Report of Bandlimited signal recovery from zero-crossings

December 24, 2023

1 Report of Bandlimited signal recovery from zero-crossings

12334125

1.1 0.1

Bandlimited signal recovery from zero-crossings. Let $y \in \mathbb{R}^n$ denote a bandlimited signal, which means that it can be expressed as a linear combination of sinusoids with frequencies in a band:

$$y_t = \sum_{j=1}^{B} a_j \cos\left(\frac{2\pi}{n}(f_{\min} + j - 1)t\right) + b_j \sin\left(\frac{2\pi}{n}(f_{\min} + j - 1)t\right), \quad t = 1, \dots, n,$$

where $f_{\min}$ is the lowest frequency in the band, B is the bandwidth, and $a, b \in \mathbb{R}^B$ are the cosine and sine coefficients, respectively. We are given $f_{\min}$ and B, but not the coefficients a, b or the signal y.

We do not know y, but we are given its sign $s = \text{sign}(y)$, where $s_t = 1$ if $y_t \geq 0$ and $s_t = -1$ if $y_t < 0$. (Up to a change of overall sign, this is the same as knowing the 'zero-crossings' of the signal, i.e., when it changes sign. Hence the name of this problem.)

We seek an estimate $\hat{y}$ of y that is consistent with the bandlimited assumption and the given signs. Of course we cannot distinguish y and αy, where $\alpha > 0$, since both of these signals have the same sign pattern. Thus, we can only estimate y up to a positive scale factor. To normalize $\hat{y}$, we will require that $\|\hat{y}\|_1 = n$, i.e., the average value of $|y_i|$ is one. Among all $\hat{y}$ that are consistent with the bandlimited assumption, the given signs, and the normalization, we choose the one that minimizes $\|\hat{y}\|_2$.

(a) Show how to find $\hat{y}$ using convex or quasiconvex optimization.

(b) Apply your method to the problem instance with data in `zero_crossings_data.*`. The data files also include the true signal y (which of course you cannot use to find $\hat{y}$). Plot $\hat{y}$ and y, and report the relative recovery error, $\|\hat{y} - y\|_2 / \|y\|_2$. Give one short sentence commenting on the quality of the recovery.

1.2 0.2

$y \in \mathbb{R}^n$

$$y_t = \sum_{j=1}^{B} a_j \cos\left(\frac{2\pi}{n}(f_{\min}+j-1)t\right) + b_j \sin\left(\frac{2\pi}{n}(f_{\min}+j-1)t\right), \quad t=1,\dots,n,$$

$f_{\min}$ B $a, b \in \mathbb{R}^B$ $f_{\min}$ B a, b y

y $s = \mathrm{sign}(y)$ $s_t = 1$ $y_t \geq 0$ $s_t = -1$ $y_t < 0$, ,

y $\hat{y}$ y αy $\alpha > 0$ y $\hat{y}$ $\|\hat{y}\|_1 = n$ $|y_i|$ $\hat{y}$ $\|\hat{y}\|_2$

(a) $\hat{y}$

(b) `zero_crossings_data.*` y $\hat{y}$ $\hat{y}$ y $\|\hat{y}-y\|_2/\|y\|_2$

1.3 1.1

(a) $\hat{y} = Ax$ $x = (a, b) \in \mathbb{R}^{2B}$ $A = [C \quad S] \in \mathbb{R}^{n\times 2B}$ $C, S \in \mathbb{R}^{n\times B}$

$$C_{tj} = \cos\left(\frac{2\pi(f_{\min}+j-1)t}{n}\right), \quad S_{tj} = \sin\left(\frac{2\pi(f_{\min}+j-1)t}{n}\right),$$

$\hat{y}$ s $s_t a_t^T x \geq 0$ $t = 1,\dots,n$ $a_1^T,\dots,a_n^T$ A $\|\hat{y}\|_1 = s^T A x = n$ l_1-

$$\begin{array}{ll} \min & \|Ax\|_2 \\ \text{s.t.} & s_t a_t^T x \geq 0, \qquad t=1,\dots,n \\ & s^T A x = n. \end{array}$$

$\hat{y} = Ax^*$ x^*

(b) python cvxpy RRMS 0.12083136047814856

zero_crossings_data.py zero_crossings_data

```
[ ]: # data for problem on bandlimited signal recovery from zero crossings
import numpy as np

def zero_crossings_data():
    n = 2048
    f_min = 4
    B = 9
```

```
s = np.array([-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,
↪-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
↪1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
↪1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
↪1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
```

```
    return n, f_min, B, s, y
```

optimization_solve.py

```
[ ]: import numpy as np
     import cvxpy as cvx

     def optimization_solve(n, f_min, B, s, y):
         #    A
         C = np.zeros((n, B))
         S = np.zeros((n, B))
         for j in range(B):
             C[:, j] = np.cos(2 * np.pi * (f_min + j) * np.arange(1, n + 1) / n)
             S[:, j] = np.sin(2 * np.pi * (f_min + j) * np.arange(1, n + 1) / n)
         A = np.hstack((C, S))

         #         L1
         x = cvx.Variable(2 * B)
         obj = cvx.norm(A @ x)
         constraints = [cvx.multiply(s, A @ x) >= 0, s.T @ (A @ x) == n]
         problem = cvx.Problem(cvx.Minimize(obj), constraints)
         problem.solve(solver=cvx.ECOS)

         y_recovered = A @ x.value

         # RRMS Relative Root Mean Square Error
         RRMS = np.linalg.norm(y - y_recovered) / np.linalg.norm(y)
         print('Optimize Success')

         return y_recovered, RRMS
```

results_output.py

```
[ ]: import matplotlib.pyplot as plt
     import numpy as np
     import pandas as pd
     import os

     def plot_and_save_with_csv(y1_data, title, xlabel, ylabel, output_folder_path,␣
      ↪fig_size=(24, 5), plot_type='plot', y2_data=None, y2_label=None,␣
      ↪legend1=None, legend2=None):
         #
         if not os.path.exists(output_folder_path):
             os.makedirs(output_folder_path)

         #
```

```
    fig, ax1 = plt.subplots(figsize=fig_size)

    if plot_type == 'plot':
        ax1.plot(y1_data, color='tab:blue', label=legend1)
    elif plot_type == 'stem':
        ax1.stem(y1_data, linefmt='tab:blue', markerfmt='bo', label=legend1)

    ax1.set_xlabel(xlabel)
    ax1.set_ylabel(ylabel)
    ax1.tick_params(axis='y')

    #    Y
    if y2_data is not None and y2_label is not None:
        ax2 = ax1.twinx()
        ax2.plot(y2_data, '--', color='tab:red', label=legend2)
        ax2.set_ylabel(y2_label)
        ax2.tick_params(axis='y')

    plt.title(title)

    #
    if legend1 is not None:
        ax1.legend(loc='upper left')
    if y2_data is not None and legend2 is not None:
        ax2.legend(loc='upper right')

    plt.tight_layout()

    #
    safe_title = title.replace(' ', '_').replace('.', '').replace('-', '_')
    image_file_name = f"{safe_title}.jpg"
    plt.savefig(os.path.join(output_folder_path, image_file_name))
    plt.show()

    #       CSV
    csv_file_name = f"{safe_title}.csv"
    if y2_data is not None:
        #
        y1_data = [y1_data] if np.isscalar(y1_data) else y1_data
        y2_data = [y2_data] if np.isscalar(y2_data) else y2_data
        combined_data = pd.DataFrame({legend1: y1_data, legend2: y2_data})
    else:
        y1_data = [y1_data] if np.isscalar(y1_data) else y1_data
        combined_data = pd.DataFrame({legend1: y1_data})
    combined_data.to_csv(os.path.join(output_folder_path, csv_file_name),␣
↪index=False, header=True)
```

main.py main.py zero_crossings_data.py optimization_solve.py

[ ]:
```python
import os
from module.zero_crossings_data import zero_crossings_data
from module.optimization_solve import optimization_solve
from module.results_output import plot_and_save_with_csv

n, f_min, B, s, y = zero_crossings_data()

y_recovered, RRMS = optimization_solve(n, f_min, B, s, y)

#
current_dir = os.path.dirname(os.path.realpath(__file__))
results_output_folder_path = os.path.join(current_dir, 'outputs/results')
error_output_folder_path = os.path.join(current_dir, 'outputs/errors')

#
#     y   y_recovered
plot_and_save_with_csv(y, 'original and recovered bandlimited signals', 'Sample Index', 'bandlimited signals', results_output_folder_path, fig_size=(24, 5), y2_data=y_recovered, y2_label='bandlimited signals', legend1='original', legend2='recovered')
plot_and_save_with_csv(RRMS, 'RRMS Error', 'Iteration', 'Error', error_output_folder_path, fig_size=(24, 5))
```

y y_recovered