1 All about Intensity (12 points)

The spatial beam profile of an ultrasonic source at 5 cm away from the source in water can be approximated as shown in the accompanied figure in the left figure. The intensity time waveform at \( x = 0 \) has been measured and is shown in the right figure. What are the \( I_{SATA} \), \( I_{SPTA} \), \( I_{SPTP} \), and \( I_{SATP} \)?

Assume the area is a circle and that the figure on the left showcases the normalized intensity over diameter of the circular surface.

Duty Cycle (DC) = \( \frac{2 \times 10^{-3} \, \text{msec}}{1 \, \text{msec}} = 2 \times 10^{-3} \)

\[ I_{SPTP} = 25 \, \text{W/cm}^2 \]
\[ I_{SPTA} = I_{SPPA} \times DC = 25 \times DC = 0.05 \, \text{W/cm}^2 = 50 \, \text{mW/cm}^2 \]
\[ I_{SATA} = \frac{\text{Power}}{\text{Area}} = \frac{\sum_i I_{SPTA} \times Area}{\text{Total Area}} = I_{SPTA} \left( \frac{0.5\pi0.4^2+0.5\pi0.2^2}{\pi0.4^2} \right) = 50 \, \text{mW/cm}^2(0.625) = \]
\[ I_{SATP} = 25 \, \text{W/cm}^2 \left( \frac{0.5\pi0.4^2+0.5\pi0.2^2}{\pi0.4^2} \right) = 25 \, \text{W/cm}^2(0.625) = 15.625 \, \text{W/cm}^2 \]
2 Acoustic Pendulum Equilibrium (9 points)

An ultrasound transducer transmits a continuous wave in the direction of a plate fixed on an axis. Consider the plate to be a perfect reflector. As a result of the wave, the plate rotates to an angle $\theta$. The plate has a mass $m$, surface area $A$, and sound velocity $c$.

(a) What causes the plate to rotate?

Radiation pressure

(b) What is the transmission intensity?

The weight of the pendulum decomposes into two components, one along the rope and one perpendicular to it: $mg\cos\theta$ and $mg\sin\theta$ respectively. The force imparted by the transducer also decomposes into two components, one along the rope, $F\sin\theta$ and one perpendicular to the plate, $F\cos\theta$. It is this component of force that balances the weight of the plate:

$$F\cos\theta = mg\sin\theta$$

$$P = \kappa \frac{I}{c}$$

for a perfect reflector, $\kappa = 2\cos^2\theta$. Modifying the above equation for force, we have:

$$F = PA = \kappa \frac{IA}{c}$$

$$\kappa IA \frac{c}{c} = mg \frac{\sin\theta}{\cos\theta} = mg \tan\theta$$

$$I = \frac{mgc \tan\theta}{\kappa A}$$

$$I = \frac{mgc \tan\theta}{2\cos^2\theta A}$$
3 Acoustic Water Fountain (9 points)

You are pointing an ultrasound transducer at the surface of a water bath and the height of the water column is 2 cm.

(a) What is the intensity of the beam?

\[ p = \rho g h = 1000 \text{ kg/m}^3 \times 9.8 \text{ m/s}^2 \times 0.02 \text{ m} = 196 \text{ N/m}^2 = 196 \text{ Pa} \]

\[ I = \frac{pc}{2} = \frac{196 \text{ Pa} \times 1480 \text{ m/s}}{2} = 145,040 \text{ W/m}^2 = 14.5 \text{ W/cm}^2 \]

(b) Is it safe to put your finger on it?

The ultrasound pulse is continuous in time, so \( I_{SPRTA} = 14.5 \text{ W/cm}^2 \). This is much higher than the threshold at which biological effects are observed (100 mW/cm²). Therefore, it is not safe to put your finger on it.

4 MR Thermometry (30 points)

Find the temperature image series. There are 3 images at each time-point: magnitude, real, and imaginary. There are 3 slices. Images 1-3 are slice 1, 4-6 are slice 2, 7-9 are slice 3. Then they repeat for 10 additional time-points. In MATLAB, you’ll use `dicomread` to read them in. The transducer is at the bottom of the image. It is not seen. There is a brighter phantom within a water bath, which is less bright. These images were acquired on a 3T MRI system, with an echo time TE of 16.1 ms. The total scan time is 3 s per slice, per time point.

(a) Reconstruct the phase difference images for slice 2 and then scale them to temperature change. Attach both the phase difference images and temperature change map to your homework.

You will notice that the phase image at the first time point is corrupted by some artifact. Skip the very first time point and compute the phase difference for each time point against the second time point. Therefore, you should have 9 images for phase differences, and 9 images for their corresponding temperature change maps.

(b) What is the artifact we see in the first time point?

(c) What is the shape of the heated area in the focal zone?

(d) Calculate a thermal dose map. Attach to your homework.

(e) Is a lethal thermal dose achieved?

(f) What is the temporal standard deviation of the temperature maps (in °C)? Look outside the hotspot to calculate this.
(g) How does it compare to the spatial standard deviation of the temperature maps (in °C)?

(h) What can degrade the temporal standard deviation?

(i) Plot the full width at half max for the phase plots per time point of slice 2. Comment the trend you see.
# HW8-MR thermometry

December 4, 2019

0.0.1 Load the data

```python
[2]:

```data_path = '/Volumes/GoogleDrive/My Drive/RAD 225/HW 8 /HW6TemperatureImages'
g = glob(data_path + '/*.dcm')
print("Total of %d DICOM images." % len(g))
```

Total of 99 DICOM images.

```python
[3]:

```# Loop over the image files and store everything into a list.
def load_scan(path):
slices = [dicom.read_file(path + '/' + s, force=True) for s in np.sort(os.listdir(path))]
return slices

```python
[4]:

```def get_pixels(scans):
    image = np.stack([scans[n].pixel_array[:, :] for n in range(len(scans))])
    return np.array(image, dtype=np.int16)

```python
[5]:

```scans = load_scan(data_path)
imgs = get_pixels(scans)
```
# parse through the slices and arrange them into magnitude, real, imaginary, # and phase components
slice_1_mag = np.asarray([imgs[j] for j in np.arange(0, 99, 9)])
slice_1_real = np.asarray([imgs[j] for j in np.arange(1, 99, 9)])
slice_1_imag = np.asarray([imgs[j] for j in np.arange(2, 99, 9)])
slice_1_phase = np.asarray([np.arctan2(slice_1_imag[j], slice_1_real[j]) for j in np.arange(0, 11, 1)])
slice_2_mag = np.asarray([imgs[j] for j in np.arange(3, 99, 9)])
slice_2_real = np.asarray([imgs[j] for j in np.arange(4, 99, 9)])
slice_2_imag = np.asarray([imgs[j] for j in np.arange(5, 99, 9)])
slice_2_phase = np.asarray([sp.arctan2(slice_2_imag[j], slice_2_real[j]) for j in np.arange(0, 11, 1)])
slice_3_mag = np.asarray([imgs[j] for j in np.arange(6, 99, 9)])
slice_3_real = np.asarray([imgs[j] for j in np.arange(7, 99, 9)])
slice_3_imag = np.asarray([imgs[j] for j in np.arange(8, 99, 9)])
slice_3_phase = np.asarray([sp.arctan2(slice_3_imag[j], slice_3_real[j]) for j in np.arange(0, 11, 1)])

0.0.2 Problem 4. a)

For this part, first you need to plot the phase differences between each time point and the 2\textsuperscript{nd} time point of slice 2. Then, you multiply these plots by a conversion factor to back out temperature difference maps:

0.0.3 Plot the phase differences:

between time points 1 and 10 (2 and 11 in MATLAB indexing)

```python
plt.title(r'$\Phi 6 - \Phi 2$ (with phase wrapping)')
plt.plot(((slice_2_phase[5] - slice_2_phase[1]))[125, :], label='Vertical View')
plt.plot(((slice_2_phase[5] - slice_2_phase[1]))[:, 125], label='Horizontal View')
plt.axvspan(90, 160, color='g', alpha=0.1)
plt.legend()
plt.show()
```
In the above figure, I have shown a trace of the phase difference between time points 6 and 2 (MATLAB indexing). Notice there is a lot of noise on the left and the right. When calculating the temperature difference, you want to crop out that noise. Also, in the region of interest, you will see sudden jumps in phase difference. Those correspond to $\pi/2$ or $-\pi/2$ angles. This effect is called **phase wrapping** and there are algorithms to correct for it. In MATLAB, there is an `unwrap` function that corrects for it. There is a Python equivalent of it in numpy library. To see the difference, take a look at the following figure:

```python
[8]: plt.title(r'$\Phi 6 - \Phi 2$')
plt.plot((np.unwrap(slice_2_phase[5] - slice_2_phase[1]))[125,:], label='Phase\_\_unwrapped')
plt.plot(((slice_2_phase[5] - slice_2_phase[1]))[125,:], label='Phase wrapped')
plt.axvspan(90, 160, color='g', alpha=0.1)
plt.legend()
plt.show()
```
You can see the two sharp peaks due to phase wrapping disappear. In the bottom figures, I am showing the raster plot for the phase-wrapped and phase-unwrapped cases:

```
[9]: plt.subplot(1, 2, 1)
   plt.title('Phase Unwrapped')

plt.subplot(1, 2, 2)
plt.title('Phase Wrapped')
plt.show()
```
Note the sharp black/white dots on the phase-wrapped raster plot. On the phase-unwrapped case we do not see those. Therefore, to compute the phase difference plots as well as temperature difference maps, I am going to use the phase unwrapped versions, while cropping out the noise region (they corrupt the phase unwrapping). I will use a pixel window of \([90 : 160, 90 : 160]\) throughout.

```
[10]:
a = 0.01
gamma = 2*pi*42.57
TE = 16.1e-3
B = 3
deltaT_factor = 1.0 / (a*gamma*B*TE)
```

```
[11]:
plt.figure(figsize=(9, 9))
for k in range(2, 11):
    plt.suptitle('Phase Difference Maps (MATLAB indexing)', fontsize=16)
    plt.subplot(3, 3, k-1)
    plt.title(r'$\Phi$ + str(k+1) + r' - $\Phi 2$')
    plt.imshow(np.unwrap((slice_2_phase[k] - slice_2_phase[1])
                     [90:160, 90:160]), extent=(90, 160, 90, 160), cmap='gray')
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```
0.0.4 Plot the temperature differences:

Here are the parameters needed in order to plot the temperature:

$\Delta \phi$, $\alpha$, $\gamma$, $B_0$, and $TE$.

$\Delta \phi$ is what we get by subtracting the phase values of the last time point and the second time point in slice 2. $\alpha = -0.01 \text{ ppm/°C}$, $\gamma = 267.52 \times 10^6 \text{ rad/s} \cdot T$, and $TE = 16.1 \times 10^{-3} \text{s}$.

Therefore, the conversion factor is:

$$\Delta T = \frac{\Delta \phi}{\alpha \gamma B_0 T E}$$
\[
\Delta T = \Delta \phi \frac{-1}{0.01 \times 10^{-6} \times 267.52 \times 10^6 \times 16.1 \times 10^{-3}}
\]

[12]:
```python
plt.figure(figsize=(9, 9))
for k in range(2, 11):
    plt.suptitle('Temperature Difference Maps (MATLAB indexing)', fontsize=16)
    plt.subplot(3, 3, k-1)
    plt.title(r'$\Phi$ + str(k+1) + r' ' - $\Phi$ 2')
    cbar = plt.colorbar(fraction=0.046, pad=0.05)
    cbar.set_label('$^\circ$C', size=15)
    plt.clim(0, 6)

plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```
\[ L = [\text{np.max}((\text{deltaT}\_\text{factor} \times \text{np.unwrap}((\text{slice}\_2\_\text{phase}[k] - \text{slice}\_2\_\text{phase}[1])[90:160, 90:160])) \text{ for } k \text{ in range}(2, 11))] \]

plt.title('Maximum $\Delta T$ per Time Point')
plt.scatter(np.arange(3, 12, 1), L)
plt.ylabel(r'\text{max } \Delta T$', size=12)
plt.xlabel('time point', size=12)
plt.text(6.5, 5.3, str(np.round(np.max(L), 2)) + '$^\circ C$', size=12)
plt.show()
Note that since the 2\textsuperscript{nd} time point is the reference that is subtracted from each time point, we are only showing $\Delta T$ for time points 3 to 11.

0.0.5 4. b) Ghosting.

0.0.6 4. c) Oval.

0.0.7 4. d) For this part, we need to use the CEM formula:

$$CEM_{43} = \int_{t=0}^{t=final} R^{(43-T)} dt$$

Since we are using a discrete case of 10 time points, we will use a discrete form of CEM:

$$CEM_{43} = \sum_{i=2}^{11} t_i R^{(43-T)}$$
The peak temperature difference we are achieving is $37 + 5.44 = 42.44^\circ C$. Therefore, the explicit equation to use is the following:

$$CEM_{43} = \sum_{i=2}^{11} t_i \left( \frac{1}{4} \right)^{(43-T_i)}$$

[15]:
```python
L = []
for k in range(2, 11):
    L.append(3 * (0.25) ** (43 - 37 - (deltaT_factor * np.
       unwrap((slice_2_phase[k] - slice_2_phase[1])[90:160, 90:160])))
L = np.asarray(L)
```

[16]:
```python
plt.figure(figsize=(9, 9))
for k in range(2, 11):
    plt.suptitle('Thermal Dose Maps (MATLAB indexing)', fontsize=16)
    plt.subplot(3, 3, k-1)
    plt.title(r'$\sum_{i=2}^{\{\text{\%s}\}} \% (\text{\str(k+1)}) +$
              r'$t_{\{\text{\str(k+1)}\}} \left( \frac{1}{4} \right)^{(43-T_{\{\text{\str(k+1)}\})}$' % (str(k+1)), pad=20)
    plt.imshow((sum(L[i] for i in range(0, k-1))),
               cmap='jet', extent=(90, 160, 90, 160))
    cbar = plt.colorbar(fraction=0.046, pad=0.05)
    cbar.set_label(r'$CEM_{43\circ}$', size=15)

plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()
```
0.0.8 4. e)  
To see whether a lethal dose is achieved, we can look at the maximum CEM at the last time point:

\[
\text{np.max}\left(\sum_{i=0}^{9} L[i] \text{ for } i \text{ in range}(0, 9)\right)
\]

The minimum threshold for lethal thermal dose is 240. We are well below 240. So, a lethal dose is not achieved.

0.0.9 4. f)  
To find the temporal standard deviation, you can just select a point outside the hotspot in one of the slices, and track its temperature values over time. For example, here I have chosen the [100,
100] pixel of slice 2 and recorded its temperature over the 11 time points:

```python
[18]:
F = []
for k in range(2, 11):
    F.append((deltaT_factor * (slice_2_phase[k] - slice_2_phase[1])[100, 100]))
F = np.asarray(F)

[19]:
plt.scatter(np.arange(3, 12, 1), F)
plt.ylabel(r'$^\circ$C', size=12)
plt.xlabel('time point')
plt.show()
```

![Graph showing temperature over time points](image)

```python
[20]:
print('Temporal standard deviation at point [100, 100]: ' + str(np.round(np.std(F), 4)))
```

```
Temporal standard deviation at point [100, 100]: 0.1172
```

0.0.10  4. g)

For this part, fix a time point and look at the standard deviation in a region outside the hot spot. But you have to make sure you are not collecting noise. For example, [90 : 160, 90 : 110] is a safe region that is both outside the noisy region and the hot spot.

print('Spatial standard deviation at region [90:160, 90:110]: '+str(np.round(np.std(G), 4)))

Spatial standard deviation at region [90:160, 90:110]: 0.2106
The temporal standard deviation is comparable to the spatial standard deviation.

\section*{4. h)
Motion}

\subsection*{4. i)

I = []
for k in range(2, 11):
    I.append((deltaT_factor * np.unwrap((slice_2_phase[k] - slice_2_phase[1])[90:160, 90:160])))
I = np.asarray(I)

plt.plot(I[1][42,:], label=r'\Phi 4 - \Phi 2', c='deepskyblue')
plt.plot(I[3][42,:], label=r'\Phi 6 - \Phi 2', c='darkorange')
plt.plot(I[8][42,:], label=r'\Phi 11 - \Phi 2', color='purple')
plt.axhline(y = np.max(I[1][42,:]) / 2, color='deepskyblue', ls=':')
plt.axhline(y = np.max(I[3][42,:]) / 2, color='darkorange', ls=': ')
plt.axhline(y = np.max(I[8][42,:]) / 2, color='purple', ls=':')
plt.legend()
plt.show()
The full width at half maximum (FWHM) starts narrow, but as the temperature increases, widens. Even as the temperature drops, the FWHM remains wide and doesn’t go back to its narrower initial width. This is due to temperature diffusion into the surrounding tissue over time.