A weighted $H^1$ seminorm regularization method for Fredholm integral equations of the first kind

Fu-Rong Lin $^a$ & Shi-Wei Yang $^a$

$^a$ Department of Mathematics, Shantou University, Shantou, Guangdong, 515063, China

Accepted author version posted online: 01 Jul 2013. Published online: 05 Aug 2013.


To link to this article: http://dx.doi.org/10.1080/00207160.2013.818137

PLEASE SCROLL DOWN FOR ARTICLE

Taylor & Francis makes every effort to ensure the accuracy of all the information (the “Content”) contained in the publications on our platform. However, Taylor & Francis, our agents, and our licensors make no representations or warranties whatsoever as to the accuracy, completeness, or suitability for any purpose of the Content. Any opinions and views expressed in this publication are the opinions and views of the authors, and are not the views of or endorsed by Taylor & Francis. The accuracy of the Content should not be relied upon and should be independently verified with primary sources of information. Taylor and Francis shall not be liable for any losses, actions, claims, proceedings, demands, costs, expenses, damages, and other liabilities whatsoever or howsoever caused arising directly or indirectly in connection with, in relation to or arising out of the use of the Content.

This article may be used for research, teaching, and private study purposes. Any substantial or systematic reproduction, redistribution, reselling, loan, sub-licensing, systematic supply, or distribution in any form to anyone is expressly forbidden. Terms &
A weighted $H^1$ seminorm regularization method for Fredholm integral equations of the first kind

Fu-Rong Lin* and Shi-Wei Yang

Department of Mathematics, Shantou University, Shantou, Guangdong 515063, China

(Received 27 November 2012; revised version received 2 April 2013; second revision received 11 June 2013; third revision received 15 June 2013; accepted 17 June 2013)

Many problems in mathematics and engineering lead to Fredholm integral equations of the first kind, e.g. signal and image processing. These kinds of equations are difficult to solve numerically since they are ill-posed. Therefore, regularization is required to obtain a reasonable approximate solution. This paper presents a new regularization method based on a weighted $H^1$ seminorm. Details of numerical implementation are given. Numerical examples, including one-dimensional and two-dimensional integral equations, are presented to illustrate the efficiency of the proposed approach. Numerical results show that the proposed regularization method can restore edges as well as details.

Keywords: Fredholm integral equation of the first kind; piecewise continuous solution; weighted $H^1$ seminorm regularization; alternative iterative algorithm; image restoration

2010 AMS Subject Classifications: 65R32; 65F22

1. Introduction

In this paper, we consider numerical methods for Fredholm integral equations of the first kind

$$
\int_{\Omega} k(x, t)f(t) \, dt = g(x), \quad x \in \Omega,
$$

where $\Omega$ is an interval on $\mathbb{R}$ or a domain in $\mathbb{R}^2$, the kernel $k$ and the right-hand side $g$ are given functions, and $f$ is the unknown function to be determined. Equation (1) is a typical ill-posed problem which has variety of applications in mathematical physics and engineering, including signal and image processing [32]. Since there is perturbation in practical applications, Equation (1) should be modified to

$$
\int_{\Omega} k(x, t)f(t) \, dt = g(x) + \eta(x), \quad x \in \Omega,
$$

where $\eta$ represents the error in the right-hand side. Equation (2) can be written in operator form

$$
Kf = q,
$$

*Corresponding author. Email: frlin@stu.edu.cn

© 2013 Taylor & Francis
where \( q = g + \eta \) and

\[
(Kf)(x) = \int_\Omega k(x,t)f(t) \, dt.
\]

It is well known that Equation (1) is ill-posed and its analytic form solution is usually not available; see for instance [26, pp. 521–524]. Due to the ill-posedness of the problem, numerical solutions are extremely sensitive to perturbation caused by observed errors or rounding errors. Therefore, regularization is required to obtain a reasonable approximate solution. In the past decades, a lot of numerical methods for the solution of Equation (1), including regularization methods and fast numerical methods, have been proposed and studied; see for instance [15,32] and references therein.

Many regularization methods [1,14,16,17,22,23,27,31] have been proposed to overcome the ill-posedness of the problem. The Tikhonov regularization method [31], the truncated singular value decomposition (TSVD) method [14], the modified TSVD method [17], the Chebyshev interpolation method [27], collocation methods [22,23], and so on, have been proposed for obtaining approximate continuous solutions of Equation (2). For discontinuous solutions of Equation (2), total variation (TV) regularization methods have been proposed and extensively studied; see for instance [1,5,7,11,13,18,19,28,33]. The drawback of the TV method is that approximate solutions often suffer the staircase effect [2] and lose fine details. In order to compensate for staircase effect and loss of details, many methods have been proposed, e.g. weighted TV (WTV) and adaptive TV methods [3,9,10,12,30], high order methods [4,8], Tikhonov nonlocal TV method [21], and combined models [25]. Some methods focus on image denoising [3,4,8,24,29,34].

The aim of this paper is to introduce a new regularization method which is capable of rapidly obtaining numerical solutions that preserve edges and fine details. The main point is that the regularization function is close to \(|\nabla f(x)|\) for possible discontinuous points (edges) of \( f(x) \), while it is close to \(|\nabla f(x)|^2 \) for continuous points. Here, \( \nabla f(x) \) denotes the gradient of \( f(x) \) at \( x \).

This paper is organized as follows. In Section 2, a new regularization functional based on a weighted \( H^1 \) seminorm is introduced. In Section 3, detailed implementation of numerical methods for approximate solutions of Equation (2) is presented. Numerical examples are presented to illustrate the effectiveness of the proposed approach in Section 4. Finally, concluding remarks are given in Section 5.

2. A weighted \( H^1 \) seminorm regularization method

In this section, we propose a regularization method which can restore not only edges but also details. Our method is based on the \( H^1 \) seminorm.

The \( H^1 \) seminorm regularization method is one of the most popular tools for obtaining smooth solutions of ill-posed problems. That is, if the solution is smooth, we can obtain a satisfactory approximate solution by minimizing the following least-squares problem:

\[
\min_f \frac{1}{2} J(f) + \frac{\alpha}{2} \|f\|_{H^1}^2,
\]

where \( \alpha > 0 \) is a regularization parameter controlling the regularization,

\[
J(f) = \|Kf - q\|_L^2
\]

with

\[
\|f\|_{L^2} \equiv \left( \int_\Omega |f(x)|^2 \, dx \right)^{1/2}
\]
denoting the two-norm of $f$, and $\|f\|_{H^1}$ denotes the $H^1$ seminorm of $f(x)$

$$\|f\|^2_{H^1} = \int_\Omega |\nabla f(x)|^2 \, dx.$$  

The $H^1$ seminorm regularization method has a strong bias against discontinuous functions. On the other hand, the TV regularization preserves the discontinuous property of the solution. The TV functional of a function $f$ is defined by

$$\text{TV}(f) = \int_\Omega |\nabla f(x)| \, dx,$$

and the minimization problem corresponding to TV regularization is given by

$$\min_{f} \frac{1}{2} J(f) + \alpha \text{TV}(f).$$

A drawback of the TV regularization method is that it tends to yield piecewise constant solution, which is called the staircase effect [2].  

Many methods have been proposed for staircasing reduction, e.g. WTV and adaptive TV methods [12], high order methods [4], Tikhonov nonlocal TV method [21], and combined models [25]. Here, we propose a simple regularization functional

$$H_w(f) \equiv \int_\Omega w(x)|\nabla f(x)|^2 \, dx,$$

where $w(x) > 0$ for $x \in \Omega$ is a weighting function. We call $H_w(f)$ a weighted $H^1$ ($WH^1$) seminorm regularization. Thus, the corresponding minimization problem is

$$\min_f T(f) = \frac{1}{2} J(f) + \frac{\alpha}{2} H_w(f).$$

We choose the weighting function $w(x)$ such that

1. $w(x)|\nabla f(x)|^2 \approx |\nabla f(x)|^2$ if $|\nabla f(x)|$ is small (we use the penalty term close to $|\nabla f(x)|^2$ for continuous points of $f(x)$);
2. $w(x)|\nabla f(x)|^2 \propto |\nabla f(x)|$ if $|\nabla f(x)|$ is large (if $x$ is not a continuous point of $f(x)$, it is better to use the TV regularization).

Motivated by Strong et al. [30], we choose the weighting function as

$$w(x) = w_\theta(x) \equiv \frac{\theta}{|\nabla f(x)| + \theta}, \quad x \in \Omega,$$

where $\theta > 0$ is a constant which can be treated as an indicator to distinguish edges and smooth regions of numerical solutions. Notice that

$$w_\theta(x)|\nabla f(x)|^2 = \frac{\theta|\nabla f(x)|^2}{|\nabla f(x)| + \theta} \approx \begin{cases} |\nabla f(x)|^2, & |\nabla f(x)| \ll \theta, \\ \theta|\nabla f(x)|, & |\nabla f(x)| \gg \theta. \end{cases}$$

3. Numerical implementation

In this section, we first deduce the discretization form of $J(f)$ and $H_w(f)$ defined by Equations (4) and (5), respectively. Then, we give an alternative iterative algorithm for minimization of $T(f)$, where $T(f)$ is the discretization form of $T(f)$ (see Equation (6)).
3.1 Discretization of one-dimensional problems

Suppose \( \Omega = [0, 1] \) and let \( \Omega \) be partitioned into \( n \) uniform subintervals \([x_i - 0.5\Delta x, x_i + 0.5\Delta x], i = 1, 2, \ldots, n\), where \( x_i = (i - 0.5)\Delta x \) with \( \Delta x = 1/n \). We apply the midpoint quadrature to discretize the integrals. Thus, the discretization form of \( J(f) = \|Kf - q\|_2^2 \) is given by

\[
J(f) = \|Kf - q\|_2^2,
\]

where \( f = (f(x_1), f(x_2), \ldots, f(x_n))^T, q = (q(x_1), q(x_2), \ldots, q(x_n))^T, \) and \( K = [\Delta x k(x_i, x_j)]_{i,j=1}^n \).

As for the weighted \( H^1 \) seminorm functional defined by Equation (5), we approximate it by

\[
H_w(f) = \sum_{i=1}^{n-1} w_i (Df_i)^2 \Delta x = \left\| \begin{pmatrix} \frac{w_1}{\Delta x} (Df_1) \\ \vdots \\ \frac{w_n-1}{\Delta x} (Df_{n-1}) \end{pmatrix} \right\|_2^2 = \|W^{1/2}D_n f\|_2^2,
\]

where \(-Df_i \equiv (f_{i+1} - f_i)/\Delta x\) is an approximation of the first derivative of \( f \) at \((x_i + x_{i+1})/2,\)

\[
D_n = \begin{bmatrix} 1 & -1 \\ 1 & \ddots & -1 \\ & \ddots & \ddots & \ddots \\ & & 1 & -1 \end{bmatrix}_{(n-1) \times n},
\]

and \( W = \text{diag}(w_1, w_2, \ldots, w_{n-1}) \) with \( w_i \) being the weighting factors given by

\[
w_i = \frac{\theta}{\Delta x(|Df_i| + \theta)}, \quad i = 1, 2, \ldots, n - 1.
\]

Thus, the discretization of the functional \( T(f) \) (defined by Equation (6)) is given by

\[
T(f) = \frac{1}{2} f(f) + \frac{\alpha}{2} H_w(f) = \frac{1}{2} \|Kf - q\|_2^2 + \frac{1}{2} \|\sqrt{\alpha} W^{1/2} D_n f\|_2^2
\]

\[
= \frac{1}{2} \left\| \begin{pmatrix} Kf - q \\ \sqrt{\alpha} W^{1/2} D_n f \end{pmatrix} \right\|_2^2 = \frac{1}{2} \left\| \begin{pmatrix} K \\ \sqrt{\alpha} W^{1/2} D_n \end{pmatrix} f - \begin{pmatrix} q \\ 0 \end{pmatrix} \right\|_2^2.
\]

Recall that the minimizer of the least-squares problem

\[
\min_f \|Af - b\|_2^2
\]

is the solution of

\[
A^T Af = A^T b.
\]

We see that for fixed \( W \), the minimizer of \( T(f) \) is the solution of

\[
\begin{pmatrix} K \\ \sqrt{\alpha} W^{1/2} D_n \end{pmatrix}^T \begin{pmatrix} K \\ \sqrt{\alpha} W^{1/2} D_n \end{pmatrix} f = \begin{pmatrix} K \\ \sqrt{\alpha} W^{1/2} D_n \end{pmatrix}^T \begin{pmatrix} q \\ 0 \end{pmatrix},
\]

i.e.

\[
(K^T K + \alpha D_n^T W D_n) f = K^T q.
\]

(10)
Remark  In our numerical tests, we modify the weights $w_1, \ldots, w_{n-1}$ to
\[ w_i = \frac{\theta_1}{|f_{i+1} - f_i| + \theta}, \]
where $\theta_1$ is chosen such that the mean value of $w_1, \ldots, w_{n-1}$ is equal to 1. Similar treatments are applied to two-dimensional problems and the WTV method.

### 3.2 Discretization of two-dimensional problems

We now consider discretization of two-dimensional problems. Suppose $\Omega = [0, 1] \times [0, 1]$, that is, Equation (3) is given by
\[ \int_0^1 \int_0^1 k(x, y, u, v)f(u, v) \, du \, dv = q(x, y), \quad (u, v) \in \Omega. \]

We consider the following weighted $H^1$ seminorm:
\[ H_w(f) = \int_0^1 \int_0^1 \left( w_x(x, y) \left( \frac{\partial f(x, y)}{\partial x} \right)^2 + w_y(x, y) \left( \frac{\partial f(x, y)}{\partial y} \right)^2 \right) \, dx \, dy, \]

where
\[ w_x(x, y) = \frac{\theta}{|\partial f(x, y)/\partial x| + \theta}, \quad w_y(x, y) = \frac{\theta}{|\partial f(x, y)/\partial y| + \theta} \]
with $\theta > 0$ being a parameter.

Let the domain $\Omega$ be partitioned into $n_x \times n_y$ uniform subdomains
\[ [x_i - 0.5 \Delta x, x_i + 0.5 \Delta x] \times [y_j - 0.5 \Delta y, y_j + 0.5 \Delta y], \quad i = 1, \ldots, n_x, \quad j = 1, \ldots, n_y, \]
where $x_i = (i - 0.5) \Delta x$ with $\Delta x = 1/n_x$ and $y_j = (j - 0.5) \Delta y$ with $\Delta y = 1/n_y$. Applying the midpoint quadrature to the functional $J(f)$, we get the discretization of $J(f)$
\[ J(f) = \| Kf - q \|_2^2, \]

where
\[ f = (f_{1,1}, \ldots, f_{n_x,1}, f_{1,2}, \ldots, f_{n_x,2}, \ldots, f_{1,n_y}, \ldots, f_{n_x,n_y})^T \]
with $f_{i,j} = f(x_i, y_j)$, $i = 1, \ldots, n_x$, $j = 1, \ldots, n_y$,
\[ q = (q_{1,1}, \ldots, q_{n_x,1}, q_{1,2}, \ldots, q_{n_x,2}, \ldots, q_{1,n_y}, \ldots, q_{n_x,n_y})^T \]
with $q_{i,j} = q(x_i, y_j)$, $i = 1, \ldots, n_x$, $j = 1, \ldots, n_y$, and
\[ K = [K_{i,j}]_{i,j=1}^{n_x n_y} \quad \text{with} \quad K_{i,j} = [\Delta x \Delta y k(x_i, y_j, x_{i'}, y_{j'})]_{i',j'=1}^{n_x n_y}. \]

As for the weighted $H^1$ seminorm regularization $H_w(f)$, we approximate it by
\[ H_w(f) = \sum_{i=1}^{n_x-1} \sum_{j=1}^{n_y} w_{x,i,j}(D_x f_{i,j})^2 \Delta x \Delta y + \sum_{i=1}^{n_x} \sum_{j=1}^{n_y-1} w_{y,i,j}(D_y f_{i,j})^2 \Delta x \Delta y, \]
where
\[ -D_x f_{i,j} = \frac{f_{i+1,j} - f_{i,j}}{\Delta x}, \quad -D_y f_{i,j} = \frac{f_{i,j+1} - f_{i,j}}{\Delta y}, \]
and
\[ w_{x,i,j} = \frac{\theta}{|D_{xf,i,j}| + \theta}, \quad w_{y,i,j} = \frac{\theta}{|D_{yf,i,j}| + \theta}. \] (11)

Let
\[ W_x = \frac{\Delta y}{\Delta x} \text{diag}(w_{x,1,1}, \ldots, w_{x,n_x-1,1}, w_{x,1,2}, \ldots, w_{x,n_x-1,2}, \ldots, w_{x,1,n_y}, \ldots, w_{x,n_x-1,n_y}), \]
\[ W_y = \frac{\Delta x}{\Delta y} \text{diag}(w_{y,1,1}, \ldots, w_{y,n_y,1}, w_{y,1,2}, \ldots, w_{y,n_y,2}, \ldots, w_{y,1,n_x-1}, \ldots, w_{y,n_y,n_x-1}), \]
and
\[ D_x = I_{n_y} \otimes D_{n_x}, \quad D_y = D_{n_y} \otimes I_{n_x}, \]
where \( D_{n_x} \) and \( D_{n_y} \) are defined by Equation (8), and \( A \otimes B \) denotes the Kronecker tensor product. For \( A = [a_{ij}]_{p \times q}, \)
\[ A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \cdots & a_{1q}B \\ a_{21}B & a_{22}B & \cdots & a_{2q}B \\ \vdots & \vdots & & \vdots \\ a_{p1}B & a_{p2}B & \cdots & a_{pq}B \end{pmatrix}. \]

Then, the discretization of the weighted \( H^1 \) seminorm regularization can be written in matrix form
\[ H_w(f) = \| W_x^{1/2} D_x f \|_2^2 + \| W_y^{1/2} D_y f \|_2^2 = \left\| \begin{pmatrix} W_x^{1/2} D_x \\ W_y^{1/2} D_y \end{pmatrix} f \right\|_2^2. \]
Thus, the discretization of \( T(f) \) is given by
\[ T(f) = \frac{1}{2} \| Kf - q \|_2^2 + \frac{1}{2} \left\| \begin{pmatrix} \sqrt{\alpha} W_x^{1/2} D_x \\ \sqrt{\alpha} W_y^{1/2} D_y \end{pmatrix} f \right\|_2^2. \]
Therefore, for fixed \( W_x \) and \( W_y \), the minimizer of \( T(f) \) is the solution of
\[ (K^T K + \alpha D_x^T W_x D_x + \alpha D_y^T W_y D_y) f = K^T q. \] (12)

### 3.3 An alternative iterative algorithm

In this subsection, we propose an alternative iterative algorithm to seek the minimizer of \( T(f) \). The steps are listed in the following Algorithm 1.

We now make remarks on computation of matrix–vector multiplications in Equation (12).

(1) It must be pointed out that for two-dimensional problems, the most expensive part of Algorithm 1 is the solution of Equation (12). If the matrix \( K \) has block Toeplitz structure, which is often the case for image deblurring and denoising problems, then the matrix–vector multiplications \( K^T (KF) \) can be done in \( O(n_x n_y \log(n_x n_y)) \) operations by using the fast Fourier transform (FFT) technique; see [6,20] for detail.
Algorithm 1 Alternative iterative method

1. Input an initial guess \( f_0 \);
2. \( \nu := 0 \);
3. Begin iterations
4. Compute the weighting factors \( w \) by Equation (9) for one-dimensional problems or by Equation (11) for two-dimensional problems;
5. Solve Equation (10) by the Gaussian elimination method or solve Equation (12) by using the conjugate gradient method with \( f_{\nu} \) as initial guess to obtain a new approximation \( f_{\nu+1} \);
6. Compute \( s = f_{\nu+1} - f_{\nu} \);
7. If \( \|s\|/\|f_{\nu+1}\| < \epsilon \), break;
8. \( \nu = \nu + 1 \);
9. End iterations
10. Output \( f_\nu \).

(2) Let \( A \) and \( B \) be \( p \times q \) and \( r \times s \) matrices, respectively, and \( f \) be a \((qs)\)-vector, then the matrix–vector product \((A \otimes B)f\) can be efficiently obtained by using

\[
(A \otimes B)f = \text{vec}(B \text{mat}(f)A^T).
\]

Here

\[
\text{mat}(f) = \begin{pmatrix}
    f_1 & f_{s+1} & \cdots & f_{(q-1)s+1} \\
    f_2 & f_{s+2} & \cdots & f_{(q-1)s+2} \\
    \vdots & \vdots & \ddots & \vdots \\
    f_s & f_{2s} & \cdots & f_{qs}
\end{pmatrix},
\]

and for \( C = [c_{ij}]_{i=1:p,j=1:r} \),

\[
\text{vec}(C) = [c_{1,1}, \ldots, c_{r,1}, c_{1,2}, \ldots, c_{r,2}, \ldots, c_{1,p}, \ldots, c_{r,p}]^T.
\]

Thus, by exploiting the sparsity of \( D_{n_x} \) and \( D_{n_y} \), and the tensor structure of \( D_x \) and \( D_y \), we can obtain the matrix–vector products \( D_x^T W_x D_x f \) and \( D_y^T W_y D_y f \) in \( O(n_x n_y) \) operations.

Based on the above discussions, we see that the matrix–vector multiplications in Equation (12) can be efficiently done in \( O(n_x n_y \log(n_x n_y)) \) operations.

4. Numerical examples

In this section, we test the approach proposed in the previous sections by several examples, including one-dimensional and two-dimensional problems. The experiments were performed by MATLAB R2010b running on a computer with an Intel Xeon CPU E5-2609 at 2.40 GHz and 64 GB of memory. The error vector \( \eta \) has normally distributed entries with zero mean.

To illustrate the efficiency of the proposed approach, we compare the performance of Algorithm 1 with several well-known methods, including the \( H^1 \) seminorm regularization method (all weighting factors are 1), the TV method, a WTV method, and two Tikhonov nonlocal total variation (NLTV) methods [21]. We will present numerical results for a one-dimensional example.
in Section 4.1, and numerical results for image restoration in Sections 4.2 and 4.3, where we consider zero boundary condition and periodic condition, respectively.

In the following, we compare the performance of different methods by the quality of the numerical solutions and the CPU time required. We will show three measurements to compare the quality of numerical solutions – the relative error, the peak-signal-to-noise ratio (PSNR), and the signal-to-noise ratio (SNR).

Let \( f_{\text{exact}} \) and \( f \) be the exact solution and the numerical solution, respectively, and let \( \| \cdot \| \) denote the vector two-norm or the Frobenius matrix norm. The relative error is defined by

\[
E_{\text{relative}} = \frac{\| f - f_{\text{exact}} \|}{\| f_{\text{exact}} \|}.
\]

The PSNR is defined by

\[
\text{PSNR} = 10 \log_{10} \frac{\text{MAX}^2}{\| f - f_{\text{exact}} \|^2 / (n_x n_y)},
\]

where MAX is the possible maximal value of \( f_{\text{exact}} \) and the SNR is defined

\[
\text{SNR} = 10 \log_{10} \frac{\| f - \bar{f} \|^2}{\| f - f_{\text{exact}} \|^2},
\]

where \( \bar{f} \) is the mean value of \( f \).

In the following tests, the TV regularization method minimizes the functional

\[
\frac{1}{2} \| K f - q \|_2^2 + \alpha \int_{\Omega} \sqrt{|f'(x)|^2 + \gamma^2} \, dx
\]

(one-dimensional case)

or

\[
\frac{1}{2} \| K f - q \|_2^2 + \alpha \int_{\Omega} \left( \left| \frac{\partial f}{\partial x} \right|^2 + \gamma^2 \right)^{1/2} + \left( \left| \frac{\partial f}{\partial y} \right|^2 + \gamma^2 \right)^{1/2} \, dx \, dy
\]

(two-dimensional case),

where \( \gamma > 0 \) is a parameter, and the WTV regularization method minimizes the functional

\[
\frac{1}{2} \| K f - q \|_2^2 + \alpha \int_{\Omega} \theta \frac{\sqrt{|f'(x)|^2 + \gamma^2}}{|f'(x)| + \theta} \, dx
\]

(one-dimensional case)

or

\[
\frac{1}{2} \| K f - q \|_2^2 + \alpha \int_{\Omega} \left( \frac{\theta}{|\partial f/\partial x| + \theta} \left( \left| \frac{\partial f}{\partial x} \right|^2 + \gamma^2 \right)^{1/2} + \frac{\theta}{|\partial f/\partial y| + \theta} \left( \left| \frac{\partial f}{\partial y} \right|^2 + \gamma^2 \right)^{1/2} \right) \, dx \, dy
\]

(two-dimensional case).

We discretize the above regularized problems in the same way as we have done in Section 3 for weighted \( H^1 \) seminorm regularized problems and solve the discretized problems by using an efficient solver for TV regularized problems – the lagged diffusivity fixed point algorithm; see for instance [32, p. 136]. For completeness, we present the main steps of the above algorithm for the WTV regularization method for two-dimensional problems.
Let
\[
\tilde{W}_x = \frac{\Delta y}{\Delta x} \text{diag} \left( \frac{w_{x,1,1}}{\phi(D_x f_{1,1})}, \ldots, \frac{w_{x,n_x-1,1}}{\phi(D_x f_{n_x-1,1})}, \frac{w_{x,1,n_y}}{\phi(D_y f_{1,n_y})}, \ldots, \frac{w_{x,n_x-1,n_y}}{\phi(D_y f_{n_x-1,n_y})} \right),
\]
(13)
\[
\tilde{W}_y = \frac{\Delta x}{\Delta y} \text{diag} \left( \frac{w_{y,1,1}}{\phi(D_y f_{1,1})}, \ldots, \frac{w_{y,n_y,1}}{\phi(D_y f_{n_y,1})}, \frac{w_{y,1,n_x-1}}{\phi(D_x f_{1,n_x-1})}, \ldots, \frac{w_{y,n_y,n_x-1}}{\phi(D_x f_{n_y,n_x-1})} \right),
\]
(14)
where \( \phi(t) = \sqrt{t^2 + \gamma^2} \). The lagged diffusivity fixed point algorithm is as follows:

**Algorithm 2 Lagged diffusivity fixed point algorithm**

1. Input an initial guess \( f_0 \);
2. \( \nu := 0 \);
3. Begin iterations
4. \( L_\nu = D_x^T \tilde{W}_x D_x + D_y^T \tilde{W}_y D_y \),
   \( \tilde{W}_x \) and \( \tilde{W}_y \) are given by Equations (13) and (14), respectively;
5. \( g_\nu = K^T (K f_\nu - q) + \alpha L_\nu f_\nu \);
6. Solve \((K^T K + \alpha L_\nu)s = -g_\nu\) by using the conjugate gradient method;
7. \( f_{\nu+1} = f_\nu + s \);
8. If \( ||s||/||f_{\nu+1}|| < \epsilon \), break;
9. \( \nu = \nu + 1 \);
10. End iterations
11. Output \( f_\nu \).

For two-dimensional problems, we solve the linear system (12) and \((K^T K + \alpha L_\nu)s = -g_\nu\) by using the conjugate gradient method with relative accuracy \(10^{-7}\). We note that the matrix–vector product \( L_\nu f\) can be efficiently obtained by using
\[
L_\nu f = D_x^T (\tilde{W}_x (D_x f)) + D_y^T (\tilde{W}_y (D_y f)).
\]
The parameter \( \epsilon \) in Step 7 of Algorithms 1 and 2 is set to \(10^{-3}\).

NLTV methods are two-step methods. In the first step, an approximate solution is obtained by solving the ill-posed linear system with a Tikhonov regularization method and the nonlocal weights are obtained by using the approximate solution. In the second step, the corresponding Euler–Lagrange equation is solved (see [21] for detail).

We downloaded the code for NLTV methods proposed in [21] from http://math.sjtu.edu.cn/faculty/xqzhang/NLIP_v1.zip. Since the downloaded code is for periodic boundary condition, we compare our weighted \(H^1\) method with NLTV methods for the case where the kernel function is periodic (see Section 4.3).

### 4.1 One-dimensional example

**Example 1** Consider Equation (2) with \( \Omega = [0, 1] \). The kernel function is given by
\[
k(x, t) = \frac{1}{\sigma \sqrt{2\pi}} e^{-(x-t)^2/2\sigma^2},
\]
Table 1. Parameter(s) chosen, relative error, PSNR, SNR, and CPU time for different methods (Example 1).

<table>
<thead>
<tr>
<th>Method</th>
<th>$(\alpha, \theta, \gamma)$</th>
<th>$E_{\text{relative}}$</th>
<th>PSNR</th>
<th>SNR</th>
<th>CPU (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H^1$</td>
<td>$(4.0e^{-3}, -, -)$</td>
<td>$1.501e^{-1}$</td>
<td>22.73</td>
<td>13.82</td>
<td>0.0118</td>
</tr>
<tr>
<td>$WH^1$</td>
<td>$(6.0e^{-1}, 1.0e^{-4}, -)$</td>
<td>$4.302e^{-2}$</td>
<td>33.59</td>
<td>24.84</td>
<td>0.4115</td>
</tr>
<tr>
<td>TV</td>
<td>$(2.0e^{-3}, -, 1.0e^{-3})$</td>
<td>$5.647e^{-2}$</td>
<td>31.22</td>
<td>22.43</td>
<td>3.1399</td>
</tr>
<tr>
<td>WTV</td>
<td>$(2.0e^{-4}, 1.0e^{-1}, 1.0e^{-3})$</td>
<td>$6.093e^{-2}$</td>
<td>30.56</td>
<td>21.83</td>
<td>0.6547</td>
</tr>
</tbody>
</table>

Figure 1. Exact solution, observed data, and numerical solutions for Example 1. (a) The exact solution $f_{\text{exact}}$, (b) an observed data $q$, (c) numerical solution ($H^1$), (d) numerical solution ($WH^1$), (e) numerical solution (TV), and (f) numerical solution (WTV).
where $\sigma = 0.05$. The right-hand side is chosen such that the exact solution is

$$f_{\text{exact}}(t) = \begin{cases} 
0.2, & t \in (0.15, 0.3), \\
0.5 - t, & t \in (0.3, 0.5), \\
3(t - 0.6)(0.9 - t), & t \in (0.6, 0.9), \\
0, & \text{otherwise}.
\end{cases}$$

In this example, the number of subintervals is set to $n = 128$, and the error vector $\eta$ has noise level $\delta = 1\%$. We use the observed data $q$ as the initial guess for all iterative methods. For

Figure 2. Exact solution, observed data, and numerical solutions for Example 2. (a) The exact solution $f_{\text{exact}}$, (b) an observed data $q$, (c) numerical solution ($H^1$), (d) numerical solution ($WH^1$), (e) numerical solution (TV), and (f) numerical solution (WTV).
Table 2. Parameter(s) chosen, relative error, PSNR, SNR, and CPU time for different methods (Example 2).

<table>
<thead>
<tr>
<th>Method</th>
<th>$\alpha, \theta, \gamma$</th>
<th>$E_{\text{relative}}$</th>
<th>PSNR</th>
<th>SNR</th>
<th>CPU (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H^1$</td>
<td>(1.0e$^{-5}$, --, --)</td>
<td>1.876e$^{-2}$</td>
<td>38.33</td>
<td>23.34</td>
<td>0.1916</td>
</tr>
<tr>
<td>WH$^1$</td>
<td>(1.0e$^{-3}$, 2.0e$^{-4}$, --)</td>
<td>1.228e$^{-2}$</td>
<td>42.18</td>
<td>27.22</td>
<td>0.7152</td>
</tr>
<tr>
<td>TV</td>
<td>(5.0e$^{-7}$, --, 1.0e$^{-3}$)</td>
<td>1.057e$^{-2}$</td>
<td>43.50</td>
<td>28.53</td>
<td>2.7224</td>
</tr>
<tr>
<td>WTV</td>
<td>(1.0e$^{-6}$, 1.0e$^{-3}$, 1.0e$^{-3}$)</td>
<td>1.015e$^{-2}$</td>
<td>43.84</td>
<td>28.91</td>
<td>7.4979</td>
</tr>
</tbody>
</table>

each method, we choose relevant parameters by several tests with the help of the discrepancy principle.

The parameter(s) chosen, the relative error, the PSNR, the SNR, and the CPU time (s) are listed in Table 1. The exact solution $f_{\text{exact}}$, an observed data $q$, and numerical results are shown in Figure 1. We can see from Figure 1 that the numerical solution obtained by using Algorithm 1 is better than those obtained by using other methods, which can be verified by comparing their relative errors, PSNRs, and SNRs. Moreover, the CPU time required by Algorithm 1 is less than those required by the TV and WTV methods.

4.2 Image restoration (zero boundary condition)

In this subsection, we present examples for restoration of images that have been contaminated by blurring and noise. We consider zero boundary condition in this subsection. All blurring matrices $K$ chosen have block Toeplitz structure, so we apply the FFT technique to compute matrix–vector multiplications, which requires $O(n_x n_y \log(n_x n_y))$ operations.

Example 2  Consider the following two-dimensional integral equation:

$$
\int_0^1 \int_0^1 k(x_1, x_2, t_1, t_2)f_{\text{exact}}(t_1, t_2) \, dt_1 \, dt_2 = g(x_1, x_2),
$$

where

$$
k(x_1, x_2, t_1, t_2) = C \, e^{- ((x_1 - t_1)^2 + (x_2 - t_2)^2)/(2\sigma^2)}, \quad \text{with} \, \sigma = 0.05, \quad C = \frac{1}{\sigma \sqrt{2\pi}},
$$

and $g(x_1, x_2)$ is chosen such that the exact solution is

$$
f_{\text{exact}}(x_1, x_2) = \begin{cases} 
2, & x_1 \in [0.1, 0.4], x_2 \in [0.1, 0.4], \\
1.5, & (x_1 - 0.75)^2 + (x_2 - 0.25)^2 \leq 0.15^2, \\
1.5x_1 + 0.3, & x_1 \in [0.1, 0.4], x_2 \in [0.65, 0.85], \\
f_4(x_1, x_2), & x_1 \in [0.6, 0.9], x_2 \in [0.6, 0.8], \\
1, & \text{elsewhere},
\end{cases}
$$

where $f_4(x_1, x_2) = 2000(x_1 - 0.6)(0.9 - x_1)(x_2 - 0.6)(0.9 - x_2)$.

In this example, we set the number of subdomains to $128 \times 128$. The exact solution $f_{\text{exact}}$, an observed data $q$ with noise level 0.1%, and numerical solutions of different methods are displayed in Figure 2. The parameter(s) chosen, the relative error, the PSNR, and the SNR of different numerical solutions, and the CPU time of different methods are shown in Table 2. From
Table 2 we see that the $WH^1$ method requires considerably less CPU time than the TV and the WTV methods.

We observe from Figure 2(d) that the $WH^1$ method can restore edges as well as details. Moreover, by comparing the restored images (Figure 2(c)–(f)), we see that among the four methods, the $WH^1$ method performs the best in restoring details (e.g. the lower-right part of the restored images).

**Example 3** Consider restoration of two 256 × 256 images, namely, Cameraman and Lena. The point spread functions for these two images are both formed by using the command ‘fspecial(“disk”,4)’ in Matlab and the noise level is 0.1%. The restored images for ‘cameraman.tif’ are shown in Figure 3 and those for ‘lena.bmp’ are shown in Figure 4. We can see from Figures 3
and 4 and Tables 3 and 4 that the restored images obtained by using the WH\textsuperscript{1} method is much better than that obtained by using the H\textsuperscript{1} method, and is better than those obtained by using the TV and the WTV methods. Moreover, the WH\textsuperscript{1} method requires considerably less CPU time than the TV and the WTV methods.

4.3 Image restoration (periodic boundary condition)

In this subsection, we use periodic kernel function to generate blurred images and then add 0.1% of noise to blurred images to get observed images. More precisely, we generate $g$ by using the
Table 3. Parameter(s) chosen, relative error, PSNR, SNR, and CPU time for different methods (Cameraman).

<table>
<thead>
<tr>
<th>Method</th>
<th>((\alpha, \theta, \gamma))</th>
<th>(E_{\text{relative}})</th>
<th>PSNR</th>
<th>SNR</th>
<th>CPU (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(H^1)</td>
<td>((2.0\cdot10^{-5}, -,-))</td>
<td>(4.809\cdot10^{-2})</td>
<td>31.94</td>
<td>19.67</td>
<td>4.9115</td>
</tr>
<tr>
<td>(WH^1)</td>
<td>((1.0\cdot10^{-3}, 1.0\cdot10^{-3}, -))</td>
<td>(3.054\cdot10^{-2})</td>
<td>35.92</td>
<td>23.94</td>
<td>9.4259</td>
</tr>
<tr>
<td>TV</td>
<td>((1.0\cdot10^{-5}, -1.0\cdot10^{-3}))</td>
<td>(3.501\cdot10^{-2})</td>
<td>34.70</td>
<td>22.41</td>
<td>18.9033</td>
</tr>
<tr>
<td>WTV</td>
<td>((1.0\cdot10^{-5}, 1.0\cdot10^{-1}, 1.0\cdot10^{-3}))</td>
<td>(3.376\cdot10^{-2})</td>
<td>34.24</td>
<td>21.98</td>
<td>27.6976</td>
</tr>
</tbody>
</table>

Table 4. Parameter(s) chosen, relative error, PSNR, SNR, and CPU time for different methods (Lena).

<table>
<thead>
<tr>
<th>Method</th>
<th>((\alpha, \theta, \gamma))</th>
<th>(E_{\text{relative}})</th>
<th>PSNR</th>
<th>SNR</th>
<th>CPU (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(H^1)</td>
<td>((4.0\cdot10^{-5}, -,-))</td>
<td>(4.248\cdot10^{-2})</td>
<td>34.61</td>
<td>20.80</td>
<td>4.3284</td>
</tr>
<tr>
<td>(WH^1)</td>
<td>((2.0\cdot10^{-4}, 6.0\cdot10^{-3}, -))</td>
<td>(3.227\cdot10^{-2})</td>
<td>36.99</td>
<td>23.20</td>
<td>7.4602</td>
</tr>
<tr>
<td>TV</td>
<td>((1.0\cdot10^{-5}, -1.0\cdot10^{-3}))</td>
<td>(3.698\cdot10^{-2})</td>
<td>35.81</td>
<td>22.00</td>
<td>18.3003</td>
</tr>
<tr>
<td>WTV</td>
<td>((1.0\cdot10^{-5}, 5.0\cdot10^{-1}, 1.0\cdot10^{-3}))</td>
<td>(3.725\cdot10^{-2})</td>
<td>34.97</td>
<td>21.16</td>
<td>21.7059</td>
</tr>
</tbody>
</table>

Table 5. Parameter(s) chosen, relative error, PSNR, SNR, and CPU time for different methods (\(f_{\text{exact}}\) given by Equation (15)).

<table>
<thead>
<tr>
<th>Method</th>
<th>((\alpha, \theta, \gamma, \sigma, \lambda))</th>
<th>(E_{\text{relative}})</th>
<th>PSNR</th>
<th>SNR</th>
<th>CPU (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(H^1)</td>
<td>((1.0\cdot10^{-4}, -,-,-,))</td>
<td>(2.805\cdot10^{-2})</td>
<td>35.02</td>
<td>20.04</td>
<td>0.7795</td>
</tr>
<tr>
<td>(WH^1)</td>
<td>((5.0\cdot10^{-2}, 2.0\cdot10^{-3}, -,-,-,))</td>
<td>(4.934\cdot10^{-3})</td>
<td>50.11</td>
<td>35.16</td>
<td>0.8433</td>
</tr>
<tr>
<td>TV</td>
<td>((1.0\cdot10^{-4}, 1.0\cdot10^{-3}, -,-,-,))</td>
<td>(4.433\cdot10^{-3})</td>
<td>51.04</td>
<td>36.09</td>
<td>1.7709</td>
</tr>
<tr>
<td>WTV</td>
<td>((5.0\cdot10^{-4}, 1.0\cdot10^{-3}, 1.0\cdot10^{-3}, -,-,-,))</td>
<td>(5.536\cdot10^{-3})</td>
<td>49.11</td>
<td>34.18</td>
<td>5.7165</td>
</tr>
<tr>
<td>NLTV(_A)</td>
<td>((-,-,-,1.0\cdot10^{-3}, 9.0\cdot10^{+3}))</td>
<td>(8.208\cdot10^{-3})</td>
<td>45.69</td>
<td>30.74</td>
<td>22.6673</td>
</tr>
</tbody>
</table>

Table 6. Parameter(s) chosen, relative error, PSNR, SNR, and CPU time for different methods (Cameraman).

<table>
<thead>
<tr>
<th>Method</th>
<th>((\alpha, \theta, \gamma, \sigma, \lambda))</th>
<th>(E_{\text{relative}})</th>
<th>PSNR</th>
<th>SNR</th>
<th>CPU (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(H^1)</td>
<td>((2.0\cdot10^{-5}, -,-,-,))</td>
<td>(4.799\cdot10^{-2})</td>
<td>31.96</td>
<td>19.68</td>
<td>3.9473</td>
</tr>
<tr>
<td>(WH^1)</td>
<td>((1.0\cdot10^{-3}, 1.0\cdot10^{-3}, -,-,-,))</td>
<td>(3.054\cdot10^{-2})</td>
<td>35.89</td>
<td>23.62</td>
<td>6.8680</td>
</tr>
<tr>
<td>TV</td>
<td>((1.0\cdot10^{-5}, -1.0\cdot10^{-3}, -,-,-,))</td>
<td>(3.489\cdot10^{-2})</td>
<td>34.73</td>
<td>22.44</td>
<td>15.6228</td>
</tr>
<tr>
<td>WTV</td>
<td>((1.0\cdot10^{-5}, 1.0\cdot10^{-1}, 1.0\cdot10^{-3}, -,-,-,))</td>
<td>(3.368\cdot10^{-2})</td>
<td>35.04</td>
<td>22.78</td>
<td>22.9726</td>
</tr>
<tr>
<td>NLTV(_A)</td>
<td>((-,-,-,1.0\cdot10^{-4}, 1.2e+3))</td>
<td>(5.123\cdot10^{-2})</td>
<td>31.39</td>
<td>19.08</td>
<td>3.4147</td>
</tr>
<tr>
<td>NLTV(_M)</td>
<td>((-,-,-,1.0\cdot10^{-4}, 1.2e+3))</td>
<td>(5.065\cdot10^{-2})</td>
<td>31.49</td>
<td>19.18</td>
<td>11.8827</td>
</tr>
</tbody>
</table>

Matlab function ‘imfilter’

\[ g = \text{imfilter}(f_{\text{exact}}, \text{ker}, \text{‘circular’}, \text{‘conv’}) \]

where ‘ker’ denotes the kernel function \(\text{fspecial(‘disk’,4)}\). The exact solution of Example 2 (i.e. \(f_{\text{exact}}\) defined by Equation (15)), Cameraman, and Lena will be tested in the following.

In this subsection, we compare the performance of six methods, namely, \(H^1\), weighted \(H^1\), TV, WTV, and two NLTV methods. All parameters of the first NLTV method are automatically selected (the standard deviation \(\sigma\) of the noised is obtained by using \(g\) and \(d\)). The parameters \(\sigma\) and \(\lambda\) in the second NLTV method are selected manually, where \(\lambda\) is the regularization parameter of the functional

\[ E(f) = TV_{\text{NL}}(f) + \frac{\lambda}{2} \|Kf - q\|_2^2 \]
Table 7. Parameter(s) chosen, relative error, PSNR, SNR, and CPU time for different methods (Lena).

<table>
<thead>
<tr>
<th>Method</th>
<th>$(\alpha, \theta, \gamma, \sigma, \lambda)$</th>
<th>$E_{relative}$</th>
<th>PSNR</th>
<th>SNR</th>
<th>CPU (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H^1$</td>
<td>$(4.0 \times 10^{-5}, -, -, -, -)$</td>
<td>$4.274 \times 10^{-2}$</td>
<td>34.56</td>
<td>20.75</td>
<td>3.5008</td>
</tr>
<tr>
<td>$WH^1$</td>
<td>$(4.0 \times 10^{-4}, 2.0 \times 10^{-3}, -, -, -)$</td>
<td>$3.203 \times 10^{-2}$</td>
<td>37.06</td>
<td>23.27</td>
<td>6.3518</td>
</tr>
<tr>
<td>TV</td>
<td>$(1.0 \times 10^{-5}, -, 1.0 \times 10^{-3}, -)$</td>
<td>$3.706 \times 10^{-2}$</td>
<td>35.79</td>
<td>21.98</td>
<td>15.5317</td>
</tr>
<tr>
<td>WTV</td>
<td>$(1.0 \times 10^{-5}, 5.0 \times 10^{-1}, 1.0 \times 10^{-3}, -)$</td>
<td>$3.750 \times 10^{-2}$</td>
<td>35.69</td>
<td>21.89</td>
<td>15.6822</td>
</tr>
<tr>
<td>NLTV$_A$</td>
<td>$-$</td>
<td>$4.731 \times 10^{-2}$</td>
<td>33.67</td>
<td>19.85</td>
<td>3.2305</td>
</tr>
<tr>
<td>NLTV$_M$</td>
<td>$(-, -,-, 3.0 \times 10^{-4}, 4.8 \times 10^{+4})$</td>
<td>$4.295 \times 10^{-2}$</td>
<td>34.51</td>
<td>20.68</td>
<td>93.7449</td>
</tr>
</tbody>
</table>

Figure 5. Restored results of $f_{exact}$ given by Equation (15), ‘Cameraman’, and ‘Lena’ obtained by using NLTV$_A$ (left column) and NLTV$_M$ (right column). (a) Restored results of $f_{exact}$ (given by Equation (15)) obtained by using NLTV methods, (b) images of ‘Cameraman’ obtained by using NLTV methods, and (c) restored images of ‘Lena’ obtained by using NLTV methods.
with $\text{TV}_{\text{NL}}(f)$ being the nonlocal WTV operator. The above NLTV methods are denoted by NLTV$_A$ and NLTV$_M$, respectively.

The parameter(s) chosen, the relative error, the PSNR, and the SNR of restored images, and the CPU time for the above methods are presented in Tables 5–7. We only present the restored images obtained by using NLTV methods (see Figure 5) since other images are similar to those in Section 4.2.

We see from Tables 5–7 that the WH$^1$ method requires more CPU time than the NLTV$_A$ method. On the other hand, the numerical solution obtained by using the WH$^1$ method is better than that obtained by using the NLTV$_A$ method, which can be seen from Figure 5 and verified by comparing their relative errors, PSNRs, and SNRs. We also see that the WH$^1$ method requires less CPU time than the NLTV$_M$ method. Moreover, the quality of the numerical solution obtained by using the WH$^1$ method is better than that obtained by using the NLTV$_M$ method.

5. Concluding remarks

In this paper, we proposed a regularization method based on a weighted $H^1$ seminorm for estimating piecewise continuous solution of Fredholm integral equations of the first kind. Detailed implementation for one-dimensional and two-dimensional problems is presented. Numerical results show that the proposed approach is effective for obtaining reasonable numerical solutions. In our future research, we will consider more efficient solution methods for normal Equations (10) and (12), say, introduce suitable preconditioners. We will also study determination methods for relevant parameters.

Acknowledgements

We thank the anonymous referees for their valuable suggestions. The research was supported by NSF of China No. 11271238 and Guangdong Provincial NSF No. 10151503101000023.

References


