Graph-based learning for Image Processing

Jeremy Budd

Course given at Caltech in Spring 2023 *
Last revised Friday 7th July, 2023

*For any remark/correction/suggestion, please send an email to `jmbudd@caltech.edu`.

Contents

Chapter 1

A brief overview of mathematical image processing

1.1 What is an image?

1.1.1 Basics of images

Definition 1.1.1 (Image)**.** *In this course, an* image $\mathcal{I}$ *will be a function from some finite set of* pixels*, which we will denote by V (for reasons which will eventually become clear), into $\mathbb{R}^d$.*

What is d? That depends on the type of image. There are three common cases. If the image is a *greyscale* image then $d = 1$ and indeed we will typically constrain the values of the image to lie in $[0, 1]$.

If the image is a colour image then typically $d = 3$. The most common case is for the three components of the image values (called the *channels* of the image) to refer respectively to the amount of red, green, and blue at that pixel–this is called an RGB image. However, other decompositions of colour images exist, such as into hue, saturation, and value channels (HSV) and into luma, blue chroma, and red chroma channels (YCbCr). We will not in this course dwell on the details of these different colour spaces. Finally, in the case of *hyperspectral* images the value each pixel will have a very large number of channels, one for each of the spectral components measured. In such cases d will be very large.

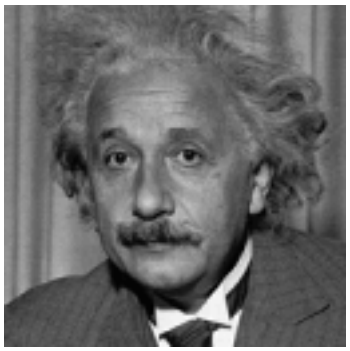

Figure 1.1: A greyscale image

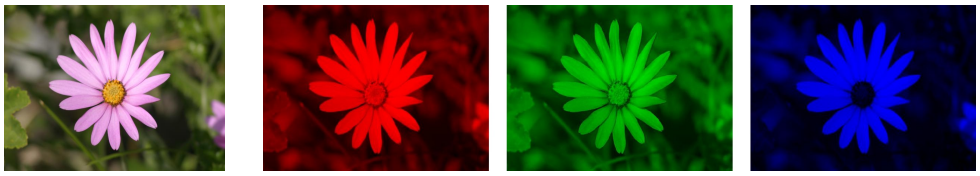

Figure 1.2: A colour image (far left) along with its R, G, and B channels. Image is from the Microsoft Research Cambridge Object Recognition Image Database, available at https://www.microsoft.com/en-us/research/project/image-understanding/.

Note. It will often be useful to think of V as a subset of some domain $\Omega \subseteq \mathbb{R}^2$. Likewise, it can be useful to think of the image $\mathcal{I} : V \to \mathbb{R}^d$ as being a restriction of a continuous image $\mathcal{I} : \Omega \to \mathbb{R}^d$.

1.1.2 What do "natural" images look like?

However, not all functions on a set of pixels correspond to realistic images. Compare the two images below.

Figure 1.3: Ordinary RGB image (left) vs. a random Gaussian function on the same pixel set (right).

Question. What features do normal images have that white noise does not?

1.2 Image compression

Images are typically never actually stored as such functions, as for typical images such a representation tends to be extremely redundant, and takes up a lot of memory.

A major topic in image processing is *image compression*, the task of storing an image in a way that takes up much less space.

1.2.1 Lossy compression: JPEG

A straightforward approach to compressing an image is to **throw away information that the human eye can't see**. This is the basic idea behind JPEG compression. Very *very* briefly, the idea of JPEG compression is to express our image $\mathcal{I}$ in a particular basis

$$\mathcal{I} = a_1\varphi_1 + a_2\varphi_2 + \cdots$$

where the basis elements φ_i encode changes with a fixed frequency in the image (and the a_i can be computed using the Discrete Cosine Transform). This is analogous to decomposing a signal into its distinct pitches. Since natural images don't change very frequently and the human eye can't see these very rapid changes, the higher frequency components can be thrown away, giving a compression.

Note. Another lossy standard JPEG2000 use the same ideas but use a different basis (a *wavelet basis*) for the decomposition.

For a very accessible description of how JPEG works, I recommend the following YouTube video https://www.youtube.com/watch?v=0me3guauq0U. For a more detailed account, see [Wal91].

1.2.2 Lossless compression: PNG

However, it is also possible to compress images without throwing away any information, by exploiting redundancy. This is how file formats like PNG work. Even more briefly than before: if you wanted to compress these lecture notes, you could do so by first exploiting how often strings of letters are repeated to replace repetitions with references to earlier strings, and then encoding what is left. PNG uses some clever tricks to apply the same ideas to images. See the same YouTuber for a very good explanation https://www.youtube.com/watch?v=EFUYNoFRHQI. For a more detailed account, see [Roe99].

1.3 Inverse problems in imaging

Many of the most important tasks in image processing involve *undoing something that has happened to an image*. These include:

- **Image Reconstruction**: The task of reconstructing an image from indirect and/or noisy measurements.
- **Image Registration**: The task of aligning distorted images.
- **Image Inpainting**: The task of filling in gaps in an image.

See Figures 1.4 and 1.5 for examples of these tasks.

Figure 1.4: Example of image denoising, a special case of image reconstruction, using the BM3D algorithm [MAF20]. Image is from the Microsoft Research Cambridge Object Recognition Image Database. Left-to-right: original image, noised version of the original image, and a BM3D denoising of the noised image.

1.3.1 Inverse problems

The general setting for all of these tasks is that of an *inverse problem*. We have some observations y of an object x^*, which are related via

$$y = \mathcal{T}(x^*) + e \tag{1.1}$$

Figure 1.5: Example of image inpainting, reproduced from Lozes *et al.* [LEL14, Figure 9].

where $\mathcal{T}$ is the *forward model*, typically a linear map, and e is an error term (e.g. a Gaussian random variable). Given y, $\mathcal{T}$, and the distribution of e, we seek to find $x \approx x^*$.

Hadamard well-posedness

In [Had02], Hadamard gave three conditions for a mathematical model to be *well-posed*:

1. A solution should exist.
2. That solution should be unique.
3. The solution should vary continuously with the input.

However, (1.1) is in general an *ill-posed problem*. The noise may bring y outside of the image of $\mathcal{T}$, solutions may be far from unique for a given y, and even if $\mathcal{T}$ is invertible and y lies in its image, the inverse map might be highly sensitive to noise. So we must proceed in a clever way.

Tikhonov–Philips variational method

A key approach to solving such problems, deriving from pioneering work by Tikhonov [Tik63] and Phillips [Phi62] in the 1960s, has been to solve a variational problem of the form

$$\arg\min_x \; R(x) + \lambda D(\mathcal{T}(x), y) \tag{1.2}$$

where R is a *regulariser*, which encodes *a priori* information about the solution x, and D is a distance term which enforces fidelity to our observed data, e.g.

$$D(\mathcal{T}(x), y) := \|\mathcal{T}(x) - y\|_2^2.$$

D encodes information about the error e. The parameter λ determines the trade-off between what we *a priori* expect to see and what our observations depict.

To explain where (1.2) comes from, it will be helpful to put things into a Bayesian setting. Suppose that we have a random variable X describing the (unknown) true object we seek to find, and a random variable Y describing our observations. Then by (1.1), X and Y are related via

$$Y = \mathcal{T}(X) + e \tag{1.3}$$

where, for example, $e \sim \mathcal{N}(0, \sigma^2 I)$. We then have an observation $Y = y$, and the idea behind the Tikhonov–Phillips approach is to find an x which is the *maximum a priori*

(MAP) estimate for X given this observation. That is, solving

$$\arg\max_x \mathbb{P}(X = x|Y = y).$$

By Bayes' theorem, this is equivalent to solving

$$\arg\max_x \mathbb{P}(X = x)\mathbb{P}(Y = y|X = x)$$

which by (1.3) further simplifies to

$$\arg\max_x \mathbb{P}(X = x)\mathbb{P}(e = y - \mathcal{T}(x)). \tag{1.4}$$

The former term, $\mathbb{P}(X = x)$, is our *prior probability* for x to be the correct reconstruction. It quantifies how "reasonable" a candidate reconstruction x is. Let us define our regulariser R such that

$$\mathbb{P}(X = x) =: e^{-R(x)}.$$

The latter term in (1.4) depends entirely on our noise model for e, which will define our data fidelity term via

$$\mathbb{P}(e = y - \mathcal{T}(x)) =: e^{-\lambda D(\mathcal{T}(x),y)}.$$

For example, if $e \sim \mathcal{N}(0, \sigma^2 I)$ then

$$\mathbb{P}(e = y - \mathcal{T}(x)) = e^{-\frac{1}{2\sigma^2}\|y-\mathcal{T}(x)\|_2^2}.$$

Therefore, (1.4) reduces to (1.2) by taking $-\log$ of the objective functional.

1.3.2 Examples

Image denoising The simplest case of image reconstruction is denoising, i.e. the task of removing noise from an image. Here we observe an image y which is related to our "true" non-noisy image $\mathcal{I}^*$ via:

$$y = \mathcal{I}^* + e$$

where e is the noise. This is exactly the setting of (1.1) with $\mathcal{T} = \mathrm{id}$, the identity map.

Medical Imaging: MRI/CT Another important example of imaging that fits into this framework perhaps surprisingly well is medical imagining. It would take too long to get into the details of the physics involved, but the measurements of Magnetic Resonance Imaging (MRI) turn out to just correspond to $\mathcal{T} = \mathrm{FT}$ the Fourier transform, and Computed Tomography (CT) imaging corresponds to a Radon transform.

Inpainting Here $\mathcal{T}$ is a projection onto the undamaged region of the image.

1.3.3 Choosing R: The Rudin–Osher–Fatemi (ROF) denoising method

The real magic in this variational approach to image reconstruction is the choice of regulariser, which as we saw encodes our prior model for what images "should" look like. The original Tikhonov work used $R(\mathcal{I}) = \frac{1}{2}\|\mathcal{I}\|_2^2$ which leads to an easy to solve minimisation problem but doesn't really capture the essence of what an image should be. One of the most important papers in image processing is that of Rudin, Osher, and Fatemi [ROF92] from 1992, which proposed using *total variation* as a regulariser.

What is total variation?

Definition 1.3.1 (Total Variation)**.** *Let $\Omega \subset \mathbb{R}^n$ be open, and let $f \in L^1(\Omega)$. Then the* total variation *of f has the following variational definition:*

$$\mathrm{TV}(f) := \sup \left\{ \int_\Omega f(x) \operatorname{div} \varphi(x)\, dx \middle| \varphi \in C_c^1(\Omega, \mathbb{R}^n), \|\varphi\|_{L^\infty(\Omega)} \leq 1 \right\}.$$

The set of $f \in L^1(\Omega)$ with $\mathrm{TV}(f) < \infty$ is denoted $\mathrm{BV}(\Omega)$.

If $f \in C^1(\bar{\Omega})$, Ω is bounded, and Ω has C^1 boundary, then the following identity holds

$$\mathrm{TV}(f) = \int_\Omega \|\nabla f(x)\|_2 \, dx. \tag{1.5}$$

Proof (sketch). We note the following fact that under the assumed conditions

$$\int_\Omega f(x) \operatorname{div} \varphi(x)\, dx = -\int_\Omega \nabla f(x) \cdot \varphi(x)\, dx$$

for all valid φ. Thus we obtain the upper bound

$$\int_\Omega f(x) \operatorname{div} \varphi(x)\, dx \leq \left| \int_\Omega \nabla f(x) \cdot \varphi(x)\, dx \right| \leq \int_\Omega \|\nabla f(x)\|_2 \|\varphi(x)\|_2\, dx \leq \int_\Omega \|\nabla f(x)\|_2\, dx.$$

Finally, we take valid φ_n approximating $-\nabla f / \|\nabla f\|_2$, which we can do because C_c^1 is dense in L^1. Then

$$\int_\Omega f(x) \operatorname{div} \varphi_n(x)\, dx \to -\int_\Omega \nabla f(x) \cdot \left(-\frac{\nabla f(x)}{\|\nabla f(x)\|_2} \right) dx = \int_\Omega \|\nabla f(x)\|_2\, dx.$$

so this upper bound is the supremum. □

Definition 1.3.2 (Discrete Total Variation)**.** *Let $\mathcal{I} : V \to \mathbb{R}$ where $V = \{1, 2, \ldots, N\}^2$. Then, inspired by (1.5), we can define the total variation of $\mathcal{I}$ by:*

$$\mathrm{TV}(\mathcal{I}) = \sum_{i=1}^{N-1} \sum_{j=1}^{N-1} \sqrt{(\mathcal{I}_{i+1,j} - \mathcal{I}_{i,j})^2 + (\mathcal{I}_{i,j+1} - \mathcal{I}_{i,j})^2}.$$

Here we can see that u is being conceptualised as a discrete approximation to a smooth function defined on a set containing the lattice points. This is only one way to discretise total variation, for other options see [Get12, §2].

Total variation as a regulariser

Earlier, we asked ourselves what ordinary images look like. The key thing we mentioned was that they have regions of gradual change with the occasional sharp edge. That is, on most of an ordinary image $\mathcal{I}$, $\nabla \mathcal{I}$ will be small, with the exception of a small number of edges. Put another way, in the discrete setting the gradient of the image will be (approximately) *sparse*. Thus it makes sense to minimise total variation, understood as the L^1 norm of the gradient, in order to encourage the gradient to be small. But why minimise the L^1 norm? Why not the L^2 norm, which leads to a much easier optimisation problem?

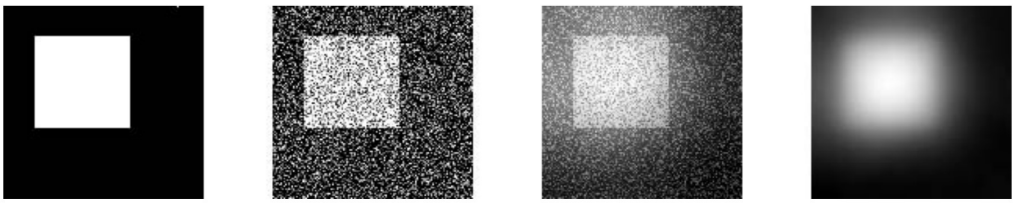

Figure 1.6: Left-to-right: image of white square on black background, the same image with Gaussian noise added, reconstruction with $R(\mathcal{I}) = \frac{1}{2}\|\mathcal{I}\|_2^2$, and reconstruction with $R(\mathcal{I}) = \frac{1}{2}\|\nabla\mathcal{I}\|_2^2$. Reproduced from [Cha00, Figure 1].

The reason is that minimising the L^2 norm does not promote sparsity in the gradient, leading to very blurry edges (see Figure 1.6). From work in compressed sensing (Candes, Romberg, and Tao [CRT06]), it was known that the L^1 norm is a good choice to promote sparsity, making total variation the best pick.

Example ROF denoisings

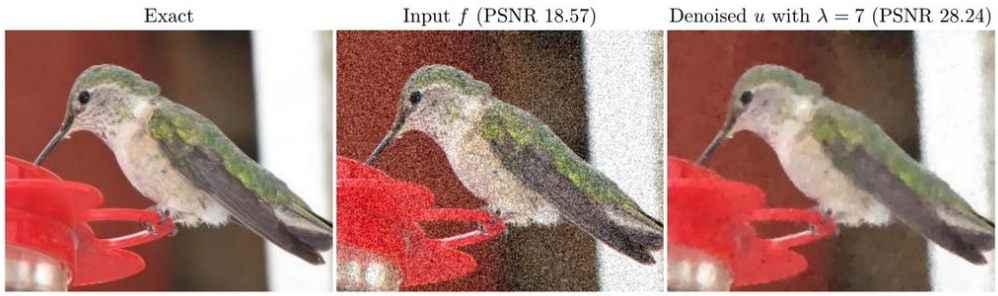

Figure 1.7: ROF denoising of an RGB image, reproduced from [Get12, p.90].

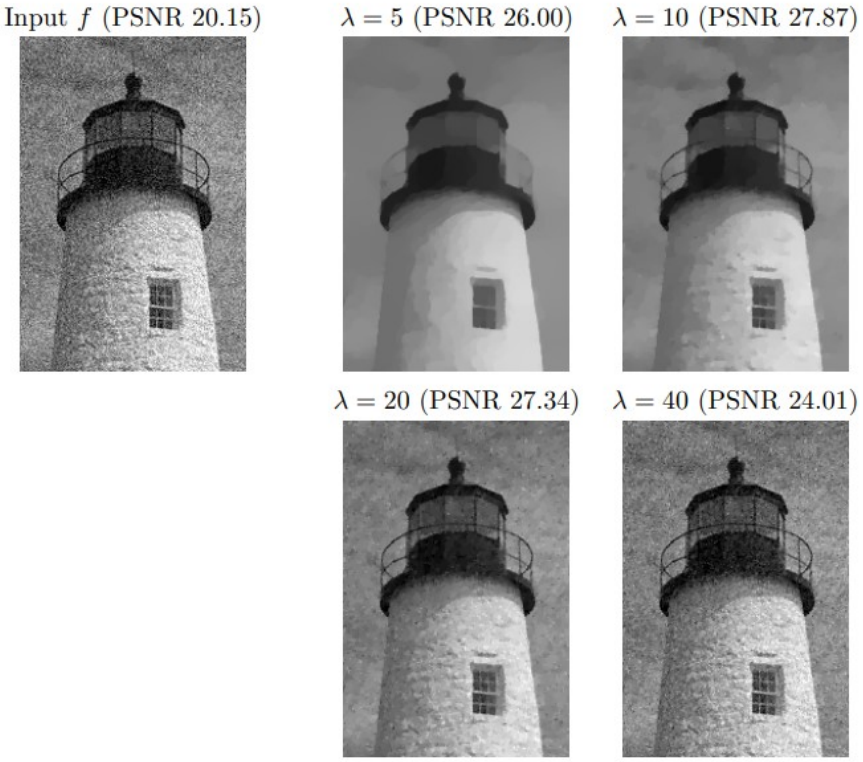

Figure 1.8: ROF denoisings of a greyscale image for various λ, reproducing [Get12, p.89].

1.3.4 Choosing R: Deep learning approaches

Nowadays image processing is being taken over by deep learning methods. This course will not be a course on deep learning, but I will briefly mention here that an important avenue of current research in this area is using training data to *learn* a prior distribution, and thereby learn R. Some example approaches for this are:

- **The Plug-and-Play Prior (P^3)** [VBW13]: This eschews an explicit regulariser altogether, via the observation that the steps involving the regulariser in most algorithms look like denoisings, and replacing these with "plug-and-play" denoisings, e.g. deep learned ones.
- **Regularisation by denoising (RED)** [REM17]: Has a similar idea, using a denoiser D to define a regulariser $R(x) = \frac{1}{2}\langle x, x - D(x)\rangle$.
- **Adversarial Regularisation** [LÖS18]: Trains a neural net to distinguish ground truth images from images which are pesudoinverses of measurements, and uses this discriminator as a regulariser.

For a detailed overview, see Arridge *et al.* [AMÖS19].

1.4 Image segmentation and data clustering/classification

1.4.1 What is image segmentation?

The task of *image segmentation* is to split up an image into is component parts. For example, in Figure 1.9 those parts might be the cows, the grass, and the sky.

Figure 1.9: Image of cows, from the Microsoft Research Cambridge Object Recognition Image Database.

In practice, one might often only care about one of these parts, e.g. the cows, with the rest of the image being regarded as "background". For example, if one were looking for a tumour in a medical scan, all one would care about is "tumour" vs. "background". This is called *binary segmentation*.

1.4.2 Data clustering and classification

Image segmentation is a special case of the task of data clustering/classification. Let $\mathcal{I} : V \to \mathbb{R}^\ell$ describe not necessarily an image but just a collection of data points (indexed by V) living in $\mathbb{R}^\ell$. Classification and clustering seek a function $u : V \to L$ where L is a finite set of labels. For binary segmentation, $L = \{0, 1\}$.

The difference between classification and clustering is that clustering is *unsupervised*, it just uses the geometry of the data to sort into into labels. Classification is *supervised* (or *(semi-)supervised*), which means we have an additional subset $Z \subseteq V$ as *training data*, with a known *a priori* labels $f : Z \to L$ to which we want u to conform. We say that the task is semi-supervised when Z is really small relative to V.

Note. In image segmentation, V is the set of pixels in an image. But we could instead take V to be a set of *images* in this setting (in which case ℓ would then become very large, equal to the number of pixels times the number of channels in each image). The data clustering task would then become the task of labelling this set of images, e.g. labelling whether each image was of a "dog" or "cat". This is another very important task in image processing.

1.4.3 Example: Mumford–Shah and Chan–Vese segmentation

The celebrated segmentation approach of Mumford and Shah [MS89] works as follows. Suppose that we wish to segment a greyscale image which we shall represent by the

continuous image $\mathcal{I} : \Omega \to \mathbb{R}$. We will do this by finding a piecewise smooth image $\mathcal{J}$ and a sum of contours Γ minimising the *Mumford–Shah functional*

$$\mathrm{MS}_{\mu,\lambda}(\mathcal{J}, \Gamma) := \int_{\Omega\setminus\Gamma} \|\nabla \mathcal{J}(x)\|_2^2 \, dx + \lambda \int_{\Omega} (\mathcal{J}(x) - \mathcal{I}(x))^2 \, dx + \mu|\Gamma|$$

where $|\Gamma|$ denotes the total length of Γ and $\mathcal{J}$ is smooth on $\Omega \setminus \Gamma$. The segments here are regions bounded by Γ. The final term in the energy may seem innocuous, but it is in fact very important, as it drives the segmentation to respect the geometry of the image.

Note. Although this segmentation energy is very natural, it turns out to be very complicated to minimise in general. We will not discuss methods for minimising this energy in this course, see e.g. [BZ87] for approaches.

A simplification of this problem considered by Mumford and Shah is to restrict $\mathcal{J}$ to be piecewise constant on $\Omega \setminus \Gamma$, where $\Gamma = \partial C$ for some closed set C, simplifying the problem to:

$$\min_{\mathcal{J},\Gamma} \int_{\Omega} (\mathcal{J}(x) - \mathcal{I}(x))^2 \, dx + \mu|\Gamma|. \tag{1.6}$$

Theorem 1.4.1 ([MS12, Theorem 5.1]). Let $\mathcal{I}$ be a bounded measurable function on Ω. Then there exists Γ and $\mathcal{J}$ (piecewise constant on $\Omega \setminus \Gamma$) minimising (1.6).

Proof. Beyond the scope of this course. □

Chan and Vese [CV01] took as their starting point the piecewise constant Mumford–Shah model, and made two changes: first, x must take the following simplified form

$$\mathcal{J}(x) = \begin{cases} c_1, & x \in C, \\ c_2, & \text{otherwise.} \end{cases}$$

Second, they added a penalisation for the area of C. Thus, the problem becomes

$$\min_{c_1,c_2,C} \mu|\partial C| + \nu \,\mathrm{Area}(C) + \lambda_1 \int_C (\mathcal{I}(x) - c_1)^2 \, dx + \lambda_2 \int_{\Omega\setminus C} (\mathcal{I}(x) - c_2)^2 \, dx. \tag{1.7}$$

Theorem 1.4.2. For $C \subseteq \Omega$ a closed set, let $\chi_C := 1$ on C and 0 on $\Omega \setminus C$. Then

$$|\partial C| = \mathrm{TV}(\chi_C).$$

Let our classifier $u = \chi_C$. Then (1.7) can be re-written:

$$\min_{c_1,c_2,u} \mu \,\mathrm{TV}(u) + \int_{\Omega} \nu u(x) + \lambda_1 u(x)(\mathcal{I}(x) - c_1)^2 + \lambda_2(1 - u(x))(\mathcal{I}(x) - c_2)^2 \, dx. \tag{1.8}$$

Then [CV01] solve this using level-set methods, which would bring us beyond the scope of the course to describe.

Chapter 2

A brief review of the graph theory that we will need

2.1 What is a graph?

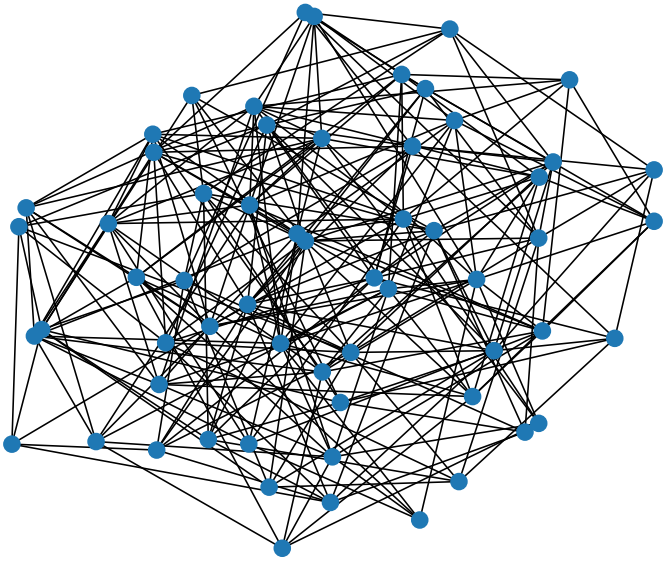

Figure 2.1: An example of a finite, simple, connected, and undirected graph.

Definition 2.1.1 (Graph)**.** *A (finite)* graph *G is a (finite) set of vertices V which are linked by edges $E \subseteq V^2$. We will primarily be concerned with graphs with the following extra properties:*

- Simple: *there is at most one edge between two vertices, and no edge connects a vertex to itself.*
- Weighted: *every edge $(i, j) \in E$ has an associated weight $\omega_{ij} > 0$. (For completeness, we extend ω to V^2 via $\omega_{ij} = 0$ if $(i, j) \notin E$.)*
- Undirected: *if $(i, j) \in E$ then $(j, i) \in E$, and $\omega_{ij} = \omega_{ji}$.*
- Connected: *For each $i, j \in V$, there is a sequence of edges that connects i to j. (More*

generally, if this is true for some $i, j \in V$ then we say that i, j lie in the same connected component *of G. These connected components partition the graph, except for any* isolated vertices.*)*

Definition 2.1.2 (Adjacency matrix/weight matrix)**.** *The* adjacency matrix *or* weight matrix *is the matrix $\omega := (\omega_{ij})$, sometimes also denoted by A. It uniquely defines the graph, up to a relabelling of the vertices (and a corresponding permutation of the row/columns of the matrix).*

Definition 2.1.3 (Degree of a vertex)**.** *For a vertex $i \in V$, we define the* degree *of i by:*

$$d_i := \sum_{j \in V} \omega_{ij}.$$

For a connected graph, $d_i > 0$ for all $i \in V$. But if $d_i > 0$ for all $i \in V$ does not necessarily mean that the graph must be connected, it only excludes isolated vertices.

Note. In this course, we will assume that $d_i > 0$ for all $i \in V$ for all of our graphs.

2.2 Functions on graphs

Definition 2.2.1 (Vertex and edge functions on graphs)**.** *On G we define the spaces ($X \subseteq \mathbb{R}$):*

$$\mathcal{V} := \{u : V \to \mathbb{R}\}, \qquad \mathcal{V}_X := \{u : V \to X\}, \qquad \mathcal{E} := \{\varphi : E \to \mathbb{R}\}.$$

Since V is finite, we can interchangeably view elements of $\mathcal{V}$ and $\mathcal{V}_X$ as functions or as real vectors. Next, we define the spaces of time-dependent vertex functions (where $T \subseteq \mathbb{R}$ an interval)

$$\mathcal{V}_{t\in T} := \{u : T \to \mathcal{V}\}, \qquad \mathcal{V}_{X,t\in T} := \{u : T \to \mathcal{V}_X\}.$$

Definition 2.2.2 (Graph gradient)**.** *For $i, j \in V$ and $u \in \mathcal{V}$ we define the* graph gradient *of u at (i, j) to be:*

$$(\nabla u)_{ij} := \begin{cases} u_j - u_i, & (i,j) \in E, \\ 0, & \text{otherwise}. \end{cases}$$

Definition 2.2.3 (Graph divergence)**.** *For $i \in V$ and $\varphi \in \mathcal{E}$ we define the* graph divergence *of φ at i to be:*

$$(\operatorname{div} \varphi)_i := \sum_{j \in V} \omega_{ij} \varphi_{ij}$$

Definition 2.2.4 (Inner products on our function spaces)**.** *For a parameter $r \in [0, 1]$ we define the following inner products on $\mathcal{V}$ and $\mathcal{E}$:*

$$\langle u, v \rangle_{\mathcal{V},r} := \sum_{i \in V} u_i v_i d_i^r, \qquad \langle \varphi, \theta \rangle_{\mathcal{E}} := \frac{1}{2} \sum_{i,j \in V} \varphi_{ij} \theta_{ij} \omega_{ij} \tag{2.1}$$

and define the inner product on $\mathcal{V}_{t\in T}$ (or $\mathcal{V}_{X,t\in T}$):

$$(u,v)_{t\in T} := \int_T \langle u(t), v(t)\rangle_{\mathcal{V},r}\ dt = \sum_{i\in V} d_i^r (u_i, v_i)_{L^2(T;\mathbb{R})}$$

where $(\cdot,\cdot)_{L^2(T;\mathbb{R})}$ is the standard continuum L^2 inner product. These inner products induce norms $\|\cdot\|_{\mathcal{V},r}$, $\|\cdot\|_{\mathcal{E}}$, and $\|\cdot\|_{t\in T}$ in the usual way.

Definition 2.2.5 (L^2 space on a graph)**.** *We define the L^2 space:*

$$L^2(T;\mathcal{V}) := \{u \in \mathcal{V}_{t\in T} \mid \|u\|_{t\in T} < \infty\},$$

which we will consider as a normed space with norm $\|\cdot\|_{t\in T}$. We also define the local L^2 space:

$$L^2_{loc}(T;\mathcal{V}) = \{u \in \mathcal{V}_{t\in T} | \forall a < b \in T, u|_{(a,b)} \in L^2((a,b);\mathcal{V})\}.$$

Proposition 2.2.1. For all $u \in \mathcal{V}$ and $\varphi \in \mathcal{E}$ such that $\varphi_{ij} = -\varphi_{ji}$,

$$-\langle \operatorname{div}\varphi, u\rangle_{\mathcal{V},0} = \langle \varphi, \nabla u\rangle_{\mathcal{E}}.$$

Proof. Expanding out the definition of $\langle \operatorname{div}\varphi, u\rangle_{\mathcal{V},0}$:

$$\begin{aligned} -\sum_{i\in V}(\operatorname{div}\varphi)_i u_i &= -\sum_{i,j\in V} \omega_{ij}\varphi_{ij}u_i = \sum_{i,j\in V}\omega_{ij}\varphi_{ji}u_i \\ &= \frac{1}{2}\sum_{i,j\in V}\omega_{ij}\varphi_{ji}u_i + \omega_{ji}\varphi_{ij}u_j \\ &= \frac{1}{2}\sum_{i,j\in V}\omega_{ij}\varphi_{ij}(u_j - u_i) = \langle\varphi,\nabla u\rangle_{\mathcal{E}}. \end{aligned}$$

□

2.3 The graph Laplacian

The most important graph operator for this course will be the graph Laplacian, defined as follows.

Definition 2.3.1 (Graph Laplacian)**.** *For a graph $G = (V,E,\omega)$ with $|V| = N$, define $D := \operatorname{diag}(d)$ (i.e. $D_{ii} := d_i$, and $D_{ij} := 0$ otherwise) to be the* degree matrix *of G. Then for parameters $r,s \in \mathbb{R}$, we define the* graph Laplacian *of G to be the operator represented by the matrix:*

$$\mathcal{L}_N^{(r,s)} := D^{-r}(D-\omega)D^{-s}. \tag{2.2}$$

That is

$$(\mathcal{L}_N^{(r,s)}u)_i = d_i^{-r}\sum_{j\in V}\omega_{ij}\left(\frac{u_i}{d_i^s} - \frac{u_j}{d_j^s}\right).$$

This operator can be thought of as a discrete diffusion operator on the graph, or simply as a matrix $\mathcal{L}_N^{(r,s)} \in \mathbb{R}^{N\times N}$. When it is clear from context, N and (r,s) may be

omitted. Common choices of (r,s) are: $(0,0)$, the *combinatorical Laplacian*, also called the *unnormalised Laplacian*; $(1,0)$, the *random walk Laplacian*; and $(1/2,1/2)$, the *symmetric normalised Laplacian*.

Exercise 1. Let $u_i \leq u_j$ for all $j \in V$ such that $\omega_{ij} > 0$ (i.e., i is a local minimiser of u). Show that $(\mathcal{L}^{(r,0)}u)_i \leq 0$.

Exercise 2. Show that

$$\langle \mathcal{L}^{(r,s)}u, v\rangle_{\mathcal{V},r-s} = \langle \nabla D^{-s}u, \nabla D^{-s}v\rangle_{\mathcal{E}}.$$

Exercise 3. Why do we call the matrix $\mathcal{L}^{(r,s)}$ a Laplacian?

Proposition 2.3.1. $\mathcal{L}^{(r,s)}$ is self-adjoint with respect to $\langle\cdot,\cdot\rangle_{\mathcal{V},r-s}$.

Proof.

$$\begin{aligned}\langle u, \mathcal{L}^{(r,s)}v\rangle_{\mathcal{V},r-s} &= \langle u, D^{-s}\mathcal{L}^{(0,0)}D^{-s}v\rangle_{\mathcal{V},0}\\ &= \langle D^{-s}\mathcal{L}^{(0,0)}D^{-s}u, v\rangle_{\mathcal{V},0}\\ &= \langle v, D^{-s}\mathcal{L}^{(0,0)}D^{-s}u\rangle_{\mathcal{V},0}\\ &= \langle v, D^{s-r}D^{-s}\mathcal{L}^{(0,0)}D^{-s}u\rangle_{\mathcal{V},r-s} = \langle v, \mathcal{L}^{(r,s)}u\rangle_{\mathcal{V},r-s}.\end{aligned}$$

□

Proposition 2.3.2. For all $r,s \in \mathbb{R}$, $\mathcal{L}^{(r,s)}$ is similar to $\mathcal{L}^{((r+s)/2,(r+s)/2)}$. It follows that for all $r,s \in \mathbb{R}$

i. $\mathcal{L}^{(r,s)}$ is similar to a diagonal matrix Λ.
ii. There exist U,V such that $\mathcal{L}^{(r,s)} = U\Lambda V^T$ and $V^TU = UV^T = I$.
iii. For all $r,s \in \mathbb{R}$, $\mathcal{L}^{(r,s)}$ is positive semi-definite with respect to $\langle\cdot,\cdot\rangle_{\mathcal{V},r-s}$, and hence every entry in Λ is non-negative.

Proof. It is easy to check that

$$\mathcal{L}^{(r,s)} = D^{-(r-s)/2}\mathcal{L}^{((r+s)/2,(r+s)/2)}D^{(r-s)/2}.$$

i. $\mathcal{L}^{((r+s)/2,(r+s)/2)}$ is a real symmetric matrix, so there exists Φ an orthogonal matrix and Λ a diagonal matrix such that $\mathcal{L}^{((r+s)/2,(r+s)/2)} = \Phi\Lambda\Phi^T$. Let $U := D^{-(r-s)/2}\Phi$. Then $\mathcal{L}^{(r,s)} = U\Lambda U^{-1}$.
ii. Let $V := D^{(r-s)/2}\Phi$. Then $\mathcal{L}^{(r,s)} = U\Lambda V^T$, $V^TU = \Phi^T\Phi = I$, and $UV^T = D^{-(r-s)/2}\Phi\Phi^TD^{(r-s)/2} = I$.
iii. Note that
$$\langle u, \mathcal{L}^{(r,s)}u\rangle_{\mathcal{V},r-s} = \langle u, \mathcal{L}^{(s,s)}u\rangle_{\mathcal{V},0} = u^T\mathcal{L}^{(s,s)}u$$
It follows from Exercise 2 that
$$u^T\mathcal{L}^{(s,s)}u = \frac{1}{2}\sum_{i,j=1}^{N}\omega_{ij}\left(\frac{u_i}{d_i^s} - \frac{u_j}{d_j^s}\right)^2 \geq 0$$
for all $u \in \mathcal{V}$. Hence if $\mathcal{L}^{(r,s)}u = \lambda u$ then $\lambda\langle u,u\rangle_{\mathcal{V},r-s} \geq 0$ and so $\lambda \geq 0$.

□

2.4 The spectrum of the graph Laplacian

From now on, we denote by

$$0 \leq \lambda_N^{(1)} \leq \lambda_N^{(2)} \leq \cdots \leq \lambda_N^{(N)}$$

the eigenvalues of $\mathcal{L}_N^{(r,s)}$, with corresponding eigenvectors

$$\xi_N^{(1)}, \cdots, \xi_N^{(N)} \in \mathcal{V}.$$

Here, we omitted the dependence on (r, s) for ease of notation. When it is clear from context, we also omit the dependence on N.

The value $\lambda^{(2)}$ is called the *algebraic connectivity* or *Fiedler value*. For a connected graph this value is strictly positive. The associated eigenvector is called the *Fiedler vector*.

We note the following spectral properties of $\mathcal{L}$.

Theorem 2.4.1.
1. The smallest eigenvalue $\lambda^{(1)}$ of $\mathcal{L}$ is zero.
2. This eigenvalue has multiplicity equal to the number of connected components and has corresponding eigenvectors $\xi^{(k)} \propto D^s \chi_{S_k}$, where S_k are the connected components.
3. The spectral radius $\rho(\mathcal{L})$ of $\mathcal{L}$ is bounded above by $2 \max_{i \in V} d_i^{1-(r+s)}$.

Exercise 4. Prove the above properties.

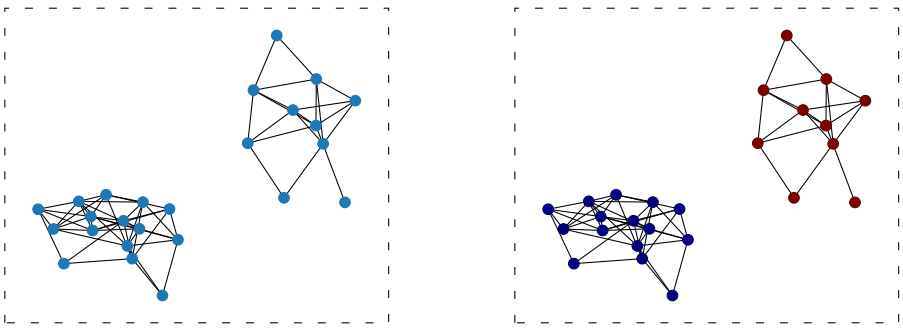

Figure 2.2: An example of a disconnected graph (left) and that same graph with nodes coloured according to the value of the Fiedler vector at that node (right).

2.5 Graph diffusion

Theorem 2.5.1 (Matrix exponential)**.** For a square matrix A, define the *matrix exponential* of A to be the matrix:

$$e^A = \sum_{n=0}^{\infty} \frac{1}{n!} A^n = I + A + \frac{1}{2} A^2 + \cdots$$

The matrix exponential has a number of key properties:

a. If $A = UBU^{-1}$, then $e^A = Ue^B U^{-1}$.
b. For $A \in \mathbb{R}^{n \times n}$, if A is symmetric then e^A is positive definite.

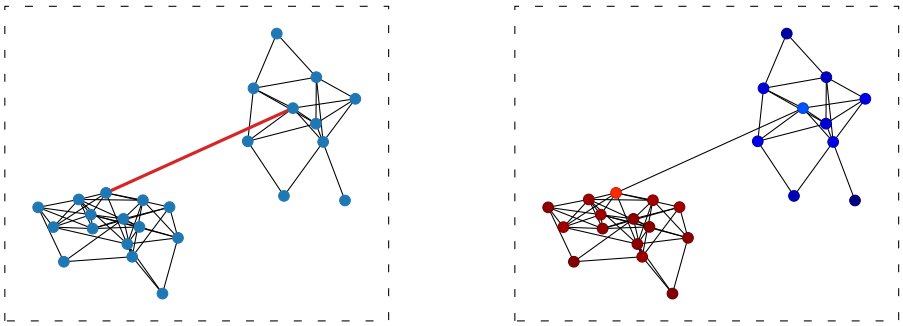

Figure 2.3: The graph from Figure 2.2 with an extra edge added to make it connected (left) and that same connected graph with nodes coloured according to the value of the Fiedler vector at that node (right).

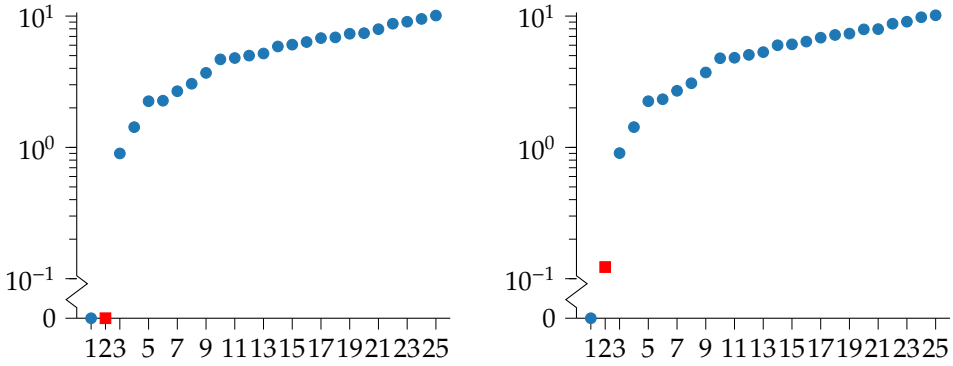

Figure 2.4: The spectra of the (unnormalised) graph Laplacians of the graphs from Figure 2.2 (left) and Figure 2.3 (right), plotted on a (crunched) log scale to emphasise the change in the second eigenvalue, indicated by a square marker.

c. If A and B commute, then
$$e^{A+B} = e^A e^B.$$
In particular, $(e^A)^{-1} = e^{-A}$.

d. $\frac{d}{dt}(e^{tA}) = Ae^{tA}$.

e. Define the *operator norm* of a matrix $A \in \mathbb{R}^{m\times n}$
$$\|A\| := \max_{x\in\mathbb{R}^n\setminus\{0\}} \frac{\|Ax\|_2}{\|x\|_2}.$$
Then if $A \in \mathbb{R}^{n\times n}$ symmetric with largest positive eigenvalue λ, then
$$\|e^A\| = e^\lambda.$$

f. If $A \in \mathbb{R}^{n\times n}$ is self-adjoint with respect to some inner product $\langle\cdot,\cdot\rangle_H$, i.e. $\langle Au, v\rangle_H = \langle u, Av\rangle_H$ for all $u, v \in RR^n$, then e^A is self-adjoint with respect to $\langle\cdot,\cdot\rangle_H$.

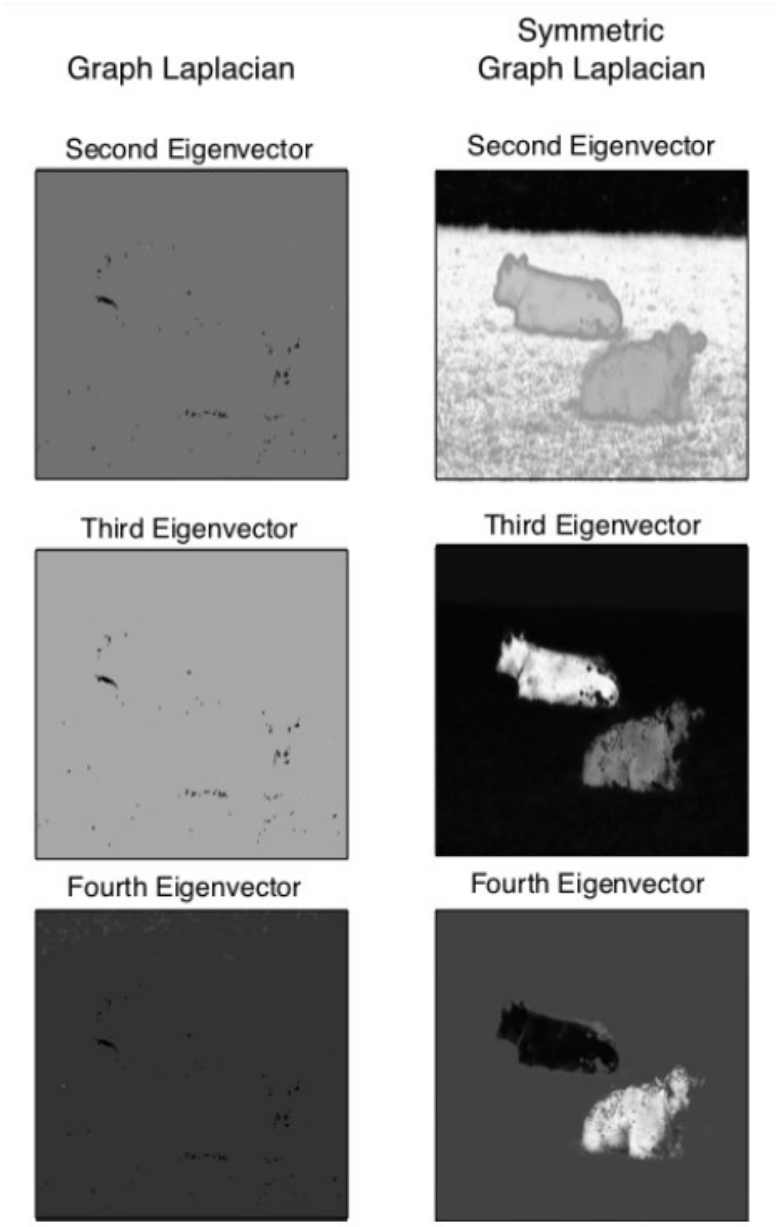

Figure 2.5: Eigenvectors of $\mathcal{L}^{(0,0)}$ (left) and $\mathcal{L}^{(1/2,1/2)}$ (right) for a graph built on the cow image from Chapter 1. Figure reproduced from [BF12, Fig. 2.1].

Proof. a.

$$\begin{aligned} e^A &= I + UBU^{-1} + \frac{1}{2}UB^2U^{-1} + \cdots \\ &= U\left(I + B + \frac{1}{2}B^2 + \cdots\right)U^{-1} \\ &= Ue^BU^{-1}. \end{aligned}$$

b. Since A is real and symmetric, $A = U\Lambda U^T$ for $U \in \mathbb{R}^{n\times n}$ orthogonal and $\Lambda \in \mathbb{R}^{n\times n}$ diagonal. Hence $e^A = Ue^\Lambda U^T$ and so $v^Te^Av = (U^Tv)^Te^\Lambda U^Tv > 0$, since e^Λ is a diagonal matrix with diagonal entries $e^{\Lambda_{ii}} > 0$.

c. Since A and B commute, we have the binomial expansion

$$(A+B)^n = \sum_{r=0}^{n}\binom{n}{r}A^rB^{n-r}.$$

Hence

$$\begin{aligned} e^{A+B} &= \sum_{n=0}^{\infty}\frac{1}{n!}(A+B)^n \\ &= \sum_{n=0}^{\infty}\sum_{r=0}^{n}\frac{1}{n!}\binom{n}{r}A^rB^{n-r} \\ &= \sum_{n=0}^{\infty}\sum_{r=0}^{n}\frac{1}{r!(n-r)!}A^rB^{n-r} \\ &= \sum_{m,k=0}^{\infty}\frac{1}{m!k!}A^mB^k \\ &= e^Ae^B, \end{aligned}$$

where the penultimate equality follows because $(m,k) = (r, n-r)$ if and only if $r = m$ and $n = m + k$.

d.

$$\begin{aligned} \frac{d}{dt}e^{tA} &= \frac{d}{dt}\left(I + tA + \frac{1}{2}t^2A^2 + \frac{1}{6}t^3A^3 + \cdots\right) \\ &= A + tA^2 + \frac{1}{2}t^2A^3 + \cdots \\ &= Ae^{tA}. \end{aligned}$$

e. As in (b), $e^A = Ue^\Lambda U^T$ and hence

$$\max_{x\in\mathbb{R}^n\setminus\{0\}}\frac{\|e^Ax\|_2}{\|x\|_2} = \max_{x\in\mathbb{R}^n\setminus\{0\}}\frac{\|Ue^\Lambda x\|_2}{\|Ux\|_2} = \max_{x\in\mathbb{R}^n\setminus\{0\}}\frac{\|e^\Lambda x\|_2}{\|x\|_2} = \max_i e^{\Lambda_{ii}}$$

where we have used that $\|Ux\|_2 = \|x\|_2$ since U is orthogonal.

f. It is easy to check by induction that A^n is self-adjoint for all n, and hence

$$\langle e^Au, v\rangle_H = \sum_{n=0}^{\infty}\frac{1}{n!}\langle A^nu, v\rangle_H = \sum_{n=0}^{\infty}\frac{1}{n!}\langle u, A^nv\rangle_H = \langle u, e^Av\rangle_H.$$

□

Note. Symmetry is important for (b). Let

$$A = \begin{pmatrix} 0 & -\pi \\ \pi & 0 \end{pmatrix} = U \begin{pmatrix} i\pi & 0 \\ 0 & -i\pi \end{pmatrix} U^{-1}.$$

Therefore

$$e^A = U \begin{pmatrix} e^{i\pi} & 0 \\ 0 & e^{-i\pi} \end{pmatrix} U^{-1} = U(-I)U^{-1} = -I$$

is negative definite. Commutativity is important for (c). Let

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \qquad \text{and} \qquad B = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}.$$

Then if we ask MATLAB to compute these:

$$e^{A+B} = \begin{pmatrix} e^2 & 0 \\ \frac{1}{2}(e^2 - 1) & 1 \end{pmatrix}$$

but

$$e^A e^B = \begin{pmatrix} e^2 & 0 \\ e - 1 & 1 \end{pmatrix} \qquad \text{and} \qquad e^B e^A = \begin{pmatrix} e^2 & 0 \\ e(e-1) & 1 \end{pmatrix}.$$

Definition 2.5.1 (Graph diffusion)**.** *From the graph Laplacian, we can define the* graph diffusion operator

$$e^{-t\mathcal{L}^{(r,s)}} u := \sum_{n=0}^{\infty} \frac{(-1)^n t^n}{n!} \left(\mathcal{L}^{(r,s)}\right)^n u.$$

Then $v(t) := e^{-t\mathcal{L}^{(r,s)}} u$ is the unique solution to the diffusion equation

$$\frac{dv}{dt} = -\mathcal{L}^{(r,s)} v(t), \qquad v(0) = u.$$

Exercise 5. Prove this.

2.5.1 The random walk on a graph

Definition 2.5.2 (Random walk)**.** *A* random walk *is a process by which a particle (a "random walker") moves through some set of locations by randomly jumping from location to location according to some rule.*

There is a natural random walk on a weighted graph, defined by the rule: **if the walker is at vertex i at step n, then it jumps to vertex j at step $n+1$ with probability** ω_{ij}/d_i. The *stationary distribution* π of a random walk on V is defined such that: $\pi_i \geq 0$ for all $i \in V$,

$$\sum_{i \in V} \pi_i = 1,$$

and if for all $i \in V$ there are k random walkers with $k\pi_i$ walkers on vertex i at step n, then for all $i \in V$:

$$\mathbb{E}[\#\text{ of walkers at vertex } i \text{ at step } n+1] = k\pi_i.$$

Proposition 2.5.2. For the above defined random walk,

$$\pi_i = \frac{d_i}{\sum_{j\in V} d_j}.$$

Proof. It is easy to see that $\pi_i \geq 0$ and that $\sum_i \pi_i = 1$. Let $W_{i,n}$ denote the number of walkers at vertex i at step n and let $W_{i,n} = kd_i/\sum_j d_j$. Then

$$\begin{aligned}
\mathbb{E}[W_{i,n+1}] &= \sum_{j\in V} \underbrace{W_{j,n}}_{\text{walkers at } j} \underbrace{\frac{\omega_{ji}}{d_j}}_{\mathbb{P}(\text{a walker at } j \text{ jumps to } i)} \\
&= \sum_{j\in V} \frac{kd_j}{\sum_{\ell\in V} d_\ell} \frac{\omega_{ji}}{d_j} \\
&= \sum_{j\in V} \frac{k\omega_{ij}}{\sum_{\ell\in V} d_\ell} \\
&= \frac{kd_i}{\sum_{\ell\in V} d_\ell} = k\pi_i.
\end{aligned}$$

□

2.5.2 The random walk perspective on graph diffusion

For $u \in \mathcal{V}$ with $u_i \geq 0$ for all $i \in V$, and $(r, s) = (1, 0)$, there is a particularly neat *random walk* perspective on graph diffusion. Suppose that at time $t = 0$ and at each vertex $i \in V$, we place $u_i d_i k$ random walkers, where $k \gg 1$ is any natural number. At each time step (of length δt), each random walker either stays put with probability $1 - \delta t$, or moves to some vertex $j \in V$ with probability $\delta t \omega_{ij}/d_i$. Let $R_i(t)$ denote the number of red walkers at vertex $i \in V$ at time $t \in \delta t\mathbb{N}$. Then $R_i(0) = u_i d_i k$ and

$$\mathbb{E}[R_i(t+\delta t)] = \mathbb{E}[R_i(t)] - \underbrace{\delta t \mathbb{E}[R_i(t)]}_{\text{walkers leaving}} + \underbrace{\delta t \sum_{j\in V} \frac{\omega_{ji}}{d_j} \mathbb{E}[R_j(t)]}_{\text{walkers arriving}}. \tag{2.3}$$

Let $\tilde{R}_i(t) := R_i(t)/d_i k$. Then by rearranging (2.3) and dividing through by $\delta t d_i k$:

$$\frac{\mathbb{E}[\tilde{R}_i(t+\delta t)] - \mathbb{E}[\tilde{R}_i(t)]}{\delta t} = -\mathbb{E}[\tilde{R}_i(t)] + \frac{1}{d_i} \sum_{j\in V} \omega_{ij} \mathbb{E}[\tilde{R}_j(t)].$$

Let $v_i(t) := \mathbb{E}[\tilde{R}_i(t)]$. Then $v_i(0) = u_i$ for all $i \in V$, and

$$\frac{v_i(t+\delta t) - v_i(t)}{\delta t} = -v_i(t) + \sum_{j\in V} \frac{\omega_{ij}}{d_i} v_j(t) = -(\mathcal{L}^{(1,0)} v(t))_i.$$

Taking $\delta t \to 0$, we recover the diffusion equation

$$\frac{dv}{dt} = -\mathcal{L}^{(1,0)} v(t), \qquad v(0) = u.$$

Note that as $k \to \infty$, $\tilde{R}_i(t) \to v_i(t)$ a.s. by the strong law of large numbers.

2.6 Graph cuts

A major approach to thinking about graph clustering comes from the following idea: if the weights ω_{ij} are supposed to describe the similarity of i and j, then the clusters we seek should be very similar *within themselves* and very dissimilar *to each other*. One way to make this quantitative is via *graph cuts*.

Definition 2.6.1 (Graph cuts). *For $S \subseteq v$, the* of S *is defined by*

$$\mathrm{Cut}(S, S^c) := \sum_{i \in S} \sum_{j \in S^c} \omega_{ij}.$$

However, this is trivially minimised by $S = \varnothing$ or $S = V$. Even excluding such trivial minimisers, minimising Cut *encourages either S or S^c to be very small. Therefore, there has been interest in various modifications to the cut functional, e.g.*

$$\mathrm{CheegerCut}(S) := \frac{\mathrm{Cut}(S, S^c)}{\min\{|S|, |S^c|\}}, \qquad \mathrm{RatioCut}(S) := \frac{\mathrm{Cut}(S, S^c)}{|S|} + \frac{\mathrm{Cut}(S, S^c)}{|S^c|}.$$

We can relate these cuts to the analysis notions from the previous lecture by defining a *graph total variation*.

Definition 2.6.2 (Graph total variation). *For $u \in \mathcal{V}$, we define the* graph total variation *of u by*

$$\mathrm{TV}_G(u) := \frac{1}{2} \sum_{i,j \in V} \omega_{ij} |(\nabla u)_{ij}| = \frac{1}{2} \sum_{i,j \in V} \omega_{ij} |u_j - u_i|.$$

Note. This total variation can also be written in a variational way akin to the continuum definition of total variation. That is

$$\begin{aligned} \mathrm{TV}_G(u) &= \langle \mathrm{sgn}(\nabla u), \nabla u \rangle_{\mathcal{E}} \\ &= \max\{\langle \varphi, \nabla u \rangle_{\mathcal{E}} \mid \varphi \in \mathcal{E} \text{ and for all } i, j \in V,\ |\varphi_{ij}| \leq 1\} \\ &= \max\{-\langle \mathrm{div}\, \varphi, u \rangle_{\mathcal{V},0} \mid \varphi \in \mathcal{E} \text{ and for all } i, j \in V,\ |\varphi_{ij}| \leq 1\}, \end{aligned}$$

where the last line follows by Proposition 2.2.1.

Theorem 2.6.1. For $S \subseteq V$, the total variation of the indicator function χ_S coincides with the graph cut of S, i.e.

$$\mathrm{TV}_G(\chi_S) = \frac{1}{2} \mathrm{Cut}(S, S^c).$$

Furthermore, it coincides with the graph *Dirichlet energy* of χ_S, i.e.

$$\mathrm{TV}_G(\chi_S) = \frac{1}{2} \sum_{i,j \in V} \omega_{ij} ((\chi_S)_i - (\chi_S)_j)^2 = ||\nabla \chi_S||_{\mathcal{E}}^2 = \langle \chi_S, \mathcal{L}^{(0,0)} \chi_S \rangle_{\mathcal{V},0}.$$

Proof. The relation to Cut is immediate from the definition. For the latter, observe that $|(\chi_S)_i - (\chi_S)_j| = ((\chi_S)_i - (\chi_S)_j)^2$ since χ_S only takes values in $\{0, 1\}$. The final equality was Exercise 2. □

Theorem 2.6.2. Let $B(u) := \min_c \sum_{i \in V} |u_i - c|$. Observe that $B(\chi_S) = \min\{|S|, |S|^c\}$. Then

$$\min_{u \in \mathcal{V}} \frac{\mathrm{TV}_G(u)}{B(u)}$$

is minimised by $u = \chi_S$, where

$$S \in \arg\min_{S \subseteq V} \mathrm{CheegerCut}(S).$$

Proof. Beyond the scope of this course. □

Chapter 3

Some graph clustering and classification methods

In this chapter, we will look at a graph clustering method and a graph classification method, both of which became popular in the early 2000s (the younger reader may therefore call these methods "classical"—the older reader may resist this suggestion).

3.1 Spectral clustering

Spectral clustering is a method with a long history, dating back to [Che69], but only became popular as a machine learning method in the early 2000s with [SM00, NJW01] and especially with von Luxburg's 'tutorial' [VL07]. The idea of spectral clustering is to use the graph Laplacian to perform a non-linear dimensionality reduction, in order to cluster data which may have complicated geometric patterns.

Let $V = \{1, ..., N\}$ and suppose we have data points $x : V \to \mathbb{R}^d$, where d may be very large. The goal of a clustering algorithm is then to sort these N feature vectors into, say, K clusters. Often, it turns out that the data points are close to a sub-manifold with much smaller dimension. This suggests the two-step procedure:

1. Reduce the data x to the lower-dimensional data $\mathbb{F}(x) : V \to \mathbb{R}^K$.
2. Cluster the $\mathbb{F}(x)$ (e.g., by K-means).

For a well-chosen $\mathbb{F}$, the clusters can then be found with a simple clustering algorithm in the second step. The idea of spectral clustering is to use the low-lying eigenvectors of the graph Laplacian on the graph built upon this data (see Chapter 6) to define $\mathbb{F}$. Let $\xi^{(k)} \in \mathcal{V}$ be the eigenvectors of $\mathcal{L}^{(r,s)}$ in increasing order of eigenvalue, and define

$$\mathbb{F}(x) : i \mapsto \left(\xi_i^{(1)}, ..., \xi_i^{(K)}\right)^T \in \mathbb{R}^K.$$

In order to develop an intuition why this is a useful choice, consider the case of K perfect clusters. That is, the graph built on the data has K disconnected components. Then, by Theorem 2.4.1, $\xi^{(k)} = D^s \chi_{S_k}$ where the $S_k \subset V$ are the clusters, and hence each $(\mathbb{F}(x))_i$ lies on the axis corresponding to the cluster which i belongs to. Then the $\mathbb{F}(x)$ are very easy to cluster.

For real data sets, we generally do not have perfectly disconnected components of course. However, if the data is clustered in roughly K groups, we would expect that the first K eigenvalues are close to zero with a spectral gap between $\lambda^{(K)}$ and $\lambda^{(K+1)}$, with the

eigenvectors still approximately indicating the location of the clusters. This is visualised in Figures 2.4 and 2.5, of which the latter somewhat dramatically shows the impact of normalisation.

The immense success of spectral clustering lies in its flexibility to deal with data of various shapes and complicated geometries. Despite its enormous popularity and success, our theoretical understanding of the performance of spectral clustering is still rather vague. While there is vast empirical evidence of clustering examples in which e.g. spectral clustering by far outperforms k-means, there exists little rigorous theoretical analysis—even for very simple cases—that would prove the superiority of spectral clustering, partly due to the two-step procedure which complicates its theoretical analysis. For recent work, see [GTHH21, GBT21, HHOS22].

3.2 Laplace learning

Laplace learning, introduced in [ZGL03], is a method which uses the graph Laplacian $\mathcal{L}$ to propagate some *a priori* labels $f : Z \to \{0,1\}$ on a (typically, very small) subset Z of V. Extend f to a function on V taking the value 0 on $V \setminus Z$. The fundamental idea is to solve the following minimisation problem:

$$\underset{u\in\mathcal{V}}{\arg\min} \qquad \mathcal{E}(u) := \frac{1}{4}\sum_{i,j\in V}\omega_{ij}\left(\frac{u_i}{d_i^s}-\frac{u_j}{d_j^s}\right)^2 \qquad \text{s.t.} \qquad u|_Z = f. \tag{3.1}$$

That is, we want to encourage similar vertices (i.e., those with a high ω_{ij} on the edge between them) to be given the same label by u, but also enforce that u agrees with our *a priori* labels f.

Theorem 3.2.1. $u \in \mathcal{V}$ solves (3.1) if and only if

$$\begin{aligned} \mathcal{L}^{(r,s)}u &= 0, \quad \text{on } V\setminus Z,\\ u &= f, \qquad \text{on } Z. \end{aligned} \tag{3.2}$$

Proof. First, recall from Exercise 2 that

$$\mathcal{E}(u) = \frac{1}{2}\langle u, \mathcal{L}^{(r,s)}u\rangle_{\mathcal{V},r-s}.$$

Then for $u|_Z = f$

$$\begin{aligned} u \text{ solves (3.1) iff } &\forall v \text{ s.t. } v|_Z = f, \mathcal{E}(u) \le \mathcal{E}(v)\\ \text{iff } &\forall v \text{ s.t. } v|_Z = f, 0 \le \langle v-u, \mathcal{L}^{(r,s)}(u+v)\rangle_{\mathcal{V},r-s}\\ \text{iff } &\forall v \text{ s.t. } v|_Z = f, 0 \le \langle v-u, \mathcal{L}^{(r,s)}(v-u)\rangle_{\mathcal{V},r-s} + 2\langle v-u, \mathcal{L}^{(r,s)}u\rangle_{\mathcal{V},r-s} \end{aligned}$$

Hence if $\mathcal{L}^{(r,s)}u = 0$ on $V \setminus Z$ then u solves (3.1). If $(\mathcal{L}^{(r,s)}u)_i \neq 0$ for some $i \in V \setminus Z$ then let $v = u + t\chi_{\{i\}}$. It follows that $v|_Z = f$ and

$$\langle v-u, \mathcal{L}^{(r,s)}(v-u)\rangle_{\mathcal{V},r-s} + 2\langle v-u, \mathcal{L}^{(r,s)}u\rangle_{\mathcal{V},r-s} = t^2\langle \chi_{\{i\}}, \mathcal{L}^{(r,s)}\chi_{\{i\}}\rangle_{\mathcal{V},r-s} + 2t d_i^{r-s}(\mathcal{L}^{(r,s)}u)_i < 0$$

for sufficiently small t of opposite sign to $(\mathcal{L}^{(r,s)}u)_i$, and hence u does not solve (3.1). □

One way of looking at (3.2) is that the desired u is a *harmonic extension* of the labels f. Finally, given u solving (3.1), we threshold u to assign labels to all of V.

3.2.1 Random walk formulation

Let $i \in V$ and consider the random walk (already introduced in Section 2.5.1):

$$X_0^i, X_1^i, X_2^i, \dots$$

defined by $X_0^i = i$ and

$$\mathbb{P}(X_{n+1}^i = j | X_n^i = k) = \frac{\omega_{jk}}{d_k} =: P_{kj}.$$

Note that $P = I - \mathcal{L}^{(1,0)}$. Let

$$\tau_i := \inf\{n \geq 0 | X_n^i \in Z\},$$

this is called the *hitting time* of Z from i.

Theorem 3.2.2. Let $s = 0$ and let u solve (3.2). Then for all $i \in V$

$$u_i = \mathbb{E}[f_{X_{\tau_i}^i}].$$

Proof (sketch). This is a consequence of Doob's optional stopping theorem. □

That is, Laplace learning can be understood as putting a bunch of random walkers on a vertex i, letting those walkers walk until they hit a labelled vertex, and then giving i the label those walkers hit most frequently.

3.3 Poisson learning

Unfortunately, for Z very small relative to V, Laplace learning degenerates, giving $u \approx c$ for some constant c on almost all of the vertices in $V \setminus Z$. This phenomenon is explained by the following exercise.

Exercise 6. Show that for Z sufficiently small, the *mixing time* for the random walk is smaller than the hitting time of Z from i, for most $i \in V \setminus Z$. Show that therefore $c = \langle f, \chi_Z \rangle_{\mathcal{V},1} / \langle \chi_Z, \chi_Z \rangle_{\mathcal{V},1}$.

In [CCTS20], a new method was proposed to avoid this issue, called *Poisson learning*. We modify this method slightly to fit it into our more general framework. Define

$$\tilde{f}^{(r,s)} := \frac{\langle f, D^s \chi_Z \rangle_{\mathcal{V},r-s}}{\langle \chi_Z, D^s \chi_Z \rangle_{\mathcal{V},r-s}},$$

the "average label". Note that $\tilde{f}^{(r,s)} = \tilde{f}^{(r,0)}$. Then consider the Poisson equation:

$$\mathcal{L}^{(r,s)} u = f - \tilde{f}^{(r,s)} \chi_Z, \qquad \langle u, D^s \mathbf{1} \rangle_{\mathcal{V},r-s} = 0. \tag{3.3}$$

Theorem 3.3.1. Let G be a connected graph. Then there exists a unique solution to (3.3).

Proof. Note that $\langle f - \tilde{f}^{(r,s)} \chi_Z, D^s \mathbf{1} \rangle_{\mathcal{V},r-s} = 0$, and that, by connectedness, $D^s \mathbf{1}$ is the only eigenvector of $\mathcal{L}^{(r,s)}$ with a zero eigenvalue. Hence

$$u^* := \sum_{k=2}^{N} \frac{1}{\lambda^{(k)}} \langle f - \tilde{f}^{(r,s)} \chi_Z, \xi^{(k)} \rangle_{\mathcal{V},r-s} \xi^{(k)}$$

solves $\mathcal{L}^{(r,s)} u^* = f - \tilde{f}^{(r,s)} \chi_Z$. Finally, if $\mathcal{L}^{(r,s)} u = f - \tilde{f}^{(r,s)} \chi_Z$ then $\mathcal{L}^{(r,s)}(u - u^*) = 0$ and hence $u - u^* \propto D^s \mathbf{1}$, so if $\langle u, D^s \mathbf{1} \rangle_{\mathcal{V},r-s} = 0$ then $u = u^*$. □

Theorem 3.3.2. Let G be a connected graph. Then u solves (3.3) if and only if u solves

$$\arg\min_{u \in \mathcal{V}} \quad \mathcal{E}(u) - \langle u, f - \tilde{f}^{(r,s)}\chi_Z\rangle_{\mathcal{V},r-s} \qquad \text{s.t.} \qquad \langle u, D^s\mathbf{1}\rangle_{\mathcal{V},r-s} = 0. \tag{3.4}$$

Proof. By Exercise 2, u minimises (3.4) if and only if u minimises

$$\langle u, \mathcal{L}^{(r,s)}u - 2(f - \tilde{f}^{(r,s)}\chi_Z)\rangle_{\mathcal{V},r-s},$$

if and only if u minimises

$$\langle u - u^*, \mathcal{L}^{(r,s)}(u - u^*)\rangle_{\mathcal{V},r-s},$$

if and only if $u = u^*$, since G is connected and $\langle u, D^s\mathbf{1}\rangle_{\mathcal{V},r-s} = 0 = \langle u^*, D^s\mathbf{1}\rangle_{\mathcal{V},r-s}$, where u^* is the unique solution to (3.3) defined in the previous theorem. □

In [CCTS20], it was shown that this method is much more effective than Laplace learning when there are very few labels Z.

3.3.1 Random walk formulation

We can derive Poisson learning from a random walk formulation in the $(r,s) = (1,0)$ case. Instead of having walkers walk until they hit a label, as in Laplace learning, instead we have walkers begin at the labelled nodes. Define

$$(w_T)_i := \sum_{t=0}^{T}\sum_{j\in Z} d_j f_j \mathbb{P}(X_t^j = i).$$

What this keeps track of is as follows: imagine that we have walkers who begin at each labelled node $j \in Z$, and carry with them a contribution $d_j f_j$, which they confer to each vertex they encounter. Then $(w_T)_i$ records the expected total contribution conferred to vertex i after T steps of the walk.

As $t \to \infty$,

$$\mathbb{P}(X_t^j = i) \to \frac{d_i}{\sum_k d_k} =: \pi_i$$

and hence for large t,

$$\sum_{j\in Z} d_j f_j \mathbb{P}(X_t^j = i) \approx \frac{d_i}{\sum_k d_k}\sum_{j\in Z} d_j f_j = \frac{d_i}{\sum_k d_k}\sum_{j\in Z} d_j \tilde{f}^{(1,0)} \approx \sum_{j\in Z} d_j \tilde{f}^{(1,0)}\mathbb{P}(X_t^j = i).$$

This long-time behaviour is undesirable, as it corresponds to when our starting location has been 'forgotten' by the random walk. We therefore subtract it off, and define

$$(u_T)_i := \frac{1}{d_i}\sum_{t=0}^{T}\sum_{j\in Z} d_j(f_j - \tilde{f}^{(1,0)})\mathbb{P}(X_t^j = i)$$

Theorem 3.3.3. For all $T \geq 0$,

$$(u_{T+1})_i = (u_T)_i - (\mathcal{L}^{(1,0)}u_T)_i + f - \tilde{f}^{(1,0)}\chi_Z$$

If the graph is connected and every eigenvalue of P (except for the 1 eigenvalue

corresponding to $P\mathbf{1} = \mathbf{1}$) lies in $(-1, 1)$, then $u_T \to u$ as $T \to \infty$ where u solves (3.3) with $(r, s) = (1, 0)$.

Proof. Let

$$(G_T)_{ij} := \frac{1}{d_i} \sum_{t=0}^{T} \mathbb{P}(X_t^j = i).$$

Then

$$\begin{aligned} d_i(G_T)_{ij} &= \delta_{ij} + \sum_{t=0}^{T-1} \mathbb{P}(X_{t+1}^j = i) \\ &= \delta_{ij} + \sum_{t=0}^{T-1} \sum_{k \in V} \frac{\omega_{ik}}{d_k} \mathbb{P}(X_t^j = k) \qquad \text{(by the definition of the random walk)} \\ &= \delta_{ij} + \sum_{k \in V} \omega_{ik} (G_{T-1})_{kj}. \end{aligned}$$

That is, recalling that $\mathcal{L}^{(1,0)} := I - D^{-1}\omega$,

$$G_T = D^{-1}(I + \omega G_{T-1}) = D^{-1} + G_{T-1} - \mathcal{L}^{(1,0)} G_{T-1}.$$

Then

$$\begin{aligned} u_{T+1} &= G_{T+1} D(f - \tilde{f}^{(1,0)} \chi_Z) \\ &= D^{-1} D(f - \tilde{f}^{(1,0)} \chi_Z) + G_T D(f - \tilde{f}^{(1,0)} \chi_Z) - \mathcal{L}^{(1,0)} G_T D(f - \tilde{f}^{(1,0)} \chi_Z) \\ &= u_T - \mathcal{L}^{(1,0)} u_T + f - \tilde{f}^{(1,0)} \chi_Z, \end{aligned}$$

as desired.

Now, let $u \in \mathcal{V}$ be the unique solution to

$$\mathcal{L}^{(1,0)} u = f - \tilde{f}^{(1,0)} \chi_Z$$

with $\langle u, \mathbf{1} \rangle_{\mathcal{V},1} = 0$. Then defining $\delta u_T := u_T - u$ we get that

$$\delta u_{T+1} = u_T - u - \mathcal{L}^{(1,0)} u_T + v = \delta u_T - \mathcal{L}^{(1,0)} \delta u_T = P \delta u_T.$$

Hence, by the assumption on the eigenvalues of P, as $T \to \infty$,

$$\delta u_T \to \frac{\langle \delta u_0, \mathbf{1} \rangle_{\mathcal{V},1}}{\langle \mathbf{1}, \mathbf{1} \rangle_{\mathcal{V},1}} \mathbf{1} = \frac{\langle u_0, \mathbf{1} \rangle_{\mathcal{V},1}}{\langle \mathbf{1}, \mathbf{1} \rangle_{\mathcal{V},1}} \mathbf{1} = 0,$$

since

$$\langle u_0, \mathbf{1} \rangle_{\mathcal{V},1} = \sum_{i \in V} \sum_{j \in Z} d_j (f_j - \tilde{f}^{(1,0)}) \mathbb{P}(X_t^j = i) = \sum_{j \in Z} d_j (f_j - \tilde{f}^{(1,0)}) = 0.$$

Hence $u_T \to u$ with u solving (3.3), as desired. □

Chapter 4

The Allen–Cahn equation and MBO scheme on graphs

4.1 Allen–Cahn and MBO

4.1.1 Ginzburg–Landau and Total Variation functionals

Definition 4.1.1 (Graph Ginzburg–Landau functional)**.** *Let $u \in \mathcal{V}$, and let $W : \mathbb{R} \to \mathbb{R}$ be a continuous* double well potential *with wells at 0 and 1. That is, $W(x) \geq 0$ for all $x \in \mathbb{R}$ and $W(0) = W(1) = 0$. Then the* graph Ginzburg–Landau functional *is defined (for parameter $\varepsilon > 0$) by:*

$$\mathrm{GL}_\varepsilon(u) := \frac{1}{2} \|\nabla u\|_{\mathcal{E}}^2 + \frac{1}{\varepsilon} \sum_{i \in V} W(u_i).$$

This can be related to the graph cut (i.e., total variation) via a special type of convergence. For a given subset of vertices $S \subset V$, define the indicator function $\chi_S : V \to \{0, 1\}$ on S by

$$\chi_S(i) = \begin{cases} 1\,, & \text{if } i \in S\,, \\ 0\,, & \text{if } i \notin S\,. \end{cases}$$

Then we relate GL_ε to TV_G.

Theorem 4.1.1 (See [vGB18, Theorem 3.1])**.** Define the following functional on $\mathcal{V}$:

$$\mathrm{TV}_0(u) := \begin{cases} \frac{1}{2}\,\mathrm{TV}_G(\chi_S), & \text{if } u = \chi_S \text{ for some } S \subseteq V, \\ \infty, & \text{otherwise.} \end{cases}$$

Then as $\varepsilon \to 0$, GL_ε Γ-converges to TV_0.

Note. For a definition of Γ-convergence, see [Bra07, Definitions 1.5 and 1.45]. For our purposes, all that matters here is that this result means that minimisers of GL_ε converge to minimisers of TV_0, and the latter are what we thought of as good clusterings.

4.1.2 Graph Ginzburg–Landau and Allen–Cahn

As we just saw, the graph Ginzburg–Landau functional is a relaxation of the graph cut, and is therefore a natural energy for binary clustering.

In the finale of this course, our focus will be on binary *classification*, where we have an additional *a priori* segmentation $f : Z \to \{0, 1\}$. We will therefore generalise this energy to include fidelity forcing to f, and also introduce some further pieces to incorporate different Laplacians.

Definition 4.1.2 (Graph Ginzburg–Landau functional with fidelity)**.** *The* graph Ginzburg–Landau functional with fidelity *is given by:*

$$\mathrm{GL}^{(r,s)}_{\varepsilon,\mu,f}(u) := \frac{1}{2}\left\|\nabla(D^{-s}u)\right\|^2_{\mathcal{E}} + \frac{1}{\varepsilon}\left\langle W \circ u, \mathbf{1}\right\rangle_{\mathcal{V},r-s} + \frac{1}{2}\left\langle u - f, M(u-f)\right\rangle_{\mathcal{V},r-s},$$

where $M := \mathrm{diag}(\mu)$ *for* $\mu \in \mathcal{V}_{[0,\infty)}$ *the* fidelity parameter *with* $Z = \mathrm{supp}(\mu)$*.*

Note that μ_i paramaterises the strength of the fidelity to the reference at vertex i. We have here extended f to be zero on $V \setminus Z$. We will suppress parameters when they are clear from context.

Theorem 4.1.2. This Ginzburg–Landau functional has the following first variation:

$$\mathrm{GL}^{(r,s)}_{\varepsilon,\mu,f}(u+\delta u) = \mathrm{GL}^{(r,s)}_{\varepsilon,\mu,f}(u) + \left\langle \mathcal{L}^{(r,s)}u + \frac{1}{\varepsilon}W' \circ u + M(u-f), \delta u\right\rangle_{\mathcal{V},r-s} + o(\delta u)$$

Proof. We first recall an important lemma (shown in Lecture 2, Exercise 1):

$$\|\nabla v\|^2_{\mathcal{E}} = \langle v, \mathcal{L}^{(0,0)}v\rangle_{\mathcal{V},0}.$$

Given this lemma, it follows that

$$\begin{aligned}
\left\|\nabla(D^{-s}u)\right\|^2_{\mathcal{E}} &= \langle D^{-s}u, \mathcal{L}^{(0,0)}D^{-s}u\rangle_{\mathcal{V},0} \\
&= \langle u, D^{-s}\mathcal{L}^{(0,0)}D^{-s}u\rangle_{\mathcal{V},0} \\
&= \langle u, D^{-(r-s)}D^{-s}\mathcal{L}^{(0,0)}D^{-s}u\rangle_{\mathcal{V},r-s} \\
&= \langle u, \mathcal{L}^{(r,s)}u\rangle_{\mathcal{V},r-s}.
\end{aligned}$$

Finally, we verify each part of the first variation separately:

$$\begin{aligned}
\frac{1}{2}\langle u+\delta u, \mathcal{L}^{(r,s)}(u+\delta u)\rangle_{\mathcal{V},r-s} &= \frac{1}{2}\langle u, \mathcal{L}^{(r,s)}u\rangle_{\mathcal{V},r-s} \\
&\quad + \langle \delta u, \mathcal{L}^{(r,s)}u\rangle_{\mathcal{V},r-s} + o(\delta u), \\
\langle W \circ (u+\delta u), \mathbf{1}\rangle_{\mathcal{V},r-s} &= \langle W \circ u + \delta u \odot (W' \circ u) + o(\delta u), \mathbf{1}\rangle_{\mathcal{V},r-s} \\
&= \langle W \circ u, \mathbf{1}\rangle_{\mathcal{V},r-s} + \langle \delta u, W' \circ u\rangle_{\mathcal{V},r-s} + o(\delta u), \\
\frac{1}{2}\langle u+\delta u - f, M(u+\delta u - f)\rangle_{\mathcal{V},r-s} &= \frac{1}{2}\langle u-f, M(u-f)\rangle_{\mathcal{V},r-s} \\
&\quad + \langle \delta u, M(u-f)\rangle_{\mathcal{V},r-s} + o(\delta u).
\end{aligned}$$

For the first and third expression, we used that $\mathcal{L}^{(r,s)}$ and M are self-adjoint with respect to $\langle\cdot,\cdot\rangle_{\mathcal{V},r-s}$. For the second expression, we used the definition of the derivative. Note that $\odot$ denotes the Hadamard (i.e., componentwise) product. □

Given these first variations, we can define the Allen–Cahn gradient flow of the Ginzburg–Landau functional.

Definition 4.1.3 (Graph Allen–Cahn equation with fidelity)**.** *The* graph Allen–Cahn equation with fidelity *is defined to be the ODE:*

$$\frac{du}{dt} = -\mathcal{L}^{(r,s)}u - \frac{1}{\varepsilon}W' \circ u - M(u - f). \tag{4.1}$$

By Theorem 4.1.2, this is the gradient flow of $\mathrm{GL}^{(r,s)}_{\varepsilon,\mu,f}$ *with respect to* $\langle \cdot, \cdot \rangle_{\mathcal{V},r-s}$.

4.1.3 The Merriman–Bence–Osher (MBO) scheme

In Chapter 2 we defined graph diffusion. To define the graph MBO scheme, we will first define graph diffusion with an extra fidelity forcing.

Definition 4.1.4 (Fidelity-forced graph diffusion)**.** *The* fidelity-forced graph diffusion *of* $u_0 \in \mathcal{V}$ *is*

$$\frac{du}{dt}(t) = -\mathcal{L}u(t) - M(u(t) - f), \qquad u(0) = u_0. \tag{4.2}$$

For $t, x \in \mathbb{R}$, let $F_t(x) := (1 - e^{-tx})/x$, and extend F_t to (real) matrix inputs via its Taylor series. Then, for any given $u_0 \in \mathcal{V}$, (4.2) has a unique solution, given by the map:

$$u(t) = \mathcal{S}_t u_0 := e^{-t(\mathcal{L}+M)}u_0 + F_t(\mathcal{L} + M)Mf.$$

Exercise 7. Prove that this is the unique solution to (4.2).

Then the graph MBO scheme (with fidelity forcing) is defined as follows.

Definition 4.1.5 (Graph MBO scheme (with fidelity forcing))**.** *The* graph MBO scheme *defines a sequence* $u_n \in \mathcal{V}_{\{0,1\}}$ *from initial condition* $u_0 \in \mathcal{V}_{[0,1]}$ *by the following diffusion-thresholding scheme with time step* $\tau \geq 0$*:*

$$(u_{n+1})_i = \begin{cases} 1, & \text{if } (\mathcal{S}_\tau u_n)_i \geq 1/2, \\ 0, & \text{if } (\mathcal{S}_\tau u_n)_i < 1/2. \end{cases} \tag{4.3}$$

That is, u_{n+1} *is defined by evolving the forced diffusion of* u_n *for time* τ*, and then thresholding.*

4.2 The SDIE scheme

4.2.1 The SDIE scheme and its variational form

Definition 4.2.1 (SDIE scheme for the Allen–Cahn equation)**.** *We define the following numerical scheme for (4.1), which we call a* semi-discrete implicit Euler (SDIE) scheme*: for time step* $\tau > 0$

$$u_{n+1} = \mathcal{S}_\tau u_n - \frac{\tau}{\varepsilon}W' \circ u_{n+1}. \tag{4.4}$$

What this is doing is diffusing u_n for time τ using the solution operator for fidelity-forced diffusion, and then taking an implicit Euler step against the potential.

4.2.2 Convexity recall

Definition 4.2.2 (Convex function)**.** *A function f on an interval $T \subseteq \mathbb{R}$ is said to be* convex *if for all $x, y \in T$ and $t \in [0,1]$*

$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y).$$

Geometrically, it always lies beneath its secant lines.

Theorem 4.2.1. Let $f : \mathbb{R} \to \mathbb{R}$ be differentiable and convex. Then x is a global minimum of f (i.e., $f(x) \leq f(y)$ for all $y \in \mathbb{R}$) if and only if $f'(x) = 0$.

Proof. Suppose that x is a global minimum of f. Then for $h > 0$

$$\frac{f(x+h) - f(x)}{h} \leq 0$$

and for $h < 0$

$$\frac{f(x+h) - f(x)}{h} \geq 0,$$

so $f'(x) = 0$. Now suppose that there exists $f(y) < f(x)$, and let $\alpha = f(x) - f(y) > 0$. Then for all $t \in [0,1]$

$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y) = f(x) - (1-t)\alpha$$

Let $tx + (1-t)y = x + h$, i.e. $h = (1-t)(y-x)$. If $y > x$, $h > 0$ and

$$\frac{f(x+h) - f(x)}{h} \leq -\frac{(1-t)\alpha}{(1-t)(y-x)} = -\frac{\alpha}{y-x}$$

If $y < x$, $h < 0$ and

$$\frac{f(x+h) - f(x)}{h} \geq -\frac{\alpha}{y-x} = \frac{\alpha}{|y-x|}.$$

In either case, taking $t \to 1$ and therefore $h \to 0$, we get $|f'(x)| \geq \alpha/|y-x|$, and so $f'(x) \neq 0$. □

4.2.3 The variational form of the SDIE scheme

Theorem 4.2.2 (Variational form of SDIE scheme)**.** Let W be differentiable, $r' \in \mathbb{R}$, and let τ, ε be such that $y \mapsto \frac{\tau}{\varepsilon} W(y) + \frac{1}{2}y^2$ is a convex function. Then u_{n+1} solves (4.4) if and only if

$$u_{n+1} \in \arg\min_{u \in \mathcal{V}} \frac{\tau}{\varepsilon} \langle W \circ u, \mathbf{1} \rangle_{\mathcal{V},r'} + \frac{1}{2} \|u - \mathcal{S}_\tau u_n\|^2_{\mathcal{V},r'}. \tag{4.5}$$

Proof. u_{n+1} solves (4.5) if and only if for all $i \in V$,

$$(u_{n+1})_i \in \arg\min_{x \in \mathbb{R}} \frac{\tau}{\varepsilon} W(x) + \frac{1}{2}(x - (\mathcal{S}_\tau u_n)_i)^2.$$

Since the objective function is differentiable and convex, this holds if and only if

$$\frac{\tau}{\varepsilon} W'((u_{n+1})_i) + (u_{n+1})_i - (\mathcal{S}_\tau u_n)_i = 0 \qquad \text{for all } i \in V$$

by the previous theorem. This is precisely (4.4). □

4.2.4 The variational form of the MBO scheme

Theorem 4.2.3. For all $r' \in \mathbb{R}$, u_{n+1} given by (4.3) solves

$$u_{n+1} \in \operatorname*{arg\,min}_{u \in \mathcal{V}_{[0,1]}} \langle \mathbf{1} - 2\mathcal{S}_\tau u_n, u \rangle_{\mathcal{V},r'} \tag{4.6}$$

which is equivalent to

$$u_{n+1} \in \operatorname*{arg\,min}_{u \in \mathcal{V}_{[0,1]}} \frac{1}{2}\langle \mathbf{1} - u, u \rangle_{\mathcal{V},r'} + \frac{1}{2}\|u - \mathcal{S}_\tau u_n\|^2_{\mathcal{V},r'}. \tag{4.7}$$

Proof. u_{n+1} solves (4.6) if and only if for all $i \in V$,

$$(u_{n+1})_i \in \operatorname*{arg\,min}_{x \in [0,1]} (1 - 2(\mathcal{S}_\tau u_n)_i)x = \begin{cases} \{0\}, & (\mathcal{S}_\tau u_n)_i < \frac{1}{2}, \\ [0,1], & (\mathcal{S}_\tau u_n)_i = \frac{1}{2}, \\ \{1\}, & (\mathcal{S}_\tau u_n)_i > \frac{1}{2}, \end{cases}$$

which is satisfied by u_{n+1} given by (4.3).

Next, we rewrite the objective functional in (4.7):

$$\begin{aligned} &\frac{1}{2}\langle \mathbf{1} - u, u \rangle_{\mathcal{V},r'} + \frac{1}{2}\|u - \mathcal{S}_\tau u_n\|^2_{\mathcal{V},r'} \\ &\quad = \frac{1}{2}\langle \mathbf{1}, u \rangle_{\mathcal{V},r'} - \frac{1}{2}\langle u, u \rangle_{\mathcal{V},r'} + \frac{1}{2}\langle u, u \rangle_{\mathcal{V},r'} - \langle u, \mathcal{S}_\tau u_n \rangle_{\mathcal{V},r'} + \frac{1}{2}\|\mathcal{S}_\tau u_n\|^2_{\mathcal{V},r'} \\ &\quad = \frac{1}{2}\langle \mathbf{1} - 2\mathcal{S}_\tau u_n, u \rangle_{\mathcal{V},r'} + \frac{1}{2}\|\mathcal{S}_\tau u_n\|^2_{\mathcal{V},r'} \end{aligned}$$

which as the same minimisers in u as $\langle \mathbf{1} - 2\mathcal{S}_\tau u_n, u \rangle_{\mathcal{V},r'}$. □

We notice that (4.5) and (4.7) are almost identical. To make them exactly identical, we desire:

1. W to equal $\frac{1}{2}x(1-x)$ on $[0,1]$.
2. W to force the minimisers to lie in $\mathcal{V}_{[0,1]}$.
3. τ equal to ε.

4.3 The double-obstacle potential, subdifferentiability, and weak differentiability

We can satisfy all of our *desiderata* by making the folowing definition.

Definition 4.3.1 (Double-obstacle potential)**.** *The* double-obstacle potential $W : \mathbb{R} \to [0,\infty]$ *is defined by*

$$W(x) = \begin{cases} \frac{1}{2}x(1-x), & x \in [0,1], \\ \infty, & \text{otherwise.} \end{cases}$$

See e.g. [BE91, BKS18, OP88] for discussion of this potential.

There is just a little problem: in the above definition of the Allen–Cahn flow and SDIE scheme, we used the derivative of W. Which this W doesn't have at 0 and 1.

Question. How can you differentiate a function that doesn't have a derivative?

4.3.1 Subdifferentiability

Definition 4.3.2 (Subdifferential, see [ET99, Definition 5.1]). *Let $f : T \to [0, \infty]$ be a convex function, where $T \subseteq \mathbb{R}$ an interval. Then $c \in \mathbb{R}$ is a* subderivative *of f at $x \in T$ if*

$$f(y) - f(x) \geq c(y - x)$$

for all $y \in T$. Geometrically, c is the slope of a line which coincides with f at x and always lies below the graph of f. The set of such subderivatives, denoted $\partial f(x)$, is called the subdifferential *of f at x.*

Example 4.3.1. Let $f(x) = |x|$. Let $x > 0$, and we desire $|y| - x \geq c(y - x)$. Taking $y = x + 1$ we get $1 \geq c$. Taking $y = x/2$ we get $-x/2 \geq -cx/2$ and so $c \geq 1$. Finally, $c = 1$ works because $|y| \geq y$ for all y. Thus $\partial f(x) = \{1\}$ for $x > 0$. Likewise, $\partial f(x) = \{-1\}$ for $x < 0$. Finally, for $x = 0$ we get $|y| \geq cy$, which is true whenever $c \in [-1, 1]$.

Theorem 4.3.1. Let $f : \mathbb{R} \to [0, \infty]$ be a convex function.

1. If f is differentiable at x, then $\partial f(x) = \{f'(x)\}$.
2. If $c_1, c_2 \in \partial f(x)$ and $c_1 < c < c_2$, then $c \in \partial f(x)$.

Exercise 8. Prove this.

Write

$$W(x) = \frac{1}{2}x(1 - x) + I_{[0,1]}(x)$$

where $I_{[0,1]}$ is the indicator function taking value 0 on $[0, 1]$ and ∞ elsewhere. Then

$$\text{“}\partial W(x)\text{”} = \frac{1}{2} - x + \partial I_{[0,1]}(x)$$

where I have been a little loose with my definitions here, as W is not convex, but the above expression for W separates it into a convex function plus a differentiable function.

Proposition 4.3.2. $I_{[0,1]}$ is a convex function and has subdifferential:

$$\partial I_{[0,1]}(x) = \begin{cases} \varnothing, & x < 0, \\ (-\infty, 0], & x = 0, \\ \{0\}, & 0 < x < 1, \\ [0, \infty), & x = 1, \\ \varnothing, & x > 1. \end{cases}$$

Proof. We wish to show that for all $x, y \in \mathbb{R}$ and $t \in [0, 1]$

$$I_{[0,1]}(tx + (1 - t)y) \leq tI_{[0,1]}(x) + (1 - t)I_{[0,1]}(y).$$

If either of $x, y \notin [0, 1]$, then the RHS is infinite so this is satisfied. If $x, y \in [0, 1]$ then $tx + (1 - t)y \in [0, 1]$ and so $LHS = RHS = 0$. $\partial I_{[0,1]}(x) = \{0\}$ for $x \in (0, 1)$ follows by the first part of the above theorem, and $\partial I_{[0,1]}(x) = \varnothing$ for $x \notin [0, 1]$ follows since $I_{[0,1]}(y) - I_{[0,1]}(x)$ is either $-\infty$ or $\infty - \infty$, so is never $\geq c(y - x)$ for any $c \in \mathbb{R}$. We will check $x = 0$ (the case $x = 1$ goes the same way). We seek $c \in \mathbb{R}$ such that

$$cy \leq I_{[0,1]}(y) = \begin{cases} 0, & y \in [0, 1], \\ \infty, & y \notin [0, 1]. \end{cases}$$

Taking $y = 1$ we get that $c \leq 0$, which is also plainly sufficient. □

4.3.2 Weak differentiability and the H^1 space

Definition 4.3.3. *Let $a < b \in \mathbb{R}$ and $f : (a,b) \to \mathbb{R}$. Then $df/dt : (a,b) \to \mathbb{R}$ is a* weak derivative *of f if it satisfies*

$$\int_a^b \frac{df}{dt}(t)\varphi(t)\,dt = -\int_a^b f(t)\frac{d\varphi}{dt}(t)\,dt$$

for all infinitely differentiable $\varphi : (a,b) \to \mathbb{R}$ which are zero outside of some closed interval contained in (a,b). Let $u \in \mathcal{V}_{t\in(a,b)}$. Then $du/dt \in \mathcal{V}_{t\in(a,b)}$ is a weak derivative of u if for all $i \in V$, $(du/dt)_i$ is a weak derivative of u_i. We define the Sobolev space

$$H^1((a,b);\mathcal{V}) := \{u \in L^2((a,b);\mathcal{V}) | du/dt \in L^2((a,b);\mathcal{V})\}$$

and the local Sobolev space*, for $T \subseteq \mathbb{R}$ an interval,*

$$H^1_{loc}(T;\mathcal{V}) := \{u \in L^2_{loc}(T;\mathcal{V}) | \forall a < b \in T, u|_{(a,b)} \in H^1((a,b);\mathcal{V})\}.$$

Sobolev spaces have a **lot** of theory attached to them, which I will do my best to skirt around. See [Bre83] for details.

4.4 Double-obstacle Allen–Cahn flow

Definition 4.4.1. *Let $u \in \mathcal{V}$. We define the set.*

$$\mathcal{B}(u) := \left\{\alpha \in \mathcal{V} \,\middle|\, \forall i \in V, \alpha_i \in -\partial I_{[0,1]}(u_i)\right\} \tag{4.8}$$

which is non-empty if and only if $u \in \mathcal{V}_{[0,1]}$. Then $\beta \in \mathcal{B}(u)$ if and only if for all $i \in V$

$$\beta_i \in \begin{cases} [0,\infty), & u_i = 0, \\ \{0\}, & 0 < u_i < 1, \\ (-\infty, 0], & u_i = 1. \end{cases}$$

Lemma 4.4.1. Let $u \in \mathcal{V}_{[0,1]}$ and $\beta \in \mathcal{B}(u)$. Then for all $\eta \in \mathcal{V}_{[0,1]}$, $\langle \beta, \eta - u\rangle_{\mathcal{V}} \geq 0$.

Proof. Consider $\beta_i(\eta_i - u_i)$. If $u_i \in (0,1)$ then $\beta_i = 0$, so this term equals 0. If $u_i = 0$ then $\beta_i \geq 0$ and $\eta_i - u_i = \eta_i \geq 0$, so the term is non-negative. If $u_i = 1$ then $\beta_i \leq 0$ and $\eta_i - u_i = \eta_i - 1 \leq 0$, so the term is non-negative. Hence for all $i \in V$, $\beta_i(\eta_i - u_i) \geq 0$. □

Definition 4.4.2 (Double-obstacle AC flow with fidelity forcing)**.** *Let T be an interval. Then a pair $(u,\beta) \in C^0(T;\mathcal{V}_{[0,1]}) \times \mathcal{V}_{t\in T}$ is a solution to double-obstacle AC flow with fidelity forcing on T if $u \in H^1_{loc}(T;\mathcal{V})$ and for almost every (a.e.) $t \in T$:*

$$\varepsilon\frac{du}{dt}(t) + \varepsilon\mathcal{L}u(t) + \varepsilon M(u(t) - f) + \frac{1}{2}\mathbf{1} - u(t) = \beta(t), \qquad \beta(t) \in \mathcal{B}(u(t)). \tag{4.9}$$

Note. Whenever I say "a.e.", feel free to ignore it if you are unfamiliar with measure theory and replace it with "every". It does not play a big role in understanding the results.

Theorem 4.4.2. If (u, β) obeys Definition 4.4.2, then for all $i \in V$ and a.e. $t \in T$,

$$\beta_i(t) = \begin{cases} \frac{1}{2} + \varepsilon(\mathcal{L}^{(r,s)}u(t))_i - \varepsilon\mu_i f_i, & \text{if } u_i(t) = 0, \\ 0, & \text{if } u_i(t) \in (0,1), \\ -\frac{1}{2} + \varepsilon(\mathcal{L}^{(r,s)}u(t))_i + \varepsilon\mu_i(1 - f_i), & \text{if } u_i(t) = 1. \end{cases} \tag{4.10}$$

Let

$$Q := \min_{i \in V} d_i^{-r} \sum_{j \in V} \omega_{ij}(d_i^{-s} - d_j^{-s}) \leq 0.$$

Then for a.e. $t \in T$,

$$\beta(t) \in \mathcal{V}_{[-1/2+\varepsilon Q, 1/2]}.$$

4.4.1 An integral form for double-obstacle Allen–Cahn

Theorem 4.4.3 (Explicit integral forms)**.** Let $(u, \beta) \in \mathcal{V}_{[0,1], t\in T} \times \mathcal{V}_{t\in T}$, and recall that F_t is $(1 - e^{-tx})/x$ extended to matrix input via its Taylor series.

Then (u, β) satisfies Definition 4.4.2 if and only if the following hold:

- β is locally integrable,
- for a.e. $t \in T$, $\beta(t) \in \mathcal{B}(u(t))$ and $\beta(t) \in \mathcal{V}_{[-1/2+\varepsilon Q, 1/2]}$, and
- for all $t \in T$ (for $B := \mathcal{L} + M - \varepsilon^{-1}I$):

$$u(t) = e^{-tB}u(0) + F_t(B)\left(Mf - \frac{1}{2\varepsilon}\mathbf{1}\right) + \frac{1}{\varepsilon}\int_0^t e^{-(t-s)B}\beta(s)\, ds. \tag{4.11}$$

Where Q is as in the previous theorem.

Proof. Let (u, β) obey Definition 4.4.2. Note that we can rewrite both (4.9) in the form:

$$\varepsilon\frac{du}{dt}(t) + \varepsilon Bu(t) - v = \beta(t) \tag{4.12}$$

where $v = \varepsilon Mf - \frac{1}{2}\mathbf{1}$. Then β is a sum of a continuous function and the derivative of a C^1 function and hence is locally integrable. The pointwise bounds on β follow from Theorem 4.4.2. Finally (4.12) can be further rewritten:

$$\varepsilon\frac{d}{dt}\left(e^{tB}u(t)\right) = e^{tB}(\beta(t) + v).$$

Hence, by the fundamental theorem of calculus on H^1 [Bre83, Theorem 8.2], if $u \in C^0(T; \mathcal{V}_{[0,1]}) \cap H^1(T; \mathcal{V})$ solves (4.9), then for all $t \in T$

$$e^{tB}u(t) - u(0) = \frac{1}{\varepsilon}\left(\int_0^t e^{sB}v\, ds + \int_0^t e^{sB}\beta(s)\, ds\right) = \frac{1}{\varepsilon}e^{tB}F_t(B)v + \frac{1}{\varepsilon}\int_0^t e^{sB}\beta(s)\, ds$$

(where we have used that $\int_0^t e^{sB}\, ds = e^{tB}F_t(B)$, which is simple to verify) and so

$$u(t) = e^{-tB}u(0) + \frac{1}{\varepsilon}F_t(B)v + \frac{1}{\varepsilon}e^{-tB}\int_0^t e^{sB}\beta(s)\, ds. \tag{4.13}$$

Thus if u solves (4.9) then u solves (4.11).

Now, let $(u,\beta) \in \mathcal{V}_{[0,1],t\in T} \times \mathcal{V}_{t\in T}$ satisfy, at a.e. $t \in T$, $\beta(t) \in \mathcal{B}(u(t))$, and $\beta(t) \in \mathcal{V}_{[-1/2+\varepsilon Q,1/2]}$ is locally integrable, and let at all $t \in T$ $u(t)$ be given by (4.13). Then, differentiating (4.13), at $t \in T$ u has formal weak derivative

$$\frac{du}{dt}(t) = -Be^{-tB}u(0) + \frac{1}{\varepsilon}e^{-tB}v + \frac{1}{\varepsilon}\beta(t) - \frac{1}{\varepsilon}B\int_0^t e^{-(t-s)B}\beta(s)\,ds$$

which can be checked to satisfy (4.12). Furthermore $u \in C^0(T;\mathcal{V}_{[0,1]}) \cap H^1(T;\mathcal{V})$, but we omit the details. □

4.4.2 Uniqueness

Theorem 4.4.4. Let $T = [0,T_0]$ or $[0,\infty)$ and let (u,β) and (v,γ) satisfy $u,v \in H^1_{loc}(T;\mathcal{V}) \cap C^0(T;\mathcal{V}_{[0,1]})$ and $u(0) = v(0)$. If (u,β) and (v,γ) solve (4.9), then $u(t) = v(t)$ for all $t \in T$ and $\beta(t) = \gamma(t)$ at a.e. $t \in T$.

4.5 The SDIE scheme for double-obstacle Allen–Cahn, and the link to the MBO scheme

4.5.1 Defining the double-obstacle SDIE scheme

As with the Allen–Cahn flow, we have to introduce subdifferential terms.

Definition 4.5.1 (Double-obstacle SDIE scheme with fidelity forcing)**.** *For $u_0 \in \mathcal{V}_{[0,1]}$, $n \in \mathbb{N}$ we define the SDIE scheme for the double-obstacle Allen–Cahn flow as the iterative scheme:*

$$\left(1 - \frac{\tau}{\varepsilon}\right)u_{n+1} - \mathcal{S}_\tau u_n + \frac{\tau}{2\varepsilon}\mathbf{1} = \frac{\tau}{\varepsilon}\beta_{n+1} \tag{4.14}$$

for $\beta_{n+1} \in \mathcal{B}(u_{n+1})$. For the rest of this chapter we will define $\lambda := \tau/\varepsilon$

4.5.2 The variational form and link to the MBO scheme

Theorem 4.5.1. Let $\lambda \in [0,1]$. If (u_{n+1},β_{n+1}) solves (4.14) with $\beta_{n+1} \in \mathcal{B}(u_{n+1})$, then, for all $r' \in \mathbb{R}$, u_{n+1} solves:

$$u_{n+1} \in \operatorname*{arg\,min}_{u\in\mathcal{V}_{[0,1]}} \lambda\langle u, \mathbf{1}-u\rangle_{\mathcal{V},r'} + \|u - S_\tau u_n\|^2_{\mathcal{V},r'}. \tag{4.15}$$

For $\lambda \in [0,1)$, (4.15) has unique solution

$$(u_{n+1})_i = \begin{cases} 0, & \text{if } (\mathcal{S}_\tau u_n)_i < \frac{1}{2}\lambda, \\ \frac{1}{2} + \frac{(\mathcal{S}_\tau u_n)_i - 1/2}{1-\lambda}, & \text{if } \frac{1}{2}\lambda \le (\mathcal{S}_\tau u_n)_i < 1 - \frac{1}{2}\lambda, \\ 1, & \text{if } (\mathcal{S}_\tau u_n)_i \ge 1 - \frac{1}{2}\lambda, \end{cases} \tag{4.16a}$$

with corresponding $\beta_{n+1} = \lambda^{-1}\left((1-\lambda)u_{n+1} - \mathcal{S}_\tau u_n + \frac{\lambda}{2}\mathbf{1}\right)$. For $\lambda = 1$, (4.15) has solutions

$$(u_{n+1})_i \in \begin{cases} \{1\}, & (\mathcal{S}_\tau u_n)_i > 1/2, \\ [0,1], & (\mathcal{S}_\tau u_n)_i = 1/2, \\ \{0\}, & (\mathcal{S}_\tau u_n)_i < 1/2, \end{cases} \tag{4.16b}$$

with corresponding $\beta_{n+1} = \frac{1}{2}\mathbf{1} - \mathcal{S}_\tau u_n$.

Hence if u_{n+1} solves (4.15) then there exists $\beta_{n+1} \in \mathcal{B}(u_{n+1})$ such that (u_{n+1}, β_{n+1}) solves (4.14). Note that this is the MBO threasholding.

Proof. Let (u_{n+1}, β_{n+1}) solve (4.14), so $\beta_{n+1} \in \mathcal{B}(u_{n+1})$. Let $z := \mathcal{S}_\tau u_n$. We seek to show that for $0 \le \lambda \le 1$

$$\lambda\langle u_{n+1}, \mathbf{1} - u_{n+1}\rangle_{\mathcal{V}} + \langle u_{n+1} - z, u_{n+1} - z\rangle_{\mathcal{V}} \le \lambda\langle \eta, \mathbf{1} - \eta\rangle_{\mathcal{V}} + \langle \eta - z, \eta - z\rangle_{\mathcal{V}}$$

for all $\eta \in \mathcal{V}_{[0,1]}$. By rearranging and cancelling this is equivalent to

$$\begin{aligned} 0 &\le \langle \eta - u_{n+1}, \lambda\mathbf{1} - 2z\rangle_{\mathcal{V}} + (1-\lambda)\left(\langle \eta, \eta\rangle_{\mathcal{V}} - \langle u_{n+1}, u_{n+1}\rangle_{\mathcal{V}}\right) \\ &= \langle \eta - u_{n+1}, \lambda\mathbf{1} - 2z + (1-\lambda)(\eta + u_{n+1})\rangle_{\mathcal{V}} \\ &= \langle \eta - u_{n+1}, 2\lambda\beta_{n+1} + (1-\lambda)(\eta - u_{n+1})\rangle_{\mathcal{V}} \\ &= 2\lambda\,\langle \eta - u_{n+1}, \beta_{n+1}\rangle_{\mathcal{V}} + (1-\lambda)\|\eta - u_{n+1}\|^2_{\mathcal{V}} \end{aligned}$$

where the second equality follows directly from (4.14). Finally, we have by Lemma 4.4.1 that this inequality holds for all $\eta \in \mathcal{V}_{[0,1]}$.

Next, let u solve (4.15). The functional in (4.15) can be rewritten as

$$\lambda\,\langle u, \mathbf{1} - u\rangle_{\mathcal{V},r'} + \|u - \mathcal{S}_\tau u_n\|^2_{\mathcal{V},r'} = \sum_{i\in V} d_i^{r'}\, g_{i,n}(u_i)$$

where

$$g_{i,n}(x) := \lambda x(1-x) + (x - (\mathcal{S}_\tau u_n)_i)^2$$

so we can reduce (4.15) to the system of 1-dimensional problems

$$(u_{n+1})_i \in \operatorname*{arg\,min}_{x\in[0,1]} g_{i,n}(x).$$

Differentiating, we get that for $0 < \lambda < 1$, $g_{i,n}$ is minimised at

$$x = \frac{(\mathcal{S}_\tau u_n)_i - \lambda/2}{1-\lambda} = \frac{1}{2} + \frac{(\mathcal{S}_\tau u_n)_i - 1/2}{1-\lambda}.$$

Therefore for $0 \le \lambda < 1$ the solution u is given by

$$u_i = \begin{cases} 0, & \text{if } (\mathcal{S}_\tau u_n)_i < \frac{1}{2}\lambda \\ \frac{1}{2} + \frac{(\mathcal{S}_\tau u_n)_i - 1/2}{1-\lambda}, & \text{if } \frac{1}{2}\lambda \le \left(e^{-\tau\Delta}u_n\right)_i < 1 - \frac{1}{2}\lambda \\ 1, & \text{if } (\mathcal{S}_\tau u_n)_i \ge 1 - \frac{1}{2}\lambda \end{cases}$$

and hence

$$\begin{aligned} &\lambda^{-1}\left((1-\lambda)u_i - (\mathcal{S}_\tau u_n)_i + \frac{\lambda}{2}\right) \\ &= \begin{cases} \frac{1}{2} - \lambda^{-1}(\mathcal{S}_\tau u_n)_i, & \text{if } (\mathcal{S}_\tau u_n)_i < \frac{1}{2}\lambda, \\ 0, & \text{if } \frac{1}{2}\lambda \le (\mathcal{S}_\tau u_n)_i < 1 - \frac{1}{2}\lambda, \\ -\frac{1}{2} + \lambda^{-1}(1 - (\mathcal{S}_\tau u_n)_i), & \text{if } (\mathcal{S}_\tau u_n)_i \ge 1 - \frac{1}{2}\lambda. \end{cases} \\ &= \begin{cases} \frac{1}{2} - \lambda^{-1}(\mathcal{S}_\tau u_n)_i, & \text{if } u_i = 0, \\ 0, & \text{if } u_i \in (0,1), \\ -\frac{1}{2} + \lambda^{-1}(1 - (\mathcal{S}_\tau u_n)_i), & \text{if } u_i = 1. \end{cases} \end{aligned}$$

Thus, noting that the top case has a non-negative value and the bottom case always has a non-positive value, we observe that $\beta := \lambda^{-1}\left((1-\lambda)u - \mathcal{S}_\tau u_n + \frac{\lambda}{2}\mathbf{1}\right) \in \mathcal{B}(u)$, so (u,β) solves (4.14).

If $\lambda = 1$ then examine the functional in (4.15) for $\lambda = 1$:

$$\begin{aligned}
&\langle u, \mathbf{1}-u\rangle_{\mathcal{V}} + \|u - \mathcal{S}_\tau u_n\|_{\mathcal{V}}^2 \\
&= \langle u, \mathbf{1}-u\rangle_{\mathcal{V}} + \langle u - \mathcal{S}_\tau u_n, u - \mathcal{S}_\tau u_n\rangle_{\mathcal{V}} \\
&= \langle u, \mathbf{1}\rangle_{\mathcal{V}} - \langle u, u\rangle_{\mathcal{V}} + \langle u, u\rangle_{\mathcal{V}} - 2\langle u, \mathcal{S}_\tau u_n\rangle_{\mathcal{V}} + \langle \mathcal{S}_\tau u_n, \mathcal{S}_\tau u_n\rangle_{\mathcal{V}} \\
&\simeq \langle u, \mathbf{1} - 2\mathcal{S}_\tau u_n\rangle_{\mathcal{V}},
\end{aligned}$$

and therefore u as a minimiser must obey

$$u_i \in \begin{cases} \{1\}, & (\mathcal{S}_\tau u_n)_i > 1/2, \\ [0,1], & (\mathcal{S}_\tau u_n)_i = 1/2, \\ \{0\}, & (\mathcal{S}_\tau u_n)_i < 1/2. \end{cases}$$

Hence $\beta \in \mathcal{B}(u)$ if and only if for each $i \in V$

$$\beta_i \in \begin{cases} [0,\infty), & (\mathcal{S}_\tau u_n)_i \le 1/2 \\ \{0\}, & (\mathcal{S}_\tau u_n)_i = 1/2, u_i \in (0,1) \\ (-\infty, 0], & (\mathcal{S}_\tau u_n)_i \ge 1/2 \end{cases}$$

and thus $\frac{1}{2}\mathbf{1} - \mathcal{S}_\tau u_n \in \mathcal{B}(u)$, so (u,β) solves (4.14). □

We note a useful consequence of this result.

Theorem 4.5.2. For $\lambda \in [0,1)$[a] and all $n \in \mathbb{N}$, if u_n and v_n are SDIE sequences defined according to Definition 4.5.1 with initial states $u_0, v_0 \in \mathcal{V}_{[0,1]}$ and ξ_1 is the smallest eigenvalue of $\mathcal{L} + M$, then

$$\|u_n - v_n\|_{\mathcal{V}} \le e^{-n\xi_1\tau}(1-\lambda)^{-n}\|u_0 - v_0\|_{\mathcal{V}}. \tag{4.17}$$

[a] For the MBO case $\lambda = 1$ the thresholding is discontinuous so we do not get an analogous property.

Exercise 9. Prove this.

4.5.3 Freezing

If τ is taken too small, the SDIE scheme "freezes". Fix $S \subseteq V$ and $\alpha \ge 0$. Then there exists τ^* (depending on S, α, $\mathcal{L}$, M, and f) such that $|(\mathcal{S}_\tau \chi_S)_i - (\chi_S)_i| \le \alpha$ for all $i \in V$ and $\tau \le \tau^*$. Then if $\alpha = \frac{1}{2}$ it follows that the only valid MBO update of $u_n = \chi_S$ is $u_{n+1} = \chi_S$ for $\tau < \tau^*$. And if $\alpha = \frac{\tau}{2\varepsilon} < \frac{1}{2}$ it follows that the only valid SDIE update of $u_n = \chi_S$ is $u_{n+1} = \chi_S$ for $\tau \le \tau^*$. Note that in this latter case τ^* has a lower value.

4.6 The $\tau \downarrow 0$ limit of the double-obstacle SDIE scheme

To simplify notation we write (4.14) in the form

$$(1-\lambda)u_{n+1} - e^{-\tau A}u_n - w = \lambda\beta_{n+1} \tag{4.18}$$

where $\lambda := \tau/\varepsilon$, $A := \mathcal{L} + M$, and $w := -\frac{1}{2}\lambda\mathbf{1} + F_\tau(A)Mf$.

We now solve the SDIE recurrence relation for the n^{th} term.

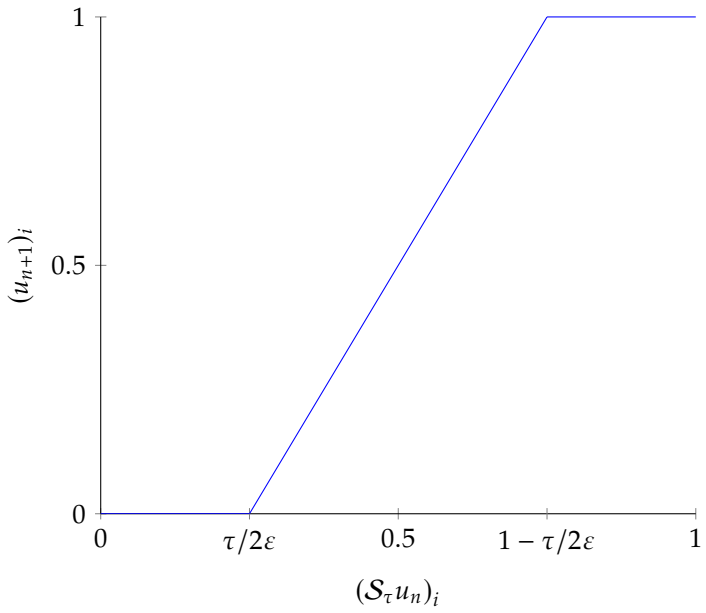

Figure 4.1: Plot of the piecewise linear thresholding of $\mathcal{S}_\tau u_n$ described by (4.16a).

Proposition 4.6.1. For $\lambda \in [0,1)$ the sequence generated by (4.18) is given by:

$$\begin{aligned} u_n =& (1-\lambda)^{-n} e^{-n\tau A} u_0 + \sum_{k=1}^{n} (1-\lambda)^{-k} e^{-(k-1)\tau A} w \\ &+ \frac{\lambda}{1-\lambda} \sum_{k=1}^{n} (1-\lambda)^{-(n-k)} e^{-(n-k)\tau A} \beta_k \end{aligned} \tag{4.19}$$

Proof. An unexciting proof by induction. □

Theorem 4.6.2. Let $t \geq 0$, $\varepsilon > 0$, $B := A - \varepsilon^{-1} I$, and $v := \varepsilon M f - \frac{1}{2}\mathbf{1}$. Then with respect to the limit of $\tau \downarrow 0$ and $n \to \infty$ with $n\tau - t \in [0, \tau)$:

1. $(1-\lambda)^{-n} e^{-n\tau A} u_0 = e^{-tB} u_0 + O(\tau)$.
2. $\sum_{k=1}^{n} (1-\lambda)^{-k} e^{-(k-1)\tau A} w = \frac{1}{\varepsilon} F_t(B) v + O(\tau)$.
3. $\frac{\lambda}{1-\lambda} \sum_{k=1}^{n} (1-\lambda)^{-(n-k)} e^{-(n-k)\tau A} \beta_k = \lambda \sum_{k=1}^{n} e^{-(n-k)\tau B} \beta_k + O(\tau)$.

Hence by (4.19), the SDIE term obeys

$$u_n = e^{-tB} u_0 + \frac{1}{\varepsilon} F_t(B) v + \lambda \sum_{k=1}^{n} e^{-(n-k)\tau B} \beta_k + O(\tau). \tag{4.20}$$

Note. The key idea will be to rewrite the sum in (4.20) as an integral, and connect to the integral form of double obstacle Allen–Cahn.

Definition 4.6.1. *Define the piecewise constant function* $z_\tau : [0,\infty) \to \mathcal{V}$

$$z_\tau(s) := \begin{cases} e^{\tau B}\beta_1^{[\tau]}, & 0 \le s \le \tau \\ e^{k\tau B}\beta_k^{[\tau]}, & (k-1)\tau < s \le k\tau \text{ for } k \in \mathbb{N} \end{cases}$$

and the function

$$\gamma_\tau(s) := e^{-sB} z_\tau(s) = \begin{cases} e^{(\tau-s)B}\beta_1^{[\tau]}, & 0 \le s \le \tau \\ e^{(k\tau-s)B}\beta_k^{[\tau]}, & (k-1)\tau < s \le k\tau \text{ for } k \in \mathbb{N} \end{cases}$$

using the bookkeeping notation of the superscript $[\tau]$ *to keep track of the time-step governing* u_n *and* β_n.

Theorem 4.6.3. For any sequence $\tau_n^{(0)} \downarrow 0$ with $\tau_n^{(0)} < \varepsilon$ for all n, there exists a function $z : [0,\infty) \to \mathcal{V}$ and a subsequence τ_n such that

(A) For all $t \ge 0$,

$$\int_0^t z_{\tau_n}(s)\, ds \to \int_0^t z(s)\, ds.$$

(B) The Cesàro sums converge pointwise: there exists $N_k \to \infty$ such that for almost every $t \ge 0$

$$\frac{1}{N_k}\sum_{n=1}^{N_k} z_{\tau_n}(t) \to z(t) \qquad \text{and} \qquad \frac{1}{N_k}\sum_{n=1}^{N_k} \gamma_{\tau_n}(t) \to \gamma(t)$$

as $k \to \infty$, where $\gamma(t) := e^{-tB} z(t)$.

Proof. Omitted. □

We thus infer convergence of the SDIE iterates.

Theorem 4.6.4. Let $\tau_n^{(0)} \downarrow 0$ with $\tau_n < \varepsilon$. Let τ_n be the subsequence from the previous theorem. Define for all $t \ge 0$:

$$\hat{u}(t) := \lim_{n\to\infty, m=\lceil t/\tau_n \rceil} u_m^{[\tau_n]}. \tag{4.21}$$

Then

$$\hat{u}(t) = e^{-tB}u_0 + \frac{1}{\varepsilon}F_t(B)v + \frac{1}{\varepsilon}\int_0^t e^{-(t-s)B}\gamma(s)\, ds. \tag{4.22}$$

Note the similarity between (4.22) and the explicit form for Allen–Cahn solutions (4.11).

Proof. By the above discussion, we can rewrite $\hat{u}(t)$ as:

$$\begin{aligned} \hat{u}(t) &= e^{-tB}u_0 + \frac{1}{\varepsilon}F_t(B)v + \lim_{n\to\infty}\frac{\tau_n}{\varepsilon}\sum_{k=1}^{m} e^{-(m-k)\tau_n B}\beta_k^{[\tau_n]} \\ &= e^{-tB}u_0 + \frac{1}{\varepsilon}F_t(B)v + \frac{1}{\varepsilon}\lim_{n\to\infty} e^{-m\tau_n B}\int_0^{m\tau_n} z_{\tau_n}(s)\, ds. \end{aligned}$$

Next, note that $m\tau_n = \tau_n\lceil t/\tau_n\rceil =: t + \eta_n$ where $\eta_n \in [0, \tau_n)$. Therefore

$$\lim_{n\to\infty} e^{-m\tau_n B}\int_0^{m\tau_n} z_{\tau_n}(s)\,ds = \lim_{n\to\infty} e^{-\eta_n B}e^{-tB}\int_0^t z_{\tau_n}(s)\,ds + e^{-\eta_n B}e^{-tB}\int_t^{t+\eta_n} z_{\tau_n}(s)\,ds$$
$$= \lim_{n\to\infty} e^{-tB}\int_0^t z_{\tau_n}(s)\,ds + e^{-tB}\int_t^{t+\eta_n} z_{\tau_n}(s)\,ds \quad \text{as } e^{-\eta_n B} = I + O(\tau_n)$$
$$= \lim_{n\to\infty} e^{-tB}\int_0^t z_{\tau_n}(s)\,ds \quad \text{as } z_{\tau_n}(s) \text{ is bounded on } [t, t + \max_{n'}\eta_{n'}] \text{ uniformly in } n$$
$$= e^{-tB}\int_0^t z(s)\,ds \quad \text{by Theorem 4.6.3(A).}$$

Finally, $e^{-tB}z(s) = e^{-(t-s)B}\gamma(s)$. □

Theorem 4.6.5. For any given $u_0 \in \mathcal{V}_{[0,1]}$, $\varepsilon > 0$, and $\tau_n^{(0)} \downarrow 0$, there exists a subsequence τ_n of $\tau_n^{(0)}$ with $\tau_n < \varepsilon$ for all n, along which the SDIE iterates $(u_m^{[\tau_n]}, \beta_m^{[\tau_n]})$ given by (4.14) with initial state u_0 converge to the double-obstacle Allen–Cahn solution with initial condition u_0 in the following sense:

- for each $t \geq 0$, as $n \to \infty$ and $m = \lceil t/\tau_n\rceil$, $u_m^{[\tau_n]} \to \hat{u}(t)$, and
- there is a sequence $N_k \to \infty$ such that for almost every $t \geq 0$, $\frac{1}{N_k}\sum_{n=1}^{N_k}\beta_m^{[\tau_n]} \to \gamma(t)$

where $(\hat{u}, \gamma)$ is the solution to (4.9) with $\hat{u}(0) = u_0$.

Proof. The only thing left to check is that $(\hat{u}, \gamma)$ is an Allen–Cahn solution, which we can do by checking that all the conditions in Theorem 4.4.3 are satisfied. We omit the details. □

Corollary 4.6.6. Let $u_0 \in \mathcal{V}_{[0,1]}$, $\varepsilon > 0$, and $\tau_n \downarrow 0$ with $\tau_n < \varepsilon$ for all n. Then for each $t \geq 0$, as $n \to \infty$, $u_{\lceil t/\tau_n\rceil}^{[\tau_n]} \to \hat{u}(t)$.

Proof. Exercise, uses uniqueness of Allen–Cahn trajectories. □

4.6.1 The well-posedness of double-obstacle Allen–Cahn

The above theorem also proves that solutions to double-obstacle Allen–Cahn always exist for any initial condition, which we hadn't proved until then.

We can also prove well-posedness.

Theorem 4.6.7. Let $u_0, v_0 \in \mathcal{V}_{[0,1]}$ define Allen–Cahn trajectories u, v by Definition 4.4.2. Then, if $\xi_1 := \min\sigma(A)$, then

$$\|u(t) - v(t)\|_{\mathcal{V}} \leq e^{-\xi_1 t}e^{t/\varepsilon}\|u_0 - v_0\|_{\mathcal{V}}. \tag{4.23}$$

Proof. Fix $t \geq 0$ and let $m := \lceil t/\tau_n\rceil$. By Corollary 4.6.6, we can take $\tau_n \downarrow 0$ such that the SDIE sequences $u_m^{[\tau_n]} \to u(t)$ and $v_m^{[\tau_n]} \to v(t)$ as $n \to \infty$. Then by (4.17):

$$\|u_m^{[\tau_n]} - v_m^{[\tau_n]}\|_{\mathcal{V}} \leq e^{-m\xi_1\tau_n}(1 - \tau_n/\varepsilon)^{-m}\|u_0 - v_0\|_{\mathcal{V}}$$

and taking $n \to \infty$ gives (4.23). □

4.6.2 Double-obstacle Allen–Cahn as a gradient flow

Theorem 4.6.8. The double-obstacle Allen–Cahn trajectory u defined by Definition 4.4.2 has $\mathrm{GL}^{(r,s)}_{\varepsilon,\mu,f}(u(t))$ monotonically decreasing in t. More precisely: for all $t > s \geq 0$,

$$\mathrm{GL}^{(r,s)}_{\varepsilon,\mu,f}(u(s)) - \mathrm{GL}^{(r,s)}_{\varepsilon,\mu,f}(u(t)) \geq \frac{1}{2(t-s)} \|u(s) - u(t)\|^2_{\mathcal{V}} .$$

Furthermore, this entails an explicit $C^{0,1/2}$ condition for u

$$\|u(s) - u(t)\|_{\mathcal{V}} \leq \sqrt{|t-s|}\sqrt{2\,\mathrm{GL}^{(r,s)}_{\varepsilon,\mu,f}(u(0))}.$$

Proof (sketch). Represent u as the limit of SDIE sequences, and use the Lyapunov energy for the SDIE scheme you will derive in Assignment 2. □

Chapter 5

Some crucial numerical linear algebra

5.1 Why we need numerical linear algebra

In this chapter we will be thinking about how one might implement the methods from the previous chapter, in particular the graph MBO scheme. This requires thinking about computing

$$\mathcal{S}_\tau u := e^{-\tau(\mathcal{L}+M)}u + F_t(\mathcal{L}+M)Mf.$$

But there is a big issue: $\mathcal{L}$ is an $N \times N$ matrix, where $N = |V|$. In the end, we will be taking V to be the set of pixels in an image, so N can easily be in the hundreds of thousands, millions, or even higher. For example, the image in Figure 5.1 has over 15 million pixels.

If V had 10^6 elements, and each entry in $\mathcal{L}$ took 1 byte of memory to store, then storing $\mathcal{L}$ alone would require 10^{12} bytes, i.e. 1 TB of memory! Never mind computing the matrix exponential of $\mathcal{L}$. Therefore, we will need to be cleverer if we ever want these methods to run in a human lifetime on ordinary computers, which is where the methods of this chapter come in.

5.2 A review of "big O notation"

A key tool for talking about the computational challenges we will need to overcome in this chapter, and the efficiency of the methods we will be describing, is the so-called "big O notation".

Definition 5.2.1. *Let $X \subseteq \mathbb{R}$ be unbounded (for our purposes X will be either $\mathbb{R}$ or $\mathbb{N}$), $f : X \to \mathbb{R}$, and $g : X \to (0, \infty)$. Then we will write*

$$f(x) = O(g(x))$$

with respect to the limit $x \to \infty$ if

$$\limsup_{x\to\infty} \frac{|f(x)|}{g(x)} := \lim_{n\to\infty} \sup_{x\in X\cap(n,\infty)} \frac{|f(x)|}{g(x)} < \infty.$$

That is, there exists $M < \infty$ such that $|f(x)| \leq Mg(x)$ for all sufficiently large x.

Figure 5.1: 3487 × 4356 image obtained from https://images.unsplash.com/photo-1679882028877-8ff92cf0abd4 (Photographer: Aaron Burden).

Note. The use of this notation in practical settings is always a little metaphorical. Technically, all O notation cares about is the behaviour in the limit, whilst in practice there are only a finite number of inputs we will ever enter into any function before the heat death of the universe. As such, technically these settings never overlap. The function

$$f(x) = \begin{cases} \sqrt{x}, & x < 10^{10^{100}}, \\ e^{e^x}, & x \geq 10^{10^{100}}, \end{cases}$$

is technically $O(e^{e^x})$ but in practice is $\sqrt{x}$, whilst the function

$$f(x) = \begin{cases} e^{e^x}, & x < 10^{10^{100}}, \\ e^{e^{10^{10^{100}}}}, & x \geq 10^{10^{100}}, \end{cases}$$

is technically $O(1)$ but in practice is e^{e^x}. Moreover, the O notation can hide massive multiplicative constants. So, the notation gives a guide for how these things grow, but it must be handled with caution.

Proposition 5.2.1 (Some key linear algebra big Os)**.**

1. A matrix $A \in \mathbb{R}^{M\times N}$ requires $O(MN)$ memory to store (is $O(MN)$ "in space").
2. If $A \in \mathbb{R}^{M\times N}$ and $B \in \mathbb{R}^{N\times K}$, then AB requires $O(MNK)$ operations to compute (is $O(MNK)$ "in time") via the naïve formula.[a] In particular, for $A \in \mathbb{R}^{M\times N}$ and $v \in \mathbb{R}^N$ a matrix-vector computation Av is $O(MN)$ in time.
3. If $A \in \mathbb{R}^{N\times N}$ then computing A^{-1} and solving the linear system $Ax = b$ are both $O(N^3)$ in time.

[a]For $M = N = K$ this can be done in $O(N^{2.8074})$ time by Strassen's algorithm [S+69], and there was a major breakthrough recently in this area by DeepMind's AlphaTensor [FBH+22].

The big advantage to using this notation is that we don't have to worry about implementation details like how much memory a matrix entry takes up or exactly how fast our computer adds or multiplies numbers. These all get abstracted away.

In the case of $\mathcal{L}$, we therefore naively require $O(N^2)$ space and $O(N^2)$ or $O(N^3)$ time to perform our desired computations. We will be supposing throughout this chapter that N is much too large for anything meaningfully above $O(N)$ in either space or time to be practical.

5.3 Computing the matrix exponential

The key quantity which we wish to compute in order to compute graph diffusion (and thereby compute the MBO or SDIE schemes) is the *matrix exponential*:

$$e^{tA} := \sum_{n=0}^{\infty} \frac{t^n}{n!} A^n \tag{5.1}$$

or rather, the matrix-vector product $e^A v$. For such a simple to express concept, the matrix exponential turns out to be remarkably tricky to compute, a state of affairs perhaps no better communicated than by the title of Moler and Van Loan's classic paper "Nineteen Dubious Ways to Compute the Exponential of a Matrix" [MVL78].

The discussion in this chapter will draw primarily from Moler and Van Loan's "Nineteen Dubious Ways to Compute the Exponential of a Matrix, Twenty-Five Years Later" [MVL03] and Higham's *Functions of Matrices: Theory and Computation* [Hig08].

5.3.1 Why not use the Taylor series?

The definition of e^{tA} is a Taylor series, why not just use that? That is, why not approximate

$$e^A \approx \sum_{n=0}^{M} \frac{1}{n!} A^n =: T_M(A)?$$

The matrix-vector products $e^A v$ could then be computed via *Horner's method*:

$$v_{M-1} := \frac{1}{(M-1)!} v + \frac{1}{M!} Av$$
$$v_k := \frac{1}{k!} v + A v_{k+1}$$

gives $v_0 = T_M(A)v \approx e^A v$.

The first issue with this is that we would still need to come up with an efficient way to compute matrix-vector products with A, but that is at least easier than with e^A.

Another, more serious, issue with this is that the Taylor series can be slow to converge, even in the scalar case. For example, to compute e^{10} to within 1% error requires 18 terms. A worse situation occurs when one seeks to compute e^{-10} via the Taylor series. This requires 35 terms to achieve 1% error, but more importantly computing negative exponentials is numerically unstable because it involves *catastrophic cancellation*.

Catastrophic cancellation is the phenomenon where a small number is computed as the difference of two much larger numbers. In the case of e^{-10}, the largest such difference is

$$-\frac{10^9}{9!} + \frac{10^{10}}{10!}$$

where $10^9/9! = 10^{10}/10! \approx 2755.7$. Compared to e^{-10}, this is massive. This causes problems numerically because any rounding errors in the computers arithmetic will be relative to the magnitude of the number computed. A 0.1% error in the computation of the large numbers could produce a huge relative error in the computation of their difference.

A matrix example from [MVL78]:

$$A = \begin{pmatrix} -49 & 24 \\ -64 & 31 \end{pmatrix}$$

has true matrix exponential

$$e^A = \begin{pmatrix} -0.74 & 0.55 \\ -1.47 & 1.10 \end{pmatrix}.$$

However, when computed using the Taylor series in "short" arithmetic (a rounding error of about 0.001%) one gets instead

$$\begin{pmatrix} -22.26 & -1.43 \\ -61.50 & -3.47 \end{pmatrix}$$

requiring 59 terms to converge; this is a massive error. This occurs because A has eigenvalues -1 and -17, so catastrophic cancellation occurs. In the case at hand, we seek to compute $e^{-t\mathcal{L}}$ or $e^{-t(\mathcal{L}+M)}$ where $\mathcal{L}$ and $\mathcal{L} + M$ are positive semi-definite, so we will face the same challenge.

There are two strategies to mitigate this issue, but neither work well in the setting where A is a very large matrix.

Compute e^{-A} and invert If A has all nonpositive eigenvalues, which are going to cause trouble, then this can be resolved by using that $e^A = (e^{-A})^{-1}$, as $-A$ will be positive semi-definite. This seems promising since in our case $A = -t\mathcal{L}$ or similar. However, computing the inverse is a big obstacle. We cannot store A or e^{-A} in memory, so we can't directly compute the inverse matrix (and we couldn't store it even if we could).

To compute $e^A v$ by this method, we would need to solve the linear system $e^{-A}w = v$ for w, with only the ability to compute matrix-vector products with e^{-A} by the Taylor series, which is expensive. Solving this linear system in such a setting would be very difficult, and would require computing many matrix-vector products.

Scaling-and-squaring *The* standard way to compute matrix exponentials (used for example in MATLAB's `expm`) is the scaling-and-squaring method. This uses the fact that $e^A = (e^{A/2^s})^{2^s}$ so e^A can be computed via squaring $e^{A/2^s}$ s times. By choosing s large enough, one can ensure that the terms in one's approximation[1] to $e^{A/2^s}$ decay rapidly, accelerating the convergence and avoiding catastrophic cancellation. However, since we can't store A in memory or compute matrix-matrix products, we can't compute the repeated squarings. One could instead try computing $e^A v = (e^{A/s})^s v$ by repeated multiplication by $e^{A/s}$, but this is much more expensive and less accurate, especially when we are computing each multiplication by evaluating the Taylor polynomial times v.

[1] The `expm` method doesn't use Taylor approximants, but rather the more accurate Padé approximants, which approximate e^x as a rational function rather than a polynomial. However, this difference is not essential to understanding the method.

5.3.2 Using the eigendecomposition

Theorem 5.3.1. ξ is an eigenvector of A with eigenvalue λ if and only if ξ is an eigenvector of e^{tA} with eigenvalue $e^{\lambda t}$.

Proof. The forward direction is very direct:

$$e^{tA}\xi = \sum_{n=0}^{\infty} \frac{t^n}{n!} A^n \xi = \sum_{n=0}^{\infty} \frac{t^n}{n!} \lambda^n \xi = e^{\lambda t}\xi.$$

The converse is left as an exercise. □

Let ξ_k be the eigenvectors of A, with corresponding eigenvalues λ_k, and express v in terms of them:

$$v = \sum_k a_k \xi_k.$$

Then

$$e^{tA}v = \sum_k a_k e^{tA}\xi_k = \sum_k a_k e^{\lambda_k t}\xi_k.$$

Thus if we can compute ξ_k and λ_k and store those pairs for which $\lambda_k t$ is largest, we can compute $e^{tA}v$ efficiently.

5.3.3 Using rank-reduction compute the matrix exponential

Definition 5.3.1 (Rank of a matrix)**.** *The* rank *of a matrix $A \in \mathbb{R}^{M\times N}$, which we will denote by* rank(A)*, is the number of linearly independent rows (or, equivalently, columns) of A. If $M = N$, it is also the number of non-zero eigenvalues (counting multiplicity) of A.*

Recall from Proposition 2.3.2 that we can write the graph Laplacian $\mathcal{L}$ in the form $U\Lambda V^T$ where $U, \Lambda, V \in \mathbb{R}^{N\times N}$, Λ is the diagonal matrix of eigenvalues, and $V^T U = UV^T = I$.

Let $A \in \mathbb{R}^{N\times N}$ be decomposed in that form $A = U\Lambda V^T$. Then we can approximate A by pulling out just K eigenvalues from Λ and the corresponding left and right eigenvectors.

$$A \approx U_K \Lambda_K V_K^T.$$

where $U_K, V_K \in \mathbb{R}^{N\times K}$, $\Lambda_K \in \mathbb{R}^{K\times K}$.

Proposition 5.3.2. $V_K^T U_K = I_K$, where I_K is the $K \times K$ identity matrix.

Proof. Up to a reordering of the columns, which is the same for both U and V, we can write $U = \begin{pmatrix} U_K & U_{N-K} \end{pmatrix}$ and $V = \begin{pmatrix} V_K & V_{N-K} \end{pmatrix}$. This reordering preserves that $V^T U = I$, so

$$\begin{pmatrix} V_K^T U_K & V_K^T U_{N-K} \\ V_{N-K}^T U_K & V_{N-K}^T U_{N-K} \end{pmatrix} = I = \begin{pmatrix} I_K & 0 \\ 0 & I_{N-K} \end{pmatrix}.$$

□

It follows that

$$\begin{aligned} e^{tA} &= I + tA + \frac{1}{2}t^2A^2 + \frac{1}{6}t^3A^3 + \cdots \\ &\approx I + tU_K\Lambda_K V_K^T + \frac{1}{2}t^2U_K\Lambda_K^2V_K^T + \frac{1}{6}t^3U_K\Lambda_K^3V_K^T + \cdots \\ &= I + U_K\left(t\Lambda_K + \frac{1}{2}t^2\Lambda_K^2 + \frac{1}{6}t^3\Lambda_K^3 + \cdots\right)V_K^T \\ &= I + U_K\left(e^{t\Lambda_K} - I_K\right)V_K^T. \end{aligned} \tag{5.2}$$

Exercise 10. What is the error of this approximation?

This allows us to compute

$$e^{tA}v \approx v + U_K\left(\left(e^{t\Lambda_K} - I_K\right)(V_K^Tv)\right)$$

in just $O(NK)$ operations.

Question. But how to (approximately) compute an accurate rank reduction without computing the whole spectrum?

5.4 The Nyström extension

The Nyström extension, originally developed by Nyström [Nys30] for integral eigenvalue problems, and popularised for numerical linear algebra in [FBCM04, WS00], is a method for rank-reducing a square matrix $A \in \mathbb{R}^{N\times N}$.

5.4.1 The continuous setting

Nyström's original work in 1930 was to do not with matrices, but with integral operators, and that is there that we will begin.

Definition 5.4.1. *The eigenvalue problem $Av = \lambda v$ can be generalised in the continuous setting to: for all $y \in [0,1]$*

$$\lambda f(y) = \int_0^1 A(y,x)f(x)\,dx, \tag{5.3}$$

where $f : [0,1] \to \mathbb{R}$ and $A : [0,1]^2 \to \mathbb{R}$.

Note. The finite dimensional case really is a special case of (5.3). For $A \in \mathbb{R}^{N\times N}$ and $v \in \mathbb{R}^N$, let

$$A(y,x) = \begin{cases} A_{ij} & \text{if } y \in [(i-1)/N, i/N), x \in [(j-1)/N, j/N) \end{cases}$$

and

$$f(x) = \begin{cases} v_i & \text{if } x \in [(i-1)/N, i/N). \end{cases}$$

Then (5.3) becomes, for $y \in [(i-1)/N, i/N)$

$$\begin{aligned}\lambda v_i &= \sum_{j=1}^{N} \int_{(j-1)/N}^{j/N} A(y,x) f(x)\, dx \\ &= \sum_{j=1}^{N} \int_{(j-1)/N}^{j/N} A_{ij} v_j\, dx \\ &= \frac{1}{N}(Av)_i.\end{aligned}$$

5.4.2 Quadrature

The key idea of the Nyström extension is to approximate such eigenfunctions f by employing a *quadrature* of the integral in (5.3).

That is, let $x_k := k/K$ for $k = 1$ to K. Then by the definition of the integral

$$\int_0^1 g(x)\, dx \approx \sum_{k=1}^{K} g(x_k)\frac{1}{K}.$$

Hence for A and f solving (5.3)

$$\lambda f(y) \approx \frac{1}{K}\sum_{k=1}^{K} A(y, x_k) f(x_k).$$

Let $v \in \mathbb{R}^K$ be defined by $v_k := f(x_k)$ and $\tilde{A} \in \mathbb{R}^{K\times K}$ by $\tilde{A}_{ij} := A(x_i, x_j)$. Then by setting $y = x_k$

$$\lambda v_k \approx \frac{1}{K}\sum_{\ell=1}^{K} \tilde{A}_{k\ell} v_\ell = \frac{1}{K}(\tilde{A}v)_k.$$

Hence we can find v by finding the λ eigenvalue of $\frac{1}{K}\tilde{A}$, and then we *extend* this eigenvector to an eigenfunction on the whole of $[0,1]$ via

$$\tilde{f}(y) := \frac{1}{\lambda K}\sum_{k=1}^{K} A(y, x_k) v_k.$$

We have here essentially interpolated f from the values $f(x_k)$, and hence we call $X := \{x_1, \ldots, x_K\}$ the *interpolation set*.

5.4.3 Returning to the matrix setting

We now wish to apply this same approach to approximate the eigendecomposition of $A \in \mathbb{R}^N$, for N large.

Recall the identification of $i \in \{1, \ldots, N\}$ with the interval $[(i-1)/N, i/N)$ in the above note. Then for $K < N$, the x_k correspond to a particular subset of $\{1, \ldots, N\}$ of size K. But the *ordering* of this identification was essentially arbitrary. Hence, X can be taken to be a *random* subset of $\{1, \ldots, N\}$ of size K. Define $Y = \{1, \ldots, N\} \setminus X$.

First, write A in the form

$$\begin{pmatrix} A_{XX} & A_{XY} \\ A_{YX} & A_{YY} \end{pmatrix},$$

where $A_{XX} := (A_{ij})_{i\in X, j\in X}$ etc.

Suppose that A_{XX} can be eigendecomposed as $A_{XX} = U_X \Lambda_X U_X^{-1}$. Now let $u_X^i \in \mathbb{R}^{K\times 1}$ be a column eigenvector of A_{XX} with eigenvalue λ_i. We seek to extend u_X^i to a vector $u^i \in \mathbb{R}^N$ by applying a discrete analogue of a quadrature. That is, let

$$u^i := \begin{pmatrix} u_X^i \\ u_Y^i \end{pmatrix}$$

be defined by the following rule:

$$\lambda_i u_k^i = \sum_{j\in X} A_{kj} u_j^i$$

which can be observed to be same quadrature trick as before. Restricting to $k \in Y$, we obtain

$$\lambda_i u_Y^i = A_{YX} u_X^i.$$

Let $U_Y := \begin{pmatrix} u_Y^1 & \cdots & u_Y^K \end{pmatrix}$. Then

$$U_Y \Lambda_X = A_{YX} U_X$$

and so (assuming that A_{XX}^{-1} exists)

$$U_Y = A_{YX} U_X \Lambda_X^{-1}.$$

Likewise, for a given row eigenvector $v_X^i \in \mathbb{R}^{1\times K}$ of A_{XX} with eigenvalue λ_i, we define $v^i := \begin{pmatrix} v_X^i & v_Y^i \end{pmatrix}$ where $\lambda_i v_Y^i := v_X^i A_{XY}$. It follows that the matrix V_Y with i^{th} row v_Y^i is given by

$$V_Y = \Lambda_X^{-1} U_X^{-1} A_{XY}.$$

Finally, we tie this all together into an approximation for A:

$$\begin{aligned} A &\approx \begin{pmatrix} U_X \\ U_Y \end{pmatrix} \Lambda_X \begin{pmatrix} U_X^{-1} & V_Y \end{pmatrix} \\ &= \begin{pmatrix} U_X \Lambda_X U_X^{-1} & U_X \Lambda_X V_Y \\ U_Y \Lambda_X U_X^{-1} & U_Y \Lambda_X V_Y \end{pmatrix} \\ &= \begin{pmatrix} A_{XX} & A_{XY} \\ A_{YX} & A_{YX} A_{XX}^{-1} A_{XY} \end{pmatrix} \\ &= \begin{pmatrix} A_{XX} \\ A_{YX} \end{pmatrix} A_{XX}^{-1} \begin{pmatrix} A_{XX} & A_{XY} \end{pmatrix} \end{aligned} \tag{5.4}$$

where in the second equality we have made use of $A_{XX} = U_X \Lambda_X U_X^{-1}$ and the above expressions for U_Y and V_Y.

The key consequence of (5.4) is that it reduces the task of storing A (which is $O(N^2)$ in space) to the task of storing A_{XX}, A_{XY}, and A_{YX}, which is $O(NK)$ in space. Furthermore, (5.4) reduces $O(N^2)$ matrix-vector products with A to $O(NK)$ products with the smaller matrices plus an $O(K^3)$ matrix inversion. When $K \ll N$ these are substantial savings.

5.5 Approximating the rank-reduced eigendecomposition using the Nyström extension

We will now consider how to extract from (5.4) an approximate rank-reduced eigendecomposition of A, which can then be used to approximate e^A via (5.2). You might think

that we are done by the above. Can't we just set

$$U = \begin{pmatrix} U_X \\ U_Y \end{pmatrix} \qquad V = \begin{pmatrix} (U_X^{-1})^T \\ V_Y^T \end{pmatrix}$$

and have $A \approx U\Lambda_X V^T$?

Unfortunately, the "extended eigenvectors" we derived above are not orthonormal, since

$$V^T U = \begin{pmatrix} U_X^{-1} & V_Y \end{pmatrix} \begin{pmatrix} U_X \\ U_Y \end{pmatrix} = I_K + V_Y U_Y = I_K + \Lambda_X^{-1} U_X^{-1} A_{XY} A_{YX} U_X \Lambda_X^{-1} \neq I_K,$$

and hence we need a further trick.

We will for simplicity restrict in this section to the case where A is a symmetric matrix, and hence we seek a rank K eigendecomposition

$$A \approx U\Sigma U^T$$

where $U \in \mathbb{R}^{N\times K}$, $\Sigma \in \mathbb{R}^{K\times K}$ have $U^T U = I_K$ and Σ diagonal.

5.5.1 The QR factorisation

Definition 5.5.1 (Gram–Schmidt orthonormalisation)**.** *Suppose that we have K linearly independent vectors $x^1, \ldots, x^K \in \mathbb{R}^N$, and we want to construct an orthonormal basis $q^1, \ldots, q^K$ for the span of these vectors. The* Gram–Schmidt algorithm *for computing such a basis works as follows.*

1. *To compute q^1 we normalise x^1, i.e. $q^1 := x^1/\|x^1\|_2$.*
2. *Suppose we have computed $q^1, \ldots, q^{j-1}$ so far. We first create a vector orthogonal to all of these q^k by zeroing the components in each of these vectors in x^j:*

$$\tilde{q}^j := x^j - \sum_{k=1}^{j-1} \langle x^j, q^k \rangle q^k.$$

3. *Next, normalise $q^j := \tilde{q}^j/\|\tilde{q}^j\|_2$.*

Exercise 11. Check that $\{q^1, \ldots, q^K\}$ is indeed orthonormal and has the same span as $\{x^1, \ldots, x^K\}$.

Now, let us rewrite this in matrix form. Let

$$A := \begin{pmatrix} x^1 & x^2 & \cdots & x^K \end{pmatrix} \in \mathbb{R}^{N\times K} \qquad Q := \begin{pmatrix} q^1 & q^2 & \cdots & q^K \end{pmatrix} \in \mathbb{R}^{N\times K}$$

i.e. $A_{ij} = (x^j)_i$ and $Q_{ij} = (q^j)_i$. Next, for $i, j \in \{1, \ldots, K\}$, define $r_{ij} := \langle q^i, x^j \rangle$ for $i < j$, $r_{ij} := 0$ for $i > j$, and $r_{jj} := \|\tilde{q}^j\|_2$. Then

$$Q_{ij} = (q^j)_i = \frac{(x^j)_i - \sum_{k=1}^{j-1} \langle x^j, q^k \rangle (q^k)_i}{\|\tilde{q}^j\|_2} = \frac{A_{ij} - \sum_{k=1}^{j-1} r_{kj} Q_{ik}}{r_{jj}}$$

and so by rearranging

$$A_{ij} = \sum_{k=1}^{K} Q_{ik} r_{kj} = (QR)_{ij}.$$

Theorem 5.5.1 (Thin QR factorisation)**.** Let $A \in \mathbb{R}^{N\times K}$ have rank K, where $K \leq N$. Then there exists $Q \in \mathbb{R}^{N\times K}$ with orthonormal columns and $R \in \mathbb{R}^{K\times K}$ upper triangular such that

$$A = QR.$$

Observe that therefore $Q^TQ = I_K$.

Note. In practice, the QR factorisation is never computed in this way, as it is numerically unstable. See [GVL96, Chapter 5] for details.

5.5.2 The Nyström-QR method

First, compute the thin QR factorisation

$$QR = \begin{pmatrix} A_{XX} \\ A_{YX} \end{pmatrix}$$

where $Q \in \mathbb{R}^{N\times K}$ has orthonormal columns, and $R \in \mathbb{R}^{K\times K}$ is upper triangular.

Note. The requirement that $\begin{pmatrix} A_{XX} \\ A_{YX} \end{pmatrix}$ has full rank is entailed by the assumption that A_{XX} is invertible. If this does not hold, it is usually wise to resample X, or restrict to the subset of X for which this does hold. Since A_{XX} is small, its condition number can be cheaply computed.

Then the Nyström approximation for A (5.4) can be written as

$$A \approx QRA_{XX}^{-1}R^TQ^T.$$

Next, compute the eigendecomposition

$$RA_{XX}^{-1}R^T = \Phi\Sigma\Phi^T,$$

where $\Phi \in \mathbb{R}^{K\times K}$ is orthogonal and $\Sigma \in \mathbb{R}^{K\times K}$ is diagonal. It follows that A can be diagonalised as

$$A \approx (Q\Phi)\Sigma(Q\Phi)^T =: U\Sigma U^T$$

where $U := Q\Phi$ and hence $U^TU = \Phi^TQ^TQ\Phi = \Phi^TI_K\Phi = I_K$.

Note. This method is $O(NK^2)$ in time and $O(NK)$ in space.

5.6 Tying this all together

Suppose we have $A \in \mathbb{R}^{N\times N}$ a symmetric matrix. Suppose we can compute on demand any particular A_{ij}. Our goal in this chapter has been to devise a method to compute

$$e^{tA}v$$

for various $t \in \mathbb{R}$ and $v \in \mathbb{R}^N$, without ever using more that $O(N)$ space or time.

The method described in this chapter is then as follows.

1. Choose $K \ll N$, and choose $X \subseteq \{1, ..., N\}$ as a random subset with $|X| = K$. Let $Y := \{1, ..., N\} \setminus X$. This step is $O(N)$ in space and time.

2. Compute A_{XX} and A_{YX}. *These are the only bits of A we will ever use.* This step is $O(NK)$ in space and time.
3. Compute the thin QR factorisation

$$QR = \begin{pmatrix} A_{XX} \\ A_{YX} \end{pmatrix}.$$

   This step is $O(NK)$ in space and $O(NK^2)$ in time.
4. Compute the eigendecomposition

$$RA_{XX}^{-1}R^T = \Phi\Sigma\Phi^T,$$

   This step is $O(K^2)$ in space and $O(K^3)$ in time.
5. Compute $U = Q\Phi$. This step is $O(NK)$ in space and $O(NK^2)$ in time.
6. For each t and v, compute (recall (5.2))

$$e^{tA}v \approx v + U((e^{t\Sigma} - I_K)(U^T v)).$$

   This step is $O(N)$ in space and $O(NK)$ in time.

Recall that this final step works because $A \approx U\Sigma U^T$ and $U^T U = I_K$.

5.7 The singular value decomposition (SVD)

5.7.1 The SVD

> *"There are two types of people in the world: those who think the SVD is the most useful thing since the wheel, and those who haven't learned about the SVD yet."*
> — J. F. Williams, talk at TU Delft in 2019.

Lemma 5.7.1. Let $r \leq N$ and $U_1 \in \mathbb{R}^{N\times r}$ have orthonormal columns. Then there exists $U_2 \in \mathbb{R}^{N\times(N-r)}$ such that $\begin{pmatrix} U_1 & U_2 \end{pmatrix}$ is orthogonal.

Proof. Let $W \subseteq \mathbb{R}^N$ be the span of the columns of U_1. Take the columns of U_2 to be an orthonormal basis of $W^\perp$, the orthogonal complement of W. □

Theorem 5.7.2 (Existence of the SVD)**.** Let $A \in \mathbb{R}^{M\times N}$ then there exist orthogonal matrices $U \in \mathbb{R}^{M\times M}$ and $V \in \mathbb{R}^{N\times N}$ and a diagonal matrix $\Sigma \in \mathbb{R}^{M\times N}$ with non-negative entries, such that

$$A = U\Sigma V^T.$$

This is called the *singular value decomposition* (SVD) of A. The columns of U are called the *left singular vectors*, the columns of V are called the *right singular vectors*, and the diagonal entries of Σ are called the *singular values*.

Proof. It is equivalent to show that orthogonal U, V exist such that $U^T AV = \Sigma$. Define

$$\|A\| := \max_{v\in\mathbb{R}^N, \|v\|_2=1} \|Av\|_2.$$

This a continuous function maximised on a compact set, so the maximum is attained at some $v \in \mathbb{R}^N$ with $\|v\|_2 = 1$. Define $u \in \mathbb{R}^M$ by $u := \|A\|^{-1}Av$, so $\|u\|_2 = 1$.

By the above lemma, there exist $U_2 \in \mathbb{R}^{M\times(M-1)}$ and $V_2 \in \mathbb{R}^{N\times(N-1)}$ such that $U := \begin{pmatrix} u & U_2 \end{pmatrix}$ and $V := \begin{pmatrix} v & V_2 \end{pmatrix}$ are orthogonal.

Then

$$U^T A V = \begin{pmatrix} u^T A v & u^T A V_2 \\ U_2^T A v & U_2^T A V_2 \end{pmatrix} = \begin{pmatrix} \|A\| & w^T \\ 0 & B \end{pmatrix} =: A_1$$

for $w \in \mathbb{R}^{N-1}$ and $B \in \mathbb{R}^{(M-1)\times(N-1)}$, since $u^T A v = \|A\| u^T u = \|A\|$ and $U_2^T A v = \|A\| U_2^T u = 0$. We seek to show that $w = 0$. Consider

$$\left\| A_1 \begin{pmatrix} \|A\| \\ w \end{pmatrix} \right\|_2^2 = \left\| \begin{pmatrix} \|A\|^2 + w^T w \\ Bw \end{pmatrix} \right\|_2^2 \geq (\|A\|^2 + w^T w)^2$$

and hence $\|A_1\|^2 \geq \|A\|^2 + w^T w$. But since U, V are orthogonal, $\|A_1\| = \|A\|$. Hence $w^T w = 0$, and so $w = 0$.

By induction, we have that there exist orthogonal matrices $U_1 \in \mathbb{R}^{(M-1)\times(M-1)}$ and $V_1 \in \mathbb{R}^{(N-1)\times(N-1)}$ and a diagonal matrix $\Sigma_1 \in \mathbb{R}^{(M-1)\times(N-1)}$ with non-negative entries, such that $U_1^T B V_1 = \Sigma_1$. Then

$$\tilde{U} := U \begin{pmatrix} 1 & 0 \\ 0 & U_1 \end{pmatrix} \qquad\qquad \tilde{V} := V \begin{pmatrix} 1 & 0 \\ 0 & V_1 \end{pmatrix}$$

are orthogonal matrices and

$$\tilde{U}^T A \tilde{V} = \begin{pmatrix} 1 & 0 \\ 0 & U_1^T \end{pmatrix} A_1 \begin{pmatrix} 1 & 0 \\ 0 & V_1 \end{pmatrix} = \begin{pmatrix} \|A\| & 0 \\ 0 & U_1^T B V_1 \end{pmatrix} = \begin{pmatrix} \|A\| & 0 \\ 0 & \Sigma_1 \end{pmatrix} =: \Sigma,$$

completing the proof. □

5.7.2 The best possible rank K approximation

Theorem 5.7.3 (Eckart–Young theorem [EY36], see also [GVL96, Theorem 2.5.3])**.** Let $A \in \mathbb{R}^{M\times N}$ have SVD

$$A = U \Sigma V^T.$$

Then with respect to $\|\cdot\|$ the spectral or Frobenius norms on $\mathbb{R}^{M\times N}$, the best rank K approximation to A, i.e.,

$$A_K := \operatorname*{arg\,min}_{X \in \mathbb{R}^{M\times N}, \operatorname{rank}(X) \leq K} \|X - A\|$$

is given by the *reduced SVD*

$$A_K = U_K \Sigma_K V_K^T$$

where $U_K = (U_{ij})_{i=1..M, j=1..K}$, $\Sigma_K = (\Sigma_{ij})_{i=1..K, j=1..K}$, and $V_K = (V_{ij})_{i=1..N, j=1..K}$.

Proof. Left as an exercise. □

Chapter 6

Image segmentation with the graph MBO scheme

6.1 Turning an image into a graph

To turn an image $\mathcal{I} : V \to \mathbb{R}^d$ into a graph, we must first specify the vertex set V and edge set set E. But now you see why we have been using V to denote the domain of a discrete image: our vertex set is precisely the set of pixels in the image. For E, for now we shall simply take $E = \{(i, j) \in V^2 \mid i \neq j\}$.

The important information of the image we shall encode in the edge weights ω. These edge weights are computed in two steps. First, the pixels of the image are mapped to *feature vectors* $z : V \to \mathbb{R}^q$. The philosophy behind these feature vectors is that pixels which are "similar" should have nearby feature vectors, where what "similar" means is application-specific.

Example 1 (Simple example)**.** Suppose that $V = \{1, ..., n_1\} \times \{1, ..., n_2\}$. One simple way to define z is that for each pixel $i =: (i_1, i_2) \in V$, the feature vector z_i can be defined as:

$$z_i := (\mathcal{K}(i_1 - j_1, i_2 - j_2)\mathcal{I}_{(j_1,j_2)})_{-m \leq i_1 - j_1, i_2 - j_2 \leq m}.$$

That is, z_i stacks together the pixel values of each pixel in the $(2m + 1) \times (2m + 1)$ square centred at i, weighted by a kernel $\mathcal{K}$.

The above example is only one way of doing things, see e.g. [BF12, CGS+17, VZ05] for a discussion of options. A particularly important recent technique is using deep learning to construct the features, see e.g. [MMS+22].

Given these feature vectors, we now compute the weights using some *similarity function* evaluated on z_i and z_j. Some common choices include:

- The Gaussian function:
$$\omega_{ij} := e^{-d(z_i,z_j)^2/\sigma^2}$$
where the choice of metric d is also important, common are the Euclidean and ℓ^1 distances.
- The Zelnik-Manor–Perona [ZMP04] approach: Given a metric $d(z_i, z_j)$, define $\sigma_i := d(z_i, z_M)$ where z_M is the M^{th} closet feature vector to z_i. Then:
$$\omega_{ij} := e^{-d(z_i,z_j)^2/\sigma_i\sigma_j}$$
This is preferred if there are many scales which we desire to segment.

- Other kernels: one might also use other kernel functions, e.g.

$$\omega_{ij} := e^{-d(z_i,z_j)/\sigma}, \qquad \text{or} \qquad \omega_{ij} := \frac{1}{\sqrt{d(z_i,z_j)^2 + c^2}}.$$

- Cosine angle (see [HSB15]):

$$\omega_{ij} := \exp\left(-\frac{1}{2\sigma^2}\left(1 - \frac{\langle z_i, z_j\rangle}{||z_i||_2||z_j||_2}\right)^2\right).$$

  Note that this, unlike all the others, is scale-independent.

These weights are then sometimes renormalised according to the formula:

$$\tilde{\omega}_{ij} = \frac{\omega_{ij}}{d_i^{1-q/2} d_j^{1-q/2}}.$$

6.2 Image segmentation as a graph classification task

Recall the image segmentation task: We have been given an image $\mathcal{I} : V \to \mathbb{R}^d$, and we wish to find a $u : V \to L$, possibly with the help of a given *a priori* segmentation $f : Z \to L$ (with $Z \subseteq V$) such that we desire u to satisfy $u|_Z \approx f$.

By the above, we can encode the image x as a graph $G = (V, E, \omega)$. The u we seek can now be re-understood as a function on our graph G. The idea of this chapter is then as follows:

- We seek a u minimising the Ginzburg–Landau energy (with fidelity) on G. This u will by construction also serve as a segmentation of the image $\mathcal{I}$.
- We will find this u by evolving the graph MBO scheme (with fidelity forcing), which we saw in Chapter 4 was a numerical scheme for the Allen–Cahn gradient flow of the Ginzburg–Landau energy.

However, there is a further question: why should the u we have derived in this way be a *good* segmentation? Recall the graph Ginzburg–Landau energy (with fidelity):

$$\mathrm{GL}^{(r,s)}_{\varepsilon,\mu,f}(u) := \frac{1}{2}||\nabla(D^{-s}u)||^2_{\mathcal{E}} + \frac{1}{\varepsilon}\langle W \circ u, \mathbf{1}\rangle_{\mathcal{V},r-s} + \frac{1}{2}\langle u - f, M(u-f)\rangle_{\mathcal{V},r-s},$$

The first term penalises the segmentation u if two vertices with a high edge weight are in different segments, encouraging the segmentation to group similar vertices together. The second term wants the segmentation to be binary. The third term penalises u for disagreeing with the *a priori* segmentation, propagating those *a priori* labels to the rest of the vertices.

6.3 The basic algorithm for image segmentation via the MBO scheme

For any time step $0 < \tau \leq \varepsilon$ note that

$$\mathcal{S}_\tau u = e^{-\tau(\mathcal{L}+M)}u + b \tag{6.1}$$

where $b := F_\tau(\mathcal{L} + M)Mf$, which is independent of u.

A starting point for an algorithm could be the following steps.

1. **Parameters:** Time step $\tau > 0$, parameter $\varepsilon > 0$, fidelity parameter $\mu \in \mathcal{V}_{[0,\infty)}$ collected in the matrix $M = \operatorname{diag}(\mu)$, and normalisation parameters r, s.
2. **Input:** Image $\mathcal{I} : V \to \mathbb{R}^\ell$, training data Z, and labels f supported on Z.
3. Compute ω from $\mathcal{I}$, and encode $\mathcal{I}$ as a graph $G = (V, E, \omega)$ with $E = \{(i,j) \in V \times V \mid i \neq j\}$.
4. Compute $\mathcal{L}$ and therefore $e^{-\tau(\mathcal{L}+M)}$ and b.
5. From some initial condition u_0, compute the MBO sequence u_n, by alternately diffusing (applying S_τ) and thresholding.
6. Stop when some stopping condition is met, at $n = n_{\text{final}}$.
7. **Output:** $u_{n_{\text{final}}}$.

But this won't work, because of the size of the matrices.

6.4 Computing graph diffusion

Recall that to compute graph diffusion, we need to compute:

$$\mathcal{S}_\tau u = \underbrace{e^{-\tau(\mathcal{L}+M)}u}_{(A)} + \underbrace{F_\tau(\mathcal{L}+M)Mf}_{(B)}.$$

6.4.1 The Strang formula

To compute (A), we use a Strang formula method.

Theorem 6.4.1 (Strang formula)**.** For all $N \in \mathbb{N}$ and $X, Y \in \mathbb{R}^{N\times N}$,

$$e^{X+Y} = (e^{Y/2k}e^{X/k}e^{Y/2k})^k + O(k^{-2}).$$

Proof. By considering the Taylor series and collecting terms, one finds that $e^{X'+Y'}$ and $e^{Y'/2}e^{X'}e^{Y'/2}$ agree until the third-order terms. Hence

$$e^{X/k+Y/k} = e^{Y/2k}e^{X/k}e^{Y/2k} + O(k^{-3}).$$

Taking the k^{th} power of both sides completes the proof. □

6.4.2 The Nyström-QR method for $\mathcal{L}$

To apply the Nyström-QR method from Chapter 5 to $\mathcal{L}^{(r,s)}$, there are a couple extra steps. First, we use (5.4) to approximate the degrees:

$$d = \omega\mathbf{1} \approx \begin{pmatrix}\omega_{XX}\\ \omega_{YX}\end{pmatrix}\omega_{XX}^{-1}\begin{pmatrix}\omega_{XX} & \omega_{XY}\end{pmatrix}\mathbf{1} =: \hat{d}.$$

Next, we define $\hat{D} = \operatorname{diag}(\hat{d})$ and apply the Nyström-QR technique to $\hat{D}^{-(r+s)/2}\omega\hat{D}^{-(r+s)/2}$ to get

$$\tilde{\omega} := D^{-(r+s)/2}\omega D^{-(r+s)/2} \approx \hat{D}^{-(r+s)/2}\omega\hat{D}^{-(r+s)/2} \approx U\Sigma U^T.$$

Finally,

$$\mathcal{L}^{((r+s)/2,(r+s)/2)} = D^{1-r-s} - D^{-(r+s)/2}\omega D^{-(r+s)/2} \approx \hat{D}^{1-r-s} - U\Sigma U^T.$$

Then we get an approximate SVD for $\mathcal{L}^{(r,s)}$

$$\mathcal{L}^{(r,s)} = D^{(s-r)/2}\mathcal{L}^{((r+s)/2,(r+s)/2)}D^{(r-s)/2} \approx \hat{D}^{1-r-s} - (\hat{D}^{(s-r)/2}U)\Sigma(\hat{D}^{(r-s)/2}U)^T.$$

6.4.3 Computing (A)

Given $\mathcal{L} \approx A - U_1 \Sigma U_2^T$ with $U_2^T U_1 = I_K$ and A a diagonal matrix, for any $u \in \mathcal{V}$ we compute (writing $\delta t := \tau/k$)

$$\begin{aligned} e^{-\tau(\mathcal{L}+M)}u &\approx e^{-\tau(A+M-U_1\Sigma U_2^T)} \\ &= \left(e^{-\frac{1}{2}\tau/k(A+M)} e^{\tau/k U_1 \Sigma U_2^T} e^{-\frac{1}{2}\tau/k(A+M)}\right)^k u + O(k^{-2}) \\ &= \left(e^{-\frac{1}{2}\delta t(A+M)} \left(I + U_1(e^{\delta t \Sigma} - I_K)U_2^T\right) e^{-\frac{1}{2}\delta t(A+M)}\right)^k u + O(\delta t^2). \end{aligned} \tag{6.2}$$

That is, we define $e^{-\tau(\mathcal{L}+M)}u = v_k$ where $v_0 = u$ and

$$\begin{aligned} v_{r+1} &:= e^{-\delta t(A+M)} v_r + e^{-\frac{1}{2}\delta t(A+M)} U_1 (e^{\delta t \Sigma} - I_K) U_2^T e^{-\frac{1}{2}\delta t(A+M)} v_r \\ &= a_1(\delta t) \odot v_r + a_3(\delta t) \odot \left(U_1 \left(a_2(\delta t) \odot \left(U_2^T \left(a_3(\delta t) \odot v_r\right)\right)\right)\right) \end{aligned} \tag{6.3}$$

where $\odot$ is the Hadamard (i.e. elementwise) product, $a_1(\delta t) := \exp(-\delta t(\mu + \operatorname{diag}(A)))$, $a_2(\delta t) := \exp(\delta t \operatorname{diag}(\Sigma)) - \mathbf{1}_K$, and $a_3(\delta t) := \exp(-\frac{1}{2}\delta t(\mu + \operatorname{diag}(A)))$ is the elementwise square root of $a_1(\delta t)$ (where exp is applied elementwise, and $\mathbf{1}_K$ is the vector of K ones).

6.4.4 Computing (B)

Computing (B) is a little less glamorous, but the upside is that it only needs to be computed once, since it is independent of u. The most straightforward way to compute it is to observe that it equals $\mathcal{S}_\tau \mathbf{0}$. Hence, we want to compute a solution to (4.2) from initial condition $\mathbf{0}$. One way to do this is compute (4.2), i.e.

$$\frac{du}{dt} = -\mathcal{L}u(t) - M(u(t) - f),$$

as in [MKB13] via the semi-implicit Euler scheme

$$\frac{u_{n+1} - u_n}{\delta t} = -\mathcal{L}u_{n+1} - M(u_n - f).$$

This scheme is implicit in $\mathcal{L}$ because this system is *stiff*, i.e. there is a large ratio between the small and large eigenvalues. This has solution

$$\begin{aligned} u_{n+1} &= (I + \delta t \mathcal{L})^{-1}(u_n - \delta t M(u_n - f)) \\ &\approx (I + \delta t A - \delta t U_1 \Sigma U_2^T)^{-1}(u_n - \delta t M(u_n - f)). \end{aligned}$$

This works best in the $r + s = 1$ case, where $A = I$ and so

$$\begin{aligned} u_{n+1} &\approx ((1 + \delta t)I - \delta t U_1 \Sigma U_2^T)^{-1}(u_n - \delta t(Mu_n - f)) \\ &\approx U_1((1 + \delta t)I_K - \delta t \Sigma)^{-1} U_2^T (u_n - \delta t(Mu_n - f)), \end{aligned}$$

where we have used the approximation $I \approx U_1 U_2^T$. Thus, we define $b = v_k$ where $v_0 = \mathbf{0}$ and

$$v_{r+1} := U_1(a_0 \odot (U_2^T(v_r - \delta t(\mu \odot v_r - f)))), \tag{6.4}$$

where $a_0 = (1 + \delta t - \delta t \operatorname{diag}(\Sigma))^{-1}$.

Note. In the general case, this can still be computed using the *Woodbury identity*:

$$(I+\delta tA-\delta tU_1\Sigma U_2^T)^{-1} = (I+\delta tA)^{-1}\left(I + \delta tU_1(-\Sigma^{-1} + \delta tU_2^T(I+\delta tA)^{-1}U_1)^{-1}U_2^T(I+\delta tA)^{-1}\right).$$

That is, let $a_4 := (1 + \delta t \operatorname{diag}(A))^{-1}$. Then

$$(I + \delta tA - \delta tU_1\Sigma U_2^T)^{-1}v = a_4 \odot (v + \delta t(U_1 w))$$

where w solves

$$(-\Sigma^{-1} + \delta tU_2^T(a_4 \odot U_1))w = U_2^T(a_4 \odot v).$$

This is more cumbersome than the $r + s = 1$ case, and becomes less well-behaved if Σ is not invertible.

6.5 Interlude: numerically testing these methods

6.5.1 Comparison of methods on a toy image

In this section, we shall compare the accuracy of the above methods for two approximation tasks. First, the task of approximating the symmetric normalised Laplacian $\mathcal{L}^{(1/2,1/2)}$, and second, the task of approximating $e^{-t\mathcal{L}^{(1/2,1/2)}}$ for $t \in \{10^{-2}, 10^{-1}, 1, 10\}$. These will both be performed on a graph built from a small enough image for us to compute the ground truth.

In particular, we will consider graphs built on the pixels of the 60×60 greyscale image from Figure 6.1.[1] Hence, $V = \{1, \ldots, 3600\}$, and for $i \neq j \in V$ we will define

$$\omega_{ij} = e^{-(\mathcal{I}_i - \mathcal{I}_j)^2},$$

where $\mathcal{I}_i \in [0, 1]$ is the intensity value of pixel i.

We will consider the following methods for these tasks, for $K \in \{50, 100, 150, \ldots, 500\}$:

i. **Optimal rank K method:** Approximate $\mathcal{L}^{(1/2,1/2)}$ by I minus the best rank K approximation of $\tilde{\omega}$, i.e.
$$\mathcal{L}^{(1/2,1/2)} \approx I - U_{\text{best}}\Sigma_{\text{best}}U_{\text{best}}^T,$$
where Σ_{best} is a $K\times K$ diagonal matrix with diagonal the largest *negative* eigenvalues of $\tilde{\omega}$, as these correspond to the largest eigenvalues of $\mathcal{L}^{(1/2,1/2)}$, and U_{best} is an $N\times K$ matrix of the corresponding eigenvectors. We approximate the matrix exponential using intead the largest *positive* eigenvalues of $\tilde{\omega}$ (as these correspond to the largest eigenvalues of $e^{t\tilde{\omega}}$) and the corresponding eigenvectors.

ii. **Nyström method:** We will compute rank K decomposition:
$$\tilde{\omega} \approx U_{QR}\Sigma_{QR}U_{QR}^T$$
using the Nyström-QR method. We will then approximate $\mathcal{L}^{(1/2,1/2)}$ and $e^{-t\mathcal{L}^{(1/2,1/2)}}$ as in the previous method. As these methods are randomised we will repeat them ten times and present the median error and mean times in the below results.

The two tests we shall perform are as follows:

[1] Defined by `[rand(30) magic(30)/max(max(magic(30))); 0.5*ones(30) min(max(0,0.5 + randn(30)/2),1);]` in MATLAB.

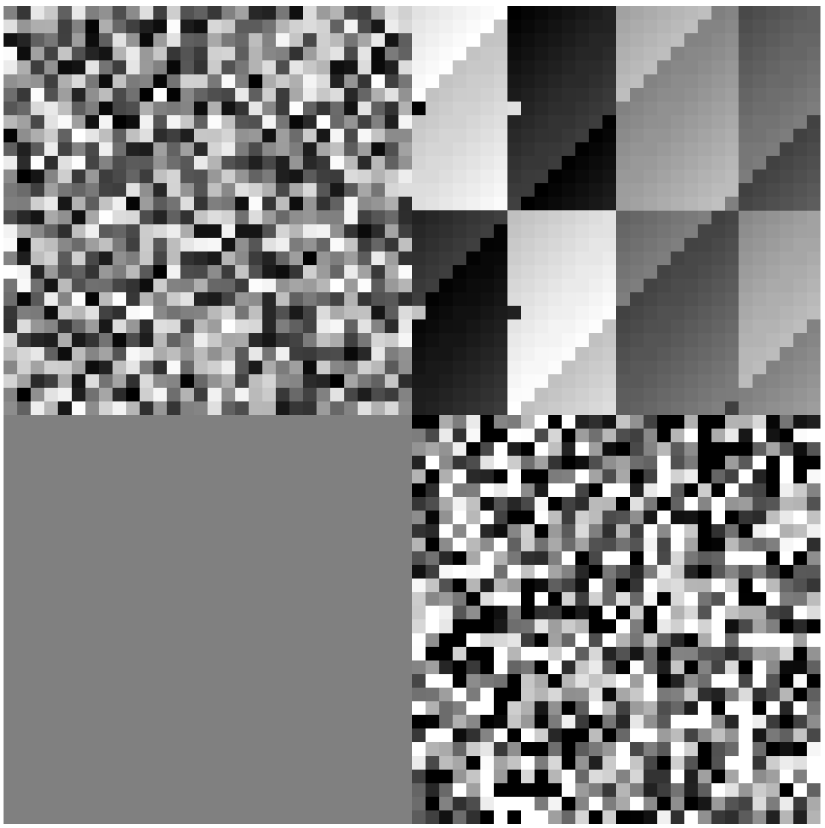

Figure 6.1: Test 60×60 image for numerical experiments.

A. Given an approximation L for $\mathcal{L}^{(1/2,1/2)}$, we will compute the *relative Frobenius error*

$$\frac{\|L - \mathcal{L}^{(1/2,1/2)}\|_F^2}{\|\mathcal{L}^{(1/2,1/2)}\|_F^2},$$

where $\|A\|_F^2 = \mathrm{tr}(A^T A) = \sum_{i,j} A_{ij}^2$ is the Frobenius norm.

B. Given an approximation E for $e^{-t\mathcal{L}^{(1/2,1/2)}}$, we likewise will compute

$$\frac{\|E - e^{-t\mathcal{L}^{(1/2,1/2)}}\|_F^2}{\|e^{-t\mathcal{L}^{(1/2,1/2)}}\|_F^2}.$$

6.5.2 Results

We present the results and timings for task (A) in Figure 6.2.

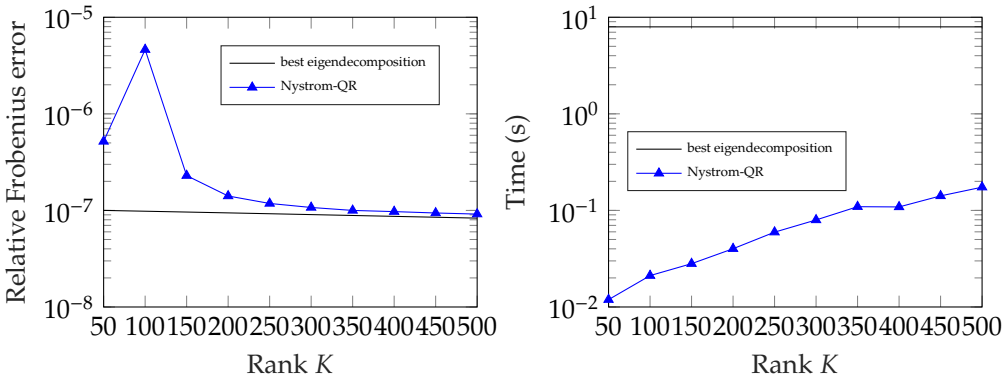

Figure 6.2: Relative Frobenius errors (left) and computation times (right) for the rank K approximate eigendecompositions of $\mathcal{L}^{(1/2,1/2)}$.

We present the results for task (B) in Figure 6.3. We omit the timings as they do not vary significantly between methods.

6.6 The full pipeline

We summarise the full pipeline in Algorithm 1.

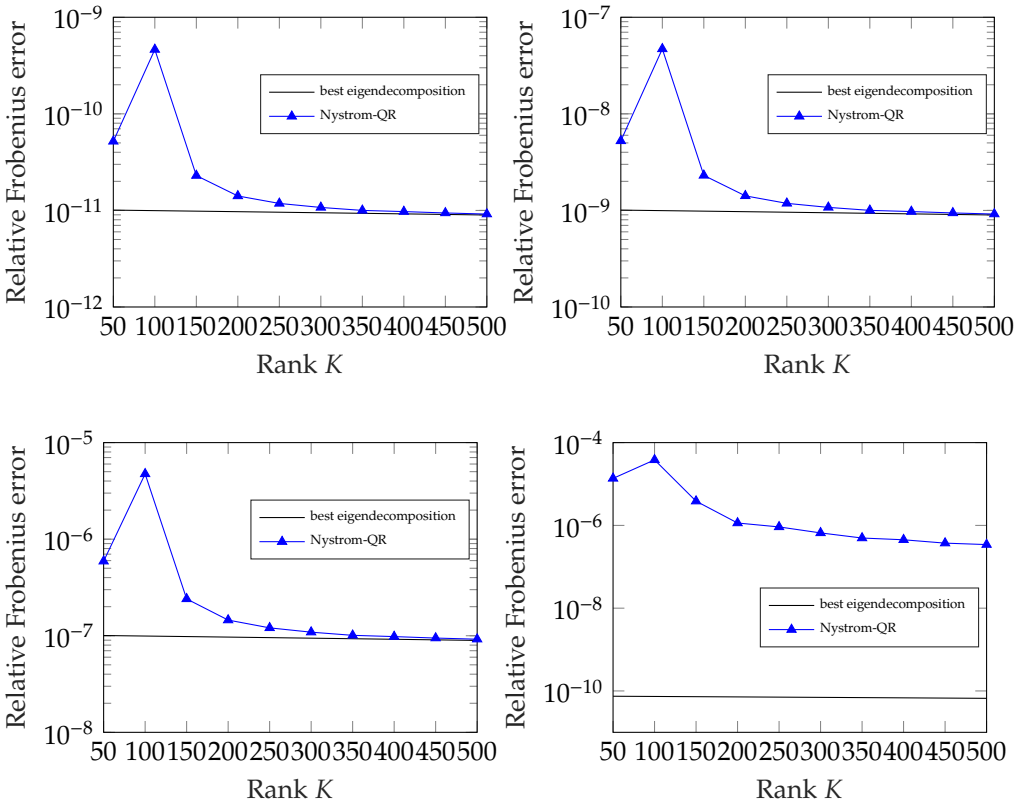

Figure 6.3: Relative Frobenius errors to $e^{-t\mathcal{L}^{(1/2,1/2)}}$ for $t = 10^{-2}$ (top left), $t = 10^{-1}$ (top right), $t = 1$ (bottom left), and $t = 10$ (bottom right)

Algorithm 1 Segmentation algorithm using the MBO scheme.

function MBOSeg($\mathcal{I}, \mu, f, V, Z, K, \alpha, \delta, \sigma, \tau, k, k_b, r, s$) ▷ Segments an image $\mathcal{I}$ by iterating an MBO scheme. $\mathcal{I} : V \to \mathbb{R}^\ell$, $\mu \in \mathcal{V}_{[0,\infty)}$, $f \in \mathcal{V}$, $k, k_b, K \in \mathbb{N}$, $\alpha, \delta, \sigma, \tau \in [0, \infty)$, $r, s \in \mathbb{R}$. *Requires:* $K \ll |V|$, $\mu|_{V\setminus Z} = f|_{V\setminus Z} = \mathbf{0}$, $r + s = 1$.

—— **Encoding $\mathcal{I}$ as a graph** ——

$z = \texttt{feature\_map}(\mathcal{I})$ ▷ Computes the feature vectors z of $\mathcal{I}$

$\Omega : (i, j) \mapsto e^{-\|z_i - z_j\|_2^2/\sigma^2}$ ▷ Defines a function that maps (i, j) to its weight ω_{ij}

—— **Nyström-QR for $\mathcal{L}^{(r,s)}$** ——

$X = \texttt{random\_subset}(V, K)$ ▷ Computes a random subset of V of size K

$\omega_{XX} = \Omega(X, X)$ ▷ Applies Ω to build the sub-matrix of ω with $i, j \in X$

$\omega_{VX} = \Omega(V, X)$ ▷ Builds the sub-matrix of ω with $i \in V$ and $j \in X$

$\hat{d} = \omega_{VX}\left(\omega_{XX}^{-1}\left(\omega_{VX}^T \mathbf{1}\right)\right)$ ▷ Uses (5.4) to approximate $d = \omega\mathbf{1}$

$\tilde{\omega}_{VX} = \hat{d}^{-(r+s)/2} \odot \omega_{VX}$ ▷ Applying $\odot$ columnwise, i.e. $(\tilde{\omega}_{VX})_{ij} = \hat{d}_i^{-(r+s)/2}(\omega_{VX})_{ij}$

$[Q, R] = \texttt{thinQR}(\tilde{\omega}_{VX})$ ▷ Computes thin QR factorisation $\tilde{\omega}_{VX} = QR$

$S = R\omega_{XX}^{-1}R^T$ ▷ Computes $S \in \mathbb{R}^{K\times K}$

$S = (S + S^T)/2$ ▷ Corrects symmetry-breaking computational errors

$[\Phi, \Sigma] = \texttt{eig}(S)$ ▷ Computes eigendecomposition $S = \Phi\Sigma\Phi^T$

$U = Q\Phi$

$U_1 = \hat{d}^{(s-r)/2} \odot U$

$U_2 = \hat{d}^{(r-s)/2} \odot U$ ▷ $\mathcal{L}^{(r,s)} \approx \hat{D}^{1-r-s} - U_1\Sigma U_2^T$

—— **Computing b** ——

▷ Method below requires $r + s = 1$; see Section 6.4.4 for general case

$a_0 = (1 + \tau/k_b - \tau/k_b \operatorname{diag}(\Sigma))^{-1}$ ▷ Reciprocation applied componentwise

$b = \mathbf{0}$

for $j = 1$ **to** k_b **do**

$b = U_1(a_0 \odot (U_2^T(b - \tau/k_b(\mu \odot (b - f)))))$ ▷ Euler scheme step, see (6.4)

end for

—— **Set-up for MBO scheme** ——

$a_1 = \exp(-\tau/k(\mu + \hat{d}^{1-r-s}))$ ▷ exp applied componentwise

$a_2 = \exp(\tau/k \operatorname{diag}(\Sigma)) - \mathbf{1}_K$ ▷ exp applied componentwise

$a_3 = \operatorname{sqrt}(a_1)$ ▷ a_1, a_2, and a_3 are for the Strang formula iterations, see (6.3)

$u^0 = \alpha\chi_{V\setminus Z} + f$ ▷ Initial condition: the *a priori* segmentation on the training data pixels, α on the rest

—— **The MBO scheme** ——

$m = 0$

while $\|u^m - u^{m-1}\|_2^2/\|u^m\|_2^2 \geq \delta$ **do** ▷ Until stopping condition met

—— **Diffuse u^m** ——

$v = u^m$

for $j = 1$ **to** k **do**

$v = a_1 \odot v + a_3 \odot \left(U_1\left(a_2 \odot \left(U_2^T\left(a_3 \odot v\right)\right)\right)\right)$ ▷ Strang formula step, see (6.3)

end for

$v = v + b$ ▷ $v \approx \mathcal{S}_\tau u^m$, see (6.1)

—— **Threshold v** ——

$V_2 = \{i \in V \mid v_i \geq \frac{1}{2}\}$

$u^{m+1} = \chi_{V_2}$ ▷ Applies the MBO thresholding

$m = m + 1$

end while

return u^m

end function

6.7 Results

We give some examples of segmentations using the above method on cow images from the Microsoft Research Cambridge Object Recognition Image Database. For more details on these examples, see Budd, Van Gennip, and Latz [BvL21] and Budd [Bud22].

6.7.1 RGB example

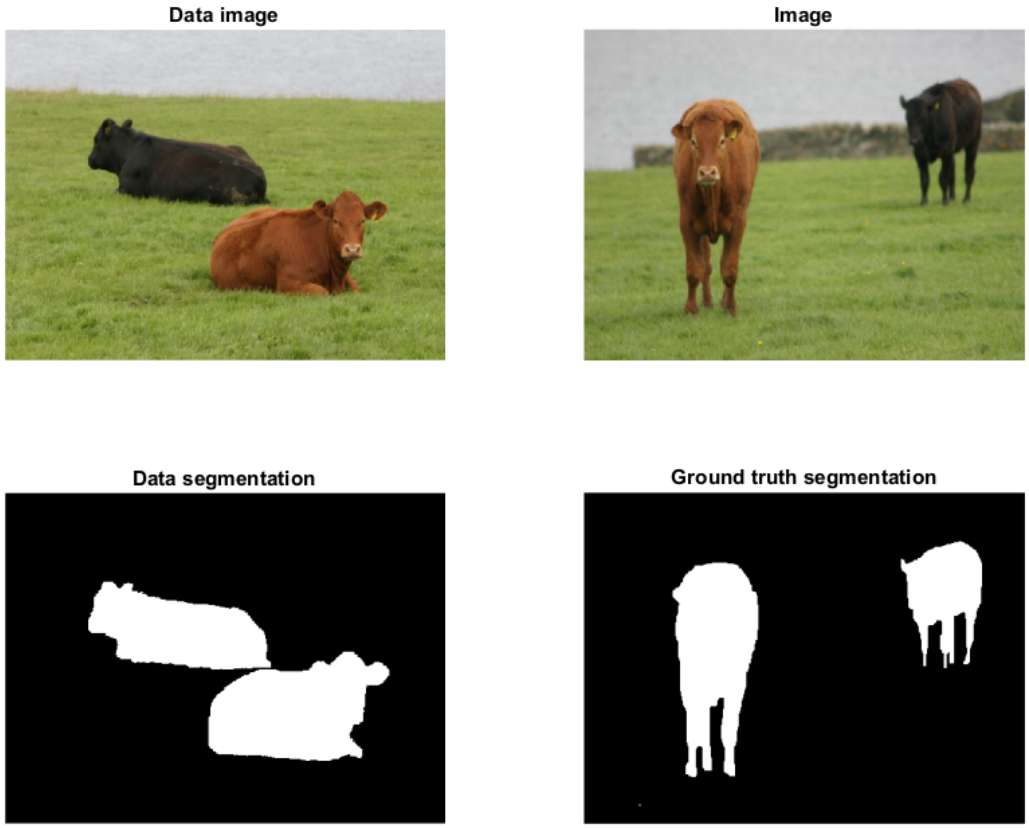

Figure 6.4: Two cows example. Top-left: the reference data image. Top-right: the image to be segmented. Bottom-left: the *a priori* segmentation f (which is a segmentation of the reference data image). Bottom-right: the ground truth segmentation of the top-right image. Both segmentations were drawn by hand by the authors.

Figure 6.5: Progression of MBO scheme over the course of its iterations. Reproduced from [Bud22, Fig. 5.11(c)].

Figure 6.6: Image masked with MBO segmentation, reproduced from [Bud22, Fig. 5.9(d)]. The segmentation accuracy is 98.4622% and run time was 2.5s.

6.7.2 Greyscale example

Figure 6.7: Greyscale two cows example. Left: reference data image. Right: image to be segmented. Ground truth segmentations are as in Figure 6.4.

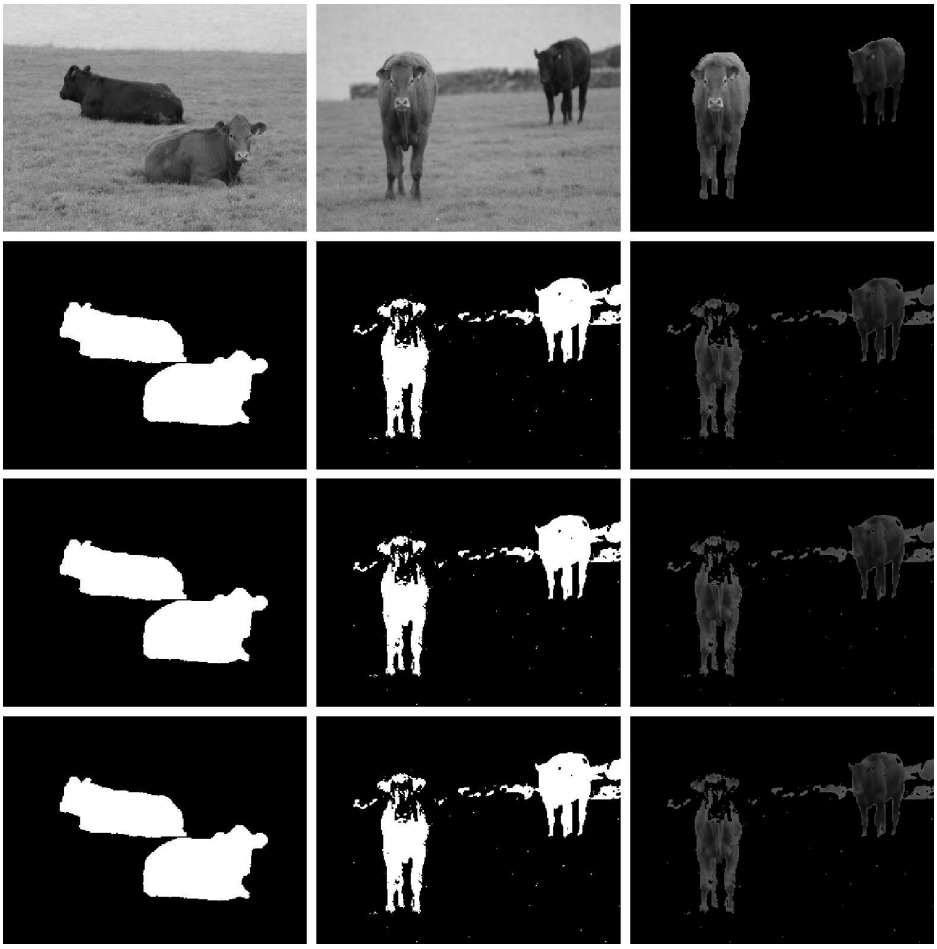

Figure 6.8: Progression of MBO segmentation of greyscale two cows. The higher run time is due to using a larger K in the Nyström-QR. Reproduced from [Bud22, Fig. 5.16(c)].

Bibliography

[AMÖS19] Simon Arridge, Peter Maass, Ozan Öktem, and Carola-Bibiane Schönlieb. Solving inverse problems using data-driven models. *Acta Numerica*, 28:1–174, 2019. 12

[BE91] J. F. Blowey and C. M. Elliott. The cahn–hilliard gradient theory for phase separation with non-smooth free energy part i: Mathematical analysis. *European Journal of Applied Mathematics*, 2(3):233–280, 1991. 39

[BF12] Andrea Bertozzi and Arjuna Flenner. Diffuse interface models on graphs for classification of high dimensional data. *Multiscale Modeling & Simulation*, 10:1090–1118, 2012. 22, 63

[BKS18] Jessica Bosch, Steffen Klamt, and Martin Stoll. Generalizing diffuse interface methods on graphs: Nonsmooth potentials and hypergraphs. *SIAM Journal on Applied Mathematics*, 78(3):1350–1377, 2018. 39

[Bra07] Andrea Braides. *Gamma-convergence for Beginners*. Oxford Scholarship Online. Oxford University Press, Oxford, 2007. 35

[Bre83] Haïm Brezis. *Analyse fonctionnelle*. Collection Mathématiques Appliquées pour la Maîtrise. [Collection of Applied Mathematics for the Master's Degree]. Masson, Paris, 1983. Théorie et applications. [Theory and applications]. 41, 42

[Bud22] Jeremy Budd. *Theory and Applications of Differential Equation Methods for Graph-based Learning*. PhD thesis, Technische Universiteit Delft, 2022. 72, 73, 74, 76

[BvL21] Jeremy Budd, Yves van Gennip, and Jonas Latz. Classification and image processing with a semi-discrete scheme for fidelity forced Allen–Cahn on graphs. *GAMM Mitteilungen*, 44(1):1–43, 2021. 72

[BZ87] Andrew Blake and Andrew Zisserman. *Visual reconstruction*. MIT press, 1987. 14

[CCTS20] J. Calder, Brendan Cook, Matthew Thorpe, and Dejan Slepcev. Poisson learning: Graph based semi-supervised learning at very low label rates. In *Proceedings of the International Conference on Machine Learning*, pages 8588–8598, 2020. 31, 32

[CGS+17] Luca Calatroni, Yves Gennip, Carola-Bibiane Schönlieb, Hannah Rowland, and Arjuna Flenner. Graph Clustering, Variational Image Segmentation Methods and Hough Transform Scale Detection for Object Measurement in Images. *Journal of Mathematical Imaging and Vision*, 57:269–291, 2017. 63

[Cha00] Antonin Chambolle. Inverse problems in image processing and image segmentation: some mathematical and numerical aspects. Technical report, 2000. 11

[Che69] J Cheeger. A lower bound for the smallest eigenvalue of the laplacian. In *Proceedings of the Princeton Conference in Honor of Professor S. Bochner*. Princeton University Press, Princeton, 1969. 29

[CRT06] Emmanuel J. Candès, Justin K. Romberg, and Terence Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics*, 59(8):1207–1223, 2006. 11

[CV01] T.F. Chan and L.A. Vese. Active contours without edges. *IEEE Transactions on Image Processing*, 10(2):266–277, 2001. 14

[ET99] Ivar Ekeland and Roger Téman. *Convex Analysis and Variational Problems*, volume 28 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics, Philadelphia, USA, 1999. 40

[EY36] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936. 62

[FBCM04] Charless Fowlkes, Serge Belongie, Fan Chung, and Jitendra Malik. Spectral grouping using the Nyström method. *IEEE transactions on pattern analysis and machine intelligence*, 26(2):214–225, 2004. 56

[FBH+22] Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J R Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, et al. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610(7930):47–53, 2022. 53

[GBT21] Alden Green, Sivaraman Balakrishnan, and Ryan J. Tibshirani. Statistical guarantees for local spectral clustering on random neighborhood graphs. *J. Mach. Learn. Res.*, 22:Paper No. [247], 71, 2021. 30

[Get12] Pascal Getreuer. Rudin–Osher–Fatemi total variation denoising using split Bregman. *Image Processing On Line*, 2012. 10, 11, 12

[GTHH21] Nicolás García Trillos, Franca Hoffmann, and Bamdad Hosseini. Geometric structure of graph Laplacian embeddings. *J. Mach. Learn. Res.*, 22:Paper No. 63, 55, 2021. 30

[GVL96] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, Maryland, third edition, 1996. 60, 62

[Had02] Jacques Hadamard. Sur les problèmes aux dérivées partielles et leur signification physique. *Princeton university bulletin*, pages 49–52, 1902. 8

[HHOS22] Franca Hoffmann, Bamdad Hosseini, Assad A. Oberai, and Andrew M. Stuart. Spectral analysis of weighted Laplacians arising in data clustering. *Appl. Comput. Harmon. Anal.*, 56:189–249, 2022. 30

[Hig08] Nicholas J. Higham. *Functions of matrices*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2008. Theory and computation. 53

[HSB15] Huiyi Hu, Justin Sunu, and Andrea L. Bertozzi. Multi-class graph mumford-shah model for plume detection using the mbo scheme. In Xue-Cheng Tai, Egil Bae, Tony F. Chan, and Marius Lysaker, editors, *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 209–222, Cham, 2015. Springer International Publishing. 64

[LEL14] François Lozes, Abderrahim Elmoataz, and Olivier Lézoray. Partial difference operators on weighted graphs for image processing on surfaces and point clouds. *IEEE Transactions on Image Processing*, 23(9):3896–3909, 2014. 8

[LÖS18] Sebastian Lunz, Ozan Öktem, and Carola-Bibiane Schönlieb. Adversarial regularizers in inverse problems. *Advances in neural information processing systems*, 31, 2018. 12

[MAF20] Ymir Mäkinen, Lucio Azzari, and Alessandro Foi. Collaborative Filtering of Correlated Noise: Exact Transform-Domain Variance for Improved Shrinkage and Patch Matching. *IEEE Transactions on Image Processing*, 29:8339–8354, 2020. 7

[MKB13] Ekaterina Merkurjev, Tijana Kostić, and Andrea Bertozzi. An MBO Scheme on Graphs for Classification and Image Processing. *SIAM Journal on Imaging Sciences*, 6:1903–1910, 2013. 66

[MMS+22] Kevin Miller, John Mauro, Jason Setiadi, Xoaquin Baca, Zhan Shi, Jeff Calder, and Andrea L. Bertozzi. Graph-based Active Learning for Semi-supervised Classification of SAR Data, 2022. 63

[MS89] David Mumford and Jayant Shah. Optimal approximations by piecewise smooth functions and associated variational problems. *Communications on Pure and Applied Mathematics*, 42(5):577–685, 1989. 13

[MS12] Jean-Michel Morel and Sergio Solimini. *Variational methods in image segmentation: with seven image processing experiments*, volume 14. Springer Science & Business Media, 2012. 14

[MVL78] Cleve Moler and Charles Van Loan. Nineteen dubious ways to compute the exponential of a matrix. *SIAM review*, 20(4):801–836, 1978. 53, 54

[MVL03] Cleve Moler and Charles Van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Rev.*, 45(1):3–49, 2003. 53

[NJW01] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14, pages 849–856. MIT Press, Cambridge, 2001. 29

[Nys30] E. J. Nyström. Über die praktische auflösung von integralgleichungen mit anwendungen auf randwertaufgaben. *Acta Math.*, 54:185–204, 1930. 56

[OP88] Y. Oono and S. Puri. Study of phase-separation dynamics by use of cell dynamical systems. I. Modeling. *Phys. Rev. A*, 38:434–453, 1988. 39

[Phi62] David L Phillips. A technique for the numerical solution of certain integral equations of the first kind. *Journal of the ACM (JACM)*, 9(1):84–97, 1962. 8

[REM17] Yaniv Romano, Michael Elad, and Peyman Milanfar. The little engine that could: Regularization by denoising (red). *SIAM Journal on Imaging Sciences*, 10(4):1804–1844, 2017. 12

[Roe99] Greg Roelofs. *PNG: the definitive guide*. O'Reilly & Associates, Inc., 1999. 7

[ROF92] Leonid I. Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259–268, 1992. 9

[S+69] Volker Strassen et al. Gaussian elimination is not optimal. *Numerische mathematik*, 13(4):354–356, 1969. 53

[SM00] Jianbo Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000. 29

[Tik63] A. N. Tikhonov. Solution of incorrectly formulated problems and the regularization method. *Soviet Math. Dokl.*, 4:1035–1038, 1963. 8

[VBW13] Singanallur V. Venkatakrishnan, Charles A. Bouman, and Brendt Wohlberg. Plug-and-Play priors for model based reconstruction. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 945–948, 2013. 12

[vGB18] Yves van Gennip and Andrea L. Bertozzi. Gamma-convergence of graph Ginzburg-Landau functionals, 2018. 35

[VL07] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17:395–416, 2007. 29

[VZ05] Manik Varma and Andrew Zisserman. A statistical approach to texture classification from single images. *International Journal of Computer Vision*, 62:61–81, 2005. 63

[Wal91] Gregory K Wallace. The jpeg still picture compression standard. *Communications of the ACM*, 34(4):30–44, 1991. 7

[WS00] Christopher Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13. MIT Press, 2000. 56

[ZGL03] Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *IN ICML*, pages 912–919, 2003. 30

[ZMP04] Lihi Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. In *Proceedings of the 17th International Conference on Neural Information Processing Systems*, NIPS'04, page 1601–1608, Cambridge, MA, USA, 2004. MIT Press. 63