1. Consider the 2D heat equation,

$$\frac{\partial^2 f(x,y,t)}{\partial x^2} + \frac{\partial^2 f(x,y,t)}{\partial y^2} = \frac{\partial f(x,y,t)}{\partial t}$$

over a rectangle of sides L_x and L_y with Neumann boundary conditions such that $x \in [0, L_x]$ and $y \in [0, L_y]$ show that the eigenfunctions ϕ_{nm} and eigenvalues λ_{nm} are given by:

$$\phi_{nm} = \cos\left(\frac{n\pi x}{L_x}\right)\cos\left(\frac{m\pi y}{L_y}\right), \quad \lambda_{nm} = \frac{n^2\pi^2}{L_x^2} + \frac{m^2\pi^2}{L_y^2}.$$

For a narrow strip $L_x = \frac{1}{2}L_y$, compute the leading six eigenvectors.

2. Coding problem: Manifold learning smackdown

Choose two of the following methods

- Isomap
- LLE
- Hessian eigenmaps
- Laplacian eigenmaps
- t-SNE
- Local tangent space alignment

(a) Briefly describe each method (what is the idea of the method? what cost function is it trying to minimize?)

(b) Apply the two methods to the three provided datasets (swiss roll, swiss roll with hole, noisy swiss roll) and plot the top 2 components. Try to find the best parameter choice for each method. Compare the performance of the two methods. Note: you may use implementations in python scikit-learn.

(c) Apply the two methods to 6000 random samples from MNIST and visualize in 3D. Color the points by their labels. Compare the performance of the two methods. The dataset can be found here `http://yann.lecun.com/exdb/mnist/`.

3. GSP coding problem: perform the following steps:

(a) Sample uniformly at random $N = 1000$ points in a square $x, y \in [-1, 1]$.

(b) Create a k-nn graph by connecting every point to its 30 nearest neighbors. Choose the weights by a Gaussian kernel, with any scaling strategy you think adequate.

(c) Compute the Fourier domain defined by the eigenvectors $\{\psi_i\}$ of the normalized graph Laplacian,

$$L = I - D^{-0.5} W D^{-0.5}$$

For the 6 least significant eigenvectors, create a scatter plot of the points in $2D$, colored by the value of the corresponding element in the eigenvector.

(d) (Do parts (d)-(h) only for the leading ~ 30 eigenvectors that should be real) Create a line plot, where the x-axis is the eigenvector number and the y-axis contains the number of zero crossing for each vector. Recall that the number of zero crossing for an eigenvector ψ is equal to the number of connected pairs i, j where $\psi(i)\psi(j) < 0$.

(e) Create a graph signal: the signal at node i is equal to,

$$s_i = \exp(-(x_i^2 + y_i^2)/\sigma)$$

Create a scatter where every node is colored by its signal. This is the signal represented in the node domain.

(f) For 3 values of σ, plot the signal in the Fourier domain, that is, plot $g_i = \langle s, \psi_i \rangle$ as a function of i.

(g) Pick one value of sigma. Each node is given a *noisy signal* equal to

$$\tilde{s}_i = s_i + \xi_i$$

where ξ_i are random samples from a Gaussian distribution $N(0, 0.1)$. Create a low pass filter in the Fourier domain by

$$\hat{g}_i = \frac{g_i}{1 + \tau\lambda_i}$$

where λ_i are the corresponding eigenvalues.

(h) Invert the signal back to the node domain by

$$\hat{s} = \sum_i \hat{g}\psi$$

and plot the l_2 norm of the error $\|\boldsymbol{s} - \hat{\boldsymbol{s}}\|$ as a function of τ, where

$$s = [s_1, \ldots, s_N]$$