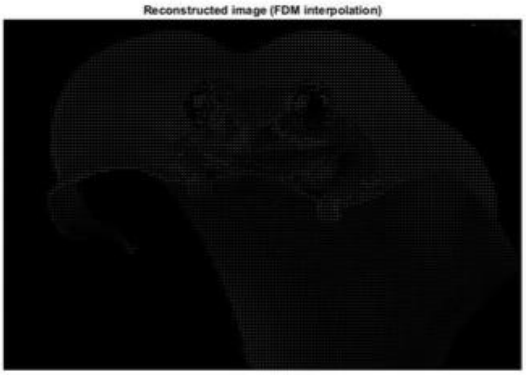
(d) FDM Heat eq.

(e) FDM Heat eq.with 50 iterations

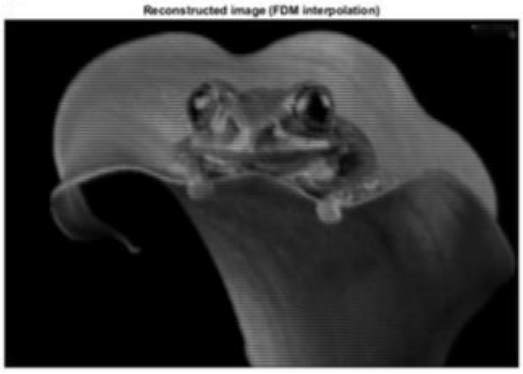


(f) FDM stencils 9p.

(g) FDM stencils 9p.with 50 iterations



(h) FDM stencils 13p.



(i) FDM stencils 13p.with 2 iterations

Figure 17: Grayscale example