RESEARCH ARTICLE   OPEN ACCESS

An Extended Krylov Subspace Method for Decoding Edge-Based Compressed Images by Homogeneous Diffusion

Volker Grimm | Kevin Liang

Institute for Applied and Numerical Mathematics, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany

Correspondence: Volker Grimm (volker.grimm@kit.edu)

ABSTRACT

The heat equation is often used to inpaint dropped data in inpainting-based lossy compression schemes. We propose an alternative way to numerically solve the heat equation by an extended Krylov subspace method. The method is very efficient with respect to the computation of the solution of the heat equation at large times. And this is exactly what is needed for decoding edge-compressed pictures by homogeneous diffusion.

1 | Introduction

Inpainting-based compression of images refers to the idea of identifying prominent data in an image and storing only this data. All other data is disregarded and, when needed, reconstructed by inpainting. In particular, we will consider edge-based compressed images, where the edges of an image, together with adjacent gray/color data, are stored [1–4]. This idea can be seen as a second-generation image coding method where the properties of the human visual system are taken into account [5]. The edge-based compression works very well for cartoon-like images [6]. To improve the quality of the reconstruction for natural images, we also compress images based on dithering [7]. Dithering also works due to the perception of images by the human visual cortex. We will work with these two basic coding techniques. But since our new contribution refers to the decoding, more advanced coding techniques [8, 9] can easily be combined with our approach. We would also like to mention that, while our proposal deals with homogeneous diffusion, the proposed method might be carried over to more general linear evolution equations, e.g., in image registration [10], and to nonlinear partial differential equations used for inpainting by the help of exponential integrators, in which the linear part is solved as proposed in this work. More information on exponential integrators might be found in the survey by Hochbruck and Ostermann [11], and information on advanced image inpainting methods by partial differential equations can be found in Schoenlieb's book [12].

To review the basic idea of inpainting-based compression of images, let $f : \Omega \to \mathbb{R}$ be a given gray-scale picture. $f(\mathbf{x})$ refers to the intensity of light and Ω is the rectangular domain of the picture. In a color picture, any channel is treated
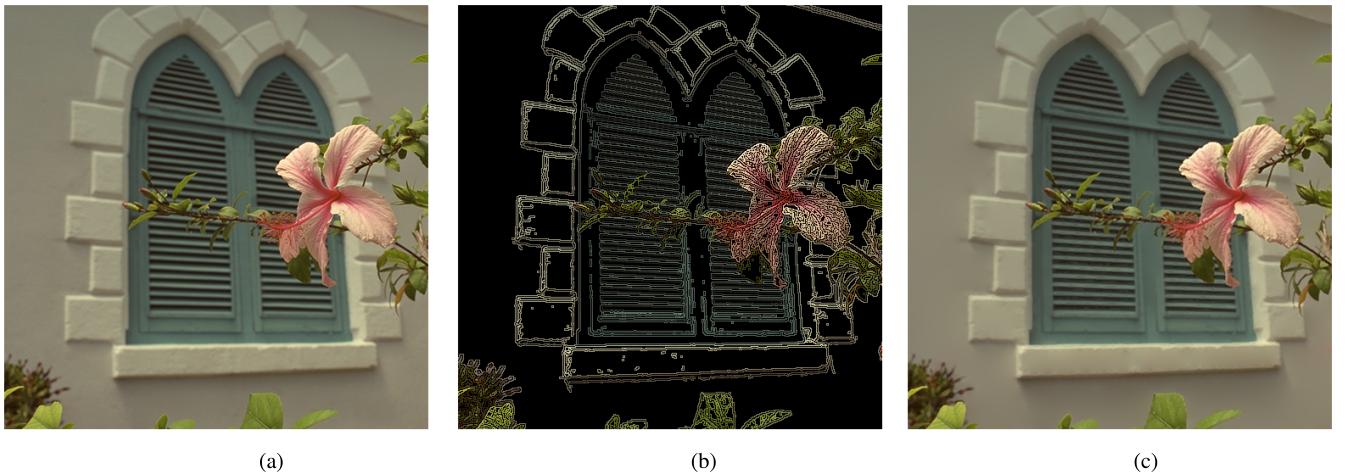
FIGURE 1 | Sketch of the compression scheme: In the encoding step, the original picture is reduced to a subset, in the decoding phase, the original is reconstructed from this subset. (a) Original, (b) compression, and (c) reconstruction.

in the same way. After the compression of the picture, the intensities are only known on a subdomain $K \subset \Omega$. This splits the image into a known part K and an unknown part $\Omega \setminus K$. To flag the stored pixels efficiently, we will use the function

$$c(\mathbf{x}) = \begin{cases} 1 & \text{for } \mathbf{x} \in K, \\ 0 & \text{for } \mathbf{x} \in \Omega \setminus K, \end{cases}$$

which we will refer to as *inpainting mask*. With the help of the inpainting mask, the compressed image f_c can be written as $f_c = cf$. In the middle of Figure 1, such a compressed picture of the picture on the left-hand side is shown. Only the pixels that are not black are stored. This data is sufficient for the reconstruction on the right-hand side of Figure 1.

For the reconstruction, we inpaint the missing data by the heat equation. The system reads

$$\partial_t u(\mathbf{x}, t) = (1 - c(\mathbf{x}))\Delta u(\mathbf{x}, t), \qquad \text{in } \Omega \times [0, \infty),$$

$$\partial_n u(\mathbf{x}, t) = 0, \qquad \text{in } \partial\Omega \times [0, \infty),$$

with Neumann boundary conditions and the compressed image

$$u(\mathbf{x}, 0) = c(\mathbf{x}) f(\mathbf{x}), \qquad \mathbf{x} \in \Omega,$$

as initial data. The reconstructed picture is the solution $u(\mathbf{x}, t)$ of the above PDE at a large time t. The new contribution is an efficient method to solve the discretized heat equation at a prescribed time t, directly. The discretization of the heat equation leads to a huge system of ordinary differential equations

$$\mathbf{y}' = A\mathbf{y}, \qquad \mathbf{y}(0) = \mathbf{b}, \tag{1}$$

where $\mathbf{b}$ are the pixels of the compressed image written as a vector according to the chosen enumeration of the pixels. Here and in the following, we will not make a difference between the image data written as a vector or as an array of pixel values. It should be clear from the context which one is meant. The reconstructed image $\mathbf{y}(t)$ at time t is given by the matrix exponential times the vector $\mathbf{b}$, i.e., $\mathbf{y}(t) = e^{tA}\mathbf{b}$.

Recently, rational Krylov subspace methods have been found to be an excellent choice for the approximation of the matrix exponential, i.e., the solution of (1) at time t. Rational Krylov subspaces were first considered by Axel Ruhe [13, 14]. They also turned out to be useful in inverse problems [15–18]. If A is an operator or an arbitrarily large matrix with a field of values in the left complex half-plane, the matrix exponential times a vector can be approximated reliably for an arbitrary time $t > 0$ under reasonable assumptions on the vector [19–22]. If the matrix A is symmetric and the field-of-values is on the negative real axis, then rational Krylov methods are known that converge fast without any restrictions on the

vector [23, 24]. For finite subintervals of the negative real line, even super-linear convergence is obtained [25]. Rational Krylov approaches are a rich class of methods that might use different poles across the rational Krylov iterations [26, 27].

Our matrix A is not symmetric, but nevertheless allows for the use of a well-chosen rational Krylov subspace such that fast convergence is obtained. We will approximate the solution of system (1) in extended Krylov subspaces of the form

$$\mathcal{E}_m((\gamma I - A)^{-1}, \mathbf{b}) = \text{span}\{\mathbf{b}, A\mathbf{b}, (\gamma I - A)^{-1}\mathbf{b}, \cdots, (\gamma I - A)^{-m+2}\mathbf{b}\},$$

where $\gamma > 0$ [19, 28, 29]. After the computation of an orthonormal basis $V_m \in \mathbb{R}^{n \times m}$ of this space and the compression $S_m = V_m^T A V_m$ of the huge matrix A to a small $m \times m$ matrix, the *Krylov approximation* $\mathbf{f}_m$ is given by

$$\mathbf{f}_m = \|\mathbf{b}\| V_m e^{t S_m} \mathbf{e}_1, \qquad \mathbf{e}_1 = \left(1, 0, \cdots, 0\right)^T \in \mathbb{R}^m. \tag{2}$$

This way, the solution of the huge system (1) is reduced to the solution of a small system of size $m \times m$, i.e., to the computation of $e^{t S_m} \mathbf{e}_1$. For this purpose, methods for small matrices can be used [30, 31]. It will turn out that a small m is sufficient for arbitrarily large matrices A. The choice of the subspace that leads to this favorable property is intricate due to two restrictions. A good error estimate is necessary to estimate the accuracy of the method, and the computation of the vectors $(\gamma I - A)^{-m}\mathbf{b}$ requires the efficient solution of linear systems. This can be done by multigrid methods.

The paper is organized as follows: After the introduction in this section, the encoding of pictures is briefly discussed in Section 2. In Section 3, the discretization of the heat equation is described. The new decoding scheme and the main result—that it works independently of the size of the picture for a given large time t—are shown in Section 4. The multigrid method adapted to our purposes as an efficient method to solve the linear systems is discussed in Section 5. Numerical experiments with the decoding scheme as an illustration of our method are conducted in Section 6. Here and everywhere else, we will use pictures of the Kodak lossless color image suite [32]. The work closes with a brief conclusion as Section 7.

2 | Encoding

In this section, we briefly describe the encoding. The basic idea is to determine a binary mask $\mathbf{c}$ of the same size as the picture that indicates which of the gray/color values are stored. The choice of this mask determines the compression. The fewer pixels we have to store, the higher the compression will be. The choice of the mask is also important for the obtainable quality of the reconstructed image. We consider two basic methods to determine a good mask for a later inpainting of the picture. For simplicity, we will not consider the generation of more elaborate masks by advanced coding techniques [8, 9]. Our new decoding algorithm works with any mask $\mathbf{c}$. For the demonstration of our approach, the two basic masks will suffice.

2.1 | Edge Detection

Edges are very important for the perception of images by the human brain [33]. Therefore, one often starts with detecting edge information in an image. One classic and often used idea is to identify edges as zero-crossings of the Laplacian of an image that has been smoothed by a Gaussian filter, the Marr–Hildreth edge detector [34]. For a color picture, $\mathbf{f} = (f_1, f_2, f_3)^T$, the Laplacian is defined as the sum of the Laplacians over all channels. The modulus of the Laplacian is defined as the sum of the absolute values of the Laplacians over all channels. The idea is to store the pixels with a large modulus of the Laplacian. To remove zero-crossings that arise from small oscillations in the image, the magnitude of the gradient of the image at every pixel is used in addition. All edges are removed where the gradient is below a certain threshold. This is basically the idea of the Canny edge detector [35]. If one uses the above idea for natural images, the highly textured parts of the image are strongly emphasized in contrast to the background. This can be seen in the reconstruction (d) of Figure 2. The background seems to be too blurred.

2.2 | Dithering

To improve the representation of smoother regions, it is proposed to choose the edge data proportional to the absolute value of the Laplacian [36]. To follow this suggestion, we use Floyd–Steinberg dithering [7] for the modulus of the Laplacian. This method also allows for a simple method to prescribe the percentage of the pixels to be stored. For example, if

(a)

(b)

(c)

(d)

FIGURE 2 | On the top left is the original picture. The modulus of the Laplacian of the Gaussian-filtered original image is shown as a gray-scale picture on the top right. The binary mask on the bottom left has been chosen such that 10% of the pixels with the largest modulus of the Laplacian (white pixels) are contained. The reconstruction according to the mask can be seen on the bottom right. (a) Original, (b) modulus of Laplacian, (c) mask, (d) reconstruction.

one wishes to store 10% of the pixels, the largest modulus of the Laplacian in the picture is scaled such that the average corresponds to a tenth of the value of a white pixel. If the maximal value of a white pixel is 255, the average corresponds to $0.1 \times 255 = 25.5$. In the course of the Floyd–Steinberg dithering, about 10% of the pixels will automatically be stored.

In the following, we will refer to the first method as *edge-based* compression and the second one as *dithering-based* compression. Dithering improves the display of the background in the reconstructed image, as can be seen in Figure 3. The edge-based compression is superior for cartoon-like images, vector graphics, pictograms, and letters, where the edges are the most important image feature. The dithering might be superior for natural images, as shown above.

3 | Discretization

For decoding the compressed pictures, the $N_x \times N_y$ pixels of the original picture are interpreted as a finite-difference approximation to the heat equation. The mask **c** turns into a binary mask, where 1 indicates that the pixel has been stored. Every such pixel is treated as a discretized Dirichlet boundary condition with the pixel value as the boundary value. Any boundary pixel, which is not a stored pixel, is treated as a homogeneous Neumann boundary. The Laplacian is discretized by the standard stencil

$$\Delta^{\mathbf{h}} \; \hat{=} \; \begin{bmatrix} 0 & \frac{1}{h_x^2} & 0 \\ \frac{1}{h_y^2} & -\left(\frac{2}{h_x^2} + \frac{2}{h_y^2}\right) & \frac{1}{h_y^2} \\ 0 & \frac{1}{h_x^2} & 0 \end{bmatrix} \tag{3}$$
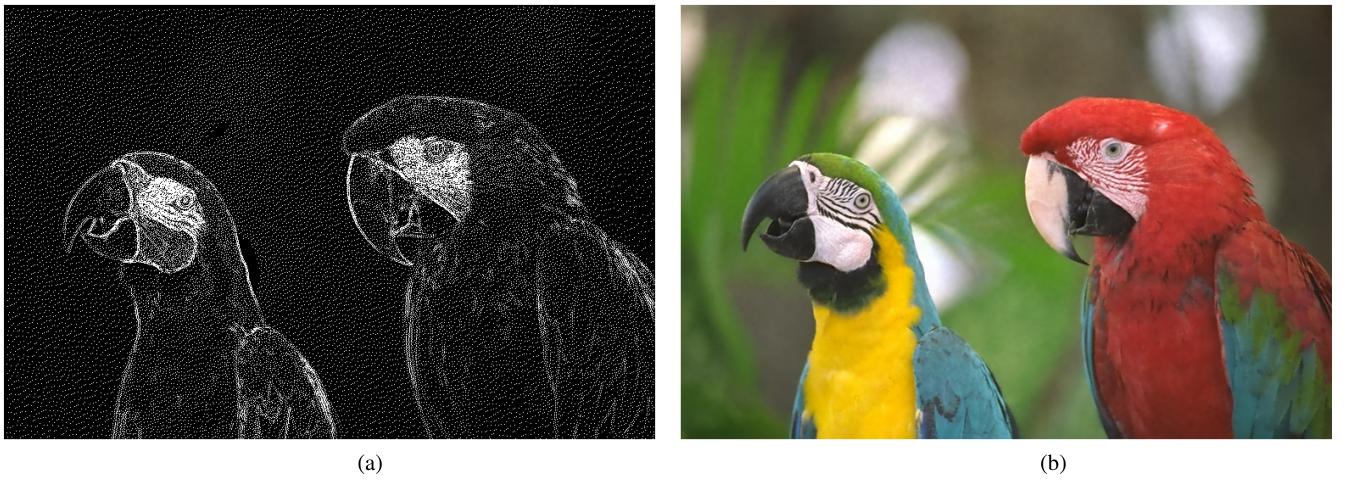
FIGURE 3 | The mask has been chosen such that the modulus of the Laplacian has been scaled to 10% and dithering has been applied. (a) Dithering (10%), (b) reconstruction.

FIGURE 4 | Example of image with dimension 3×2 and its row-major numbering.

with grid constants h_x, h_y in x, y-direction, respectively. The stencil is applied to every pixel that has not been stored. As usual in image processing, we will assume that the grid constants are one in both directions on the finest grid, which corresponds to the original picture. We illustrate the discretization by the small example in Figure 4. The gray pixels indicate the stored pixels, i.e., the Dirichlet boundary data, where the mask **c** is one. The discretized heat equation reads

$$\mathbf{y}' = A\mathbf{y}, \qquad \mathbf{y}(0) = \mathbf{b}, \tag{4}$$

where A and **b** are as follows,

$$A = \begin{bmatrix} -2 & 1 & 0 & 1 & 0 & 0 \\ 1 & -3 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & -2 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ f_3^h \\ 0 \\ f_5^h \\ 0 \end{bmatrix}.$$

The matrix R,

$$R = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{m \times n}, \qquad m \leq n,$$

selects, by multiplication from the left-hand side, the rows of the matrix A that correspond to inner pixels where the Laplacian stencil is applied. R^T blows a vector that corresponds to the inner pixels up to the full size of our picture while setting the boundary pixels to zero. The projector $P = R^T R$ projects to the orthogonal complement of the space spanned by **b**. As a consequence, $PA = A$. Furthermore, $RR^T = I_m$, where I_m designates the $m \times m$ identity matrix. With the help

of the matrices R and R^T, respectively, a symmetric matrix A_{sym} can be extracted from the matrix A as well as a reduced vector $\mathbf{b}_{\text{sym}}$,

$$
A_{\text{sym}} = RAR^T = \begin{bmatrix} -2 & 1 & 1 & 0 \\ 1 & -3 & 0 & 0 \\ 1 & 0 & -2 & 0 \\ 0 & 0 & 0 & -2 \end{bmatrix}, \qquad \mathbf{b}_{\text{sym}} = RA\mathbf{b},
$$

that allow for an alternative representation of the solution of system (4) given in Lemma 1.

Lemma 1. *The exact solution of Equation (4) can be written as*

$$
\mathbf{y}(t) = e^{tA}\mathbf{b} = \mathbf{b} + R^T\left(t\varphi_1(tA_{\text{sym}})\mathbf{b}_{\text{sym}}\right), \tag{5}
$$

where $\varphi_1(z) = (e^z - 1)/z$.

Proof. It is well known that the matrix exponential solves the ordinary differential equation (4). Hence, from here, we obtain

$$
\mathbf{y}(t) = e^{tA}\mathbf{b} = \mathbf{b} + t\varphi_1(tA)A\mathbf{b} = \mathbf{b} + t\varphi_1(tPA)PA\mathbf{b} = \mathbf{b} + R^T\left(t\varphi_1(tA_{\text{sym}})\mathbf{b}_{\text{sym}}\right),
$$

by $PA = A$ and the fact that

$$
\varphi_1(R^TRA)R^T = R^T\varphi_1(RAR^T)
$$

according to Corollary 1.34 on page 21 in Higham's book [31]. □

In the alternative representation (5) of the solution, one can see that the stored pixels in $\mathbf{b}$ are never altered due to the properties of R^T, which is also true for the exact solution of (4), of course. It follows from the Gershgorin disk theorem that A_{sym} has only negative eigenvalues and is therefore invertible. (Strictly speaking, A_{sym} has only negative eigenvalues as soon as at least one boundary pixel exists in the interior of the rectangular domain of the picture.) From the stencil, one can easily read off that the matrix A_{sym} is symmetric. Based on these facts, the following theorem shows that $(\gamma I - A)$ is invertible for all $\gamma > 0$, which is crucial for our decoding method.

Lemma 2. *For $\gamma > 0$, we have*

$$
(\gamma I - A)^{-1} = \frac{1}{\gamma}I + \frac{1}{\gamma}R^T(\gamma I - A_{\text{sym}})^{-1}RA.
$$

Proof. We compute

$$
(\gamma I - A)\left(\frac{1}{\gamma}I + \frac{1}{\gamma}R^T(\gamma I - A_{\text{sym}})^{-1}RA\right) = \frac{1}{\gamma}(\gamma I - A) + \frac{1}{\gamma}(\gamma I - A)R^T(\gamma I - A_{\text{sym}})^{-1}RA
$$

$$
= I - \frac{1}{\gamma}A + \frac{1}{\gamma}P(\gamma I - A)R^T(\gamma I - A_{\text{sym}})^{-1}RA = I - \frac{1}{\gamma}A + \frac{1}{\gamma}R^T(\gamma I - A_{\text{sym}})(\gamma I - A_{\text{sym}})^{-1}RA
$$

$$
= I - \frac{1}{\gamma}A + \frac{1}{\gamma}PA = I.
$$

□

4 | Decoding by the Extended Krylov Subspace Method

Extended Krylov subspaces for invertible matrices use the matrix as well as its inverse [19, 28, 29]. Let C be an invertible matrix and $\mathbf{b}$ a vector of a suitable dimension. Then the extended Krylov subspace $\mathcal{E}_p^q(C, \mathbf{b})$ is defined as

$$
\mathcal{E}_p^q(C, \mathbf{b}) = \text{span}\left\{\mathbf{b}, C^{-1}\mathbf{b}, \dots, C^{-q}\mathbf{b}, C\mathbf{b}, \dots, C^{p-1}\mathbf{b}\right\}.
$$

Due to Lemma 2, $\gamma I - A$ is invertible and hence we can set $C = (\gamma I - A)^{-1}$. We will use extended Krylov subspaces where q is always one of the following forms

$$
\mathcal{E}_m((\gamma I - A)^{-1}, \mathbf{b}) = \mathcal{E}_{m-1}^1((\gamma I - A)^{-1}, \mathbf{b}) = \text{span}\{\mathbf{b}, A\mathbf{b}, (\gamma I - A)^{-1}\mathbf{b}, \cdots, (\gamma I - A)^{-m+2}\mathbf{b}\}.
$$

Note, that we have used that span$\{\mathbf{b}, (\gamma I - A)\mathbf{b}\}$ = span$\{\mathbf{b}, A\mathbf{b}\}$, here. We start by computing an orthonormal basis $V_m \in \mathbb{R}^{n \times m}$ of the extended Krylov subspace by Algorithm 1. Then, we compute the compression $S_m = V_m^T A V_m \in \mathbb{R}^{m \times m}$ of the large matrix A, and finally the *Krylov approximation* $\mathbf{f}_m$ as given in Equation (2). To understand the properties of the Krylov algorithm, we study the algorithm via the symmetric matrix A_{sym}, the initial vector $\mathbf{b}_{\text{sym}}$, and the alternative solution representation (5).

The computation of the basis V_m by Algorithm 1 can be compared with the symmetric Lanczos method for a related Krylov subspace as outlined in Lemma 3.

Lemma 3. *Let* $W_{m-1} = [\mathbf{w}_1, \cdots, \mathbf{w}_{m-1}]$ *be the (symmetric) Lanczos basis for the Krylov subspace* $\mathcal{K}_{m-1}((\gamma I - A_{\text{sym}})^{-1}, \mathbf{b}_{\text{sym}})$ *according to Algorithm 2. Then, the orthonormal basis for the extended Krylov subspace*

$$\mathcal{E}_m((\gamma I - A)^{-1}, \mathbf{b}) = \text{span}\{\mathbf{b}, A\mathbf{b}, (\gamma I - A)^{-1}\mathbf{b}, \cdots, (\gamma I - A)^{-m+2}\mathbf{b}\}$$

according to Algorithm 1 reads

$$V_m = [\mathbf{v}_1, \ldots, \mathbf{v}_m] = [\mathbf{v}_1, R^T W_{m-1}], \qquad \mathbf{v}_1 = \frac{1}{\|\mathbf{b}\|}\mathbf{b}.$$

Proof. The idea of the following proof is to compare the (non-symmetric) Arnoldi-like algorithm with the (non-symmetric) Arnoldi algorithm (cf. algorithm 6.1 in Saad's book [37]) for the standard Krylov space $\mathcal{K}_{m-1}((\gamma I - A_{\text{sym}})^{-1}, \mathbf{b}_{\text{sym}})$. The (non-symmetric) Arnoldi-like algorithm and the (non-symmetric) Arnoldi algorithm are the same as the symmetric Lanczos-like Algorithm 1 and the symmetric Lanczos algorithm 2, respectively, with the difference that the for-loop j runs from $j = 0, \ldots, m$. In the (non-symmetric) Arnoldi-like algorithm, we compute h_{jm} for $j = 1, \ldots, m$, which guarantees that the generated vectors are perpendicular to each other. The (non-symmetric) Arnoldi-like algorithm leads to the following. Let $\mathbf{v}_1 = \frac{1}{\|\mathbf{b}\|}\mathbf{b}$, which is the obvious start. Then

---

ALGORITHM 1 | Symmetric Lanczos-like algorithm.

Set $\mathbf{v}_1 = \mathbf{b}/\|\mathbf{b}\|$, $\mathbf{v}_0 = \mathbf{0}$
for m = 1,2,3,... **do**
    if $m = 1$ **then**
        $\mathbf{u} = A\mathbf{v}_1$
    else
        $\mathbf{u} = (\gamma I - A)^{-1}\mathbf{v}_m$
    end if
    for $j = m - 1, m$ **do**
        $h_{jm} = (\mathbf{u}, \mathbf{v}_j)$
    end for
    $\tilde{\mathbf{v}}_{m+1} = \mathbf{u} - h_{mm}\mathbf{v}_m - h_{m-1,m}\mathbf{v}_{m-1}$
    $\mathbf{v}_{m+1} = \tilde{\mathbf{v}}_{m+1}/h_{m+1,m}$, $h_{m+1,m} = \|\tilde{\mathbf{v}}_{m+1}\|$
end for

ALGORITHM 2 | Symmetric Lanczos.

Set $\mathbf{w}_1 = \mathbf{b}_{\text{sym}}/\|\mathbf{b}_{\text{sym}}\|$, $\mathbf{w}_0 = \mathbf{0}$
for m = 1,2,3,... **do**
    $\mathbf{u} = (\gamma I - A_{\text{sym}})^{-1}\mathbf{w}_m$
    for $j = m - 1, m$ **do**
        $\widehat{h}_{jm} = (\mathbf{u}, \mathbf{w}_j)$
    end for
    $\tilde{\mathbf{w}}_{m+1} = \mathbf{u} - \widehat{h}_{mm}\mathbf{w}_m - \widehat{h}_{m-1,m}\mathbf{w}_{m-1}$
    $\mathbf{w}_{m+1} = \tilde{\mathbf{w}}_{m+1}/\widehat{h}_{m+1,m}$, $\widehat{h}_{m+1,m} = \|\tilde{\mathbf{w}}_{m+1}\|$
end for

$$A\mathbf{v}_1 = \frac{1}{\|\mathbf{b}\|}A\mathbf{b} = \frac{1}{\|\mathbf{b}\|}R^T\mathbf{b}_{\text{sym}},$$

$$h_{11} = \mathbf{v}_1^T A\mathbf{v}_1 = \frac{1}{\|\mathbf{b}\|^2}\mathbf{b}^T A\mathbf{b} = \frac{1}{\|\mathbf{b}\|^2}\mathbf{b}^T P A\mathbf{b} = 0,$$

$$\tilde{\mathbf{v}}_2 = A\mathbf{v}_1 - h_{1,1}\mathbf{v}_1 = \frac{1}{\|\mathbf{b}\|}R^T\mathbf{b}_{\text{sym}},$$

$$h_{21} = \|\tilde{\mathbf{v}}_2\| = \frac{\|\mathbf{b}_{\text{sym}}\|}{\|\mathbf{b}\|},$$

$$\mathbf{v}_2 = \frac{1}{h_{21}}\tilde{\mathbf{v}}_2 = R^T\frac{1}{\|\mathbf{b}_{\text{sym}}\|}\mathbf{b}_{\text{sym}} = R^T\mathbf{w}_1.$$

Our statement is proved for $m = 2$. For $m \geq 3$, we have

$$h_{1m} = \mathbf{v}_1^T(\gamma I - A)^{-1}\mathbf{v}_m = \mathbf{v}_1^T\left(\frac{1}{\gamma}I + \frac{1}{\gamma}R^T(\gamma I - A_{\text{sym}})^{-1}RA\right)\mathbf{v}_m = \frac{1}{\gamma}\mathbf{v}_1^T\mathbf{v}_m + \frac{1}{\gamma}(R\mathbf{v}_1)^T(\gamma I - A_{\text{sym}})^{-1}RA\mathbf{v}_m = 0,$$

since $\mathbf{v}_1^T\mathbf{v}_m = 0$ and $R\mathbf{v}_1 = \mathbf{0}$. For $j = 2, \ldots, m$, we obtain

$$h_{jm} = \mathbf{v}_j^T(\gamma I - A)^{-1}\mathbf{v}_m = \mathbf{v}_j^T\left(\frac{1}{\gamma}I + \frac{1}{\gamma}R^T(\gamma I - A_{\text{sym}})^{-1}RA\right)\mathbf{v}_m = \frac{1}{\gamma}\mathbf{v}_j^T\mathbf{v}_m + \frac{1}{\gamma}(R\mathbf{v}_j)^T(\gamma I - A_{\text{sym}})^{-1}RA\mathbf{v}_m$$

$$= \frac{1}{\gamma}(R^T\mathbf{w}_{j-1})^T R^T\mathbf{w}_{m-1} + \frac{1}{\gamma}(RR^T\mathbf{w}_{j-1})^T(\gamma I - A_{\text{sym}})^{-1}RAR^T\mathbf{w}_{m-1} = \frac{1}{\gamma}\mathbf{w}_{j-1}^T\mathbf{w}_{m-1} + \frac{1}{\gamma}\mathbf{w}_{j-1}^T(\gamma I - A_{\text{sym}})^{-1}A_{\text{sym}}\mathbf{w}_{m-1}.$$

With the help of the relation

$$(\gamma I - A_{\text{sym}})^{-1}A_{\text{sym}} = -I + \gamma(\gamma I - A_{\text{sym}})^{-1}, \tag{6}$$

we conclude

$$h_{jm} = \frac{1}{\gamma}\mathbf{w}_{j-1}^T\mathbf{w}_{m-1} + \frac{1}{\gamma}\mathbf{w}_{j-1}^T\left(-I + \gamma(\gamma I - A_{\text{sym}})^{-1}\right)\mathbf{w}_{m-1} = \mathbf{w}_{j-1}^T(\gamma I - A_{\text{sym}})^{-1}\mathbf{w}_{m-1} = \hat{h}_{m-1,j-1}, \quad j = 2, \ldots, m.$$

By (6), we also have

$$(\gamma I - A)^{-1}\mathbf{v}_m = \left(\frac{1}{\gamma}I + \frac{1}{\gamma}R^T(\gamma I - A_{\text{sym}})^{-1}RA\right)\mathbf{v}_m = R^T\left(\frac{1}{\gamma}I + \frac{1}{\gamma}(\gamma I - A_{\text{sym}})^{-1}A_{\text{sym}}\right)\mathbf{w}_{m-1} = R^T(\gamma I - A_{\text{sym}})^{-1}\mathbf{w}_{m-1}.$$

Finally, we find

$$\tilde{\mathbf{v}}_{m+1} = (\gamma I - A)^{-1}\mathbf{v}_m - \sum_{j=2}^{m}h_{jm}\mathbf{v}_j - h_{1m}\mathbf{v}_1 = R^T(\gamma I - A_{\text{sym}})^{-1}\mathbf{w}_{m-1} - \sum_{j=2}^{m}\hat{h}_{j-1,m-1}R^T\mathbf{w}_{j-1}$$

$$= R^T\left((\gamma I - A_{\text{sym}})^{-1}\mathbf{w}_{m-1} - \sum_{j=1}^{m-1}\hat{h}_{j,m-1}\mathbf{w}_j\right) = R^T\tilde{\mathbf{w}}_m$$

as well as $h_{m+1,m} = \|\tilde{\mathbf{v}}_{m+1}\| = \|\tilde{\mathbf{w}}_m\| = \hat{h}_{m,m-1}$ and

$$\mathbf{v}_{m+1} = \frac{\tilde{\mathbf{v}}_{m+1}}{h_{m+1,m}} = R^T\frac{\tilde{\mathbf{w}}_m}{\hat{h}_{m,m-1}} = R^T\mathbf{w}_m.$$

We have now proved the statement of our lemma for the (non-symmetric) Arnoldi algorithms. Since the matrix $(\gamma I - A_{\text{sym}})^{-1}$ is symmetric, the (non-symmetric) Arnoldi algorithm automatically reduces to the symmetric Lanczos algorithm (cf. section 6.6 of Saad's book [37]). Hence, $\hat{h}_{j-1,m-1} = 0$ for $j - 1 \leq m - 3$, and we obtain $h_{jm} = 0$ for $j \leq m - 2$. □

The first m steps of Algorithm 1 and the first $m - 1$ steps of Algorithm 2 can be written compactly as

$$\left[A\mathbf{v}_1, (\gamma I - A)^{-1}[\mathbf{v}_2, \cdots, \mathbf{v}_m]\right] = [\mathbf{v}_1, \cdots, \mathbf{v}_{m+1}]\underline{H}_m \quad \text{or} \quad (\gamma I - A_{\text{sym}})^{-1}[\mathbf{w}_1, \cdots, \mathbf{w}_{m-1}] = [\mathbf{w}_1, \cdots, \mathbf{w}_m]\underline{\hat{H}}_{m-1}, \tag{7}$$

respectively, where H_m is of dimension $(m+1) \times m$ and $\widehat{H}_{m-1}$ is of dimension $m \times (m-1)$. Here and elsewhere, we assume that the algorithm did not break down. In case of a breakdown, the breakdown is lucky. The exact solution of the heat equation is available in this case. Both matrices, H_m and $\widehat{H}_{m-1}$, are unreduced upper Hessenberg matrices that contain the values computed in the algorithms and are otherwise set to zero. Hence, more exactly, both matrices are tridiagonal due to Lemma 3. With $\underline{H}_m$ and $\widehat{H}_{m-1}$, we designate the matrices with dimensions $m \times m$ and $(m-1) \times (m-1)$, respectively. We then have

$$\underline{H}_m = \begin{bmatrix} 0 & 0 \\ \tilde{\mathbf{u}} & \widehat{H}_{m-1} \end{bmatrix}, \quad \tilde{\mathbf{u}} = h_{21}\mathbf{e}_1 = \frac{\|\mathbf{b}_{\mathrm{sym}}\|}{\|\mathbf{b}\|}\mathbf{e}_1, \quad \text{and} \quad \widehat{H}_{m-1}^T = \widehat{H}_{m-1}.$$

The following lemma relates the compression $S_m = V_m^T A V_m$ of A in the extended Krylov subspace to the compression $\widetilde{S}_{m-1} = W_{m-1}^T A_{\mathrm{sym}} W_{m-1}$ of A_{sym} in the Krylov space $\mathcal{K}_{m-1}((\gamma I - A_{\mathrm{sym}})^{-1}, \mathbf{b}_{\mathrm{sym}})$.

Lemma 4. *Let $\widetilde{S}_{m-1} = W_{m-1}^T A_{\mathrm{sym}} W_{m-1}$. Then, for $S_m = V_m^T A V_m$, one can find*

$$S_m = \begin{bmatrix} 0 & 0 \\ \tilde{\mathbf{u}} & \tilde{S}_{m-1} \end{bmatrix}, \quad \tilde{\mathbf{u}} = \frac{1}{\|\mathbf{b}\|}W_{m-1}^T\mathbf{b}_{\mathrm{sym}} = \frac{\|\mathbf{b}_{\mathrm{sym}}\|}{\|\mathbf{b}\|}\mathbf{e}_1,$$

with $\mathbf{e}_1 \in \mathbb{R}^{m-1}$, and therefore, $\gamma I - S_m$ is invertible for all $\gamma > 0$.

Proof. By Lemma 3, one obtains

$$S_m = V_m^T A V_m = \begin{bmatrix} \frac{1}{\|\mathbf{b}\|}\mathbf{b} & R^T W_{m-1} \end{bmatrix}^T A \begin{bmatrix} \frac{1}{\|\mathbf{b}\|}\mathbf{b} & R^T W_{m-1} \end{bmatrix} = \begin{bmatrix} \frac{1}{\|\mathbf{b}\|}\mathbf{b}^T \\ W_{m-1}^T R \end{bmatrix} \begin{bmatrix} \frac{1}{\|\mathbf{b}\|}A\mathbf{b} & AR^T W_{m-1} \end{bmatrix} = \begin{bmatrix} \frac{1}{\|\mathbf{b}\|^2}\mathbf{b}^T A\mathbf{b} & \frac{1}{\|\mathbf{b}\|}\mathbf{b}^T AR^T W_{m-1} \\ \frac{1}{\|\mathbf{b}\|}W_{m-1}^T RA\mathbf{b} & W_{m-1}^T RAR^T W_{m-1} \end{bmatrix}.$$

Since for any $\mathbf{v} \in \mathbb{R}^n$, $A\mathbf{v}$ has zeros where $\mathbf{b}$ has entries and vice versa, $\mathbf{b}^T A\mathbf{b} = 0$ and $\frac{1}{\|\mathbf{b}\|}\mathbf{b}^T AR^T W_{m-1} = \mathbf{0}$ is a zero row vector of length $m-1$. Furthermore,

$$\tilde{\mathbf{u}} = \frac{1}{\|\mathbf{b}\|}W_{m-1}^T RA\mathbf{b} = \frac{1}{\|\mathbf{b}\|}W_{m-1}^T\mathbf{b}_{\mathrm{sym}} = \frac{\|\mathbf{b}_{\mathrm{sym}}\|}{\|\mathbf{b}\|}W_{m-1}^T\mathbf{w}_1 = \frac{\|\mathbf{b}_{\mathrm{sym}}\|}{\|\mathbf{b}\|}\mathbf{e}_1$$

and

$$\tilde{S}_{m-1} = W_{m-1}^T RAR^T W_{m-1} = W_{m-1}^T A_{\mathrm{sym}} W_{m-1}.$$

Hence

$$\gamma I - S_m = \begin{bmatrix} \gamma & 0 \\ -\tilde{\mathbf{u}} & \gamma I - \tilde{S}_{m-1} \end{bmatrix}$$

is invertible, since $\gamma > 0$ and $\tilde{S}_{m-1}$ is a symmetric, negative-definite matrix, by Lemma 5, and $\gamma I - \tilde{S}_{m-1}$ therefore is a symmetric positive-definite matrix. $\square$

Lemma 5. *$\tilde{S}_{m-1}$ is a symmetric and negative-definite matrix.*

Proof. The symmetry follows directly from the symmetry of A_{sym}:

$$\tilde{S}_{m-1}^T = \left(W_{m-1}^T A_{\mathrm{sym}} W_{m-1}\right)^T = W_{m-1}^T A_{\mathrm{sym}}^T W_{m-1} = W_{m-1}^T A_{\mathrm{sym}} W_{m-1} = \tilde{S}_{m-1}.$$

Since A_{sym} is symmetric and has only negative eigenvalues, the field-of-values of A_{sym} is given as

$$F(A_{\mathrm{sym}}) = [\lambda_{\mathrm{min}}, \lambda_{\mathrm{max}}] \subset (-\infty, 0),$$

where

$$\lambda_{\mathrm{min}} = \min_{\lambda \in \sigma(A_{\mathrm{sym}})} \lambda, \qquad \lambda_{\mathrm{max}} = \max_{\lambda \in \sigma(A_{\mathrm{sym}})} \lambda,$$

and $\sigma(A_{\text{sym}})$ is the set of eigenvalues. Hence, for the field-of-values of $\tilde{S}_{m-1}$,

$$F(\tilde{S}_{m-1}) = \left\{ \frac{\mathbf{y}^H \tilde{S}_{m-1} \mathbf{y}}{\mathbf{y}^H \mathbf{y}} \;\middle|\; 0 \neq \mathbf{y} \in \mathbb{C}^{m-1} \right\} \subseteq F(A_{\text{sym}}) = [\lambda_{\min}, \lambda_{\max}],$$

since

$$\frac{\mathbf{y}^H W_{m-1}^H A_{\text{sym}} W_{m-1} \mathbf{y}}{\mathbf{y}^H \mathbf{y}} = \frac{\mathbf{y}^H W_{m-1}^H A_{\text{sym}} W_{m-1} \mathbf{y}}{\mathbf{y}^H W_{m-1}^H W_{m-1} \mathbf{y}} = \frac{\mathbf{x}^H A_{\text{sym}} \mathbf{x}}{\mathbf{x}^H \mathbf{x}}, \qquad \mathbf{x} = W_{m-1} \mathbf{y}.$$

Hence, since $\sigma(\tilde{S}_{m-1}) \subseteq F(\tilde{S}_{m-1})$, all eigenvalues of $\tilde{S}_{m-1}$ are negative and $\tilde{S}_{m-1}$ is negative definite. $\qquad\square$

Lemma 6. *Let $S_m = V_m^T A V_m$ be the compression of A. With $E = [\mathbf{0}, I_{m-1}] \in \mathbb{R}^{(m-1)\times m}$, let $\widehat{H}_{m-1} = E H_m E^T$. Then $\widehat{H}_{m-1}$ is invertible and one may compute the compression S_m by the values computed in Algorithm 1 as follows*

$$S_m = \begin{bmatrix} 0 & 0 \\ \tilde{\mathbf{u}} & \tilde{S}_{m-1} \end{bmatrix}, \quad \tilde{\mathbf{u}} = \frac{\|\mathbf{b}_{\text{sym}}\|}{\|\mathbf{b}\|}\mathbf{e}_1 \in \mathbb{R}^{m-1}, \quad \text{and} \quad \tilde{S}_{m-1} = \gamma I - \widehat{H}_{m-1}^{-1} + h_{m+1,m}^2(\mathbf{v}_{m+1}^T A \mathbf{v}_{m+1} - \gamma)\widehat{H}_{m-1}^{-1}\mathbf{e}_{m-1}\mathbf{e}_{m-1}^T \widehat{H}_{m-1}^{-1}.$$

Proof. According to Lemma 4, the structure is as stated in this lemma, where

$$\tilde{S}_{m-1} = W_{m-1}^T A_{\text{sym}} W_{m-1}.$$

By formula (5.8) in Grimm [21], we obtain that $\widehat{H}_{m-1}$ in the relation on the right-hand side of (7) is invertible as well as

$$\tilde{S}_{m-1} = W_{m-1}^T A_{\text{sym}} W_{m-1} = \gamma I - \widehat{H}_{m-1}^{-1} + \widehat{h}_{m,m-1}^2(\mathbf{w}_m^T A_{\text{sym}} \mathbf{w}_m - \gamma)\widehat{H}_{m-1}^{-1}\mathbf{e}_{m-1}\mathbf{e}_{m-1}^T \widehat{H}_{m-1}^{-1}.$$

As outlined in the discussion following formula (7), the values H_m computed in Algorithm 1 relate to the ones computed by Algorithm 2, collected in $\widehat{H}_{m-1}$, as $\widehat{H}_{m-1} = E H_m E^T$. Therefore, $\widehat{h}_{m,m-1} = h_{m+1,m}$ and $\mathbf{v}_{m+1}^T A \mathbf{v}_{m+1} = \mathbf{w}_m^T R A R^T \mathbf{w}_m = \mathbf{w}_m^T A_{\text{sym}} \mathbf{w}_m$ conclude the proof of our lemma. $\qquad\square$

The following theorem states that the boundary pixels are correctly set in the first Krylov step and not altered afterwards due to the properties of the matrix R^T. An alternative representation of the Krylov approximation is given with the help of the φ_1-function analogous to Lemma 1.

Theorem 1. *The Krylov approximation to the matrix exponential times vector, $e^{tA}\mathbf{b}$, in the Krylov subspace $\mathcal{E}_m((\gamma I - A)^{-1}, \mathbf{b})$ reads*

$$\|\mathbf{b}\| V_m e^{tS_m}\mathbf{e}_1 = \mathbf{b} + R^T\left(\|\mathbf{b}_{\text{sym}}\| W_{m-1} t\varphi_1(t\tilde{S}_{m-1})\mathbf{e}_1\right).$$

Proof. With the help of Lemma 4, one obtains

$$e^{tS_m} = \sum_{k=0}^{\infty} \frac{t^k}{k!} S_m^k = \sum_{k=0}^{\infty} \frac{t^k}{k!}\begin{pmatrix} 0 & 0 \\ \tilde{\mathbf{u}} & \tilde{S}_{m-1} \end{pmatrix}^k = I + \sum_{k=1}^{\infty} \frac{t^k}{k!}\begin{pmatrix} 0 & 0 \\ \tilde{S}_{m-1}^{k-1}\tilde{\mathbf{u}} & \tilde{S}_{m-1}^k \end{pmatrix},$$

and hence

$$e^{tS_m}\mathbf{e}_1 = \mathbf{e}_1 + \sum_{k=1}^{\infty} \frac{t^k}{k!}\begin{pmatrix} 0 & 0 \\ \tilde{S}_{m-1}^{k-1}\tilde{\mathbf{u}} & \tilde{S}_{m-1}^k \end{pmatrix}\mathbf{e}_1 = \mathbf{e}_1 + \begin{bmatrix} 0 \\ \sum_{k=1}^{\infty}\frac{t^k}{k!}\tilde{S}_{m-1}^{k-1}\tilde{\mathbf{u}} \end{bmatrix} = \begin{bmatrix} 1 \\ t\varphi_1(t\tilde{S}_{m-1})\tilde{\mathbf{u}} \end{bmatrix} = \begin{bmatrix} 1 \\ \frac{\|\mathbf{b}_{\text{sym}}\|}{\|\mathbf{b}\|}t\varphi_1(t\tilde{S}_{m-1})\mathbf{e}_1 \end{bmatrix}.$$

Finally, one obtains

$$\|\mathbf{b}\| V_m e^{tS_m}\mathbf{e}_1 = \|\mathbf{b}\|\begin{bmatrix} \mathbf{v}_1 & R^T W_{m-1} \end{bmatrix}\begin{bmatrix} 1 \\ \frac{\|\mathbf{b}_{\text{sym}}\|}{\|\mathbf{b}\|}t\varphi_1(t\tilde{S}_{m-1})\mathbf{e}_1 \end{bmatrix} = \|\mathbf{b}\|\mathbf{v}_1 + \|\mathbf{b}\| R^T W_{m-1}\left(\frac{\|\mathbf{b}_{\text{sym}}\|}{\|\mathbf{b}\|}t\varphi_1(t\tilde{S}_{m-1})\mathbf{e}_1\right)$$

$$= \mathbf{b} + R^T\left(\|\mathbf{b}_{\text{sym}}\| W_{m-1} t\varphi_1(t\tilde{S}_{m-1})\mathbf{e}_1\right).$$

$\qquad\square$

The comparison of the exact solution with the Krylov approximation via the alternative representation by the φ_1-function leads to the error bound in Theorem 2. The φ_1-function can be approximated uniformly for matrices/operators with field-of-values in the left half-plane [21]. Here we obtain even better bounds due to the symmetry of the matrix at which the function is evaluated [24, 38, 39].

Theorem 2. *The error of the Krylov approximation* $\mathbf{f}_m = \|\mathbf{b}\| V_m e^{tS_m} \mathbf{e}_1$ *in the extended Krylov subspace* $\mathcal{E}_m((\gamma I - tA)^{-1}, \mathbf{b})$ *with* $\gamma > 0$ *to the solution of (1) reads, for* $m \geq 2$ *and* $t > 0$,

$$\|e^{tA}\mathbf{b} - \mathbf{f}_m\| \leq 2t E_m(\gamma)\|\mathbf{b}_{\mathrm{sym}}\|,$$

with

$$E_m(\gamma) = \min_{r \in \mathcal{R}_{m-1}} \|\varphi_1 - r\|_{(-\infty,0]},$$

where $\| \cdot \|_{(-\infty,0]}$ *designates the supremum norm on* $(-\infty, 0]$ *and the space*

$$\mathcal{R}_{m-1} = \left\{ \frac{p_{m-2}}{(\gamma - \cdot)^{m-2}} \mid p_{m-2} \in \mathcal{P}_{m-2} \right\}$$

is the space of rational functions of the indicated form and dimension $m - 1$. *Here,* $\mathcal{P}_{m-2}$ *is the space of polynomials with degree less than or equal to* $m - 2$.

Proof. With Lemma 1 and Theorem 1, one obtains

$$e^{tA}\mathbf{b} - \mathbf{f}_m = tR^T\left(\varphi_1(tA_{\mathrm{sym}})\mathbf{b}_{\mathrm{sym}} - \|\mathbf{b}_{\mathrm{sym}}\| W_{m-1}\varphi_1(t\tilde{S}_{m-1})\mathbf{e}_1\right).$$

Hence, by the exactness property in the Krylov space $\mathcal{K}_{m-1}((\gamma I - tA_{\mathrm{sym}})^{-1}, \mathbf{b}_{\mathrm{sym}})$, namely

$$r(tA_{\mathrm{sym}})\mathbf{b}_{\mathrm{sym}} = \|\mathbf{b}_{\mathrm{sym}}\| W_{m-1}r(t\tilde{S}_{m-1})\mathbf{e}_1$$

for all $r \in \mathcal{R}_{m-1}$, one obtains

$$\|e^{tA}\mathbf{b} - \mathbf{f}_m\| \leq t\|\left(\varphi_1(tA_{\mathrm{sym}}) - r(tA_{\mathrm{sym}})\right)\mathbf{b}_{\mathrm{sym}}\| + t\|\mathbf{b}_{\mathrm{sym}}\|\|\left(r(t\tilde{S}_{m-1}) - \varphi_1(t\tilde{S}_{m-1})\right)\mathbf{e}_1\|$$

$$\leq 2t\|\mathbf{b}_{\mathrm{sym}}\| \cdot \min_{r \in \mathcal{R}_{m-1}} \max_{z \in (-\infty,0]} |\varphi_1(z) - r(z)|.$$

$\square$

Note that $E_m(\gamma)$ depends neither on the chosen $t > 0$ nor on the size of the matrix A. With the same ideas as in van den Eshof and Hochbruck [24], Table 1 of optimal values of γ with respect to the minimization of the error can be numerically computed with the help of a simple transform and the Remez algorithm. Note that there is a subtle issue about scaling. One has to use the Krylov subspace as given in Theorem 2. Alternatively, one might use the space $\mathcal{E}_m((\tilde{\gamma}I - A)^{-1}, \mathbf{b})$ with $\tilde{\gamma} = \gamma_{\mathrm{opt}}/t$ for the simple reason that $\mathcal{E}_m((\gamma I - tA)^{-1}, \mathbf{b}) = \mathcal{E}_m((\frac{\gamma}{t}I - A)^{-1}, \mathbf{b})$. We refer the reader to van den Eshof and Hochbruck [24] for details.

We illustrate the bounds and the necessity of the scaling numerically. We use an all-white square gray-scale picture of size 1024×1024. The Canny-like edge detector then produces the mask with all boundary pixels set to one and all interior points set to zero. That is, the compressed picture has a white boundary and all pixels in the interior are black. For this simple example, the solution of the inpainting by the heat equation can be computed at any time $t > 0$ by fast transforms. In Figure 5, we show the error bound of Theorem 1 with the optimal choices of γ according to Table 1 as a black solid line and the error of the approximation with respect to the Krylov subspace $\mathcal{E}_m((\tilde{\gamma}I - A)^{-1}, \mathbf{b})$ with $\tilde{\gamma} = \gamma_{\mathrm{opt}}/t$ as in the theorem as green circle-marked line for $t = 25$, $t = 10^2$, and $t = 10^4$, respectively. The red diamond-marked line corresponds to the extended Krylov subspace $\mathcal{E}_m((I - A)^{-1}, \mathbf{b})$. That is, γ is set to one and not scaled. For $t = 25$, the approximation for small dimensions of the space with γ fixed to one is clearly worse than the error bound, in contrast to the properly scaled Krylov subspace. For $t = 10^2$, and $t = 10^4$, the approximation of the space with γ fixed to one does not improve for larger dimensions of the Krylov subspace, either.

Since the eye can not distinguish very small deviations, the error bound in Theorem 1 and our experiment suggest that an extended Krylov subspace of a small dimension and with the optimal choice of γ is sufficient for image processing purposes. This will be confirmed in Section 6 by numerical experiments with real pictures.

TABLE 1 | Numerical approximation to the optimal value of γ, γ_{opt}, and the error $E_m(\gamma_{\text{opt}})$ for the optimal choice of γ, dimension m of the extended Krylov subspace, and #*lss* linear system solves by the multigrid method.

| #*lss*	m	$E_m(\gamma_{\text{opt}})$	γ_{opt}	#*lss*	m	$E_m(\gamma_{\text{opt}})$	γ_{opt}
1	3	$2.6 \cdot 10^{-2}$	1.5	11	13	$5.3 \cdot 10^{-7}$	8.5
2	4	$6.6 \cdot 10^{-3}$	3.5	12	14	$1.8 \cdot 10^{-7}$	10
3	5	$2.2 \cdot 10^{-3}$	5.5	13	15	$5.7 \cdot 10^{-8}$	11.5
4	6	$6.9 \cdot 10^{-4}$	3.5	14	16	$2.5 \cdot 10^{-8}$	10
5	7	$2.0 \cdot 10^{-4}$	5	15	17	$8.6 \cdot 10^{-9}$	11.5
6	8	$8.9 \cdot 10^{-5}$	7	16	18	$3.1 \cdot 10^{-9}$	13
7	9	$2.8 \cdot 10^{-5}$	8.5	17	19	$1.3 \cdot 10^{-9}$	11.5
8	10	$1.0 \cdot 10^{-5}$	6.5	18	20	$4.8 \cdot 10^{-10}$	13
9	11	$3.8 \cdot 10^{-6}$	8.5	19	21	$1.9 \cdot 10^{-10}$	14.5
10	12	$1.1 \cdot 10^{-6}$	10	20	22	$8.3 \cdot 10^{-11}$	16
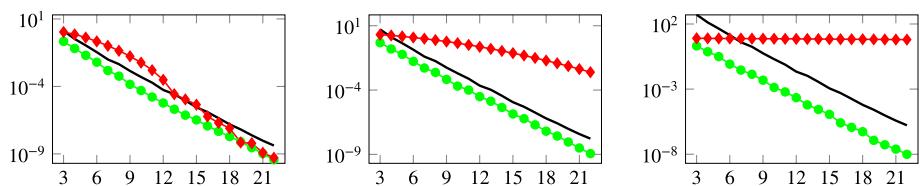
FIGURE 5 | Error bound of Theorem 1 (black) solid line for the optimal choice of γ, error of the extended Krylov subspace approximation to the compressed all-white picture for the optimal γ (green) circle-marked line, and error of the extended Krylov subspace with γ fixed to one (red) diamond-marked line vs. dimension of the Krylov subspaces for $t = 25$ on left-hand side, $t = 10^2$ in the middle, $t = 10^4$ on the right-hand side.

5 | Implementation Details

For the efficient computation of $\mathbf{v}^{\mathbf{h}} = (\gamma I - A^{\mathbf{h}})^{-1}\mathbf{b}^{\mathbf{h}}$, we use the multigrid method applied to the system

$$B^{\mathbf{h}}\mathbf{v}^{\mathbf{h}} = \mathbf{b}^{\mathbf{h}}, \qquad B^{\mathbf{h}} = (\gamma I - A^{\mathbf{h}}).$$

The superscript $\mathbf{h}$ indicates that the matrices and vectors belong to the finest grid, which corresponds to the original image. To efficiently implement the multigrid method, we operate on discrete images [6, 40]. For the application of the multigrid method, we look at a fine grid $\Omega^{\mathbf{h}}$ with $N^{\mathbf{h}} = N_x^{\mathbf{h}} \times N_y^{\mathbf{h}}$ pixels, where $N_x^{\mathbf{h}}$ and $N_y^{\mathbf{h}}$ correspond to the number of pixels in x- and y-direction, respectively. The grid spacing is denoted by $\mathbf{h} = (h_x, h_y)^T$. On the finest grid $(h_x, h_y)^T = (1, 1)^T$, which is a popular choice in image processing. For the next coarser grid, one would like to double the grid spacing in both directions. This is only possible for powers of two. To include other grids, we define the coarser grid $\Omega^{\mathbf{H}}$ with the spacings $\mathbf{H} = (H_x, H_y)^T$, with

$$H_x = h_x \frac{N_x^{\mathbf{h}}}{N_x^{\mathbf{H}}}, \qquad H_y = h_y \frac{N_y^{\mathbf{h}}}{N_y^{\mathbf{H}}},$$

where $N_x^{\mathbf{H}} = \lceil N_x^{\mathbf{h}}/2 \rceil$ and $N_y^{\mathbf{H}} = \lceil N_y^{\mathbf{h}}/2 \rceil$ are the number of pixels in each direction in the coarse grid, where $\lceil \cdot \rceil$ denotes the ceiling function. For the restriction, the coarse pixel is the average of the fine pixels according to the area of the fine pixel that contributes to the coarse pixel. The prolongation reverses this process. For the restriction matrix $I_{\mathbf{h}}^{\mathbf{H}}$ and the prolongation matrix $I_{\mathbf{H}}^{\mathbf{h}}$, one has the relation

$$I_{\mathbf{h}}^{\mathbf{H}} = \alpha(I_{\mathbf{H}}^{\mathbf{h}})^T, \qquad \alpha = \frac{N_x^{\mathbf{H}} N_y^{\mathbf{H}}}{N_x^{\mathbf{h}} N_y^{\mathbf{h}}} = \frac{h_x h_y}{H_x H_y}.$$

Restriction and prolongation are illustrated in Figure 6.
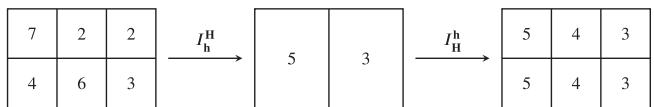
**FIGURE 6** | Example for the restriction and prolongation.
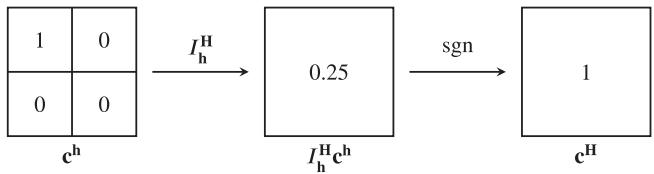
FIGURE 7 | Example for the restriction applied to the inpainting mask.
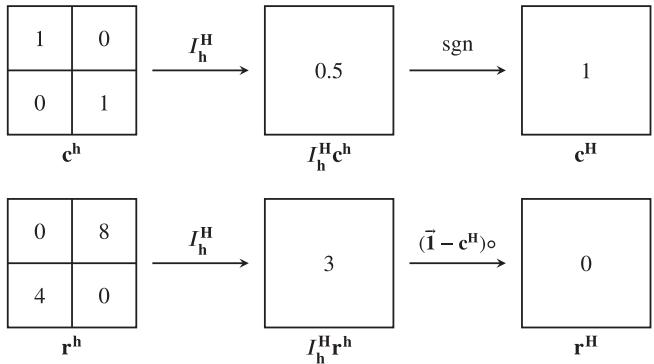

FIGURE 8 | Computation of the coarse residual $\mathbf{r^H}$. As for the fine residual, the coarse residual is set to zero for known pixels.

Since we have two different sorts of pixels, we also have to apply the restriction and prolongation to our binary inpainting mask $\mathbf{c}$. We therefore adapt the restriction of the inpainting by applying the element-wise sign function to the restricted inpainting mask

$$\mathbf{c^H} = \text{sgn}(I_\mathbf{h}^\mathbf{H}\mathbf{c^h}).$$

This is also illustrated in Figure 7.

With these two definitions, we obtain a natural definition of the coarse matrix $A^\mathbf{H}$. We can simply use the standard stencil for the Laplace operator with respect to the grid spacing $\mathbf{H} = (H_x, H_y)^T$. For the multigrid method, we also need to compute the restriction of the fine residual $\mathbf{r^h}$. With the Hadamard product $\circ$, the restriction of the fine residual to the coarse residual is

$$\mathbf{r^H} = (\vec{\mathbf{1}} - \mathbf{c^H})\circ(I_\mathbf{h}^\mathbf{H}\mathbf{r^h}), \qquad \vec{\mathbf{1}} = (1, \cdots, 1)^T.$$

The residual needs to be set to zero for known pixels (known according to the coarse inpainting mask $\mathbf{c^H}$). This is illustrated in Figure 8.

To apply nested iteration to obtain a good starting value for the multigrid cycles, we also need a restriction for the right-hand side $\mathbf{b^h}$. Here we use

$$\mathbf{b^H} = \left(I_\mathbf{h}^\mathbf{H}(\mathbf{c^h}\circ\mathbf{b^h})\right) \oslash \left(I_\mathbf{h}^\mathbf{H}\mathbf{c^h}\right),$$

where $\oslash$ is element-wise division with the exception that a division by zero leads to zero. This is illustrated in Figure 9.

To motivate this choice of the restriction for the right-hand side, we illustrate the restriction of $\mathbf{b}$ with the reweighting and without in Figure 10. Since division by very small numbers might be instable, the values of $I_\mathbf{h}^\mathbf{H}\mathbf{c^h}$ are set to zero below a certain tolerance ϵ, $0 < \epsilon \ll 1$, before the operator $\oslash$ is applied. More exactly, we use

$$c_{ij}^\mathbf{H} = \begin{cases} 1 & \text{if } (I_\mathbf{h}^\mathbf{H}\mathbf{c^h})_{ij} > \epsilon, \\ 0 & \text{else.} \end{cases}$$
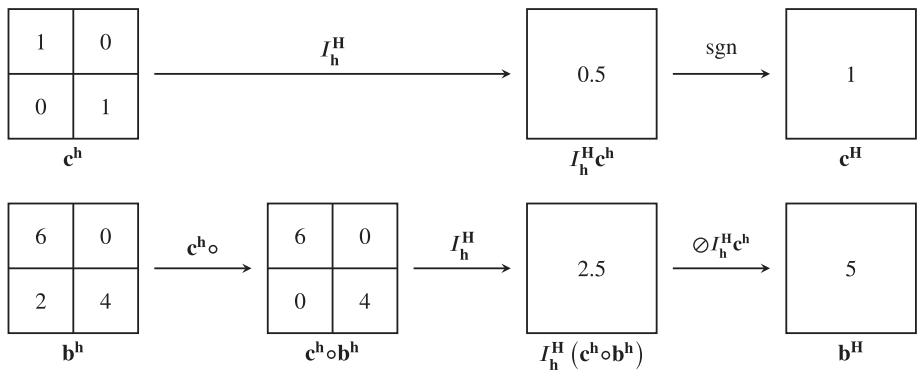
FIGURE 9 | Computation of the coarse right-hand side $\mathbf{b}^{\mathbf{H}}$.


(a)

(b)
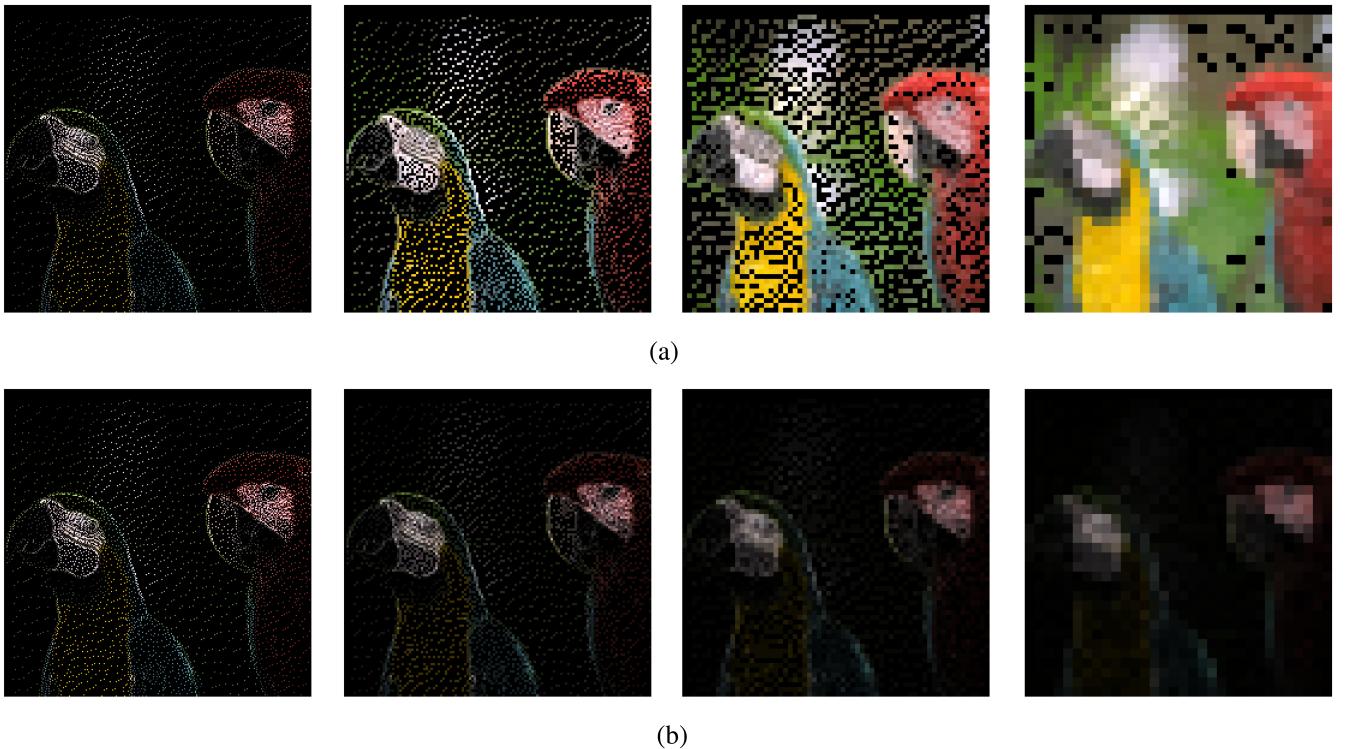
FIGURE 10 | Restriction of the right-hand side $\mathbf{b}^{\mathbf{h}}$. The top row shows the restriction with the reweighting, the bottom row without. Without reweighting, the colors lose intensity on the coarser grids. (a) $\mathbf{b}^{\mathbf{h}}$, $\mathbf{b}^{\mathbf{H}} = \left(I_h^H(\mathbf{c}^{\mathbf{h}} \circ \mathbf{b}^{\mathbf{h}}) \right) \oslash \left(I_h^H \mathbf{c}^{\mathbf{h}} \right), \ldots$ with normalization, (b) $\mathbf{b}^{\mathbf{h}}$, $I_h^H \mathbf{b}^{\mathbf{h}}, \ldots$ without normalization.

With these preparations, we state the full multigrid method as Algorithm 5. It consists of nested iteration (cf. Algorithm 4) for a good starting vector followed by several μ-cycles (cf. Algorithm 3). The μ-cycle for $\mu = 1$ is also called V-cycle and the μ-cycle with $\mu = 2$ is also called W-cycle. The damped Jacobi method is used for the pre- and post-relaxation steps with a damping constant $2/3$. The default is $\nu_1 = 4$ and $\nu_2 = 4$. As a stopping criterion for the multigrid method, we use the relative norm of the residual set to a small value (`default` $= 10^{-10}$).

6 | Numerical Experiments

In this section, we present some experiments with our decoding scheme. In the first subsection, we show that the new method outperforms standard time-integration methods. That the method compares to other (linear) edge-compressing schemes is shown in the second subsection. Finally, we demonstrate the use of our scheme on a real-world device.

ALGORITHM 3 | μ-cycle ($\mathbf{v}^{\mathbf{h}} = \text{MG}(\mathbf{v}^{\mathbf{h}}, \mathbf{b}^{\mathbf{h}})$).

---

if $\Omega^{\mathbf{h}}$ = coarsest grid **then**

    solve $B^{\mathbf{h}}\mathbf{v}^{\mathbf{h}} = \mathbf{b}^{\mathbf{h}}$ by direct solver

else

    relax ν_1 times $B^{\mathbf{h}}\mathbf{u}^{\mathbf{h}} = \mathbf{b}^{\mathbf{h}}$ with given starting value $\mathbf{v}^{\mathbf{h}}$

    $\mathbf{r}^{\mathbf{H}} = (\vec{\mathbf{1}} - \mathbf{c}^{\mathbf{H}}) \circ I_{\mathbf{h}}^{\mathbf{H}}(\mathbf{b}^{\mathbf{h}} - B^{\mathbf{h}}\mathbf{v}^{\mathbf{h}})$

    $\mathbf{v}^{\mathbf{H}} = 0$

    $\mathbf{v}^{\mathbf{H}} = \text{MG}(\mathbf{v}^{\mathbf{H}}, \mathbf{r}^{\mathbf{H}})$ μ times

    correct $\mathbf{v}^{\mathbf{h}} = \mathbf{v}^{\mathbf{h}} + I_{\mathbf{H}}^{\mathbf{h}}\mathbf{v}^{\mathbf{H}}$

    relax ν_2 times $B^{\mathbf{h}}\mathbf{u}^{\mathbf{h}} = \mathbf{b}^{\mathbf{h}}$ with starting value $\mathbf{v}^{\mathbf{h}}$

end if

ALGORITHM 4 | Nested iteration ($\mathbf{v}^{\mathbf{h}} = \text{NI}(\mathbf{b}^{\mathbf{h}})$).

---

if $\Omega^{\mathbf{h}}$ = coarsest grid **then**

    solve $B^{\mathbf{h}}\mathbf{v}^{\mathbf{h}} = \mathbf{b}^{\mathbf{h}}$ by direct solver

else

    $\mathbf{b}^{\mathbf{H}} = \left(I_{\mathbf{h}}^{\mathbf{H}}(\mathbf{c}^{\mathbf{h}} \circ \mathbf{b}^{\mathbf{h}}) \right) \oslash (I_{\mathbf{h}}^{\mathbf{H}}\mathbf{c}^{\mathbf{h}})$

end if

$\mathbf{v}^{\mathbf{H}} = \text{NI}(\mathbf{b}^{\mathbf{H}})$

prolongate $\mathbf{v}^{\mathbf{h}} = I_{\mathbf{H}}^{\mathbf{h}}\mathbf{v}^{\mathbf{H}}$

$\mathbf{v}^{\mathbf{h}} = \text{MG}(\mathbf{v}^{\mathbf{h}}, \mathbf{b}^{\mathbf{h}})$ ν_0 times

---

ALGORITHM 5 | Full multigrid ($\mathbf{v}^{\mathbf{h}} = \text{FMG}(\mathbf{v}^{\mathbf{h}}, \mathbf{b}^{\mathbf{h}})$).

$\mathbf{v}^{\mathbf{h}} = \text{NI}(\mathbf{b}^{\mathbf{h}})$

$\mathbf{v}^{\mathbf{h}} = \text{MG}(\mathbf{v}^{\mathbf{h}}, \mathbf{b}^{\mathbf{h}})$ k times

---

6.1 | Performance of the Integrator

Basically, we have to compute the solution $\mathbf{y}(t)$ of the system of ordinary differential equations (1) for a large time t. After subdividing the interval $[0, t]$ in n subintervals, the standard and most-used methods to approximate this solution are the implicit (or backward) Euler method

$$\mathbf{y}(t) \approx \left(\gamma(\gamma I - A)^{-1} \right)^n \mathbf{b}_0, \quad \gamma = \frac{n}{t}, \tag{8}$$

and the Crank–Nicolson method

$$\mathbf{y}(t) \approx \left((\gamma I + A)(\gamma I - A)^{-1} \right)^n \mathbf{b}_0, \quad \gamma = \frac{2n}{t}. \tag{9}$$

The larger n, the more accurate the approximation. To apply both methods, we have to solve n linear systems of the same type as for the Krylov method. Since this is the largest workload, we compare the methods with respect to the number of necessary solutions of linear systems of this type. For our edge-compressed all-white square test picture of Section 4, the relative error in the Euclidean norm is shown in the top row of Figure 11. For the edge-compressed image obtained from the original image on the left-hand side of Figure 12, the relative error is shown in the bottom row of Figure 11. For $t = 25$, $t = 10^2$, and $t = 10^3$, the errors of the methods vs. the number of required solutions of the large linear system are shown. For larger t, the Crank–Nicolson method becomes worse (which is a known behavior due to stability considerations), while the backward Euler scheme remains unaffected. For large t and an approximation error of about 10^{-3}, the implicit Euler scheme needs to solve 1000 linear systems of the type $(\gamma I - A)\mathbf{x} = \mathbf{b}$, while the Krylov method only needs 8. This is a factor of 125 times faster. This clearly demonstrates our main contribution that the Krylov method can solve homogeneous inpainting problems with a significantly improved speed.
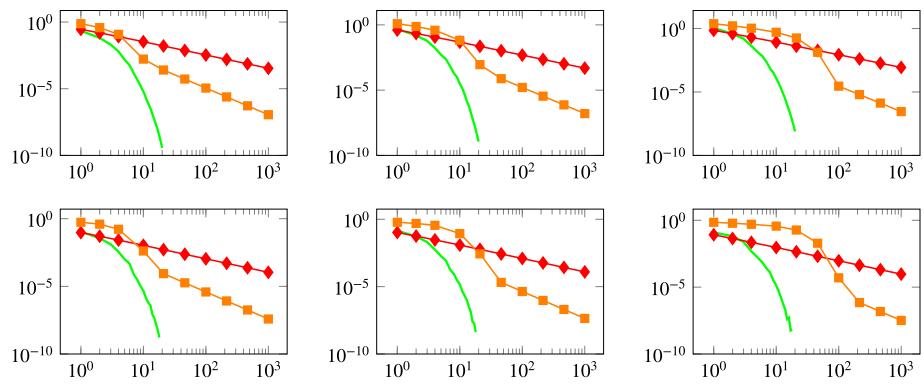
FIGURE 11 | Error of the implicit Euler scheme (red) diamond-marked line, error of the Crank–Nicolson method (orange) square-marked line, and the rational Krylov subspace method with optimal gamma (green) solid line vs. number of solutions of the linear system for $t = 25$ on the left-hand side, $t = 10^2$ in the middle, $t = 10^3$ on the right-hand side. The top row corresponds to the edge-compressed all-white square image and the bottom row corresponds to the edge-compressed image obtained from the original image on the left-hand side of Figure 12.

FIGURE 12 | (a) Shows the original image and (b) shows the reconstruction by computing the solution of the heat equation up to time $t = 10^7$. (a) Original image, (b) reconstruction at $t = 10^7$.

6.2 | Quality of Compression

To ensure that decoding the edge-compressed pictures by the Krylov method does not affect the quality of the recovered image, we provide experiments with pictures of the Kodak lossless true color image suite (cf. Franzen [32]). To measure the deviation of the decoded compressed picture from the original picture, we use the mean-square error (MSE). For two color pictures $\mathbf{u}, \mathbf{v} \in \mathbb{R}^{M,N,3}$ with three color channels and dimension $M \times N$, the mean-square error is given as

$$\text{MSE}(\mathbf{u}, \mathbf{v}) := \frac{1}{3MN} \sum_{k=1}^{3} \sum_{i=1}^{M} \sum_{j=1}^{N} (u_{i,j,k} - v_{i,j,k})^2.$$

We use $\mu = 2$ in Algorithm 3, which corresponds to the W-cycle, and 4 pre- as well as post-relaxation steps. For the nested iteration to obtain a good starting value, we use $\nu_0 = 1$ in Algorithm 4. For these pictures, we use 7 levels in the multigrid method.

We found that for a large time $t = 10^7$, the extended Krylov subspace with dimension $m = 3$ is sufficient to provide a good reconstruction. This means that only one solution of a linear system with the multigrid method is necessary. For the optimal $\gamma = 1.5/t$ and picture kodim07 of the test suite, the original picture and the reconstruction can be seen in Figure 12 on the left-hand and right-hand side, respectively.

TABLE 2 | Comparison of the quality of the reconstruction for the proposed decoding scheme for dithering-based as well as edge-based compressed images of the test suite.

	dithering-based			edge-based				dithering-based			edge-based		
img	bpp	MSE	PSNR	bpp	MSE	PSNR	img	bpp	MSE	PSNR	bpp	MSE	PSNR
01	2.37	**161.42**	**26.05**	**2.17**	164.39	25.97	13	**2.70**	**261.84**	**23.95**	2.99	286.27	23.56
02	2.23	**26.63**	**33.88**	**1.99**	64.59	30.03	14	2.63	**77.51**	**29.24**	**2.50**	101.81	28.05
03	2.18	**14.23**	**36.60**	**1.66**	50.11	31.13	15	2.34	**26.98**	**33.82**	**1.86**	74.44	29.41
04	2.37	**26.31**	**33.93**	**2.11**	47.98	31.32	16	2.09	**35.29**	**32.65**	**1.81**	63.26	30.12
05	2.74	**144.67**	**26.53**	**2.43**	165.88	25.93	17	2.28	**22.83**	**34.55**	**1.92**	53.52	30.85
06	2.34	**86.72**	**28.75**	**2.00**	148.47	26.41	18	2.71	**74.10**	**29.43**	**2.57**	120.92	27.31
07	2.38	**22.65**	**34.58**	**1.45**	62.19	30.19	19	2.30	**52.76**	**30.91**	**1.70**	106.74	27.85
08	2.68	**271.95**	**23.79**	**1.99**	282.04	23.63	20	2.05	**22.22**	**34.66**	**1.28**	68.80	29.76
09	2.16	**20.77**	**34.96**	**1.19**	51.70	31.00	21	2.36	**49.72**	**31.17**	**1.72**	111.10	27.67
10	2.22	**22.90**	**34.53**	**1.61**	46.91	31.42	22	2.59	**42.20**	**31.88**	**2.42**	72.10	29.55
11	2.41	**53.89**	**30.82**	**2.06**	86.52	28.76	23	2.36	**9.86**	**38.19**	**1.88**	39.84	32.13
12	2.13	**20.24**	**35.07**	**1.77**	39.67	32.15	24	2.57	**102.71**	**28.01**	**2.18**	153.04	26.28
							avg	2.38	**68.77**	**31.58**	**1.97**	102.60	28.77

The image has been compressed by the dithering-based method. The same observation turned out to be true for the whole test set and for dithering-based as well as edge-based compression of the images. We present the results in Table 2. Here, we also state the peak signal-to-noise ratio (PSNR),

$$\text{PSNR}(\mathbf{u}, \mathbf{v}) := 10 \cdot \log_{10}\left(\frac{255^2}{\text{MSE}(\mathbf{u}, \mathbf{v})}\right)[\text{dB}],$$

which is the most commonly used measure for the quality of reconstructions in lossy compression schemes. We also state the compression rate in *bits per pixel (bpp)*, which refers to the average number of bits needed to encode each image pixel. The original pictures are RGB pictures using 8 bits per color channel, which gives 24 bpp in the original pictures. Better values are marked in bold. The averages (avg) over all values are shown in the last row.

The results in Table 2 show that the decoding method is sufficiently accurate.

6.3 | Performance on an Everyday Device

High compression rates are particularly important for embedded devices like smartphones, smart TV sets, and smart watches, where storage is limited. Nowadays, these devices include embedded GPUs (Graphics Processing Units), which allow for accelerating image processing tasks considerably. The industry standard to accelerate graphics by the use of these GPUs is OpenGL ES (Open Graphics Library for Embedded Systems) managed by the non-profit Khronos group [41]. For our experiment, we will use the version OpenGL ES 3.2, which is available on 88.6% of the devices running Android as of the April 23, 2024 [42] as well as the version OpenGL ES 3.1 with the extension GL_EXT_color_buffer_float, which allows rendering to float textures attached to a framebuffer. Our approach, with the implementation details given in Section 5, perfectly fits the OpenGL application interface. Pictures are treated as textures that are operated on in a parallel manner by the use of vertex and fragment shaders. Turning the matrices into sparse formats would not lead to algorithms that can be easily ported to embedded GPUs. We first used a desktop computer with an NVIDIA GeForce RTX 3060/PCIe/SSE2. OpenGL ES 3.2 is available on this GPU. The arithmetic mean of ten run-times of our program to decode the 512×512 RGB picture in the middle of Figure 1 to the picture on the right-hand side of Figure 1 was 0.014 s. The dimension of the system (1) is 786,432 for this picture. On a notebook with the integrated graphics processor Intel(R) HD Graphics 620 (KBL GT2), the arithmetic mean of ten run-times was 0.049 s. OpenGL ES 3.2 is also available on this graphics processor. As an embedded device, we used a Samsung Galaxy J4+ smartphone running Android version 9 (Pie) with a Qualcomm Adreno 308 GPU, with the same picture. This phone allows for version OpenGL ES 3.1 with GL_EXT_color_buffer_float

FIGURE 13 | Experiment on a smartphone. (a) Compression, (b) reconstruction.

extension. The C code was compiled with the native development toolkit for Android systems [43]. The transition from the left-hand side of Figure 13 to the right-hand side of Figure 13 took about 0.718 s in average. With less than a second, this seems to be fast enough to decode edge-compressed pictures stored on this phone in a real-life application. We also tested our algorithm on a high-end smartphone with a Qualcomm Adreno 740 GPU running Android version 14. This phone supports OpenGL ES 3.2 and the decoding took 0.034 s, measured as the arithmetic mean of ten runs. At that speed, touching the compression on the left-hand side of Figure 13 immediately turns the image to the reconstruction on the right-hand side of Figure 13. The experience is smooth, and one does not even notice the slightest delay.

7 | Conclusion

We presented an efficient method to solve inpainting problems by homogeneous diffusion based on extended Krylov subspaces. Error bounds for the approximation of the linear heat equation up to a prescribed large time t are given and numerically verified. The method is applicable to all inpainting problems of this type. We studied the problem of decoding edge-based and dithering-based compressed images, where the boundaries for the inpainting problems are especially challenging.

References

1. S. Carlsson, "Sketch Based Coding of Grey Level Images," *Signal Processing* 15, no. 1 (1988): 57–83, https://doi.org/10.1016/0165-1684(88)90028-X.

2. J. H. Elder, "Are Edges Incomplete?," *International Journal of Computer Vision* 34 (1999): 97–122, https://doi.org/10.1023/A:1008183703117.

3. R. Hummel and R. Moniot, "Reconstructions From Zero-Crossings in Scale Space," *IEEE Transactions on Acoustics, Speech, and Signal Processing* 37 (1989): 2111–2130, https://doi.org/10.1109/29.45555.

4. Y. Zeevi and D. Rotem, "Image Reconstruction From Zero-Crossings," *IEEE Transactions on Acoustics, Speech, and Signal Processing* 34 (1986): 1269–1277, https://doi.org/10.1109/TASSP.1986.1164922.

5. M. M. Reid, R. J. Millar, and N. D. Black, "Second-Generation Image Coding: An Overview," *ACM Computing Surveys* 29 (1997): 3–29, https://doi.org/10.1145/248621.248622.

6. M. Mainberger, A. Bruhn, J. Weickert, and S. Forchhammer, "Edge-Based Compression of Cartoon-Like Images With Homogeneous Diffusion," *Pattern Recognition* 44, no. 9 (2011): 1859–1873, https://doi.org/10.1016/j.patcog.2010.08.004.

7. R. W. Floyd and L. Steinberg, "An Adaptive Algorithm for Spatial Grey Scale," *Proceedings of the Society of Information Display* 17 (1976): 75–77.

8. L. Hoeltgen, S. Setzer, and J. Weickert, "An Optimal Control Approach to Find Sparse Data for Laplace Interpolation," in *Energy Minimization Methods in Computer Vision and Pattern Recognition.* Lecture notes in Computer Science, vol. 8081, ed. A. Heyden, F. Kahl, C. Olsson, M. Oskarsson, and X. C. Tai (Springer, 2013), 151–164, https://doi.org/10.1007/978-3-642-40395-8_12.

9. M. Mainberger, S. Hoffmann, J. Weickert, et al., "Optimising Spatial and Tonal Data for Homogeneous Diffusion Inpainting," in *Scale Space and Variational Methods in Computer Vision. SSVM 2011. Lecture Notes in Computer Science*, vol. 6667, ed. A. M. Bruckstein, B. M. ter Haar Romney, A. M. Bronstein, and M. M. Bronstein (Springer, 2012), 27–37, https://doi.org/10.1007/978-3-642-24785-9_3.

10. V. Grimm, S. Henn, and K. Witsch, "A Higher-Order PDE-Based Image Registration Approach," *Numerical Linear Algebra With Applications* 13, no. 5 (2006): 399–417, https://doi.org/10.1002/nla.467.

11. M. Hochbruck and A. Ostermann, "Exponential Integrators," *Acta Numerica* 19 (2010): 209–286, https://doi.org/10.1017/S0962492910000048.

12. C. B. Schönlieb, "Partial Differential Equation Methods for Image Inpainting," in *Cambridge Monographs on Applied and Computational Mathematics*, vol. 29 (Cambridge University Press, 2015), https://doi.org/10.1017/CBO9780511734304.

13. A. Ruhe, "Rational Krylov Sequence Methods for Eigenvalue Computation," *Linear Algebra and Its Applications* 58 (1984): 391–405, https://doi.org/10.1016/0024-3795(84)90221-0.

14. A. Ruhe, "Rational Krylov: A Practical Algorithm for Large Sparse Nonsymetric Matrix Pencils," *SIAM Journal on Scientific Computing* 19 (1998): 1535–1551, https://doi.org/10.1137/S1064827595285597.

15. C. Brezinski, P. Novati, and M. Redivo-Zaglia, "A Rational Arnoldi Approach for Ill-Conditioned Linear Systems," *Journal of Computational and Applied Mathematics* 236, no. 8 (2012): 2063–2077, https://doi.org/10.1016/j.cam.2011.09.032.

16. A. Buccini, M. Donatelli, and L. Reichel, "Iterated Tikhonov Regularization With a General Penalty Term," *Numerical Linear Algebra With Applications* 24, no. 4 (2017): e2089, https://doi.org/10.1002/nla.2089.

17. V. Grimm, "A Conjugate-Gradient-Type Rational Krylov Subspace Method for Ill-Posed Problems," *Inverse Problems* 36, no. 1 (2020): 015008, https://doi.org/10.1088/1361-6420/ab5819.

18. R. Ramlau and L. Reichel, "Error Estimates for Arnoldi-Tikhonov Regularization for Ill-Posed Operator Equations," *Inverse Problems* 35, no. 5 (2019): 055002, https://doi.org/10.1088/1361-6420/ab0663.

19. T. Göckler and V. Grimm, "Convergence Analysis of an Extended Krylov Subspace Method for the Approximation of Operator Functions in Exponential Integrators," *SIAM Journal on Numerical Analysis* 51, no. 4 (2013): 2189–2213, https://doi.org/10.1137/12089226X.

20. T. Göckler and V. Grimm, "Uniform Approximation of φ-Functions in Exponential Integrators by a Rational Krylov Subspace Method With Simple Poles," *SIAM Journal on Matrix Analysis and Applications* 35, no. 4 (2014): 1467–1489, https://doi.org/10.1137/140964655.

21. V. Grimm, "Resolvent Krylov Subspace Approximation to Operator Functions," *BIT Numerical Mathematics* 52, no. 3 (2012): 639–659, https://doi.org/10.1007/s10543-011-0367-8.

22. V. Grimm and T. Göckler, "Automatic Smoothness Detection of the Resolvent Krylov Subspace Method for the Approximation of C_0-Semigroups," *SIAM Journal on Numerical Analysis* 55, no. 3 (2017): 1483–1504, https://doi.org/10.1137/15M104880X.

23. J. E. Andersson, "Approximation of e^{-x} by Rational Functions With Concentrated Negative Poles," *Journal of Approximation Theory* 32, no. 2 (1981): 85–95, https://doi.org/10.1016/0021-9045(81)90106-4.

24. J. van den Eshof and M. Hochbruck, "Preconditioning Lanczos Approximations to the Matrix Exponential," *SIAM Journal on Scientific Computing* 27, no. 4 (2006): 1438–1457, https://doi.org/10.1137/040605461.

25. B. Beckermann and S. Güttel, "Superlinear Convergence of the Rational Arnoldi Method for the Approximation of Matrix Functions," *Numerische Mathematik* 121, no. 2 (2012): 205–236, https://doi.org/10.1007/s00211-011-0434-8.

26. R. U. Börner, O. G. Ernst, and S. Güttel, "Three-Dimensional Transient Electromagnetic Modelling Using Rational Krylov Methods," *Geophysical Journal International* 202, no. 3 (2015): 2025–2043, https://doi.org/10.1093/gji/ggv224.

27. K. Bergermann and M. Stoll, "Adaptive Rational Krylov Methods for Exponential Runge–Kutta Integrators," *SIAM Journal on Matrix Analysis and Applications* 45, no. 1 (2024): 744–770, https://doi.org/10.1137/23M1559439.

28. V. Druskin and L. Knizhnerman, "Extended Krylov Subspaces: Approximation of the Matrix Square Root and Related Functions," *SIAM Journal on Matrix Analysis and Applications* 19, no. 3 (1998): 755–771, https://doi.org/10.1137/S0895479895292400.

29. L. Knizhnerman and V. Simoncini, "A New Investigation of the Extended Krylov Subspace Method for Matrix Function Evaluations," *Numerical Linear Algebra With Applications* 17, no. 4 (2010): 615–638, https://doi.org/10.1002/nla.652.

30. A. H. Al-Mohy and N. J. Higham, "Computing the Action of the Matrix Exponential, With an Application to Exponential Integrators," *SIAM Journal on Scientific Computing* 33, no. 2 (2011): 488–511, https://doi.org/10.1137/100788860.

31. N. J. Higham, *Functions of Matrices. Society for Industrial and Applied Mathematics (SIAM)* (SIAM, 2008), https://doi.org/10.1137/1.9780898717778.

32. R. Franzen, "Kodak Lossless True Color Image Suite," (2024), http://r0k.us/graphics/kodak.

33. D. Marr, "Early Processing of Visual Information," *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences* 275, no. 942 (1975): 483–519, https://doi.org/10.1098/rstb.1976.0090.

34. D. Marr and E. Hildreth, "Theory of Edge Detection," *Proceedings of the Royal Society of London* B207 (1980): 187–217, http://www.jstor.org/stable/35407.

35. J. Canny, "A Computational Approach of Edge Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence* 8, no. 6 (1986): 679–698, https://doi.org/10.1109/TPAMI.1986.4767851.

36. Z. Belhachmi, D. Bucur, B. Burgeth, and J. Weickert, "How to Choose Interpolation Data in Images," *SIAM Journal on Applied Mathematics* 70, no. 1 (2009): 333–352, https://doi.org/10.1137/080716396.

37. Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed. (Society for Industrial and Applied Mathematics, 2003), https://doi.org/10.1137/1.9780898718003.

38. Y. Saad, "Analysis of Some Krylov Subspace Approximations to the Matrix Exponential Operator," *SIAM Journal on Numerical Analysis* 29, no. 1 (1992): 209–228, https://doi.org/10.1137/0729014.

39. B. Beckermann and L. Reichel, "Error Estimates and Evaluation of Matrix Functions via the Faber Transform," *SIAM Journal on Numerical Analysis* 47, no. 5 (2009): 3849–3883, https://doi.org/10.1137/080741744.

40. A. Bruhn, J. Weickert, C. Feddern, T. Kohlberger, and C. Schnörr, "Variational Optic Flow Computation in Real-Time," *IEEE Transactions on Image Processing* 14, no. 5 (2005): 608–615, https://doi.org/10.1109/TIP.2005.846018.

41. Khronos Group, 2025, https://www.khronos.org/.

42. Distribution Dashbord, https://developer.android.com/about/dashboards/index.html#OpenGL.

43. Android NDK, 2025, https://developer.android.com/ndk.