MITSUBISHI ELECTRIC RESEARCH LABORATORIES
Cambridge, Massachusetts

Edge-enhancing filters with negative weights

Andrew Knyazev (knyazev@merl.com)

Outline:

- Traditional graph-based filter via graph Laplacian
- Iterated filter as the power method
- Traditional graph Laplacian and the mass-spring model
- 1D signals, cosine transform, and vibration modes of a string
- Low frequency eigenmodes with flat ends need sharpening
- Negative weights in the adjacency matrix as repulsion
- Edge-preserving vs. edge-enhancing 1D filter performance
- Conclusions
- Future work

The main claim:

Negative weights in the adjacency matrix used for graph-based edge-preserving signal smoothing enhance the edges in the signal!

Traditional graph-based filter via graph Laplacian

A discrete function $x[j]$, $j \in \{1, 2, \ldots, N\}$, is an input signal to filter. The output signal $y[i]$ is a weighted average of the signal values $x[j]$:

$$y[i] = \sum_j \frac{w_{ij}}{\sum_j w_{ij}} x[j].$$

The weights $w_{ij} \geq 0$ are the entries of a symmetric matrix W, which is interpreted as an adjacency matrix of a graph. Let D be the diagonal matrix with the non-zero diagonal entries $d_i = \sum_j w_{ij}$. The filter can then be equivalently written as the following vector transform

$$y = D^{-1} W x.$$

The symmetric nonnegative definite matrix $L = D - W$ is called the (graph) Laplacian, and $D^{-1}L = I - D^{-1}W$ in the normalized Laplacian,

$$y = x - D^{-1} L x.$$

Bilateral, guided, and total variation filters can be written in this form.

Iterated filter as the power method

The filter transform $y = D^{-1}Wx$ can be applied iteratively,

$$x_{i+1} = D^{-1}Wx_i,\ i = 0, 1, \ldots, m < \infty,$$

which is the so-called *power method.* Two options:

1. Linear filter: using some fixed weights, calculated from a guidance signal, for all iterations.
2. Nonlinear filter: updating the weights w_{ij} at least at some iterations using the result of the previous iterations.

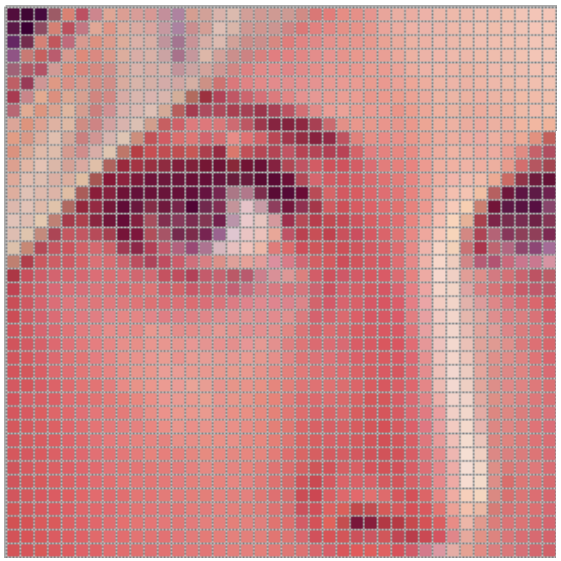
In the linear case, the iterative matrix $F = D^{-1}W$ is fixed and diagonalizable, thus, the power method mathematically gives

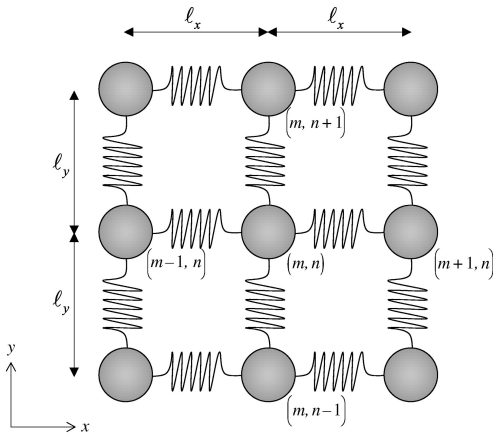
$$x_m = (F)^m\, x_0 = \Sigma_j\, \mu_j^m \left(v_j^T D x_0\right) v_j,$$

where $1 = |\mu_1| \geq |\mu_2| \geq \ldots$ are the eigenvalues of the matrix $D^{-1}W$ corresponding to the eigenvectors v_j, i.e. it suppresses the noisy part of the signal—contributions corresponding to the smallest eigenvalues μ.

Traditional graph Laplacian and the mass-spring model I

Possible simple setup for 2D imaging using a guidance signal g,





Bilateral Filter

$$w_{ij} = \exp\left(-\frac{(g[i]-g[j])^2}{2\sigma_r^2}\right)$$

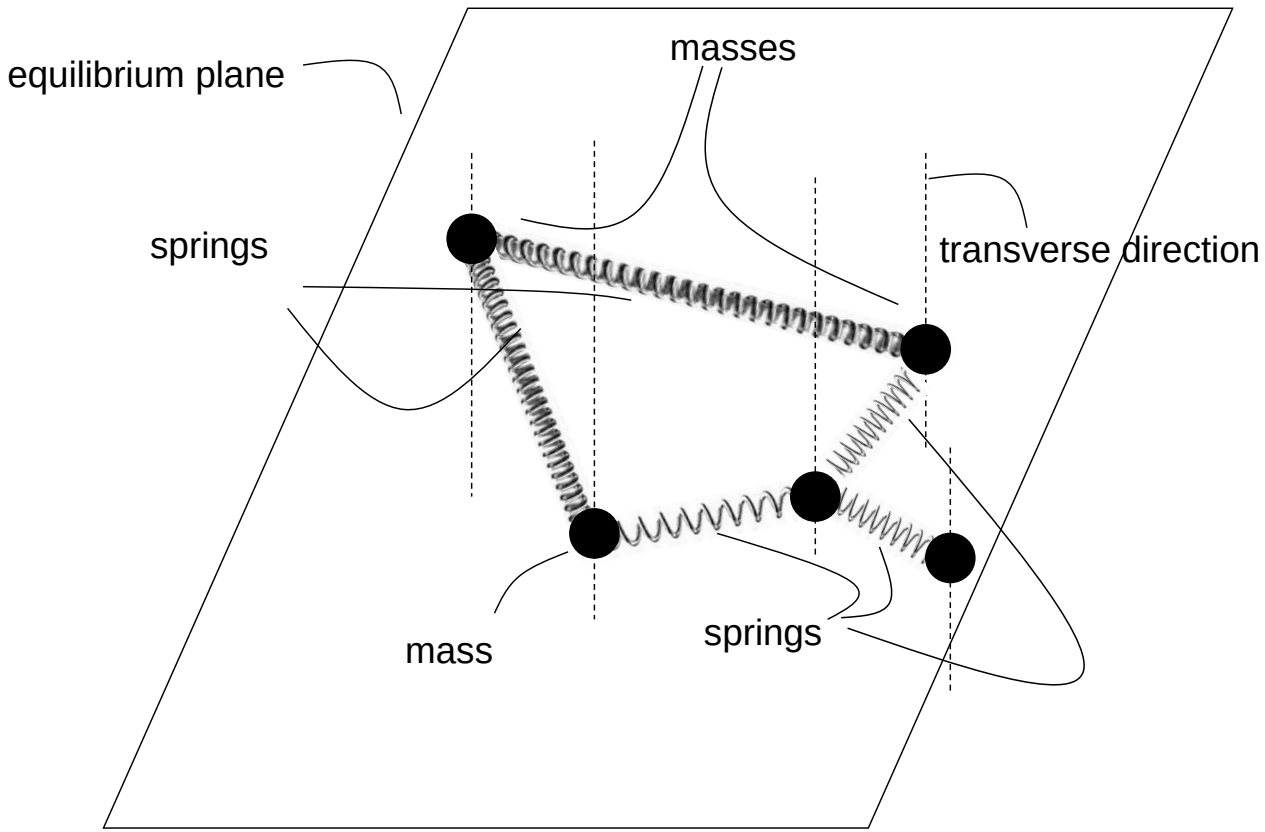
Spring stiffness

$$k_{ij} = w_{ij}$$

Pixels/masses i and j that are weakly correlated, i.e. with very different values $g[i]$ and $g[j]$, are weakly connected, i.e. the corresponding graph weight w_{ij}/spring stiffness k_{ij} is relatively small.

Traditional graph Laplacian and the mass-spring model II

The eigenvectors of the graph Laplacian are mathematically the same as the eigenmodes of the transversal vibrations of the mass-spring system!



1D signals, cosine transform, and vibration modes of a string I

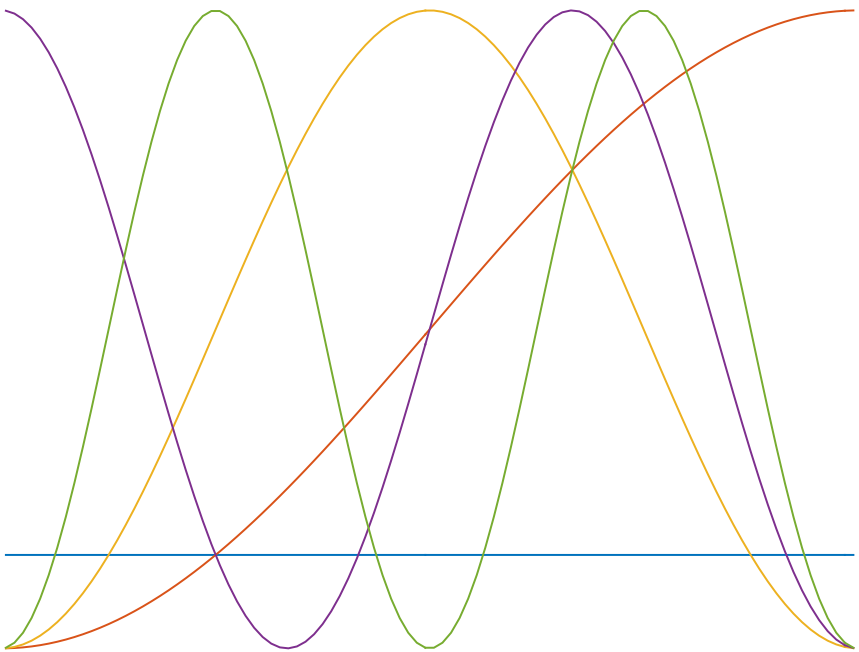
Take a scalar signal on a one-dimensional uniform grid, where the weights w_{ij} are computed only for the nearest neighbors and set to zero otherwise. Let us start with a constant guidance signal g, with $g[i] - g[j] = 0$. Then, $w_{i-1\,i} = w_{i\,i} = w_{i\,i+1} = 1$ and the graph Laplacian $L = D - W$, based on g, is the following tridiagonal matrix

$$L = \begin{pmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 1 \end{pmatrix}$$

This is a standard three-point-stencil finite-difference approximation of the negative second derivative of functions with homogeneous Neumann boundary conditions, i.e. vanishing first derivatives at the end points of the interval. This is a discrete model of a string with free ends.

1D signals, cosine transform, and vibration modes of a string II

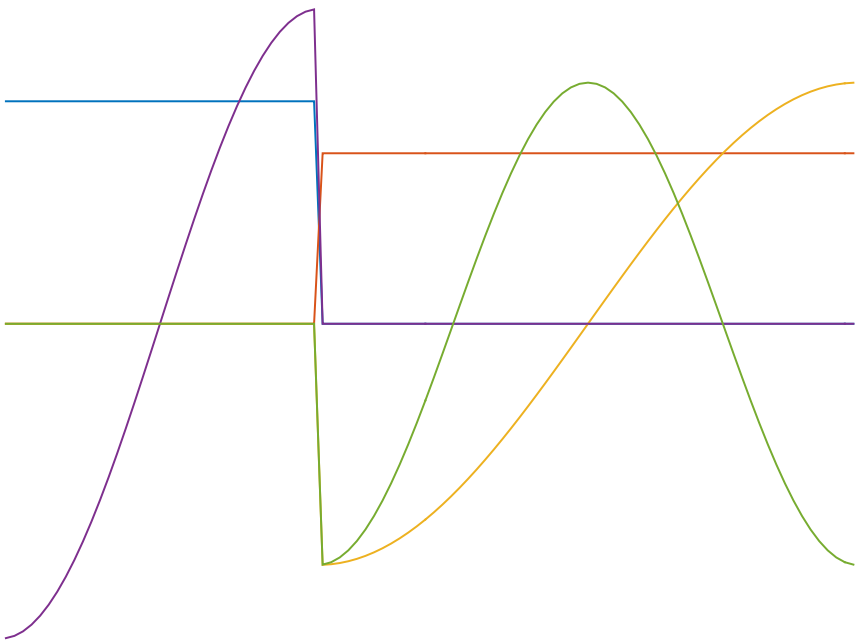
Eigenvectors of the Laplacian in this case are the basis vectors of the Discrete Cosine Transform (DCT). Displayed below are the first five low frequency eigenmodes (the eigenvectors corresponding to the smallest eigenvalues) of L—these are vibration modes of a detached string.



The graph-based bilateral denoising filter turns into the classical DCT low-pass filter in this example.
Notice that the end points of the eigenmodes are flat—the ends of the string are free, i.e. not attached.

Low frequency eigenmodes with flat ends need sharpening I

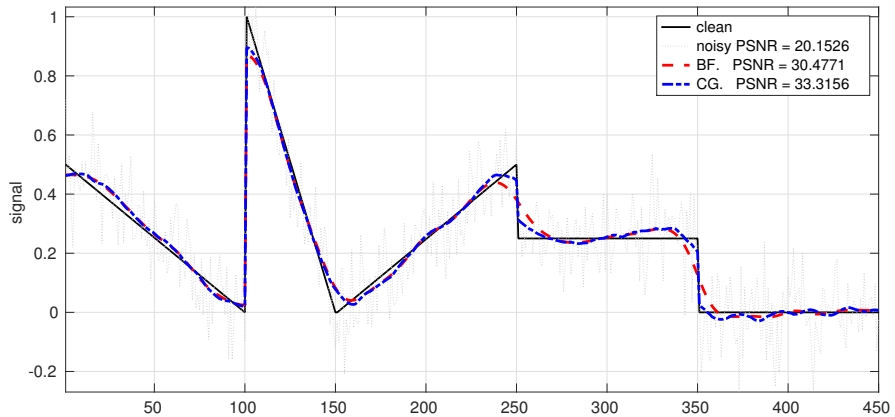
Next, we consider a piece-wise constant guidance signal, where $g[i]$ is very different from $g[i+1] = 0$ for some index i. BF weights change, there is now a very small value $w_{i\,i+1} = w_{i+1\,i}$ for some index i, keeping all other entries of the matrix W the same as before. Low frequency eigenmodes of L react to such a change in the adjacency matrix W:

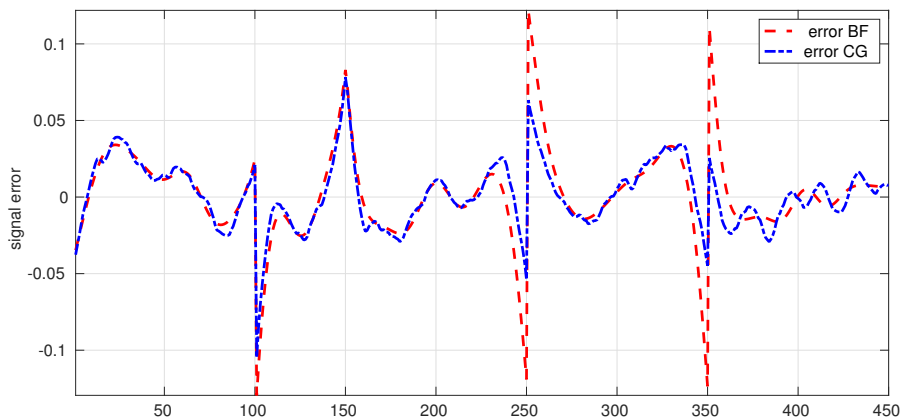


The eigenmodes now have a jump between the indexes i and $i+1$, leading to the edge preserving denoising low-pass filter, as we want. The end points of the eigenmodes are flat at the jump, since the 2 sub-strings are barely attached.

Low frequency eigenmodes with flat ends need sharpening II

Let us check it out, using iterated BF and CG BF, as the previous talk:



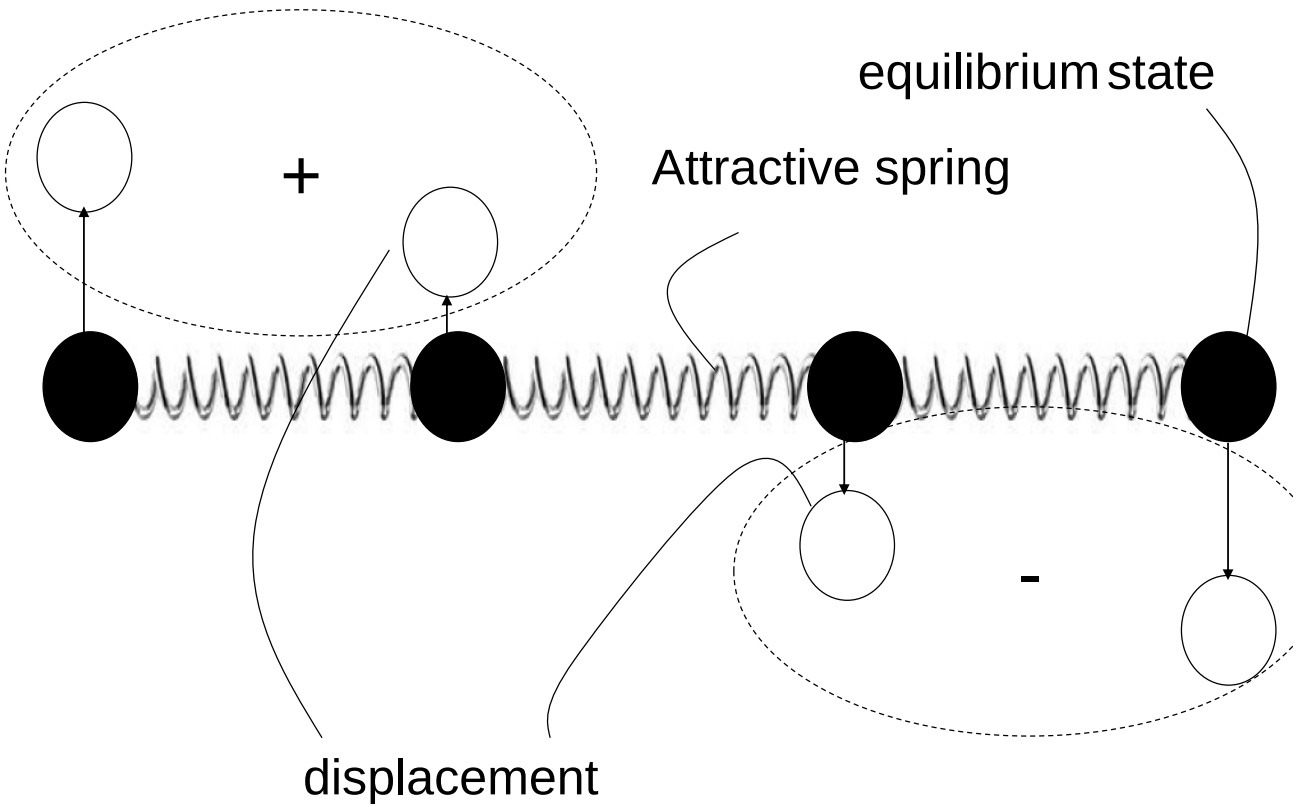


The end points of the eigenmodes are flat at every jump in the signal, no matter if the signal is flat or has sharp corners at the jump. Thus, the sharp corners are difficult to keep sharp while denoising the signal.

To better reproduce the corners, we need to sharpen the flat ends of the eigenmodes at the jump.

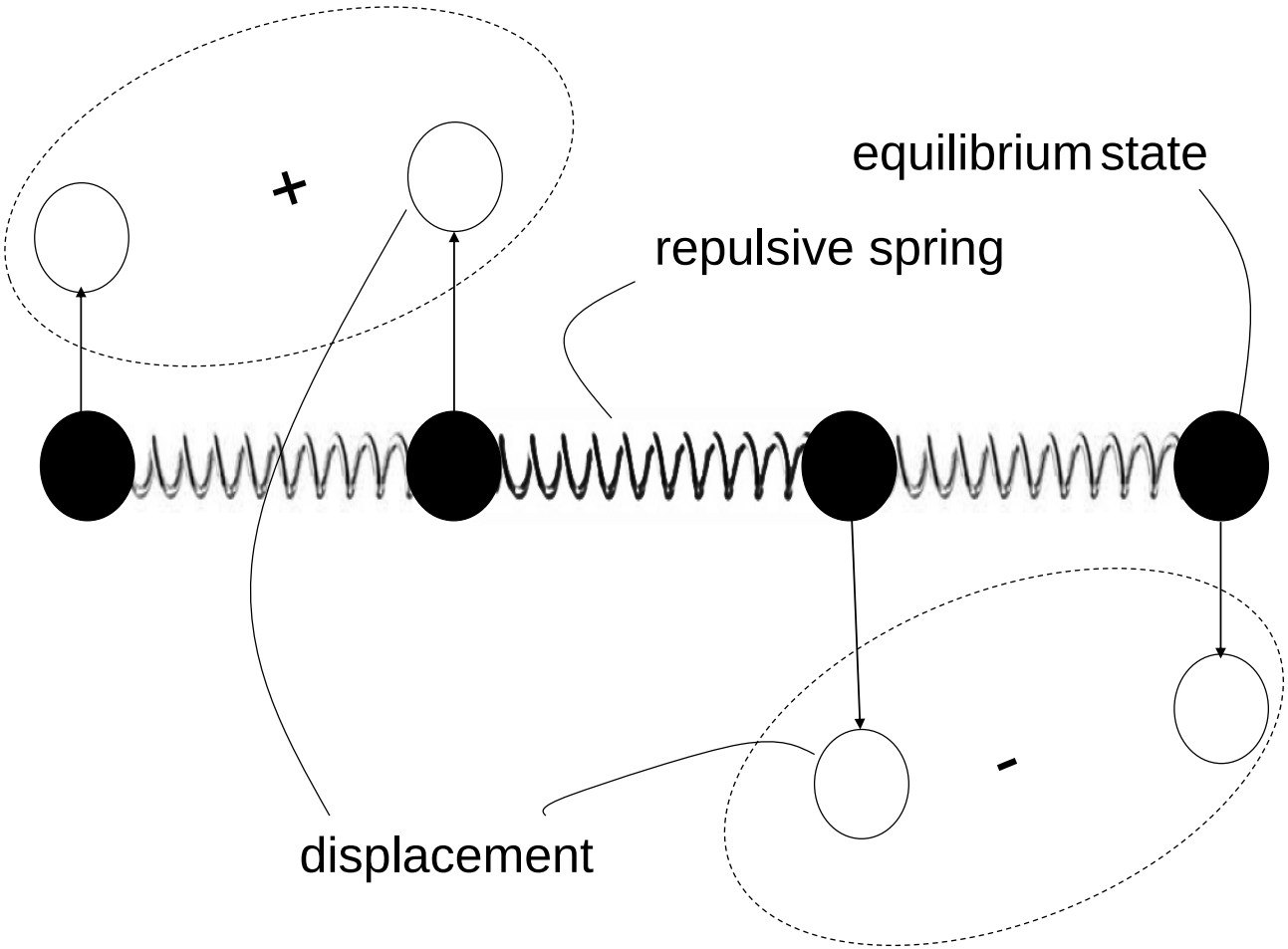
Low frequency eigenmodes with flat ends need sharpening III

Back to the drawing board, let us use the mass-spring model and intuition. Notice that the standard model uses only attractive springs.



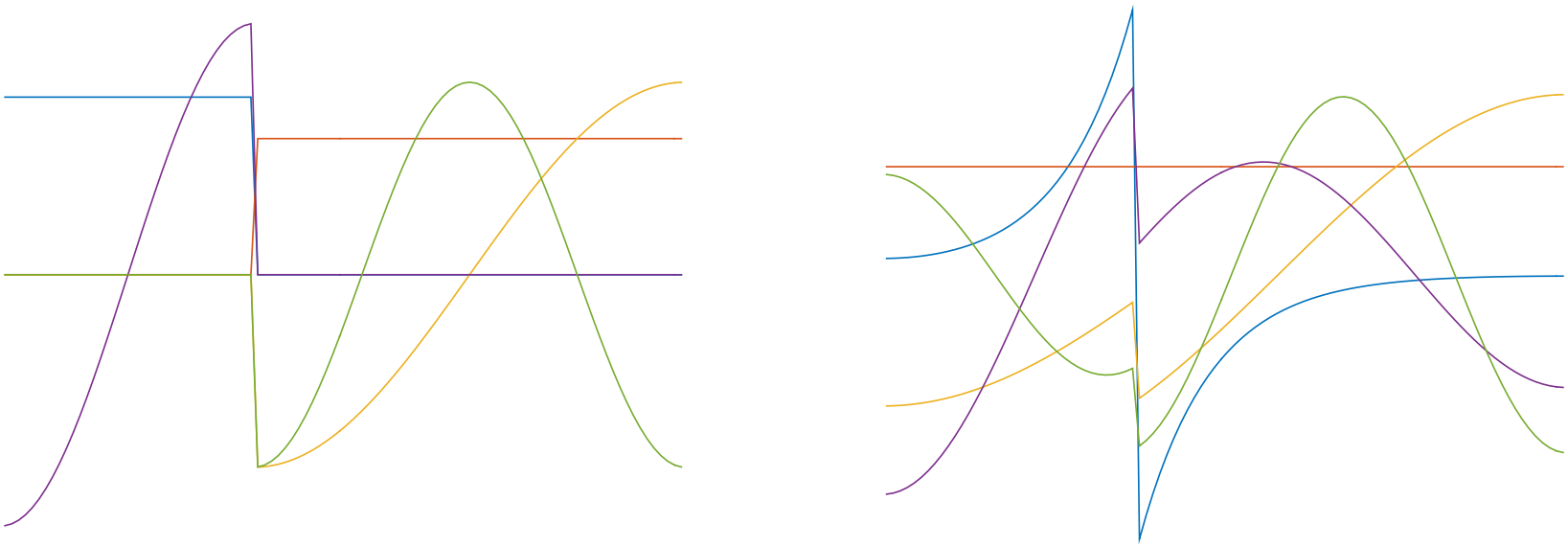
Negative weights in the adjacency matrix as repulsion I

Let us substitute one attractive spring with a repulsive spring!



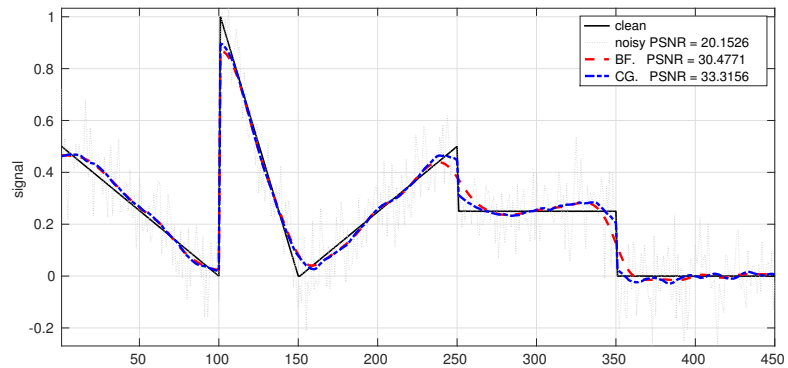
Negative weights in the adjacency matrix as repulsion II

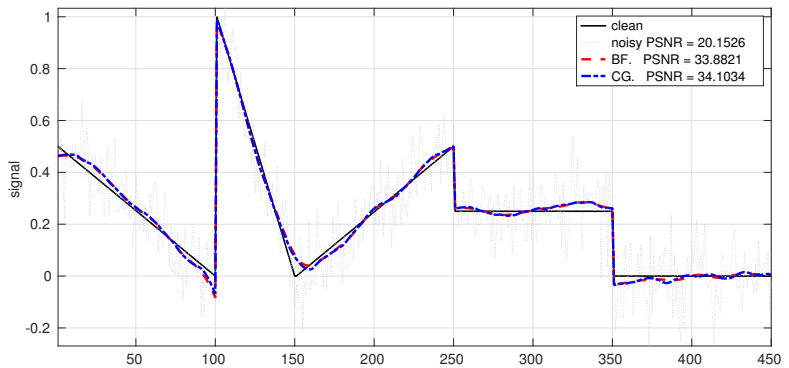
A repulsive spring corresponds to a negative edge weight in the graph, connecting (repulsing, we must say) the two corresponding vertices. The flat ends of the string eigenmodes (left panel) turn upwards and now form sharp corners (right panel, $w_{i\,i+1} = w_{i+1\,i} = -0.05$)! Great! Voila!

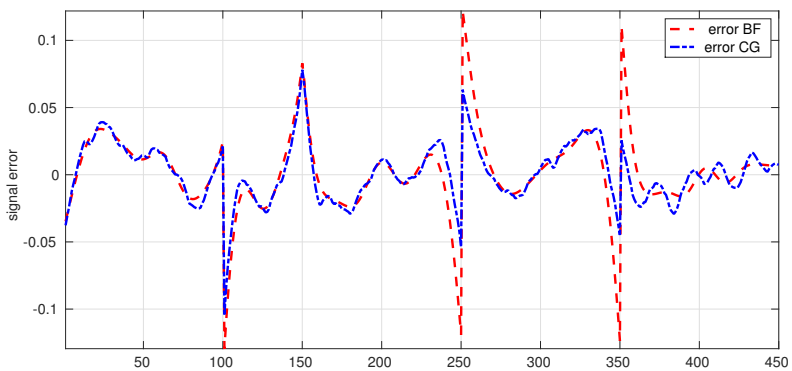


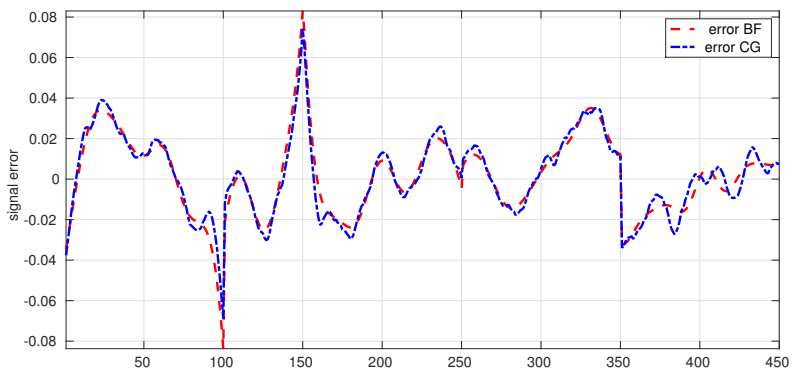
Idea: use negative graph weights at the points of the signal jumps to enhance the edges in the denoised signal by graph-based low-pass filters.

Edge-preserving vs. edge-enhancing 1D filter performance









Dramatic improvement both in PSNR and edge matching!

Full disclosure: the negative weights are hand-tuned $-2 \cdot 10^{-3}$, -10^{-3}, and -10^{-8} at the known jumps $i = 100$, 250, and 350, correspondingly.

Conclusions

- The novel concept of negative graph weights introduced
- Connection made to the standard mass-spring model, but with some springs repulsive rather than attractive
- The influence of negative weights on the shapes of the "low-frequency" eigenvectors of the graph Laplacian explained by comparing to mass-spring vibration modes
- Repulsing springs/negative weights result in sharpening of the otherwise flat ends of the eigenvectors at the interface of the jump in the guidance signal, used to generate the weights, i.e., the entries of the graph adjacency matrix
- Negative weights in low-pass graph-based signal filters can dramatically improve edge matching, especially sharp corners
- Negative weights appear naturally as correlations for vector signals

Future work

- Specific formulas for calculating the negative graph weights for scalar and vector signals
- Testing and use in applications
- Implications of negative graph weights for spectral clustering

Thank you!