Lecture 07: Matrix Re-ordering; Image De-Noising

June 18, 2025

Outline

Introduction

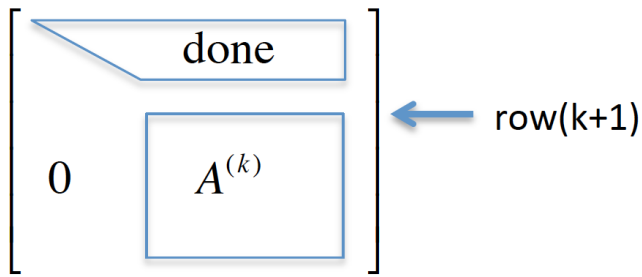
We have seen so far that a graph provides a useful abstraction of the structure of symmetric matrices. The graph offers insight into where fill-in can occur during factorization. In the previous lecture we discussed envelope methods, such as (reverse) Cuthill-McKee, which are a family of reorderings that can reduce fill by minimizing the envelope.

- Envelope/Level set methods,
- (Reverse) Cuthill McKee,
- Markowitz,
- Minimum Degree.

In this lecture we will look at the last two reordering schemes above.

Matrix Re-ordering - Markowitz Reordering

- Markowitz [Markowitz 1957] is a local rule that tries to (approximately) minimize fill on the current step only.
- In other words, it greedily minimizes fill-in for the current step.
- After k steps of LU factorization we have:



where $A^{(k)}$ indicates the lower right block matrix after k steps of LU.

Matrix Re-ordering - Markowitz Reordering

- Consider the fill that can occur during one step of LU factorization from this point.
- Normally, we subtract multiples of current row $(k+1)$ from rows below $(k+2, \ldots, n)$, if there is a non-zero in the column to be zeroed out.
- Specially, fill-in does not occur in rows that already have a zero in the column, e.g.,

$$A^{(k)} = \begin{bmatrix} \times & \times & & \times \\ \times & \times & & \times \\ 0 & & & \\ \times & \times & & \times \end{bmatrix}, \tag{1}$$

  where blue $\times$ denotes fill-in that has occurred.
- The worst case fill-in at this step (using this pivot $\times$) is

$$\begin{aligned} & 4 \text{ entries} \\ = \; & (2 \text{ other non-zeros in row}) \times (2 \text{ other non-zeros in column}). \end{aligned}$$

Matrix Re-ordering - Markowitz Reordering

- We can **swap rows and columns on the fly** to reduce the resulting fill-in.
- Consider all entries $a_{ij}^{(k)}$ in the lower right block $A^{(k)}$.
- The idea is to determine the entry that would minimize the (worst-case) fill.
- Then, swap it into the top-left (pivot) position of $A^{(k)}$.
- The row that produces the least fill on this current step is determined using the **Markowitz product**.

Matrix Re-ordering - Markowitz Reordering

- Let $r_i^{(k)} = \texttt{nnz}$ (number of nonzeros) in row i of $A^{(k)}$ and $c_j^{(k)} = \texttt{nnz}$ in column j of $A^{(k)}$.
- The maximum possible fill using $a_{ij}^{(k)}$ as the pivot is

$$(r_i^{(k)} - 1)(c_j^{(k)} - 1),$$

  which is called the **Markowitz product**.
- **Brief Explanation:** Consider the rows below the pivot row first (i.e. the count of non-zero entries in the chosen column). Zero entries below the pivot cannot cause fill-in, because we don't have to use the pivot row to eliminate a zero entry.
- The Markowitz reordering algorithm swaps rows/columns to choose the pivot $a_{\ell m}^{(k)}$ that minimizes the Markowitz product, i.e.,

$$(\ell, m) = \arg\min_{k \le i,j} (r_i^{(k)} - 1)(c_j^{(k)} - 1).$$

Matrix Re-ordering - Markowitz Reordering

- See the Lecture Notes for a concrete example.
- If A is symmetric, then we select $a_{\ell\ell}^{(k)}$ with

$$\ell = \arg\min_{k \le i} (r_i^{(k)} - 1),$$

  since $\arg\min_{k \le i} r_i^{(k)} = \arg\min_{k \le j} c_j^{(k)}$.
- So we only need to consider diagonal entries for symmetric matrices.
- By symmetrically swapping both rows and columns, $a_{\ell\ell}^{(k)}$ becomes the new pivot.
- This approach has the following features:
  - preserves symmetry and **diagonal dominance**,
  - corresponds to node reordering.

Matrix Re-ordering - Markowitz Reordering

- **Aside:** A matrix is (weakly) **diagonally dominant** if for every row, the magnitude of the diagonal entry is larger than or equal to the sum of the magnitudes of all the other entries in that row. That is,

$$|a_{ii}| \geq \sum_{i \neq j} |a_{ij}|, \text{ for all } i.$$

- A matrix is (strongly) **diagonally dominant** if for every row, the magnitude of the diagonal entry is strictly larger than the sum of the magnitudes of all the other entries in that row. That is,

$$|a_{ii}| > \sum_{i \neq j} |a_{ij}|, \text{ for all } i.$$

- We will see more on this when we discuss **iterative methods**.

Matrix Re-ordering - Minimum Degree Reordering

- The symmetric case of Markowitz reordering inspires an algorithm called **minimum degree reordering**.
- Consider the $(r_i^{(k)} - 1)$ for diagonal entries of this symmetric matrix

$$\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & & & \\ \times & & \times & \times & \\ \times & & \times & \times & \times \\ \times & & & \times & \times \end{bmatrix}. \tag{2}$$

- Because the matrix is symmetric, therefore we only need to consider diagonal entries.
- We have that

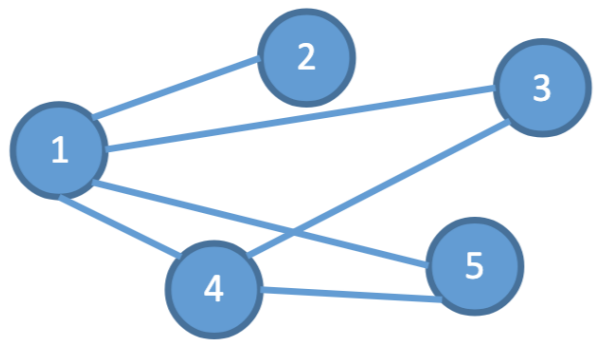
$$\begin{aligned} a_{11} &\mapsto 4, \\ a_{22} &\mapsto 1, \\ a_{33} &\mapsto 2, \\ a_{44} &\mapsto 3, \\ a_{55} &\mapsto 2, \end{aligned}$$

so we would swap to use a_{22} as the pivot.

Matrix Re-ordering - Minimum Degree Reordering

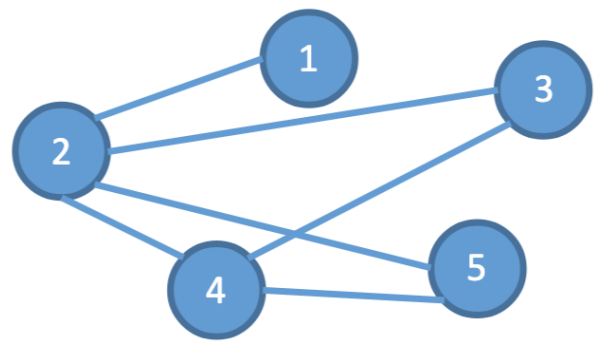
- But what do these values correspond to in the graph view?
- The value of $(r_i^{(k)} - 1)$ is number of off-diagonal non-zero entries in the row, which is the same as the **degree** of the corresponding node!
- The original matrix in (2) gives the following graph:

$$\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & & & \\ \times & & \times & \times & \\ \times & & \times & \times & \times \\ \times & & & \times & \times \end{bmatrix} \longleftrightarrow$$



- While the reordering with a_{22} as the pivot gives:

$$\begin{bmatrix} \times & \times & & & \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \\ & \times & \times & \times & \times \\ & \times & & \times & \times \end{bmatrix} \longleftrightarrow$$



Matrix Re-ordering - Minimum Degree Reordering

- Minimum degree ordering **chooses the node with (current) minimum degree as the pivot element**, at each step of factorization.
- When multiple nodes have same degree we need a strategy to break ties.
- Some possible strategies are:
  1. select the node with smallest node number in the original ordering,
  2. pre-order with RCM, then select the node that is numbered earlier according to an RCM ordering (computed in advance),
  3. Various others, e.g. "multiple minimum degree" chooses multiple nodes that don't interact and eliminate them at once.
- In practice, tie breaking may have a significant impact on the order.
- Minimum degree reordering is optimal for trees, i.e., MD produces no fills on trees.
- See the Lecture Slides for an explanation of this claim, pseudocode, and useful examples.

Matrix Re-ordering - Minimum Degree Reordering

- Finally, we mention some of the many possible improvements of MD:
  - "supervariables"/indistinguishable nodes: nodes with identical adjacency structure (neighbours) can be eliminated simultaneously,
  - multiple elimination: non adjacent nodes of same degree can also be safely eliminated simultaneously,
  - approximate minimum degree: use an approximation to the degree updates of neighbours, which improves the run time,
  - quotient graph: smarter graph representation to reduce storage.
- Remember that our overall goal of reordering is to **minimize computation and storage costs** of factorization on sparse matrices by limiting fill.
- Minimum degree ordering tries to greedily minimize fill at each step by eliminating the node with least degree.
- MD often outperforms RCM but is still just a heuristic!

Stability (Optional) - Matrix Condition Numbers

- Here we will discuss some stability issues for factorization. We consider another use for row/column swaps to now help with stability.
- How do small error/changes in a matrix problem $Ax = b$ affect the (exact) solution?
- The **matrix condition number**, $\kappa(A) = \|A\|\|A^{-1}\|$ (where $\|A\| = \max \frac{\|Ax\|}{\|x\|}$), provides a measure for this.
- Note that $\kappa \geq 1$.
- The condition number $\kappa(A)$ can provide an upper bound on the change in x due to the relative change δ in b and/or A.
- Specifically, if

$$\max\left(\frac{\|\Delta A\|}{\|A\|}, \frac{\|\Delta b\|}{\|b\|}\right) \leq \delta,$$

then

$$\frac{\|\Delta x\|}{\|x\|} \leq 2\kappa(A)\delta + O\left(\delta^2\right).$$

Stability (Optional)

- Stability is a property of the numerical algorithm, which is distinct from conditioning of the problem.
- Essentially, stability is concerned with how errors or changes in input to the numerical algorithm affect the output.
- For example, do small errors magnify or shrink during computation?
- It is important to note that a highly stable algorithm cannot prevent issues due to a poorly conditioned problem.
- Furthermore, an unstable algorithm can give useless results, even for a well-conditioned problem.

Stability (Optional)

- Here we will discuss the stability of LU factorization.
- The basic goal is to find L and U whose "size" remains under control.
- Huge entries will also inflate round off error and produce useless results.
- For example, we do not want re-multiplying LU together to give a new $\hat{A}$ that is far from the input A.

Pivoting (Optional) - Unnecessary Pivoting

- Pivoting improves stability of the factorization, but tends to reduce sparsity.
- There is a tradeoff that needs to be considered.
- But for certain matrices pivoting is never necessary.

Theorem 1

If A is symmetric positive definite, then during LU factorization the pivot $a_{kk}^{(k-1)} > 0$ for all k.

Proof.

See Lecture Notes. □

Pivoting (Optional) - Unnecessary Pivoting

- Pivoting is also unnecessary for:
  1. Row strictly diagonally-dominant matrices

$$|a_{kk}| > \sum_{j \neq k} |a_{kj}| \qquad \text{for} \qquad k = 1, \ldots, n,$$

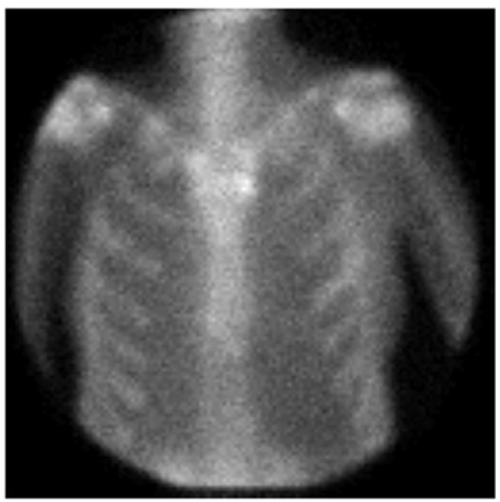
  2. Column strictly diagonally-dominant matrices

$$|a_{kk}| > \sum_{j \neq k} |a_{jk}| \qquad \text{for} \qquad k = 1, \ldots, n.$$

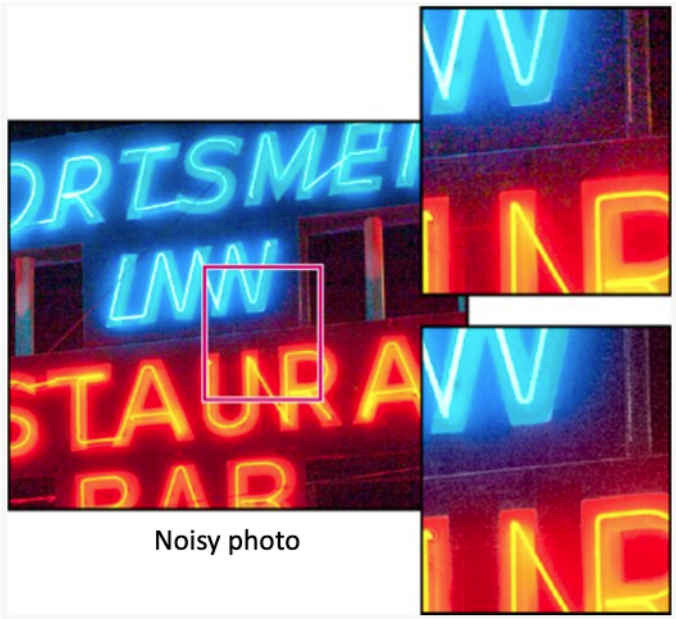
- That is, matrices whose diagonal entry is bigger in magnitude than the sum of remaining entries in the row/column (respectively).

Image De-Noising

- Images often contain random "noise" (small errors), arising from the sensors, capture method, or (lighting) conditions. See for example Figure 1.



Noisy medical image



Noisy photo

Figure: Example noisy images.

Image De-Noising

- Synthetic images generated by raytracing have noise unless you run them for a very long time. An alternative approach is to raytrace for a short time, then clean up with some de-noising. (see Figure 2).

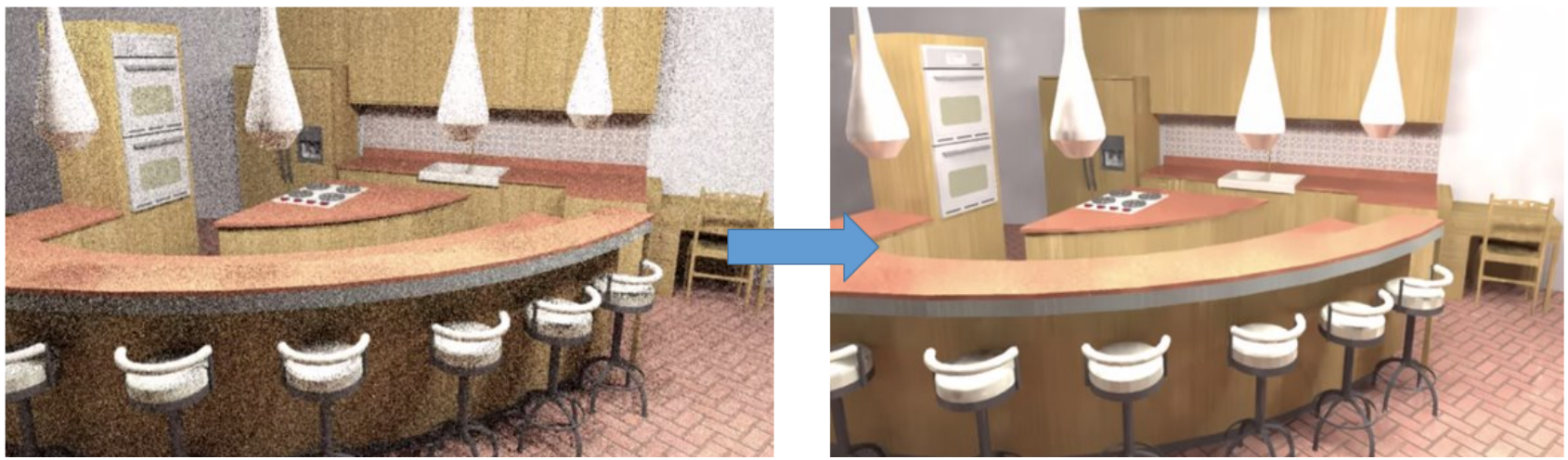


Figure: Example noisy synthetic image (left) and denoised image (right) from [Kalantari et al. SIGGRAPH 2015].

- Often there is enough "signal" amidst the noise that we can try to recover a version with the noise removed/reduced.

Image De-Noising - Inverse Problems

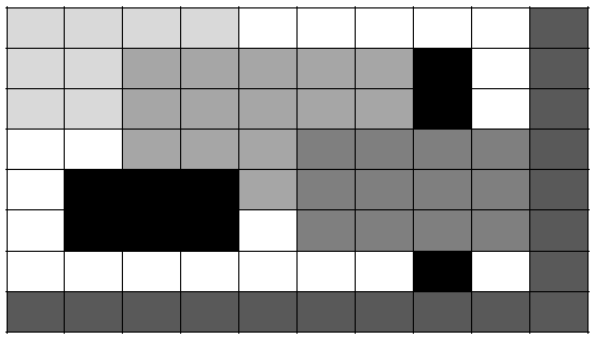
- Image denoising is an **inverse problem**.
- That is, given some observations we want to reconstruct the source/factors that generated them.
- Given some (noisy) observation u^0 of some signal u^* we want to recover the clean signal u^*, i.e.,

$$u^0 = u^* + n,$$

  where n is the noise.
- Thus, we want to decompose the observation u^0 into the sum of two components: the clean signal u^* and the noise n.

Image De-Noising - Inverse Problems

- The observed image is u^0 is given.
- The goal is to find an approximation of u^*.
- We treat grayscale images as 2D scalar functions



u_{ij} = pixel intensity value at row i, column j.

Image De-Noising - Inverse Problems

Two key assumptions enabling us to solve the inverse problem:

1. noise is not too large, i.e., observation u^0 is "close" to signal u^*
2. signal u^* has some **structure** that we can exploit.

Image De-Noising - Regularization Models

- We seek u satisfying

$$\min_u \alpha R(u) + \|u - u^0\|_2^2,$$

  where $R(u)$ is the **regularization model**.
- In this form, the $\|u - u^0\|_2^2$ term can be thought of as a measure of the discrepancy between the observation u^0 and the numerical solution u.
- (The notation $\|\cdot\|_2^2$ should remind us of the square of the 2-norm, i.e. the square of the Euclidean distance, i.e. the sum of the squares of the distances along each axis.)

Image De-Noising - Regularization Models

- The parameter $\alpha > 0$ is called the **regularization constant**, which controls the trade-off between
  - **regularity** ("smoothness") and
  - **fit** (fidelity to data u^0).
- The regularization constant balances the two goals:
  - $\alpha \to 0$: ignores the first term (regularization) implying $u \approx u^0$, so this basically outputs the observation u^0,
  - $\alpha \to \infty$: ignores the second term (observation) implying $u \approx$ (minimizer of $R(u)$) giving a perfectly "regular" image.
- Good recovery of a denoised image relies on
  - an appropriate tuning of α, and on
  - the regularization model $R(u)$.
- We will discuss three options here:
  1. **Tikhonov**,
  2. **Laplacian**, and
  3. **Total Variation** regularizations.

Image De-Noising - Regularization Models - Tikhonov Regularization

- For **Tikhonov regularization** $R(u)$ is a measure of the total sum of pixel intensity

$$R(u) = \|u\|_2^2.$$

- Therefore our optimization becomes

$$\min_u \alpha\|u\|_2^2 + \|u - u^0\|_2^2.$$

Image De-Noising - Regularization Models - Tikhonov Regularization

- Solving this quadratic optimization (e.g., via Euler-Lagrange equations - all details skipped) leads to

$$\begin{aligned} \alpha u + (u - u^0) &= 0, \text{ so} \\ (\alpha + 1)u &= u^0. \end{aligned}$$

- Hence, the new pixel intensities are given by

$$u = \frac{u^0}{\alpha + 1}. \tag{3}$$

- From (3) we see that the solution with Tikhonov regularization gives
  - $u \to u^0$ as $\alpha \to 0$ and
  - $u \to 0$ when $\alpha \to \infty$.
- Thus, α indeed balances matching the input data and being close to a perfectly regular image (image of all zeros).
- However, this is **really not what we want** from our regularization since it is pushing us towards a zero intensity image.

Image De-Noising - Regularization Models - Laplacian Regularization

- Consider a noisy "image" (signal) in 1D shown in Figure 3.

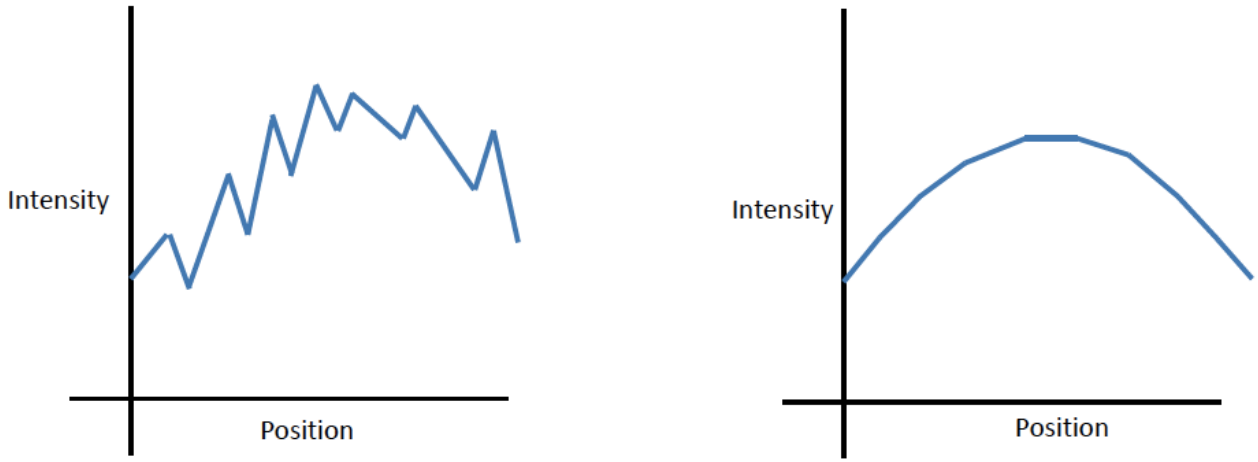


Figure: Example of a noisy signal (left) and smoother signal (right).

- It can be seen from the noisy 1D image that there is drastic change in slope throughout the image.
- If one compared to the smoother 1D image the slope changes more continuously.
- Therefore, we should try to penalize changes in slopes/derivatives, ∇u, instead of pixel values u.

Image De-Noising - Regularization Models - Laplacian Regularization

- The **Laplacian regularization model** $R(u)$ is a measure of the total sum of intensity **gradients**

$$R(u) = \|\nabla u\|_2^2.$$

- So the optimization problem becomes

$$\min_u \alpha\|\nabla u\|_2^2 + \|u - u^0\|_2^2.$$

- The optimal solution (minimizer) satisfies the linear PDE

$$\begin{aligned} -\alpha\nabla\cdot\nabla u + (u - u^0) &= 0, \\ -\alpha\Delta u + u &= u^0, \end{aligned} \tag{4}$$

where $\nabla\cdot\nabla = \Delta$ is the Laplace operator (from our PDE application).

Image De-Noising - Regularization Models - Laplacian Regularization

- We can apply finite differences to (4) to compute a numerical approximation of the minimizer u_{ij} at each pixel (i, j).
- Using the finite difference approximation of the Laplacian Δ, previously discussed in Lecture 4, we have

$$\frac{\alpha}{h^2}(4u_{ij} - u_{i-1,j} - u_{i+1,j} - u_{i,j-1} - u_{i,j+1}) + u_{ij} = u^0_{ij}.$$

- This gives a matrix equation of the form $(\alpha A + I)u = u^0$.

Image De-Noising - Regularization Models - Laplacian Regularization

- If the solution remains too noisy we can try iterating as

$$(\alpha A + I)u^{k+1} = u^k, \qquad \text{for} \qquad k = 1, 2, \ldots, K.$$

- However, **the drawback of Laplacian regularization is that it tends to smear out edges** as shown in Figure 4.

Figure: Laplacian regularization for image denoising of the noisy image (left). The result on the right is smeared out throughout the image.

Image De-Noising - Regularization Models - Total Variation Regularization

- To avoid smearing edges the **total variation regularization** takes
$$R(u) = \|\nabla u\|_1.$$
- This still minimizes the slopes but with a different measure that does not punish them too much (i.e., 1-norm, without squaring).
- So our optimization roughly becomes
$$\min_u \alpha\|\nabla u\|_1 + \|u - u^0\|_2^2.$$
- The minimization problem leads to another PDE to solve, namely
$$-\alpha\nabla \cdot \left(\frac{1}{\|\nabla u\|_1}\right)\nabla u + u = u^0. \tag{5}$$
- The PDE (5) is similar to the Laplacian regularization PDE, but with 1 instead of $\frac{1}{\|\nabla u\|_1}$.

Image De-Noising - Regularization Models - Total Variation Regularization

- The effect is that the matrix coefficients (amount of smoothing) depends on gradients in the image themselves.
  1. Near big intensity jumps (edges of objects in an image)

$$\|\nabla u_{ij}\| \text{ is large } \Rightarrow \frac{1}{\|\nabla u_{ij}\|} \text{ is small!}$$

Therefore the 1st term becomes negligible giving $u \approx u^0$, which leaves edges nearly unchanged staying close to data u^0.

  2. However, in "flat" regions, where intensity is roughly constant,

$$\|\nabla u_{ij}\| \text{ is small } \Rightarrow \frac{1}{\|\nabla u_{ij}\|} \text{ is large!}$$

This implies more diffusion at pixel (i, j) since effectively we have

$$-C\nabla \cdot \nabla u_{ij} + u_{ij} = u^0_{ij}, \qquad \text{where } C \text{ is some large value.}$$

The increase in diffusion makes these regions flatter/smoother.

Image De-Noising - Regularization Models - Total Variation Regularization

To summarize, the behaviour of total variation regularization is

1. Edge-like regions are smoothed less, and
2. flatter regions are smoothed more.

So we get smoothing that roughly "stays within the lines".

Image De-Noising - Regularization Models - Total Variation Regularization

- Figure 5 shows results from the original paper of a noisy image (top) and total variation denoised image (bottom).

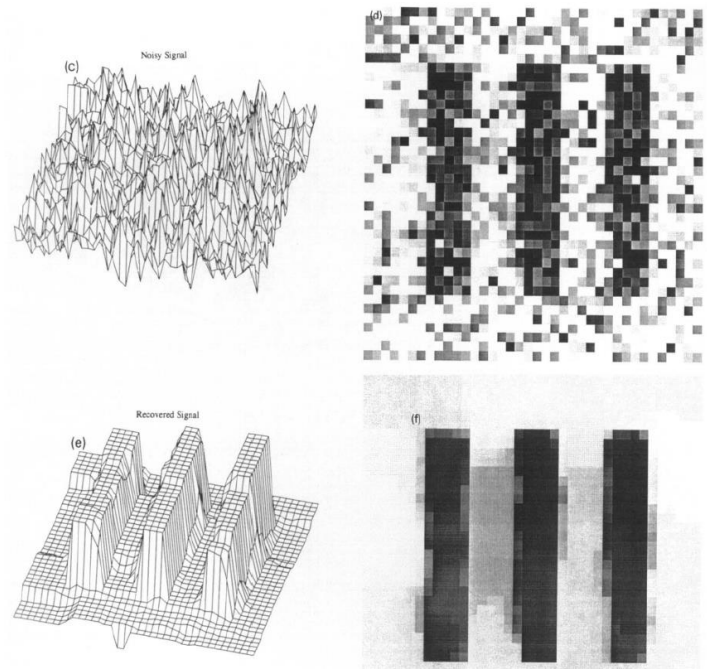


Figure: Noisy image (top) and denoised image (bottom) using total variation regularization [Rudin et al. 1992].

Image De-Noising - Regularization Models - Total Variation Regularization

- We will now discuss how to compute a numerical approximation to (5).
- We can apply a finite difference discretization again of the form

$$\alpha A(u)u + u = u^0,$$

  but this equation is **nonlinear**.
- The coefficients in the matrix $A(u)$ depend on the solution u itself.
- We need to solve this equation numerically: we cannot solve the PDE directly, as we did for the earlier techniques.

Image De-Noising - Regularization Models - Total Variation Regularization

- A simple approach to solve nonlinear equations is the **fixed point iteration**.
- We freeze the coefficients to make the equations linear, solve, update, and repeat.
- That is, solve

$$\begin{aligned} & \alpha A(u^k)u^{k+1} + u^{k+1} &&= u^0, \\ \Rightarrow \quad & (\alpha A(u^k) + I)u^{k+1} &&= u^0, \qquad \text{for } k = 0, 1, \ldots, K. \end{aligned}$$

- We pick an initial guess and compute an approximate solution by solving the system.
- The matrix $A(u^k)$ is then recomputed for the next iteration.

Image De-Noising - Regularization Models - Total Variation Regularization

- Note that such an iteration **does not always converge to the solution** in general.
- Fortunately, in this case the fixed point iteration does converge.
- There are different approaches to determine when to stop iterating, i.e., what is K?
- One approach is to stop iterating when the approximation is not changing much anymore, i.e.

$$\|u^{k+1} - u^k\| < \text{tol},$$

for some small tolerance.

Image De-Noising - Regularization Models - Total Variation Regularization

Noisy Image Laplacian Total Variation

Figure: Comparison denoising an image with Laplacian and total variation regularization (images from Mathworks Matlab manual).

- Figure 6 compares the Laplacian and total variation regularization.
- The Laplacian regularization is unable to remove as much noise as the total variation regularization.
- Increasing the regularization parameter α for the Laplacian regularization will just blur the image instead of removing noise.

Image De-Noising - Regularization Models - Total Variation Regularization

- The effect of increasing α for total variation regularization is shown in Figure 7.
- Edges are still well preserved as α increases and the image becomes smoother.

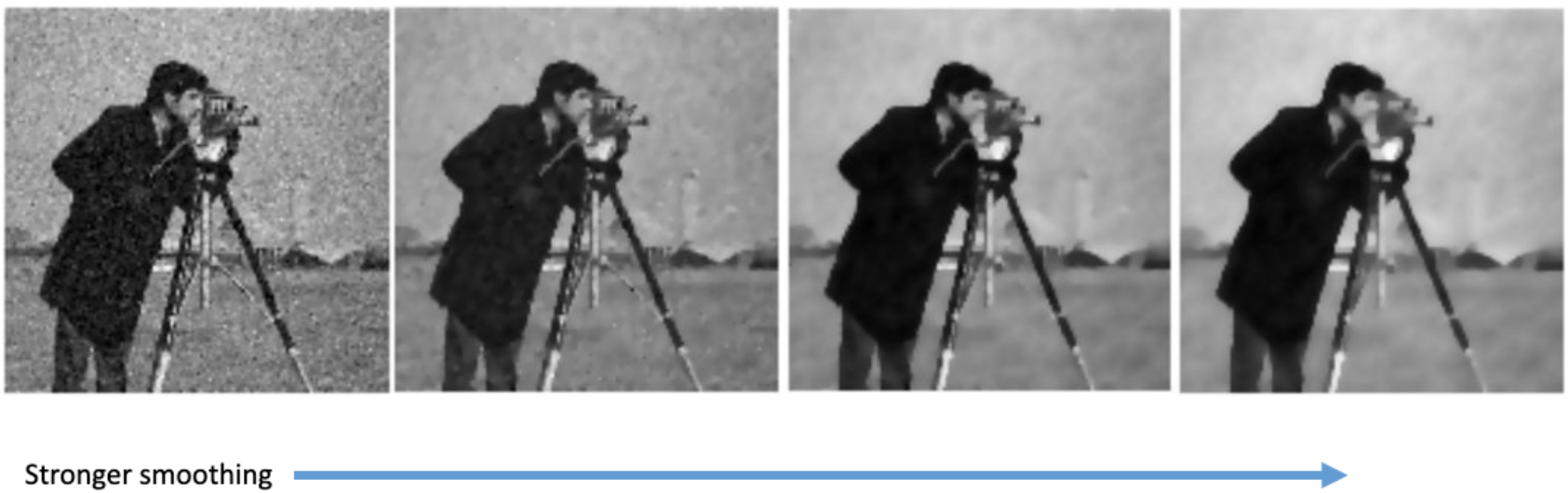


Figure: Effect of increasing the regularization parameter α with total variation regularization.

Summary

- Many questions about how to complete these computations have been left unanswered so far.
- When we work out such examples on the Crowdmark assignments, all the required details will be specified.