Computational Medicine
Computational Anatomy
Homework 2

1 Invertible Matrices

Consider a vector $x = \binom{x}{y} \in \mathbb{R}^2$, and a transformation $\varphi(x) = Ax = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \binom{x}{y}$, with Jacobian matrix $D\varphi(x) = \begin{pmatrix} \frac{\partial\varphi_1(x)}{\partial x} & \frac{\partial\varphi_1(x)}{\partial y} \\ \frac{\partial\varphi_2(x)}{\partial x} & \frac{\partial\varphi_2(x)}{\partial y} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} = A$.

Show that if the matrix A is invertible, then the transformation φ is 1 to 1 and onto. That is:

1.1

Show that $\varphi(x) = \varphi(x') \implies x = x'$

1.2

Show that $\forall y \in \mathbb{R}^2$, $\exists x \in \mathbb{R}^2$ such that $\varphi(x) = y$.

2 Jacobian Chain Rule

2.1 Scalar transformations

Consider a point $x \in \mathbb{R}$ and the transformation $f : x \mapsto f(x) \in \mathbb{R}$. Let the inverse transformation be $f^{-1} : x \mapsto f^{-1}(x) \in \mathbb{R}$.

Show that $\frac{d}{dx} f^{-1}(x) = \frac{1}{\frac{d}{dx} f\big|_{f^{-1}(x)}}$.

Hint: prove and use the chain rule $\frac{d}{dx} f \circ f^{-1}(x) = \frac{d}{dx} f\big|_{f^{-1}(x)} \frac{d}{dx} f^{-1}(x)$.

2.2 Vector transformations

Consider a point $x = \left(\begin{smallmatrix} x \\ y \\ z \end{smallmatrix}\right) \in \mathbb{R}^3$, and the transformation $\varphi : x \mapsto \varphi(x) \in \mathbb{R}^3$. Let the inverse transformation be defined by $\varphi^{-1} : x \mapsto \varphi^{-1}(x) \in \mathbb{R}^3$.

Show that $D\varphi^{-1}(x) = \left(D\varphi\big|_{\varphi^{-1}(x)} \right)^{-1}$.

Hint: prove and use the chain rule $D(\varphi \circ \varphi^{-1}(x)) = D\varphi|_{\varphi^{-1}(x)} D\varphi^{-1}(x)$.

The Jacobian matrix is defined as $D\varphi(x) = \begin{pmatrix} \frac{\partial \varphi_1(x)}{\partial x} & \frac{\partial \varphi_1(x)}{\partial y} & \frac{\partial \varphi_1(x)}{\partial z} \\ \frac{\partial \varphi_2(x)}{\partial x} & \frac{\partial \varphi_2(x)}{\partial y} & \frac{\partial \varphi_2(x)}{\partial z} \\ \frac{\partial \varphi_3(x)}{\partial x} & \frac{\partial \varphi_3(x)}{\partial y} & \frac{\partial \varphi_3(x)}{\partial z} \end{pmatrix}$

3 Splines on the real line

In this problem, we will be considering one-dimensional "images" that are simply a series of Gaussians centered at "landmark points," t_i, given by $\sum_{i=1} \frac{1}{2\pi} \exp(-\frac{1}{2}(t - t_i)^2)$. The atlas image, $I_{atlas}(t)$ has landmarks $\{t_1 = 20, t_2 = 40, t_3 = 60, t_4 = 80\}$. The target image (and its landmarks $\{t'_1, t'_2, t'_3, t'_4\}$), is simply a leftward shift of the atlas $I_{target}(t) = I_{atlas}(t + s)$ for some s.

Our goal is to match the landmarks between the two images using splines, in particular, the spline associated with the Green's kernel $k(t - t_0) \doteq \frac{1}{2a} \exp(-a * |t - t_0|)$, where $a \in \mathbb{R}^+$.

Let the transformation vector field be given by $v(t) = \sum_{j=1}^{n} k(t - t_j)p_j$ where the p_j's are chosen such that $v(t_i) = t'_i - t_i$.

3.1

Plot the images $I_{target}(t)$, and $I_{atlas}(t)$ for $s = 2$. Note that if you are plotting in matlab, you will need to sample the variable `t` at a high resolution, such as `t = 0:0.01:100` (in Python, `t = numpy.linspace(0, 100, num=10000)`).

3.2

Calculate the required p_j's for $s = 1$ and $a = 0.5$.

Plot the transformed target image, $I_{target}(x + v(x))$, on top of $I_{target}(t)$, and $I_{atlas}(t)$. The transformed target should take the same values as the atlas at all t_is. Also plot $v(t)$ vs. t. Would you say the landmark matching was successful?

3.3

Calculate the required p_j's for $s = 10$ and $a = 0.5$.

Plot the transformed target image, $I_{target}(x + v(x))$, on top of $I_{target}(t)$, and $I_{atlas}(t)$. Also plot $v(t)$ vs. t. Does the transformed target still resemble the original target?

3.4

What is a qualitative difference between the $v(t)$ vs. t plots in Problems 3.2 and 3.3? Repeat problem 3.3 with $a = 0.1$, did things change?

4 Inverses and compatibility

Consider matrices A, B and vector x. From linear algebra we know that $(B \circ A) \cdot x = B \cdot (A \cdot x)$.

In words, "If you transform x with with the matrix $B \circ A$, you will get the same result as if you first transform x with A and then transform it with B."

This is called "compatibility" between the group of matrices, and their action on the vector x.

In computational anatomy we consider the action of transformations φ, ψ on images I. A transformation acts on an image according to $\varphi \cdot I = I \circ \varphi^{-1}$, and one transformation combines with another according to $\psi \circ \varphi(x) = \psi(\varphi(x))$.

Prove that $(\psi \circ \varphi) \cdot I = \psi \cdot (\varphi \cdot I)$. Show that this does not hold if the action of the transformation on an image did not include the inverse.

Hint: recall that $(\psi \circ \varphi)^{-1} = \varphi^{-1} \circ \psi^{-1}$.

5 Functions on the real line

5.1 Fourier transform of Convolution

The Fourier transform of a continuous function $f(t)$ is defined as $F(\omega) = \int_{-\infty}^{\infty} f(x)e^{-i\omega t}dx$. The convolution of two functions $f(x)$, $g(x)$ is defined as $f(x) * g(x) = \int_{\infty}^{\infty} f(\tau)g(x-\tau)d\tau$. Write the Fourier transform of $f(x) * g(x)$ in terms of the Fourier transforms of $f(x)$ and $g(x)$. Derive this result.

5.2 Convolution of Fourier transforms

Write the convolution of the Fourier transforms $(F(\omega) * G(\omega))$ in terms of the original functions, $f(x)$, and $g(x)$. Derive this result.

5.3 Convolution

Prove using convolution $\frac{1}{2}\exp(-|x|) = \exp(-x)u_s(x) * \exp(x)u_s(-x)$. Here u_s is the step function, $u_s(x) = 0$ for $x < 0$ and $u_s(x) = 1$ for $x > 0$.

5.4 Convolution via Fourier transforms

Prove using Fourier transforms $\frac{1}{2}\exp(-|x|) = \exp(-x)u_s(x) * \exp(x)u_s(-x)$.

Hints: Write $\frac{1}{2}\exp(-|x|) = \frac{1}{2}\exp(x)u_s(-x)\frac{1}{2}\exp(-x)u_s(x)$.

Recall that if $u(x)$ and $U(f)$ are Fourier transform pairs, then $u(-x)$ and $U(-f)$ are also Fourier transform pairs.

Last note that $\frac{1}{2}\left(\frac{1}{1+i2\pi f}\right) + \frac{1}{2}\left(\frac{1}{1-i2\pi f}\right) = \frac{1}{1+(2\pi f)^2}$.

6 Differential operators

Define $\langle f, g\rangle = \int_{-\infty}^{\infty} f(x)g(x)dx$.

Let $A = -\frac{\partial^2}{\partial x^2} + id$ and let $L = \frac{\partial}{\partial x} + id$.

Prove that $\langle Aw, w\rangle = \langle Lw, Lw\rangle$ for a function w that vanishes at infinity (i.e. $lim_{x\to\infty} w(x) = lim_{x\to-\infty} w(x) = 0$).

7 Calculating Linear Transformations and Jacobians

In this set of exercises you will calculate transformations to match one pair of landmarks to another, visualize the transformation as a grid, and calculate the Jacobian. You will demonstrate that that the model $\phi(x) = x + v(x)$ may result in transformations which do not have an inverse.

The assignment can be performed in MATLAB or Python. Python boilerplate code has been provided in the appendix. Please hand in any code you write, in addition to anything else the problems ask for.

7.1 Generate a grid

We will calculate the value of our transformations φ at each point on a grid. Generate a grid as in the previous homework.

```
% The location of each pixel
nX = 200; % number of columns
nY = 200; % number of rows
xj = 1 : nX; % x location of each column
yi = 1 : nY; % y location of each row
[xij,yij] = meshgrid(xj,yi); % x,y location of each pixel
```

7.2 Generate the landmarks

Start with a pair of landmarks at locations $(125, 100)$ and $(150, 50)$. Store each landmark as a column in the variable `X`.

For $\theta = 30$ degrees, generate a counterclockwise rotation matrix stored in the variable `R`.

Define a pair of target landmarks `Y` by rotating `X` by 30 degrees.

7.3 Plot the landmarks and grid

Show the landmarks `X` as a scatterplot in cyan. Show the landmarks `Y` in the same scatterplot in red. Draw an untransformed grid in the same plot as follows.

```
down = 10; % downsampling is important so you can see things clearly
xijdown = xij(1:down:end,1:down:end);
yijdown = yij(1:down:end,1:down:end);
% This is a trick to plot a grid easily.
% We actually plot a 3D surface,
% but view it directly from above so it looks 2D
surf(xijdown,yijdown,ones(size(xijdown)),'facecolor','none','edgecolor','k');
```

7.4 Calculate an optimal 2×2 matrix transformation

Find the $A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$, a 2×2 matrix, which brings `X` to `Y` by minimizing the error

$$E = \sum_{i=1}^{2} |AX(i) - Y(i)|^2 = \sum_{i=1}^{2}\sum_{j=1}^{2} |\sum_{k=1}^{2} a_{jk}X_k(i) - Y_j(i)|^2 \tag{1}$$

Minimize this error analytically. You should be able to derive a matrix equation whose solution gives the optimal A.

In matlab, numerically calculate A for these landmarks.

Note that you should already know what the optimal A is for this example (what is it?), use this to check your work.

Hint: Do you recognize this problem?

7.5 Plot the transformed landmarks and grid

Transform your landmarks `X` by left multiplying with `A`. Show the transformed landmarks, `AX`, as a scatterplot in blue. Show the landmarks `Y` in the same scatter plot in red. Draw a transformed grid in the same plot.

```
Axij = % ... you should implement this
Ayij = % ... you should implement this
Axijdown = Axij(1:down:end,1:down:end);
Ayijdown = Ayij(1:down:end,1:down:end);
surf(Axijdown,Ayijdown,ones(size(Axijdown)),'facecolor','none','edgecolor','k');
```

7.6 Calculate the Jacobian

Use MATLAB's `gradient` function (or Python's `numpy.gradient`) to calculate the Jacobian of this transformation (`Axij` and `Ayij`) and its determinant everywhere on the 200×200 grid. Visualize the Jacobian determinant as an image with a colorbar.

What do you expect the value of the Jacobian determinant to be? You should use this to check your work.

7.7 Calculate an optimal Gaussian kernel transformation

Chose the standard deviation $\sigma = 50$ for this exercise.

We calculate a displacment vector field of the form

$$\begin{aligned} v_1(x) &= \sum_{i=1}^{2} \exp\left(-\frac{1}{2\sigma^2}|x - X(i)|^2\right) p_1(i) \\ v_2(x) &= \sum_{i=1}^{2} \exp\left(-\frac{1}{2\sigma^2}|x - X(i)|^2\right) p_2(i) \end{aligned} \tag{2}$$

while satisfying boundary conditions.

The boundary conditions

$$\begin{aligned} v_1(X(1)) = Y_1(1) - X_1(1) &= \sum_{i=1}^{2} \exp\left(-\frac{1}{2\sigma^2}|X(1) - X(i)|^2\right) p_1(i) \\ v_1(X(2)) = Y_1(2) - X_1(2) &= \sum_{i=1}^{2} \exp\left(-\frac{1}{2\sigma^2}|X(2) - X(i)|^2\right) p_1(i) \end{aligned} \tag{3}$$

can be written as a 2×2 matrix vector equation for the x component of the transformation, $V_1 = \hat{K}P_1$, where $\hat{K}$ is a 2×2 matrix, and V_1 and P_1 are 2×1 vectors storing x components of v and p respectively. Write out this equation for the x component of the $p(i)$. You should solve it analytically, and computationally in matlab.

Do the same for the y component of $p(i)$ by writing the boundary conditions

$$\begin{aligned} v_2(X(1)) = Y_2(1) - X_2(1) &= \sum_{i=1}^{2} \exp\left(-\frac{1}{2\sigma^2}|X(1) - X(i)|^2\right) p_2(i) \\ v_2(X(2)) = Y_2(2) - X_2(2) &= \sum_{i=1}^{2} \exp\left(-\frac{1}{2\sigma^2}|X(2) - X(i)|^2\right) p_2(i) \end{aligned}$$

as a matrix equation.

7.8 Plot the transformed landmarks and grid

Transform your landmarks by adding $v(X(i))$ to them. Plot these in blue as a scatterplot. On the same plot, show `Y` as a scatterplot in red.

Calculate the transformation at every point on your grid.

```
% initialize to identity, we will add the displacment v
phix = xij;
phiy = yij;
for i = 1 : nY
    for j = 1 : nX
        % add the displacement for each p(k) in the sum
        for k = 1 : size(X,2) % number of landmarks
            Kij = % ... implement this, the kernel evaluated at (j,i) - X(k)
            phix(i,j) = phix(i,j) + % ... add the x component for p(k)
            phiy(i,j) = phiy(i,j) + % ... add the y component for p(k)
        end
    end
end
```

Visualize the deformed grid as above.

```
phixdown = phix(1:down:end,1:down:end);
phiydown = phiy(1:down:end,1:down:end);
surf(phixdown,phiydown,ones(size(phixdown)),'facecolor','none','edgecolor','k');
```

7.9 Calculate the Jacobian

Calculate the Jacobian of `phix` and `phiy` and its determinant as above. Visualize it as an image with a colorbar.

7.10 Repeat the exercise for $\theta = 45$ degrees

Describe what you notice about the deformed grid, the determinant of the Jacobian, and the invertibility of the transformation.

Appendix - Python Boilerplate

7.1

```
import numpy as np
# The location of each pixel
nX = 200 # number of columns
nY = 200 # number of rows
xj = range(0, nX) # x location of each column
yi = range(0, nY) # y location of each row
xij, yij = np.meshgrid(xj, yi) # x,y location of each pixel
```

7.3

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
```

```
down = 10 # downsampling is important so you can see things clearly
xijdown = xij[0:-1:down, 0:-1:down]
yijdown = yij[0:-1:down, 0:-1:down]

# This is a trick to plot a grid easily.
# We actually plot a 3D surface,
# but view it directly from above so it looks 2D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_wireframe(xijdown,yijdown,-1.*np.ones(xijdown.shape),color='k',alpha=0.5);
ax.view_init(azim=0,elev=90), ax.w_zaxis.line.set_lw(0.);
ax.set_zticks([]), ax.grid(False);
```

7.5

```
Axij = # ... you should implement this
Ayij = # ... you should implement this
Axijdown = Axij[0:-1:down, 0:-1:down]
Ayijdown = Ayij[0:-1:down, 0:-1:down]

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_wireframe(Axijdown,Ayijdown,-1.*np.ones(Axijdown.shape),color='k',alpha=0.5);
ax.view_init(azim=0, elev=90), ax.w_zaxis.line.set_lw(0.);
ax.set_zticks([]), ax.grid(False);
```

7.8

```
# initialize to identity, we will add the displacement v
phix = xij
phiy = yij
for i in range(0, nY):
    for j in range(0, nX):
    # add the displacement for each p(k) in the sum
        for k in range(0, X.shape[1]): # number of landmarks
            Kij = # ... implement this, the kernel evaluated at (j,i) - X(k)
            phix[i,j] = phix[i,j] + # ... add the x component for p(k)
            phiy[i,j] = phiy[i,j] + # ... add the y component for p(k)

# Visualize the deformed grid as above.
phixdown = phix[0:-1:down, 0:-1:down]
phiydown = phiy[0:-1:down, 0:-1:down]

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_wireframe(phixdown,phiydown,-1*np.ones(phixdown.shape),color='k',alpha=0.5);
ax.view_init(azim=0, elev=90), ax.w_zaxis.line.set_lw(0.);
ax.set_zticks([]), ax.grid(False);
```