Numerical Optimization
Tikhonov Regularization

Stephan Gelever[*]

December 5, 2018

Abstract

Final project on image restoration/deblurring.

1 Goal

Given blurry image

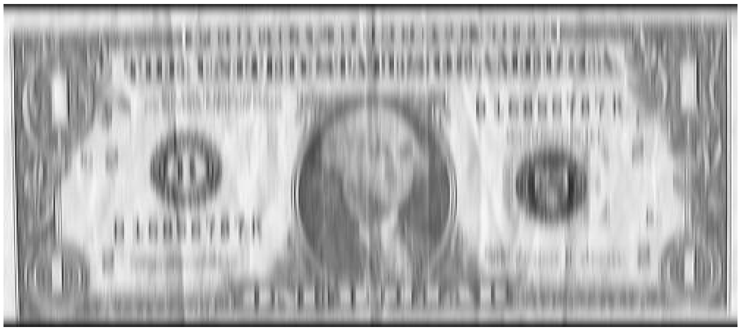

Figure 1: Blurry Image

and a blurring matrix A, reconstruct an approximate image $\tilde{X}$ such that the serial number is visible. Also, implement the CG algorithm to solve the linear system.

[*] Email: gelever@pdx.edu

2 Results

The most clear image was obtained with $\lambda \approx 5.15e-06$. This value was initialy obtained by doing truncated SVD and checking when the image becomes clear.

The serial number is clearly `B16856787K`. Using the table found at

`http://www.onedollarbill.org/decoding.html`

this dollar bill was printed in New York.

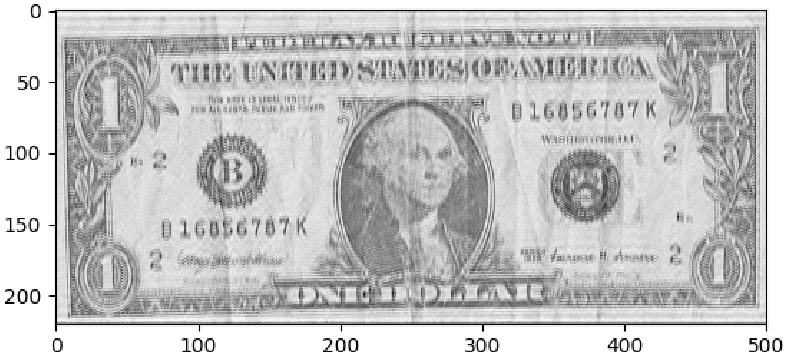

Figure 2: Restored Image - $\lambda \approx 5.15e-06$

3 Program

```
# Copyright Stephan Gelever 2018
# Numerical Optimization - Final Project

import matplotlib.pyplot as plt
import numpy as np

def block_cg(A, b, x=None, tol=1e-8, max_iter=2000):
    if not x:
        x = np.zeros_like(b)

    return np.asarray([cg(A, b_i, x_i, tol, max_iter)
                  for b_i, x_i in zip(b.T, x.T)]).T

def cg(A, b, x=None, tol=1e-15, max_iter=2000):
    if x is None:
        x = np.zeros_like(b)

    r = b - A.dot(x)
    p = r

    num_iter = 0
    tol_sqr = tol * tol

    while r.dot(r) > tol and num_iter < max_iter:
        Ap = A.dot(p)

        pAp = p.dot(Ap)
        r_r = r.dot(r)

        alpha = r_r / pAp

        x = x + (alpha * p)
        r = r - (alpha * Ap)

        r_r_next = r.dot(r)

        if r_r_next < tol_sqr:
            break

        beta = r_r_next / r_r

        p *= beta
        p += r
        num_iter += 1

    return x

def build_A(n, L=0.45):
    B = np.zeros((n,n))

    for i in range(n):
        B[i, i] = 1.0 - (2.0 * L)

    for i in range(n - 1):
        B[i, i + 1] = L
        B[i + 1, i] = L

    return np.linalg.matrix_power(B, 25)


def tikhonov(lmbda, A, D):
    AT = A.transpose()

    ATA = AT.dot(A)
    AT_d = AT.dot(D)

    lmbda_I = (l*l) * np.eye(n)
    ATAI = ATA + lmbda_I

    #x_hat = np.linalg.solve(ATAI, AT_d)
    x_hat = block_cg(ATAI, AT_d)

    return x_hat


D = np.loadtxt("dollarblur.m", dtype=float)
n,m = D.shape
A = build_A(n)

for l in np.linspace(1e-7, 1e-4, 80):
    print("l: {}".format(l))

    x_hat = tikhonov(l, A, D)

    imgplt = plt.imshow(x_hat)
    imgplt.set_cmap('Greys_r')
    plt.show()
```