Principles of MRI

Peder E. Z. Larson

Sep 12, 2026

CONTENTS

This is a book and associated simulations for learning the principles of magnetic resonance imaging (MRI). It is based on about a decade of teaching MRI to 1st year graduate students coming from a broad range of educational backgrounds (primarily natural sciences and engineering).

Learning Goals

Upon completing this book, the following Learning Goals should be achieved

1. Describe the 4 fundamental components of a MRI scan and why they are necessary
2. Understand what MRI is measuring
3. Describe the concepts of polarization, resonance, excitation, and relaxation
4. Describe how various types of MRI contrast are created
5. Describe how images are formed
6. Understand the most popular pulse sequences and their acronyms
7. Understand the constraints and tradeoffs in MRI
8. Manipulate MRI sequence parameters to improve performance
9. Identify artifacts and how to mitigate them
10. Manipulate and analyze MRI data

Getting Started

This book jumps pretty quickly into concepts, for a quick overview of MRI, here's my attempt to explain MRI in under 10 minutes:

How MRI Works (Explained In under 10 minutes)

Principles of MRI Course Lectures

You can also access a full playlist of my lectures

Principles of MRI Recorded Lectures

Using MRI-Education-Resources code

There is a set of some associated code that goes along with this book, it is available at the git repository https://github.com/LarsonLab/MRI-education-resources, and also the GitHub icon above. I recommend cloning the repository via the GitHub Desktop application, this way you get all the code at once and then can pull any updates made to the repository.

Acknowledgements

This book is greatly influenced by many other great resources that I want to acknowledge up front:

- *Principles of Magnetic Resonance Imaging* by Dwight Nishimura - I learned from this great text and borrowed a lot of its material. For some of my students with less engineering/signal processing background, this book on its own was challenging.
- Net magnetizations animation from Bill Overall and the SpinBench simulation and pulse sequence design tool
- Miki Lustig's MRI course notes
- Educational resources from the Danish Research Centre for Magnetic Resonance, which includes a great on-line Bloch Simulator

Part I

Magnetic Resonance Physics

CHAPTER

ONE

SPIN PHYSICS

To understand MRI, we must delve into physics and understand what is fundamentally being measured by MRI, which are nuclear "spins", and how they behave in a magnetic field which includes the resonance phenomenom that forms the basis of MRI. A key concept is the "Net Magnetization" $\vec{M}$ which is the sum of all the spins in a given volume. This chapter provides a higher level overview of these concepts, and a more rigorous mathematical description is provided in the Bloch Equation chapter.

1.1 Learning Goals

1. Understand what MRI is measuring
    - Define a nuclear spin
    - Define the concept of the net magnetization
2. Describe the concepts of polarization, resonance, excitation, and relaxation

1.2 Nuclear Spin

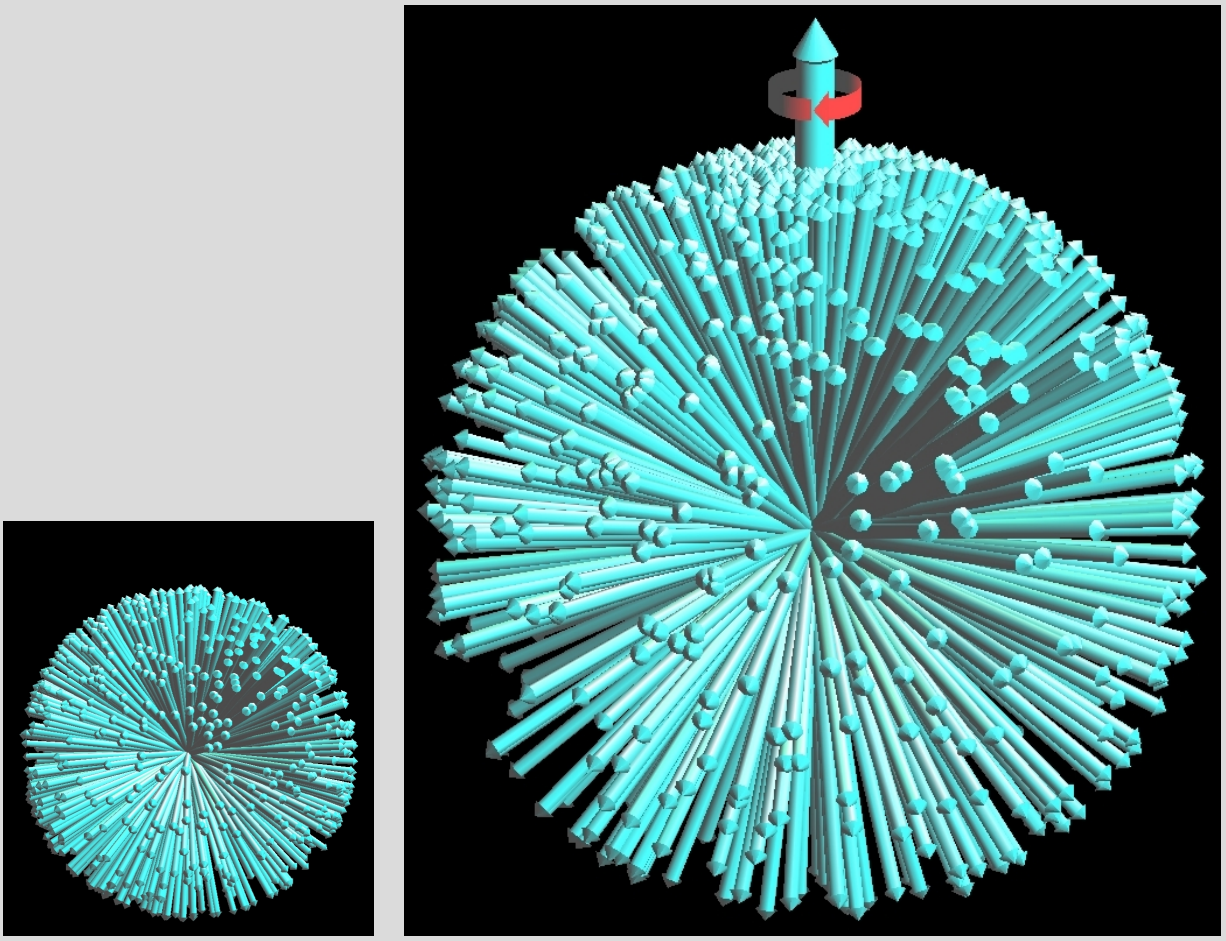
A nuclear spin is defined as a nucleus with a non-zero net angular momentum. This means that the nucleus has a net magnetic moment, μ, meaning it can act as a small magnet. It also experiences magnetic resonance, which means it will respond to a specific frequency of electromagnetic fields. This resonance is the basis for the MRI signal.

In MRI, we are tyipcally imaging hydrogen nuclei ($^1 H$) which are about 80% of the atoms in our bodies. The primary source of $^1 H$, also referred to as protons, is water, although protons in lipids are also imaged.

Other nuclei also have spin, including $^{13}C, ^{19}F, ^{23}Na, ^{31}P$, but these are much less abundant and thus have less available signal compared to protons.

1.3 Polarization and the Net Magnetization

In the absence of a magnetic field, nuclear spins will be randomly aligned. However when spins experience a magnetic field, they will preferentially align with that magnetic field. In MRI, we cannot measure individual spins, but we observe the ensemble of spins, which is is known as the Net Magnetization, $\vec{M}(\vec{r},t)$.



Distribution of spins in the absense of a magnetic field

Distribution of spins after application of a strong magnetic field that polarizes the spins as reflected in increased density of arrows in the direction of the field (here vertical). The large arrow pointing up represents the sum of all these spins, or the Net Magnetization.

Source https://www.drcmr.dk/mr

1.3.1 Equilibrium Magnetization

In MRI, we define the main magnetic field, B_0, to be oriented along the z-axis. When a subject is placed in a MRI scanner, the equilibrium magnetic field is

$$\vec{B}(\vec{r},0) = \begin{bmatrix} 0 \ 0 \ B_0 \end{bmatrix}$$

Under this condition of equilibrium, the net magnetization is oriented in the direction of the main magnetic field

$$\vec{M}(\vec{r},0) = \begin{bmatrix} 0 \ 0 \ M_0(\vec{r}) \end{bmatrix}$$

And has an amplitude of that is defined as the Equilibrium Magnetization:

$$M_0(\vec{r}) = \frac{N(\vec{r}) \bar{\gamma}^2 h^2 I_Z (I_Z +1) B_0}{3 k T}$$

where $N(\vec{r})$ is the spin density, h is Planck's constant (6.63e-34 J/Hz), $\bar{\gamma}$ is the gyromagnetic ratio (42.58e6 Hz/T for $^1 H$), B_0 is the magnetic field, I_Z is the spin number (1/2 for hydrogen), k is Boltzmann's constant (8.62e-5 eV/K or 1.38e-23 J/K), and T is temperature (human body temperature = 310 K).

1.3.2 Fractional Polarization

Another helpful formulation to describe the preference of spins to align with the magnetic field is to look at the fraction of spins that, if observed, would be aligned with the magnetic field. This can be done using the Boltzman distribution that describes the ratio of spins in the spin-up, $N_\uparrow$, to the spin-down, N_-, states, shown here for spin 1/2 nuclei ($I_Z = 1/2$):

$$\frac{N_\uparrow}{N_\downarrow} = e^{h \bar{\gamma} B_0 / kT}$$

This includes the separation between the two energy states, $\Delta E = h \bar{\gamma} B_0$. It can also be used to derive the equilibrium magnetization above.

From this distribution, a fractional polarization can be defined as the difference between spin up and spin down states, divided by the total number of spins, which describes the fraction of spins that contribute to the MR signal:

$$ \frac{N_\uparrow - N_\downarrow}{N_\uparrow + N_\downarrow} = \frac{ 1 - e^{-h \bar{\gamma} B_0 / kT} }{ 1 + e^{-h \bar{\gamma} B_0 / kT} }$$

If the exponent is small (or, when the polarization is low), this simplifies to

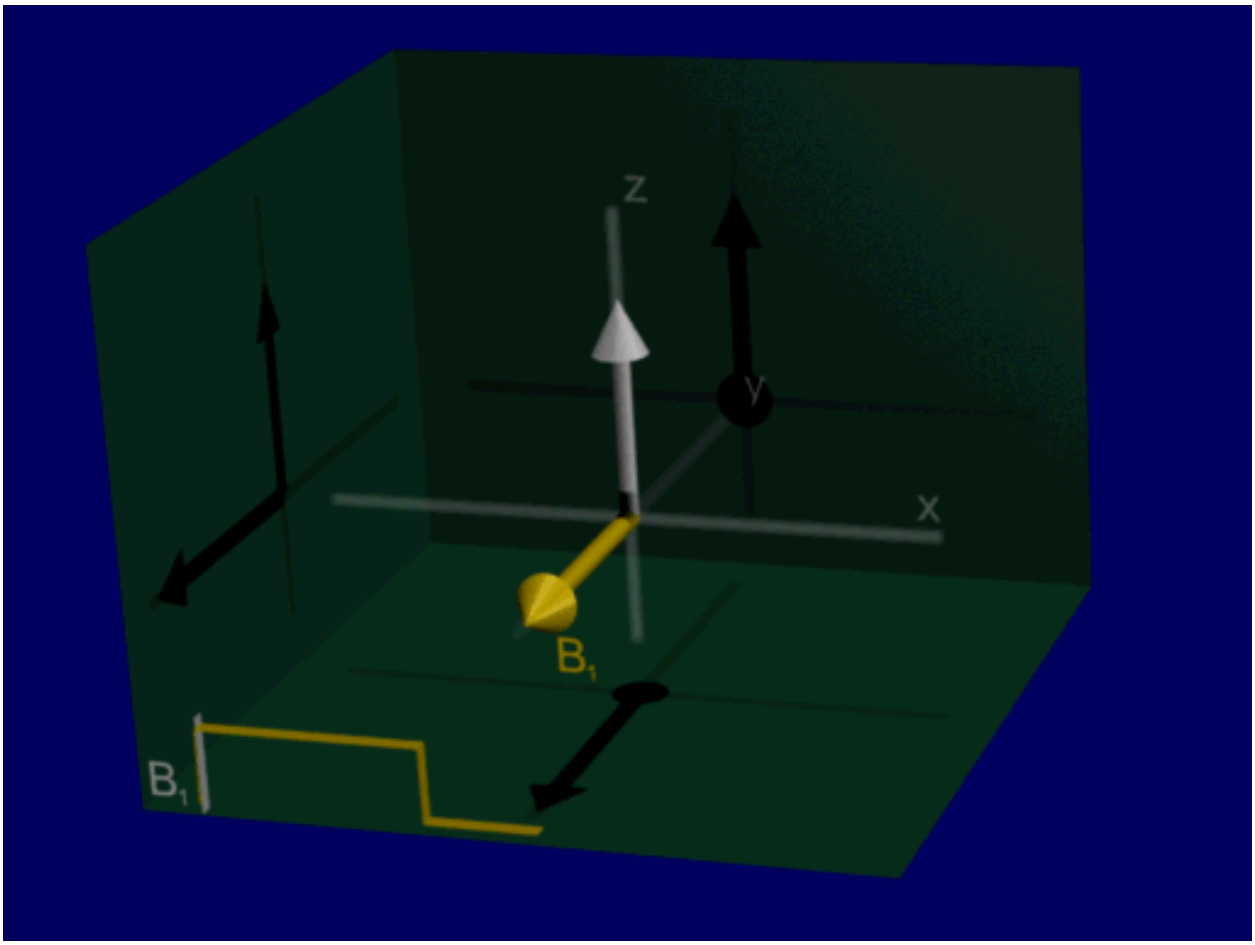
$$ \frac{N_\uparrow - N_\downarrow}{N_\uparrow + N_\downarrow} \approx \frac{h \bar{\gamma} B_0}{2 kT}$$

1.4 Resonance

When in a mangetic field, spins and therefore the net magnetization experience magnetic resonance, which means they will respond strongly to a specific frequency of electromagnetic fields.

An analogy for resonance is to consider a swing: In order to swing higher and higher, you must move your legs back and forth at the right rate. If you move your legs at the wrong rate, you will simply not move much higher.

In MRI, we can perturb the net magnetization from equilibrium by applying a time-varying magnetic field (B_1) at the “resonance frequency”. This will perturb or “tip” the net magnetization away its equilibrium where it is in alignment with the magnetic field.



1.4.1 Resonance Frequency

The magnetic resonance frequency is proportional to the magnetic field amplitude

$$f = \bar{\gamma} |\vec{B}|$$

When the magnetic field is simply our main field, B_0, then we get the so-called Larmor frequency

$$f_0 = \bar{\gamma} B_0$$

1.4.2 Simulation of Resonance

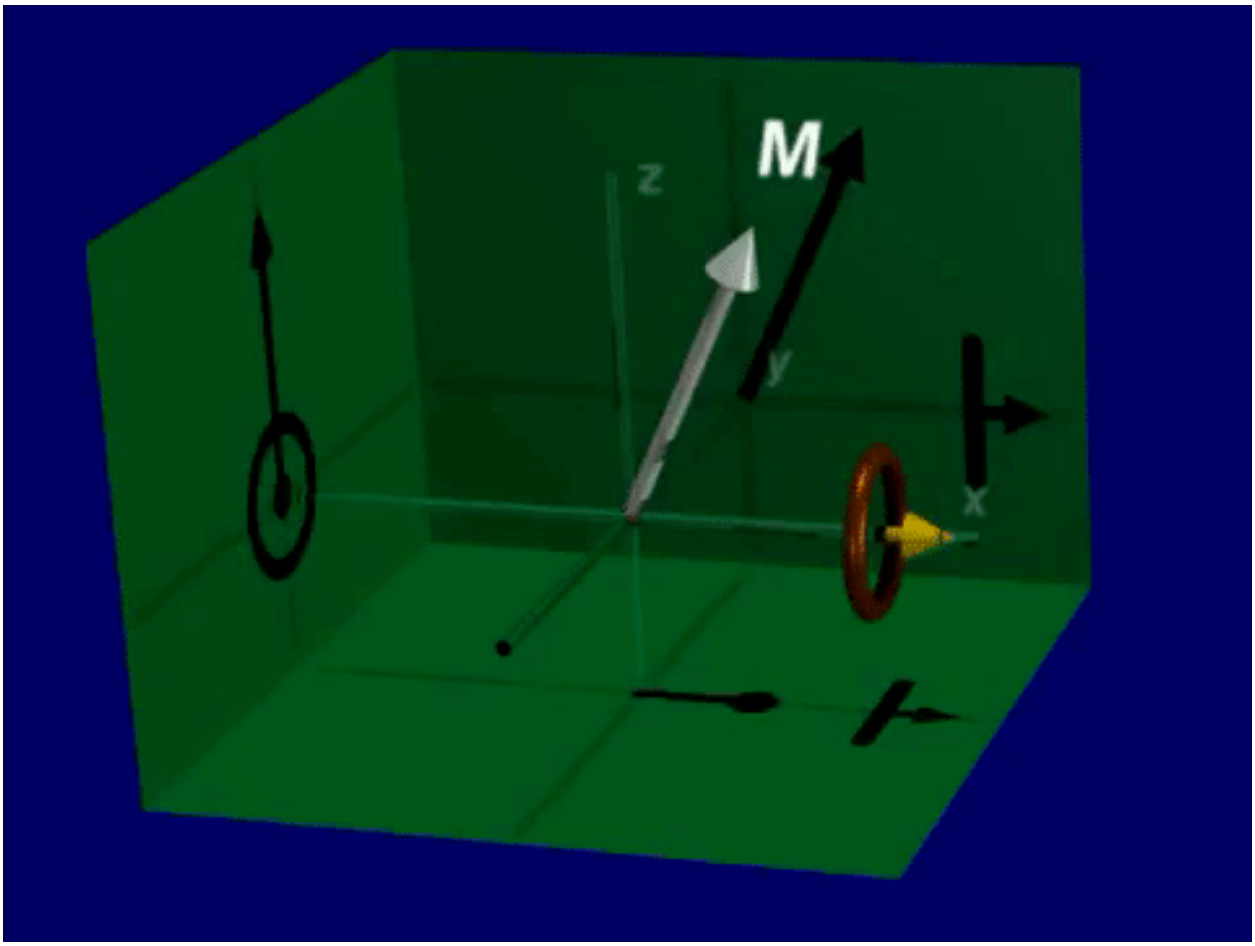
For this simulation use the CompassMR simulator for visualizing resonance. To simulate

1. Start in the default compass mode, with an applied B_0 field(default)
2. Move the B_1 magnetic field slider - this should change the angle of the compass slightly
3. Now adjust the B_1 freq(uency) slider - if you choose the correct frequency, then you should be able to create large oscillations in the compass
4. Change the main field, B_0, and see that now you need a new B_1 freq(uency) to achieve resonance
5. Switch to using the Coil for B_1 in the top right checkbox and repeat. This is one step closer to a real experiment.

1.5 Excitation and Precession

1.5.1 Precession

Precession refers to the rotation of the spins and therefore the net magnetization around the magnetic field. This precession is part of the magnetic resonance phenomenom and is what creates signal in MRI. The spins and net magnetization will rotate around the magnetic field, with the resonance frequency.



1.5.2 Excitation

In order for precession to occur, the spins or net magnetization must not be aligned with the magnetic field. As the nuclear spins are only preferentially aligned with the magnetic field, many of them will experience precession at any given time. However, the net magnetization will be aligned with the magnetic field at equilibrium, and therefore will not precess.

To cause the net magnetization to precess, an time-varying magnetic field (B_1) is applied at the resonance frequency. This perturbs the net magnetization away from alignment with the magnetic field, and it will continue to precess after the B_1 field is turned off.

Once we have precession of the net magnetization, it creates a time-varying magnetic field that can then be detected as MRI signal.

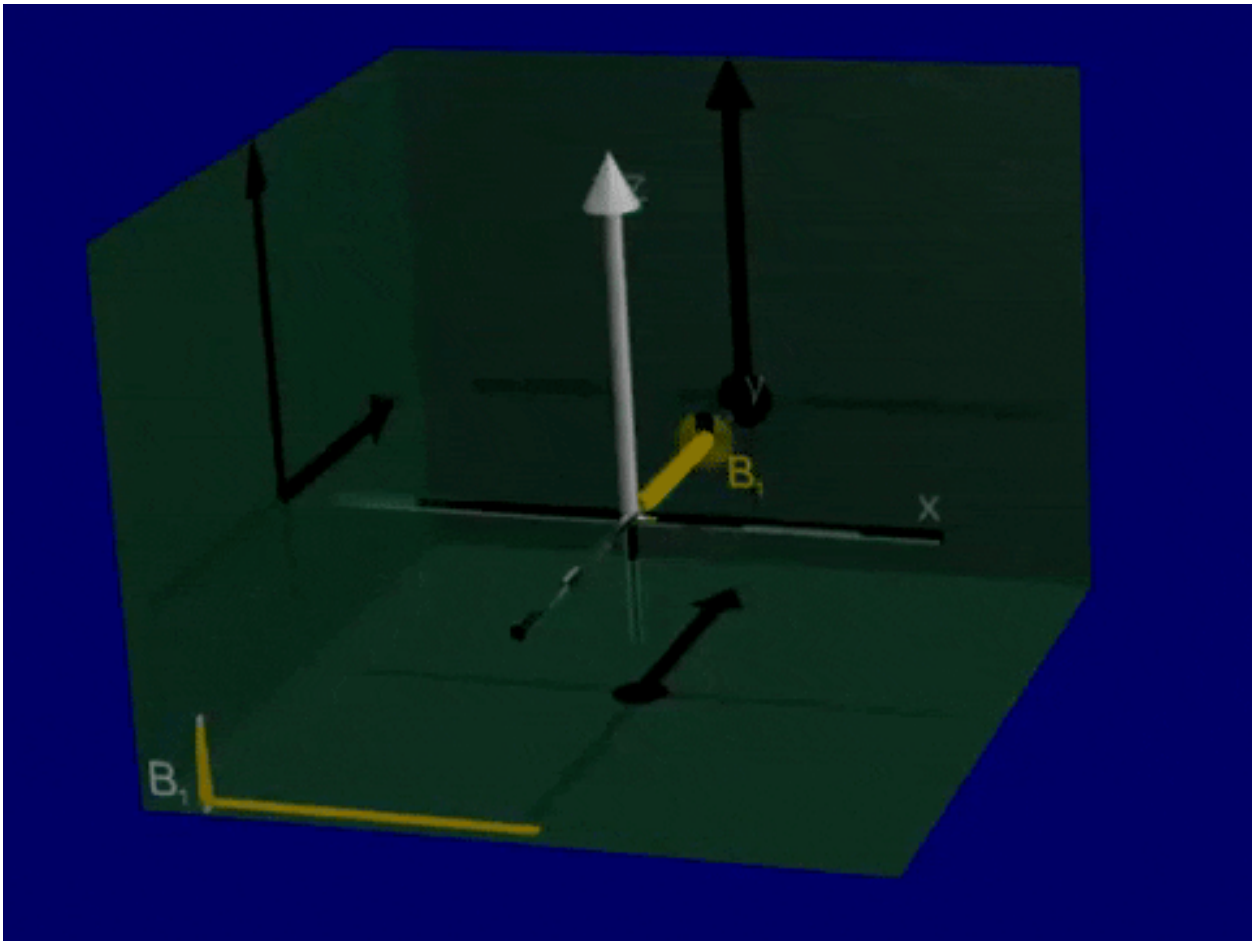
1.5.3 Simulation of Excitation and Precession

Again, use the CompassMR simulator for visualizing resonance and precession.

1. Start in the default compass mode again
2. Hit the Push button in the top right. Notice how the needle now precesses at a specific frequency
3. Switch on using the Coil for B_1 in the top right checkbox, and hit Push again. Now the blue trace highlights the signal created in this coil by the precessing needle. This is because the needle will also create a time-varying magnetic field that can be detected
4. Now switch to the Spin mode in the top right checkbox, and Push the spin. It will now precess in 3D around the up-down direction of B_0

1.6 Relaxation

Relaxation refers to the process by which the net magnetization, our ensemble of individual spins, returns to thermal equilibrium. This is governed by 2 time constants, $T_1(\vec{r})$ - the longitudinal (M_Z) or spin-lattice relaxation time constant, and $T_2(\vec{r})$ - the transverse (M_{XY}) or spin-spin relaxation time constant. These time constants differ depending on tissue composition and structure, and both of these relaxation time constants are valuable sources of MRI contrast.



1.6.1 Simulation of Relaxation

One last time in this chapter, use the CompassMR simulator.

1. Start in the default compass mode
2. Hit the Push button in the top right. The needle will begin to precesses but then returns to alignment with the B_0 magnetic field (in the vertical direction). This return to equilibrium is relaxation

CHAPTER

TWO

BLOCH EQUATION

The Bloch Equation, named after physicist Felix Bloch who won a Nobel Prize for discovery of magnetic resonance phenomena, describes almost all behavior seen in MRI. Here, we will describe this equation and derive key results to use for explaining the behavior of the net magnetization and its interaction with magnetic fields.

2.1 Learning Goals

1. Understand what MRI is measuring
    - Mathematically define the net magnetization
2. Describe the concepts of polarization, resonance, excitation, and relaxation
    - Describe the rotation of the net magnetization due to interaction with magnetic fields
    - Understand how to use the rotating frame to simplify descriptions of magnetic resonance
    - Describe the process of relaxation

2.2 THE Bloch Equation

The Bloch equation describes the behavior of the net magnetization, $\vec{M}(\vec{r},t)$, in a magnetic field, $\vec{B}(\vec{r},t)$, and can describe the vast majority of all MRI phenomena. While the complete Bloch Equation is presented here, we can typically use results derived from this equation to analyze and design MRI experiments.

$$ \frac{d\vec{M}(\vec{r},t)}{dt} = \gamma \vec{M}(\vec{r},t) \times \vec{B}(\vec{r},t) + \begin{bmatrix} -1/T_2(\vec{r})& 0 & 0 \ 0 & -1/T_2(\vec{r}) & 0 \ 0 & 0 & -1/T_1(\vec{r}) \end{bmatrix} \vec{M}(\vec{r},t) + \begin{bmatrix} 0 \ 0 \ M_0(\vec{r})/T_1(\vec{r}) \end{bmatrix} $$

The beavhior depends on constants of:

- $M_0(\vec{r})$ - the equilibrium magnetization
- $T_1(\vec{r})$ - the longitudinal (M_Z) or spin-lattice relaxation time constant
- $T_2(\vec{r})$ - the transverse (M_{XY}) or spin-spin relaxation time constant

All of which can vary across our subject (and all a valuable source of contrast!).

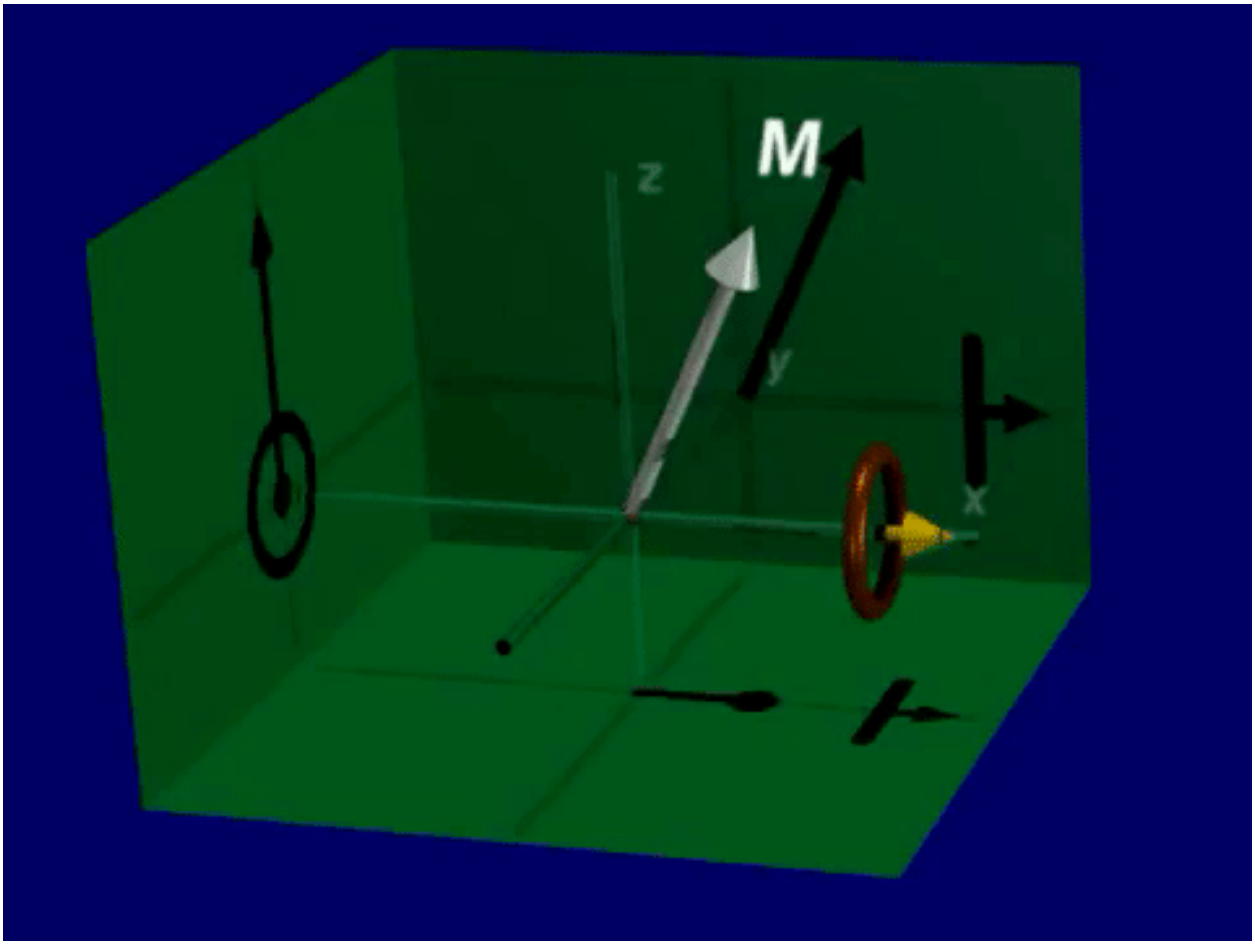
There is a very full-featured, interactive Bloch Equation Simulator available online, that is valuable to understand the behavior of the net magnetization:
Bloch Equation Simulator

2.3 Precession

The first term in the Bloch equation means that the net magnetization, $\vec{M}$ rotates around the magnetic field, $\vec{B}$, with a left-handed rotation. This rotation is sometimes referred to as precession.

To observe this, we neglect relaxation ($T_1, T_2 \rightarrow \infty$):

$$ \frac{d\vec{M}(\vec{r},t)}{dt} = \gamma \vec{M}(\vec{r},t) \times \vec{B}(\vec{r},t) $$



2.3.1 Simulation of Precession

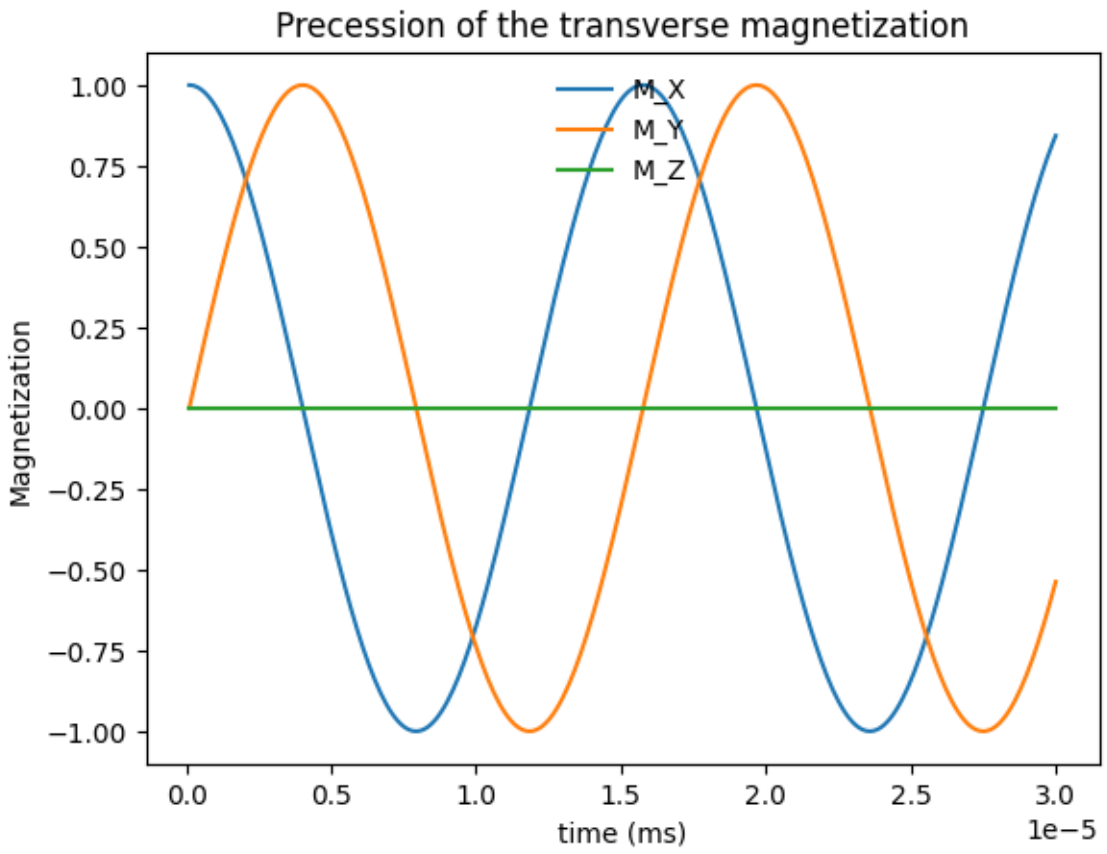
Open up the Bloch Equation Simulator. You will see a visualization of a net magnetization vector that is precessing around the magnetic field (thin line).

```
import numpy as np
import matplotlib.pyplot as plt

# Parameters
B0 = 1.5e3  # 1.5 T = 1500 mT
dt = 0.1e-6  # 0.1 ns = 0.1e-6 ms
N = 300  # Number of time steps
t = np.arange(1, N + 1) * dt  # Time vector in ms
```

(continues on next page)

(continued from previous page)

```
# Initial magnetization
Mstart = np.array([1, 0, 0])  # After RF excitation

# Static magnetic field
Bstatic = np.array([0, 0, B0])  # Magnetic field in mT

# Placeholder for magnetization over time
Mall = np.zeros((3, N))
Mall[:, 0] = Mstart

# Define the Bloch rotation function
def bloch_rotate(M, dt, B):
    gamma = 42.58  # Gyromagnetic ratio in kHz/mT
    omega = gamma * np.linalg.norm(B) * 2 * np.pi  # Angular frequency in radians/ms
    theta = omega * dt  # Rotation angle
    if np.linalg.norm(B) == 0:
        return M  # No rotation if B is zero
    axis = B / np.linalg.norm(B)  # Rotation axis
    cos_theta = np.cos(theta)
    sin_theta = np.sin(theta)
    R = np.array([
        [cos_theta + axis[0]**2 * (1 - cos_theta), axis[0] * axis[1] * (1 - cos_
 ↪theta) - axis[2] * sin_theta, axis[0] * axis[2] * (1 - cos_theta) + axis[1] * sin_
 ↪theta],
        [axis[1] * axis[0] * (1 - cos_theta) + axis[2] * sin_theta, cos_theta +␣
 ↪axis[1]**2 * (1 - cos_theta), axis[1] * axis[2] * (1 - cos_theta) - axis[0] * sin_
 ↪theta],
        [axis[2] * axis[0] * (1 - cos_theta) - axis[1] * sin_theta, axis[2] * axis[1]␣
 ↪* (1 - cos_theta) + axis[0] * sin_theta, cos_theta + axis[2]**2 * (1 - cos_theta)]
    ])
    return R @ M

# Simulate magnetization over time
for It in range(N - 1):
    Mall[:, It + 1] = bloch_rotate(Mall[:, It], dt, Bstatic)

# Plot the results
plt.plot(t, Mall[0, :], label='M_X')
plt.plot(t, Mall[1, :], label='M_Y')
plt.plot(t, Mall[2, :], label='M_Z')
plt.xlabel('time (ms)')
plt.ylabel('Magnetization')
plt.legend(loc='upper center', frameon=False)
plt.title('Precession of the transverse magnetization')
plt.show()
```



2.4 RF Excitation

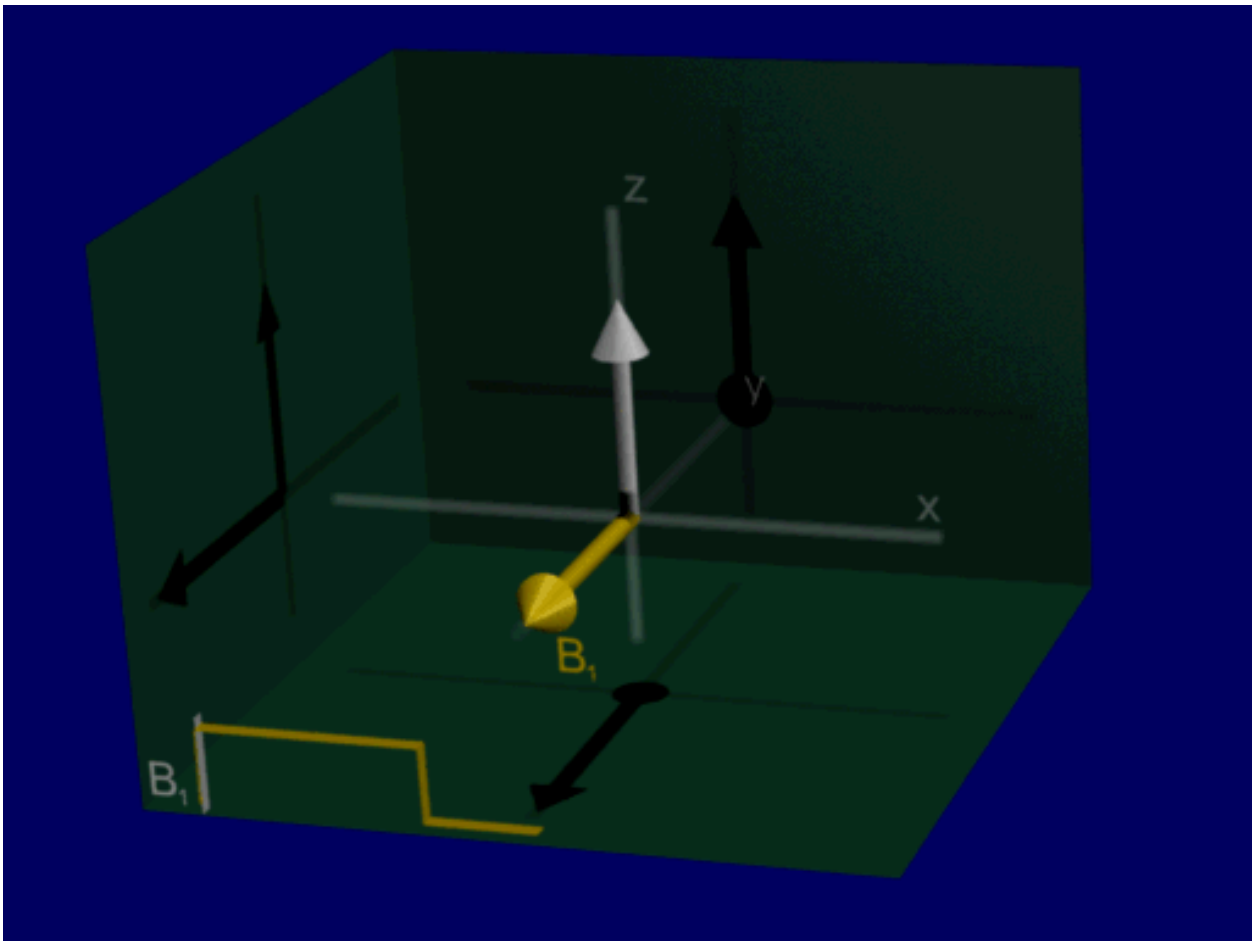
RF excitation occurs when an oscillating magnetic field is applied orthogonally to the main magnetic field. If we apply RF at the Larmor frequency, the magnetic field would be

$$\vec{B}(\vec{r},t) = \begin{bmatrix} B_{1}^+(\vec{r},t) \cos(2\pi\bar{\gamma} B_0 t) \ B_{1}^+(\vec{r},t) \sin(2\pi\bar{\gamma} B_0 t) \ B_0 \end{bmatrix}$$

When applied at the Larmor frequency, the applied RF energy and the net magnetization are in resonance, and we achieve excitation.

2.4.1 Example: Constant amplitude pulse

A good first RF excitation pulse to consider is a constant amplitude RF pulse, $B_{1}^+(\vec{r},t) = B_{1,0}$. When this is applied on resonance, the Bloch equation results in a progressive rotation of the net magnetization away from the z-axis, as illustrated below:



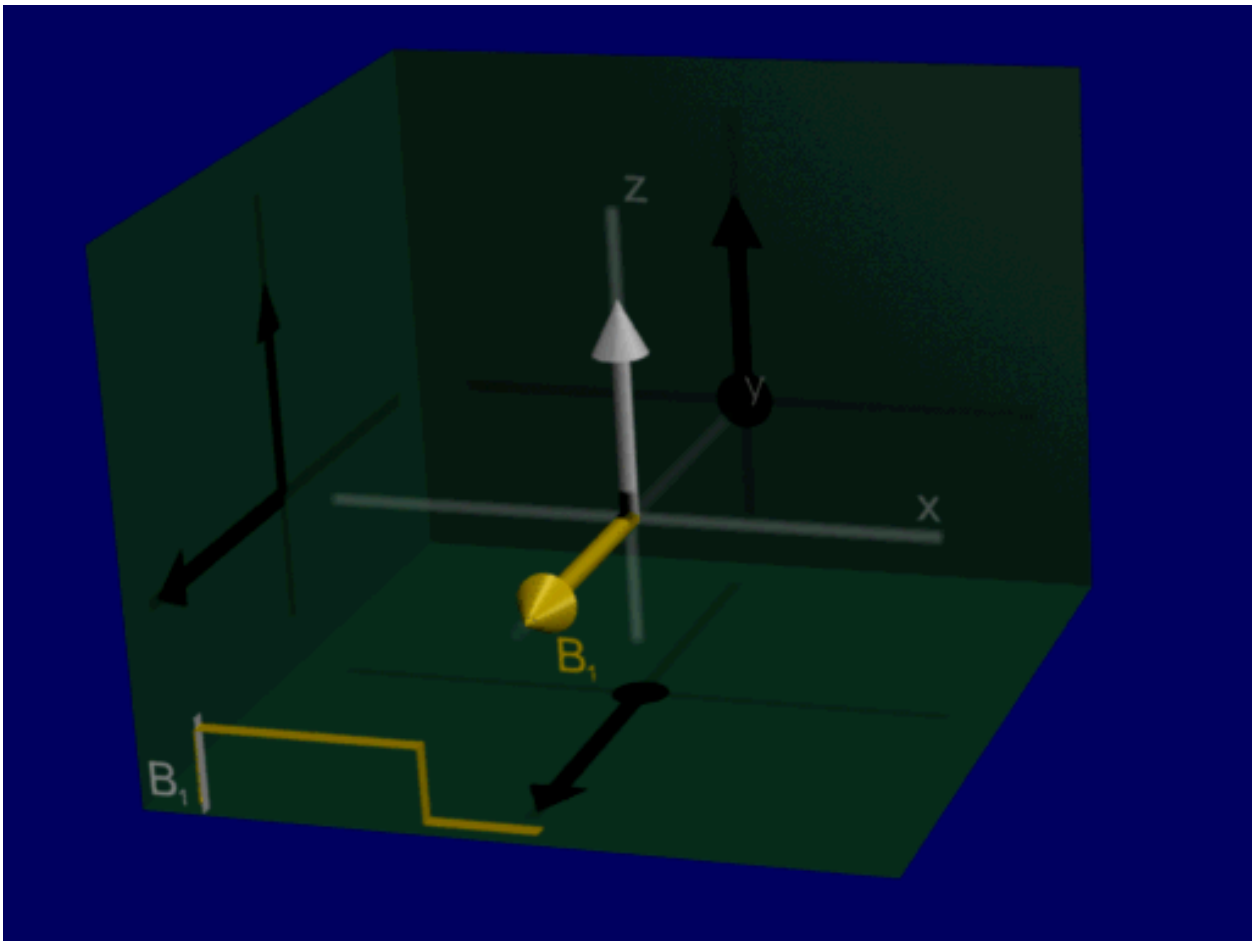
2.4.2 Lab versus Rotating Frame

For simplification, we use the so-called “rotating frame” - a reference frame that is rotating around the main field (z-axis) at the Larmor frequency, $f_0 = \bar{\gamma} B_0$, which greatly simplifies the interpretation, visualization, and math when analyzing the Bloch equation. This is in contrast to the “lab frame” or “stationary frame”.

In the rotating frame, the magnetic field vector is transformed to both have the rotation of the RF pulse terms removed. In this transformation of reference frames, the main magnetic field, B_0, is also removed. This change in the representation removes the precession of $\vec{M}$ around the z-axis, as we should expect in the rotating frame.

$$\vec{B}(t) = \begin{bmatrix} B_{1,X}^+(\vec{r},t) \ B_{1,Y}^+(\vec{r},t) \ 0 \end{bmatrix}$$

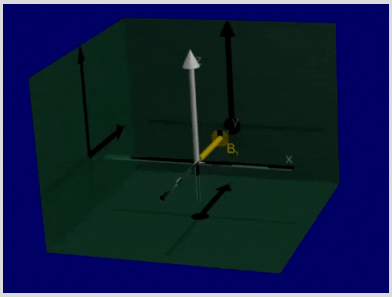
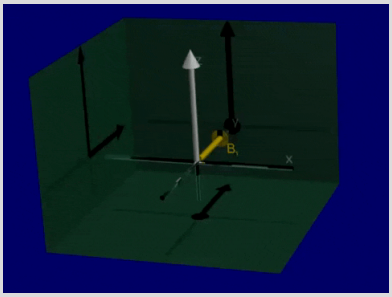
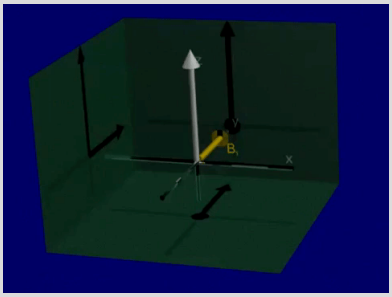
Now we can analyze RF excitation as a rotation around RF magnetic field amplitude and do not have to include Larmor frequency in our analysis, shown in the illustration below:



2.4.3 Common flip angle RF excitations

For a constant amplitude RF pulse, the flip angle depends on the duration of the RF pulse, T_{rf} and the strength of the RF magnetic field, $b_{1,0}$:

$$\theta = \gamma b_{1,0} T_{rf} $$

		
45-degree flip	90-degree flip	180-degree flip

2.4.4 Simulations of RF Excitation

Again, open up the Bloch Equation Simulator.

1. Lab versus rotating frame: The default view is in the lab (stationary) frame. If you select the 'B0' option from the 'Frame' in the top left it will change to the rotating frame.
2. RF Excitation: Change to 'Equilibrium' scene in bottom left. Then, use the '90x hard' button to apply a constant amplitude pulse.
3. Other flip angles: Go back to 'Equilibrium' scene, and try the other hard RF pulse flip angles.

Below are additional Bloch equation simulations and associated code for RF excitation show non-resonant magnetic fields, resonant magnetic fields, and excitation in the rotating frame.

First, when a non-resonant magnetic field is applied. An additional magnetic field is applied orthogonal to the main magnetic field, but *not* applied at the Larmor frequency, and there is no creation of transverse magnetization.

To achieve excitation the RF pulse is applied at the Larmor frequency, $\omega_0 = \gamma B_0$. With a resonant RF pulse, we have excitation of the net magnetization away from the direction of the main magnetic field, and creation of transverse magnetization, M_X and M_Y.

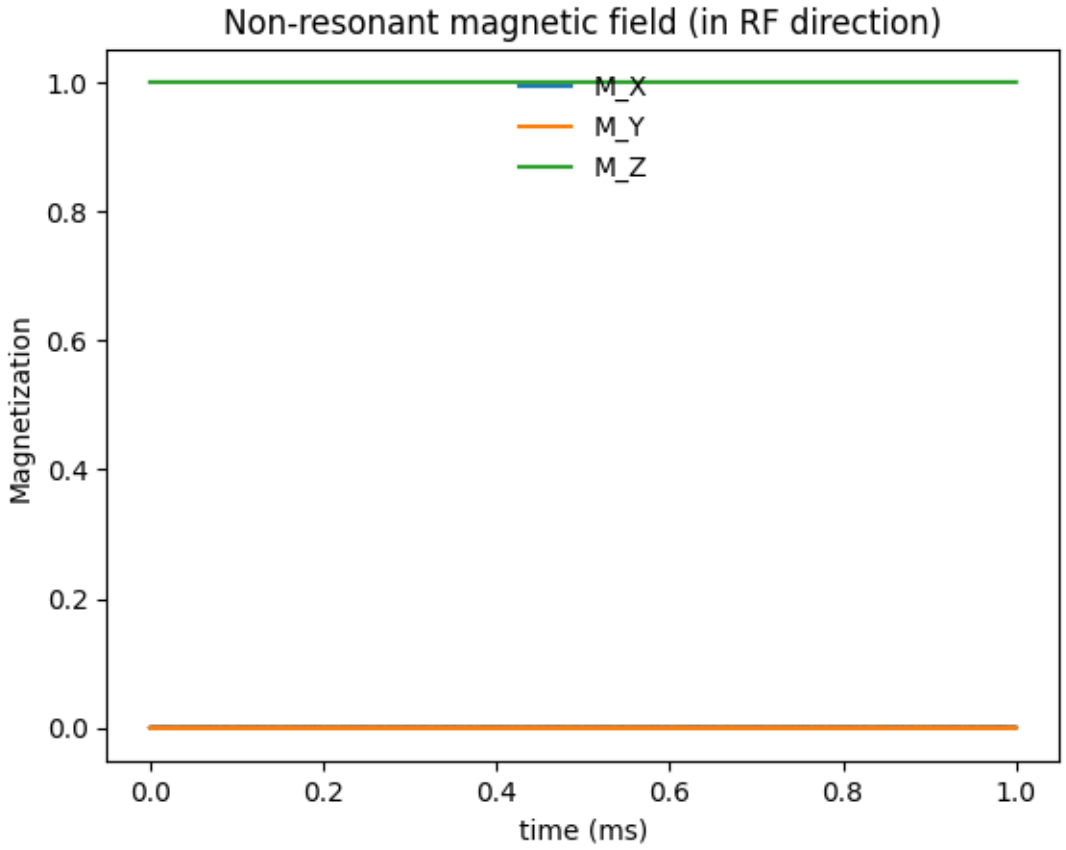
Finally, the simulation is converted into the rotating frame. It is hard to visualize the transverse magnetization in the lab because it is rotating at the Larmor frequency. The excitation is more clearly visualized in the rotating frame.

```
import numpy as np
import matplotlib.pyplot as plt

# Parameters
gammabar = 42.58  # kHz/mT
B0 = 10  # 10 mT for simplicity to visualize rotation
f0 = gammabar * B0  # kHz

M0 = 1.0
M_equilibrium = np.array([0, 0, M0])  # Initial magnetization at equilibrium

# RF pulse parameters
T_RF = 1  # ms
t = np.linspace(0, T_RF, 4000)  # Time vector

RF_flip_angle = np.pi / 2  # radians
B10 = RF_flip_angle / (2 * np.pi * gammabar * T_RF)  # mT

# Define the Bloch rotation function
def bloch_rotate(M, dt, B):
    gammabar = 42.58  # kHz/mT
    omega = gammabar * np.linalg.norm(B) * 2 * np.pi  # Angular frequency in radians/
 ↪ms
    theta = omega * dt  # Rotation angle
    if np.linalg.norm(B) == 0:
        return M  # No rotation if B is zero
    axis = B / np.linalg.norm(B)  # Rotation axis
    cos_theta = np.cos(theta)
    sin_theta = np.sin(theta)
    R = np.array([
        [cos_theta + axis[0]**2 * (1 - cos_theta), axis[0] * axis[1] * (1 - cos_
 ↪theta) - axis[2] * sin_theta, axis[0] * axis[2] * (1 - cos_theta) + axis[1] * sin_
 ↪theta],
        [axis[1] * axis[0] * (1 - cos_theta) + axis[2] * sin_theta, cos_theta +␣
 ↪axis[1]**2 * (1 - cos_theta), axis[1] * axis[2] * (1 - cos_theta) - axis[0] * sin_
```

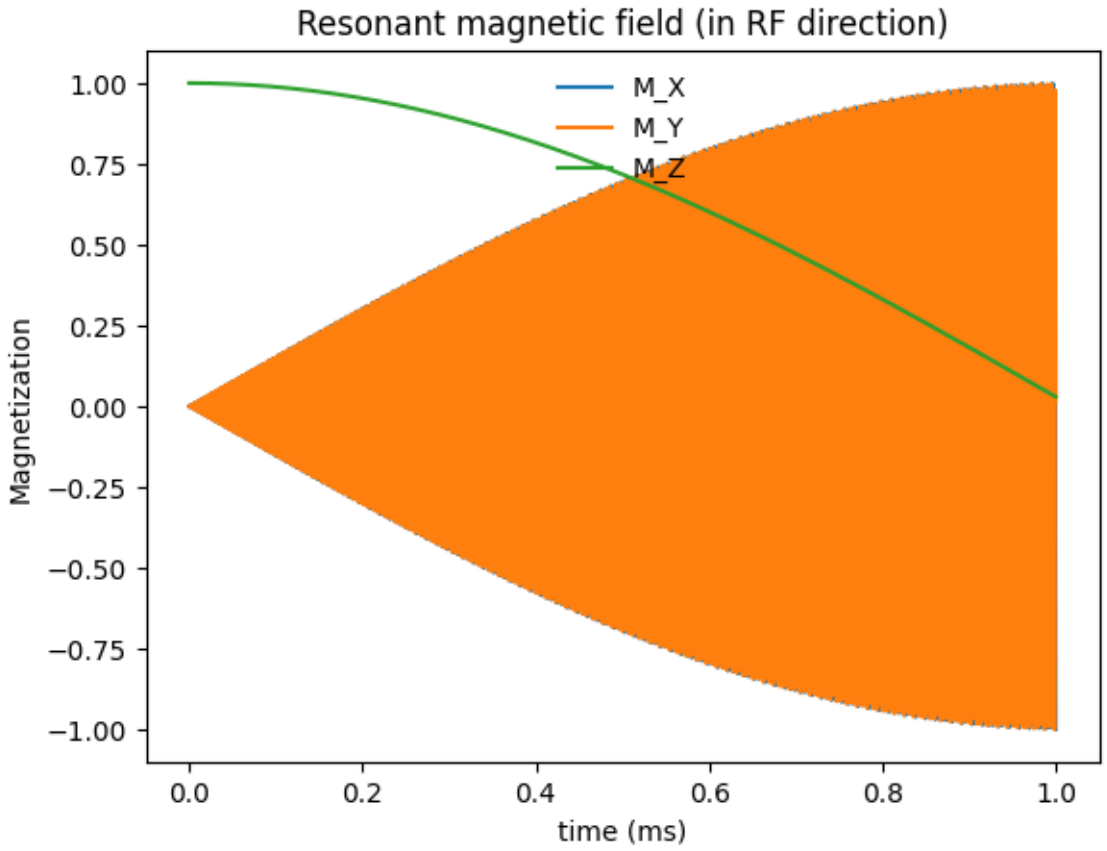
(continues on next page)

(continued from previous page)

```
↪theta],
        [axis[2] * axis[0] * (1 - cos_theta) - axis[1] * sin_theta, axis[2] * axis[1]␣
↪* (1 - cos_theta) + axis[0] * sin_theta, cos_theta + axis[2]**2 * (1 - cos_theta)]
    ])
    return R @ M

# RF not applied at resonance frequency (constant magnetic field in X)
B = np.array([B10, 0, B0])  # Magnetic field vector

Mall = np.zeros((3, len(t)))
Mall[:, 0] = M_equilibrium
for It in range(len(t) - 1):
    dt = t[It + 1] - t[It]
    Mall[:, It + 1] = bloch_rotate(Mall[:, It], dt, B)

# Plot the results
plt.plot(t, Mall[0, :], label='M_X')
plt.plot(t, Mall[1, :], label='M_Y')
plt.plot(t, Mall[2, :], label='M_Z')
plt.xlabel('time (ms)')
plt.ylabel('Magnetization')
plt.legend(loc='upper center', frameon=False)
plt.title('Non-resonant magnetic field (in RF direction)')
plt.show()
```



```
# RF at Larmor resonance frequency
B = np.array([B10 * np.cos(2 * np.pi * f0 * t), B10 * np.sin(2 * np.pi * f0 * t), B0␣
↪* np.ones(len(t))])

Mall = np.zeros((3, len(t)))
Mall[:, 0] = M_equilibrium
for It in range(len(t) - 1):
    dt = t[It + 1] - t[It]
    Mall[:, It + 1] = bloch_rotate(Mall[:, It], dt, B[:,It])


# Plot the results
plt.plot(t, Mall[0, :], label='M_X')
plt.plot(t, Mall[1, :], label='M_Y')
plt.plot(t, Mall[2, :], label='M_Z')
plt.xlabel('time (ms)')
plt.ylabel('Magnetization')
plt.legend(loc='upper center', frameon=False)
plt.title('Resonant magnetic field (in RF direction)')
plt.show()
```

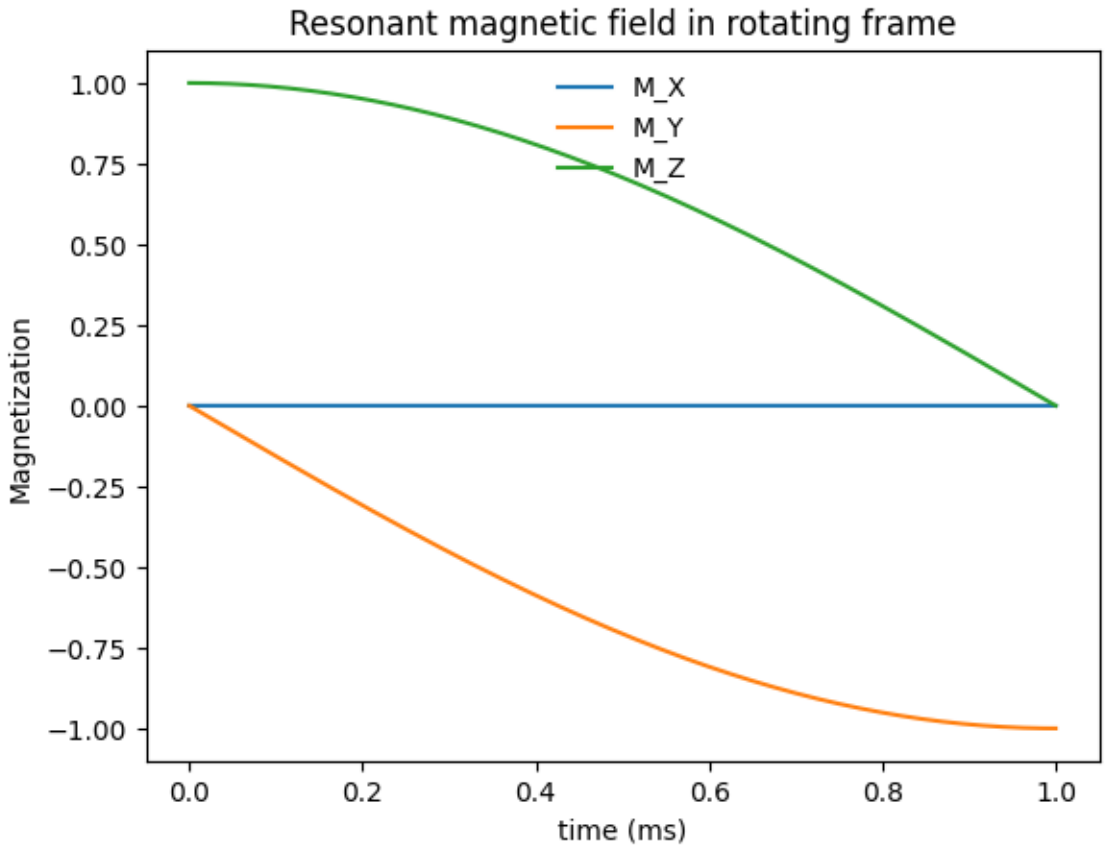



```
# RF at Larmor resonance frequency
B = np.array([B10, 0, 0])

Mall = np.zeros((3, len(t)))
Mall[:, 0] = M_equilibrium
```

(continues on next page)

(continued from previous page)

```
for It in range(len(t) - 1):
    dt = t[It + 1] - t[It]
    Mall[:, It + 1] = bloch_rotate(Mall[:, It], dt, B)


# Plot the results
plt.plot(t, Mall[0, :], label='M_X')
plt.plot(t, Mall[1, :], label='M_Y')
plt.plot(t, Mall[2, :], label='M_Z')
plt.xlabel('time (ms)')
plt.ylabel('Magnetization')
plt.legend(loc='upper center', frameon=False)
plt.title('Resonant magnetic field in rotating frame')
plt.show()
```



2.5 Relaxation

The other key result of the Bloch equation is that the net magnetization will return to equilibriuam over time with specific time constants, called T_1 and T_2. In the rotating frame, without any applied RF energy, the Bloch equation is simplified to

$$ \frac{d\vec{M}(\vec{r},t)}{dt} = \begin{bmatrix} -1/T_2(\vec{r})& 0 & 0 \ 0 & -1/T_2(\vec{r}) & 0 \ 0 & 0 & -1/T_1(\vec{r}) \end{bmatrix} \vec{M}(\vec{r},t) + \begin{bmatrix} 0 \ 0 \ M_0(\vec{r})/T_1(\vec{r}) \end{bmatrix} $$

This has solutions of

$$ \vec{M}(\vec{r},t) = \begin{bmatrix} e^{-t/T_2(\vec{r})} & 0 & 0 \ 0 & e^{-t/T_2(\vec{r})} & 0 \ 0 & 0 & 1 \end{bmatrix} \vec{M}(\vec{r},0) + \begin{bmatrix} 0 \ 0 \ M_0(\vec{r})(1-e^{-t/T_1(\vec{r})}) \end{bmatrix} $$

It is extremely convenient to divide up the net magnetization vector into the separate transverse (XY) and longitudinal (Z) components. The longitudinal component, M_Z, by convention is always aligned with B_0, the main magnetic field, whereas the transverse components, M_X and M_Y are both perpendicular to the direction of B_0.

For the transverse magnetization, using the shorthand complex notation for the transverse magnetization, $M_{XY}(\vec{r},t) = M_X(\vec{r},t) + i M_Y(\vec{r},t)$:

$$M_{XY}(\vec{r},t) = M_{XY}(\vec{r},0) e^{-t/T_2(\vec{r})}$$

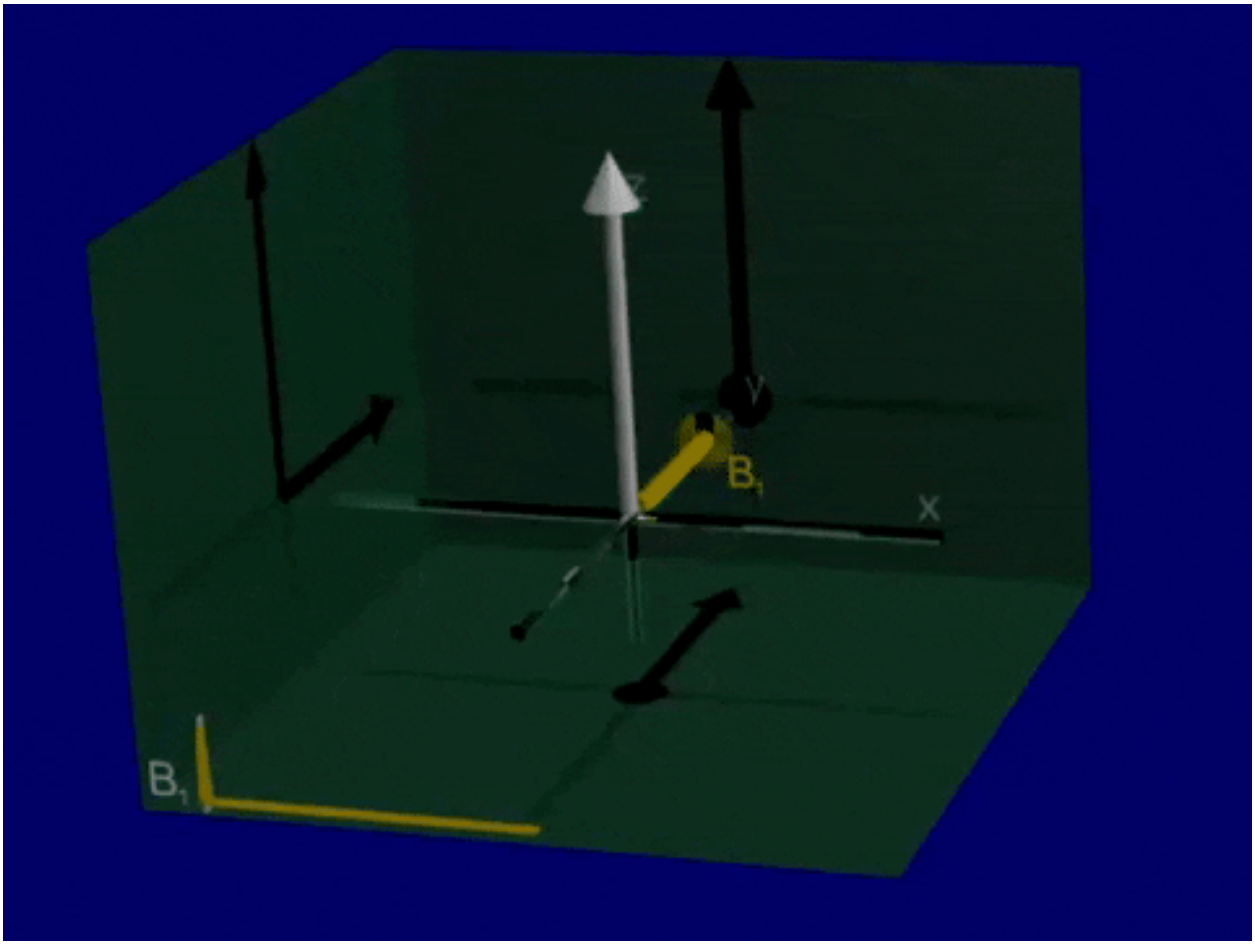
This equation means that the transverse magnetization decays from its original value (when $t=0$) down to 0, with an exponential time constant $T_2(\vec{r})$.

For the longitudinal magnetization:

$$M_Z(\vec{r},t) = M_Z(\vec{r},0)e^{-t/T_1(\vec{r})} + M_0(\vec{r})(1- e^{-t/T_1(\vec{r})})$$

This equation means that the longitudinal magnetization returns or recovers to its equilibrium value, $M_0(\vec{r})$, with an exponential time constant $T_1(\vec{r})$.

T1 and T2 relaxation after RF Excitation



2.5.1 Simulation of Relaxation

One more time, open up the Bloch Equation Simulator. Under the 'Relaxation' options in the top left, you can adjust T1 and T2 relaxation rates.

1. Experiment with different T1 and T2 relaxation values
2. When the magnetization turns to equilibrium, use a RF pulse and you will see relaxation occurring again.
3. From Equilibrium, try a 180-degree flip angle and try adjusting both T1 and T2. Which parameter influence the relaxation in this situation?

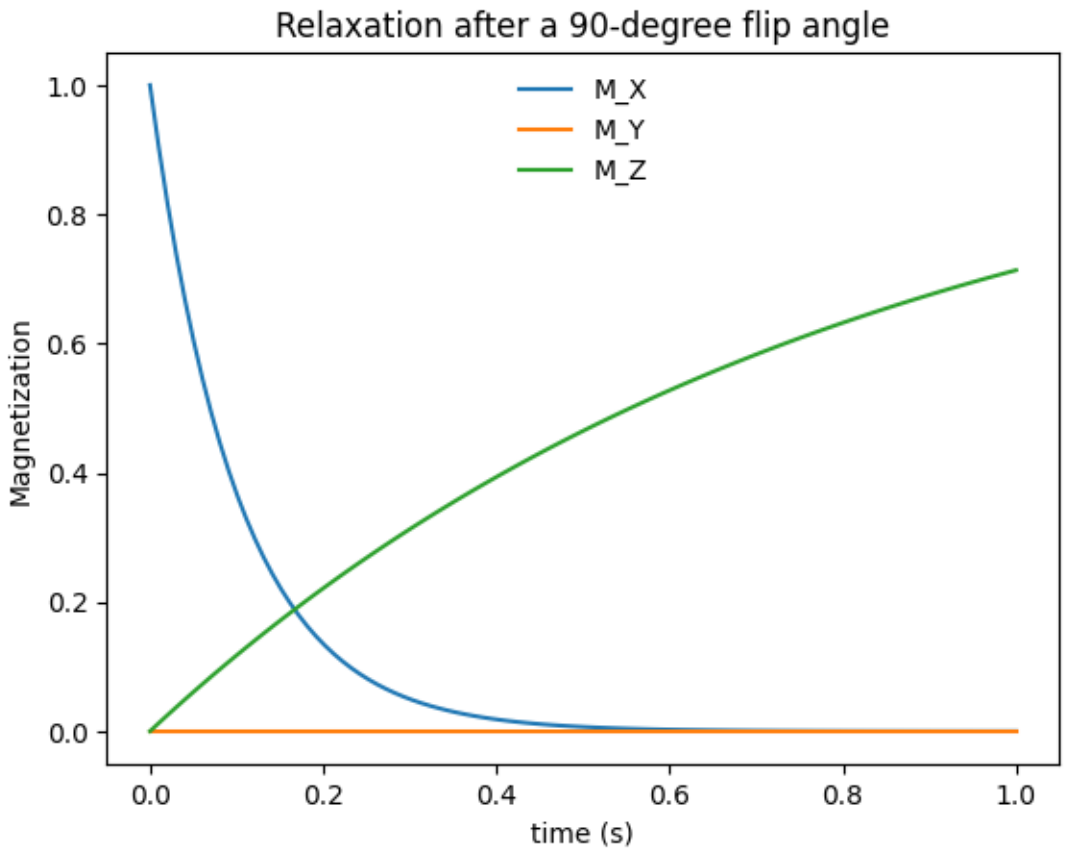
Below are additional Bloch equation simulations and associated code of relaxation.

```
import numpy as np
import matplotlib.pyplot as plt

# Time vector
t = np.linspace(0, 1, 100)  # seconds

# Relaxation parameters
M0 = 1.0
T1 = 0.8  # seconds
T2 = 0.1  # seconds
```

(continues on next page)

(continued from previous page)

```
# Initial magnetization
M_equilibrium = np.array([0, 0, M0])

# Flip angle
flip = 90  # degrees

# Gyromagnetic ratio
gammabar = 42.58  # kHz/mT

# RF pulse duration
T = 1e-3  # 1 ms

# Calculate RF pulse amplitude (milliTesla)
B10 = (flip * np.pi / 180) / (2 * np.pi * gammabar * T)

# Define the Bloch RF tip function
def bloch_rftip(M, T, B1):
    # Apply RF tip (simplified for this example)
    theta = gammabar * B1 * T * 2 * np.pi  # Flip angle in radians
    R = np.array([
        [np.cos(theta), 0, np.sin(theta)],
        [0, 1, 0],
        [-np.sin(theta), 0, np.cos(theta)]
    ])
    return R @ M

# Define the Bloch relaxation function
def bloch_relax(M_start, t, M0, T1, T2):
    E1 = np.exp(-t / T1)
    E2 = np.exp(-t / T2)
    Mx = M_start[0] * E2
    My = M_start[1] * E2
    Mz = M_start[2] * E1 + M0 * (1 - E1)
    return np.array([Mx, My, Mz])

# Apply RF tip
M_start = bloch_rftip(M_equilibrium, T, B10)

# Compute magnetization over time
Mall = np.zeros((3, len(t)))
for It in range(len(t)):
    Mall[:, It] = bloch_relax(M_start, t[It], M0, T1, T2)

# Plot the results
plt.plot(t, Mall[0, :], label='M_X')
plt.plot(t, Mall[1, :], label='M_Y')
plt.plot(t, Mall[2, :], label='M_Z')
plt.xlabel('time (s)')
plt.ylabel('Magnetization')
plt.legend(loc='upper center', frameon=False)
plt.title(f'Relaxation after a {flip}-degree flip angle')
plt.show()
```


Relaxation after a 90-degree flip angle
M_X
M_Y
M_Z
Magnetization
time (s)
0.0
0.2
0.4
0.6
0.8
1.0

CHAPTER

THREE

IN VIVO SPIN PHYSICS

In living organisms, there are several important spin physics phenoma that influence the MRI signal. Most significantly these include magnetic susceptibility differences, and chemical shift differences between water and fat protons. There are multiple other phenoma of interest, including flow, diffusion, and magnetization transfer, but they are not covered here.

3.1 Learning Goals

1. Describe the concepts of polarization, resonance, excitation, and relaxation
    - Describe the concept of magnetic susceptibility
    - Describe how magnetic susceptibility differences affect resonance
    - Describe the concept of chemical shift
    - Describe how chemical shift differences affect resonance

3.2 Magnetic Susceptibility

The magnetic susceptibility is a measure of how magnetized a substance gets when placed in a magnetic field, and denoted by the Greek symbol χ. Most importantly for MRI, tissue gets magnetized when placed in a magnetic field. Substances are classified into three types based on their susceptibility:

1. Diamagnetic substances ($\chi < 0$) – When placed in an external magnetic field, a weak magnetic field is induced in the opposite direction to B0. As a result, the effective magnetic field is reduced. They are basically nonmagnetic. The vast majority of tissues in the body have this property.
2. Paramagnetic substances ($\chi > 0$): They become magnetized while the external magnetic field is on and become demagnetized once the field has been turned off. Their induced magnetic field is in the same direction as the external magnetic field. causes an increase in the effective magnetic field. Air, oxygen, and gadolinium are examples of paramagnetic substances.
3. Ferromagnetic substances ($\chi >> 0$) – These are strongly attracted to the magnetic field and dangerous in MRI. They become permanently magnetized even after the magnetic field has been turned off. Three types of ferromagnets are known: iron (Fe), cobalt (Co), and nickel (Ni).

Magnetic susceptibility changes the magnetic field in the vicinity of the substance. This can include changes within the substance, but also in surrounding tissues. In particular, there can be signficant distortions of the magnetic field at the boundary between two substances with different susceptibilities. These changes in magnetic field change the resonance frequency. They also change relaxation rates, particularly T2*, which are described in more detail later.

The following effects of magnetic susceptibility are important in MRI:

1. Air-Tissue interfaces - Tissue primarily consists of water, which is slightly diamagnetic, while air is slightly paramagnetic. This causes a significant distortion of the magnetic field at the air-tissue interface, which can lead to signal loss and artifacts in MRI images.
2. Gadolinum-based contrast agents - Gadolinium is a paramagnetic substance that is commonly used as a contrast agent in MRI. This leads to changes in relaxation rates wherever the agent goes, creating valuable contrast.
3. Blood oxygen level dependent (BOLD) contrast - Oxygen is paramagnetic, and the presence or lack of oxygen in blood leads to changes in relaxation rates. This phenomenon is exploited in functional MRI (fMRI) to measure brain activity based on changes in blood flow and oxygenation.
4. Tissue iron content - Iron is a ferromagnetic substance that can accumulate in tissues, such as in cases of hemochromatosis or hemosiderosis. This can lead to distortions of the magnetic field and changes in relaxation rates.
5. Implants - Medical implants contain various materials that can affect the magnetic field. Modern orthopedic implants are made of non-ferromagnetic materials like titanium or stainless steel, which are safe for MRI, but also are strongly paramagnetic so can lead to significant distortions of the mangetic field and create significant artifacts. However, others may contain ferromagnetic materials, such as iron, which can pose a safety risk in the MRI scanner. It is crucial to check the safety of any implant before performing an MRI.

3.3 Chemical Shift

A fundamental concept in magnetic resonance and NMR is chemical shift. This is the phenomenon where the resonance frequency of a nucleus due to shielding created by the orbital motion of surrounding electrons in response to a magnetic field. Thus the magnetic field experienced by a nucleus is

$$ B_{eff} = B_0 (1 + \sigma)$$

where σ is the shielding constant that is dependent on the local chemical environment. For example, a specific atom in a molecule will have a shielding constant that depends on the electrons in the surrouding atoms of the molecule.

The chemical shift, δ, is defined with respect to a reference frequency, which is typically the resonance frequency of a standard molecule, and is in units of parts per million (ppm). The is approximately the difference in shielding constant between the reference and the molecule of interest. This leads to a change in resonance frequency due to chemical shift that is given by

$$ \Delta f_{cs} = \delta_{cs} \bar\gamma B_0$$

3.3.1 Fat versus Water

In MRI, chemical shift most commonly affects the images due to the difference in resonance frequency between fat and water protons. Fat is composed of lipids which have a relatively complex set of chemical shifts that also varies by the type of lipid. In MRI a generic “fat” signal is typically modeled as a single peak that has a -3.5 parts per million (ppm) shift from water. This leads to a resonance frequency shift Δf_{cs} of approximately 220 Hz at 1.5 T and 440 Hz at 3 T.

3.4 Variations in the Magnetic Field

In vivo MRI studies experience changes in the main magnetic field, B_0, primarily due to three factors. These are described as "off-resonance" when they change the resonance frequency for all spins.

1. Main field B_0 inhomogeneity (< 1 ppm) – *Off-resonance source.* The main magnetic field is imperfect.
2. Magnetic susceptibility differences (up to 10 ppm) – *Off-resonance source.* Changes in the magnetic field at boundaries of materials (e.g. air-tissue, metal implants) as described above.
3. Chemical shift (-3.5 ppm) – different resonance frequencies in fat and water as described above.

These all change the longitudinal component of the magnetic field, which we will mathematically describe as:

$$B_Z(\vec r) = B_0 + \Delta B_0(\vec r) + \delta_{cs} B_0$$

where $\Delta B_0(\vec r)$ is the off-resonance term that includes both main field imperfections and magnetic susceptibility effects.

Part II

MRI System

CHAPTER

FOUR

THE MRI SYSTEM

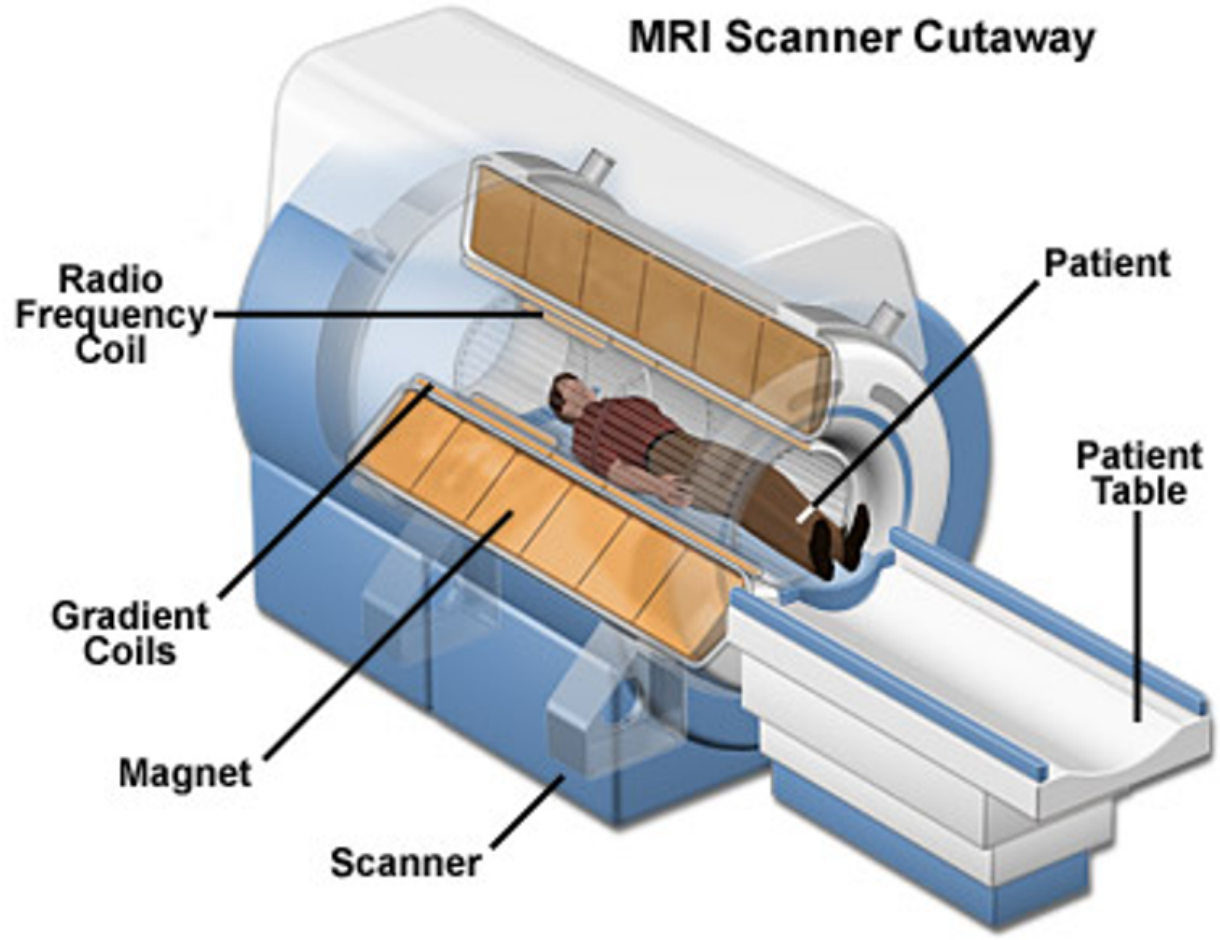
Every MRI system consists of several major components, required to create measureable magnetic resonance phenoma and to perform spatial localization. This section will deconstruct a MRI scanner and see what is inside. This will give insight into the fundamentals of a MRI experiment and what is hardware is required.

4.1 Learning Goals

1. Describe the 4 fundamental components of a MRI scan and why they are necessary
    - What are the required components of MRI system?
    - What is the purpose of these components?
    - What calibrations are required for a MRI scan?
    - Answer these questions:
        - Why is MRI so loud?
        - Why is it dangerous to bring certain metals into a MRI scanner?

4.2 Components of a MRI Scanner

If we cut open a typical MRI scanner, it would look like this



The scanner consists of 3 hardware components that are required for all MRI scans

1. Main magnet - B_0
2. Radiofrequency (RF) coils, including a transmit RF coil - $B_1^+(\vec{r},t)$ - and a receive RF coil - $B_1^-(\vec{r},t)$
3. Magnetic field gradient coils - $\vec{G}(t)$

4.2.1 Main magnet - B_0

The purpose of the main magnetic field is "Polarization". This refers to the preferential alignment of nuclear spins with a magnetic field. The more alignment there is of spins, the greater the MRI signal. The stronger the magnetic field, the more alignment and therefore the stronger signal that will be received.

This magnetic field is typically 1.5 Tesla (1.5 T) or 3 T. It is designed to be homogeneous. To achieve this large of a mangetic field typically requires using a superconducting magnet, which consists of a superconducting alloy bathed in liquid helium, and this type of magnet is always on.

Due to B_0, there is a large magnetic field inside and around the MRI scanner, making it very dagerous to bring certain materials near the scanner. These materials, typically metals that are ferromagnetic like iron, can be sucked into the scanner by the powerful magnetic field.

4.2.2 Radiofrequency (RF) coils - $B_1^+(\vec{r},t), B_1^-(\vec{r},t)$

The purpose of the RF coils is "Excitation" and "Acquisition". RF coils are antennas designed to detect changes in electromagnetic fields in the radiofrequency range. These coils and associated components must be *tuned* to the resonance frequency for the MRI system in order effectively transmit and receive signals at the resonance frequency.

Excitation refers to depositing energy at a specific frequency that perturbs the spins, causing them to resonate and release energy that is captured as the MRI signal. This is performed using a transmit RF coil, described by the magnetic field $B_1^+(\vec{r},t)$. This energy is deposited in the radio-frequency (RF) range, typically in the 100 MHz range for MRI, which is the resonance frequency.

After Excitaiton, the spins release energy, also in the RF range, as they return to thermal equilibrium. This is measured in Acquisition where the signal is detected by the receive RF coil, described by the magnetic field $B_1^-(\vec{r},t)$.

The RF coils are typically separated into separate transmit and receive coils because of different requirements: the transmit coils for excitation are designed to be homogeneous in depositing energy across the field-of-view (FOV), while receive coils are designed for maximum maximum sensitivity to the relatively small signals detectable from the spins.

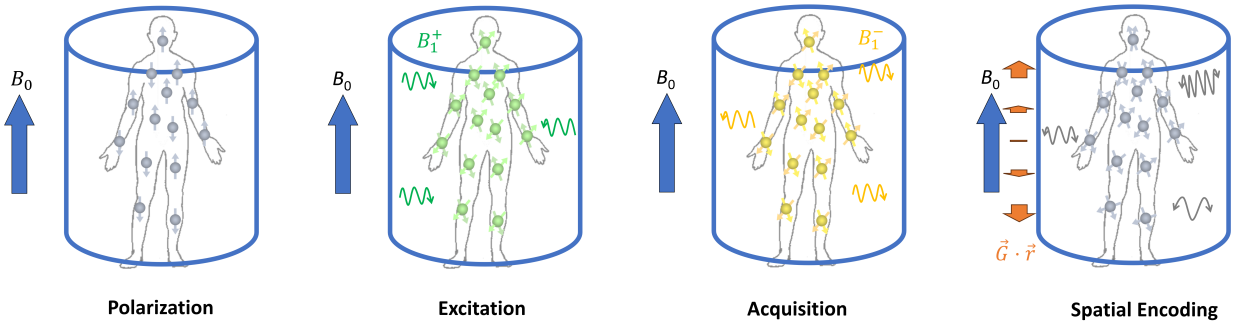
4.2.3 Magnetic field gradient coils - $\vec{G}(t)$

The purpose of the magnetic field gradient coils is for "Spatial Encoding", which enables creation of images. These coils, commonly refered to simply as "gradients", are designed to create linearly varying magnetic fields. These varying magnetic fields add and subtract from the main magnetic field. Since the magnetic field determines the resonance frequency ($f = \bar{\gamma} |\vec{B}|$), this will create a spatial variations in the magnetic field that can be used to separate signals from different locations and ultimately create images.

The noises during a MRI scan are due to the rapid switching of electric currents in the gradient coils. The change in electric currents and resulting change in magnetic field interacts with the main magnetic field and exerts a force (known as Lorentz force) on the gradient coil. Thus the gradient coils will vibrate as they are switched on and off, creating loud noises.

4.3 Fundamental MRI Experiment Procedure

1. Polarization: Place subject in a large magnetic fiel
    - Components required: main magnet, B_0
2. Excitation: Perturb the spins from equilibrium by applying RF energy
    - Components required: transmit RF coil, $B_1^+(\vec{r},t)$
3. Acquisition: Detect the changing magnetic field RF energy from excited spins
    - Components required: receive RF coil, $B_1^-(\vec{r},t)$
4. Spatial Encoding: Create spatial variations in the magnetic field to distinguish spins at different locations. This can occur during excitation and acquisition.
    - Components required: Magnetic field gradient coils, $\vec{G}(t)$
5. Repeat Excitation, acquisition, and encoding as needed to acquire enough data to form an image



4.4 Calibrations

Calibrations are performed of the individual components of a MRI scanner both when the system is installed and during each scan.

4.4.1 Installation Calibrations

1. Main magnet: Minor adjustments to the main magnetic field, called “shimming”, is performed to make the main magnetic field more homogeneous
2. RF systems: Checking for sources of RF interference entering the scanner room
3. Gradient coils: Calibration of the gradient strength and mapping of the gradient fields to ensure accurate spatial encoding.

4.4.2 Pre-scan Calibrations

Different people and different places in the body will create differences in both the main magnetic field as well as the RF fields due to loading and magnetic susceptibility, thus requiring calibrations to be performed for each scan. This is sometimes referred to as “prescan”.

1. Main magnetic field: Measurements of the resonance frequency across the imaging field-of-view are used to perform shimming and choosing the correct RF frequency (“center frequency”)
2. RF systems: The transmit RF field created, B_1^{+}, is calibrated by modifying the RF transmit power, implemented via amplifier gains. Often, the RF receive coil profiles, B_1^{-}, are also measured for intensity correction and accelerated imaging methods.

CHAPTER

FIVE

MAGNETIC FIELDS IN MRI

The magnetic fields created and manipulated during a MRI experiment critically determine the entire MRI experiment. We will provide some background on creating and detecting magnetic fields, and then describe the magnetic fields present in MRI systems.

5.1 Learning Goals

1. Describe the 4 fundamental components of a MRI scan and why they are necessary
    - Define the magnetic fields created by various components
    - Describe the purpose of each magnetic field created
2. Understand what MRI is measuring
    - Describe how RF coils receive MRI signal

5.2 Summary of Magnetic Fields

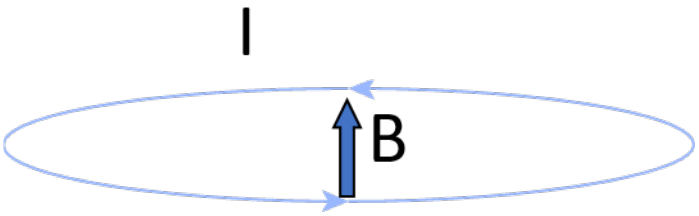
Each of the main components of a MRI system creates magnetic fields, but with different orientations, magnitudes, and frequencies, which are summarized in the following table

Component	Notation	Direction	Frequency	Strength	Purpose
Main Field	B_0	z	0 Hz	≈ 1 T	Polarization
Main Field Inhomogeneities	$\Delta B_0(\vec{r})$	z	0 Hz	$\approx 1-10$ ppm	-
Magnetic Field Gradients	$\vec G(t)$	z	≈ 1 kHz	≈ 10 mT	Spatial Encoding
Transmit RF Coils	$B_1^+(\vec{r},t)$	x,y	≈ 100 MHz	$\approx 10 \mu T$	Excitation
Receive RF Coils	$B_1^-(\vec{r},t)$	x,y	≈ 100 MHz	$\approx 1 \mu T$	Reception
Net Magnetization	$\vec{M}(\vec{r},t)	z,y,z	≈ 100 MHz	$\approx 1 \mu \mathrm{T}$	Signal Source

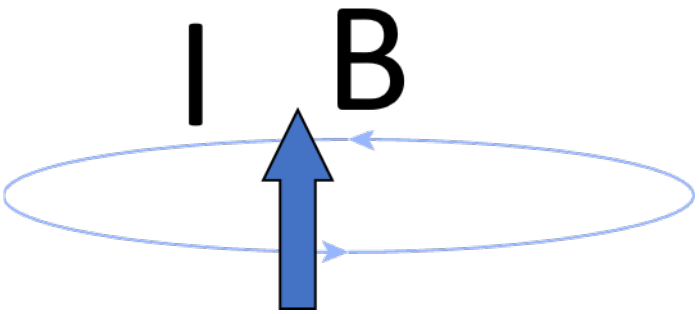
Also included is the Net Magnetization, $\vec{M}(\vec{r},t)$, which is the magnetic fields created from nuclear spins that we measure in MRI.

5.3 Loop currents and magnetic fields

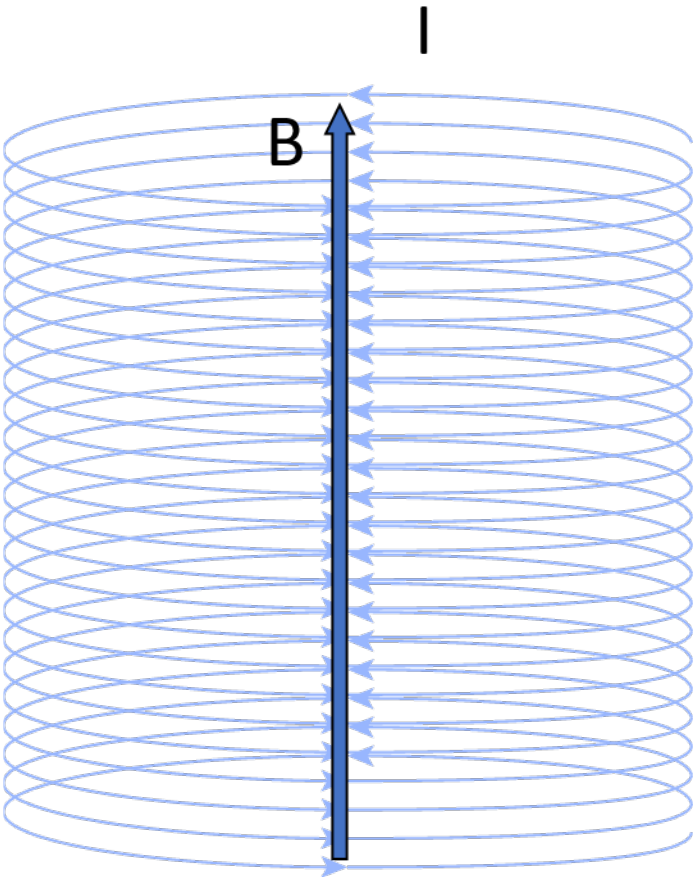
Magnetic fields, $\vec{B}$, can be created by current, I, through loop of wire, governed by the principles of electromagnetism in Maxwell's equations. The the "left-hand rule" determines the direction of the magnetic field.



By increasing the current, the magnetic field is increased



We can also link many loops together (a "solenoid") to increase the extent of the magnetic field.



A solenoid design is typically used to create the main magnetic field for MRI, B_0, to create a large region in space with a homogeneous magnetic field, as required for MRI across a volume.

5.4 Main Magnet - B_0

The main magnetic field is created with solenoid coils, shown above, that can provide a large region in space where there is large, homogeneous magnetic field.
The large magnetic field is required to create stronger polarization. The large region is space determines where we can create an image. And having a homogeneous magnet ensures there are no image distortion artifacts.

This magnet creates magnetic fields of usually 1.5 or 3 T, by convention oriented in the z-direction, and this field is always on during the scan. Mathematically, we can add this to our magnetic field vector:

$$\vec{B}(\vec{r},t) = \begin{bmatrix} 0 \ 0 \ B_0 \end{bmatrix}$$

Note that this magnetic field is never perfect, which is captured by the main field inhomogeneities, $\Delta B_0(\vec r)$, and that the field changes due to the molecular environment which is known as chemical shift. These concepts and how they change B_Z are described in *In Vivo Spin Physics*.

5.5 Magnetic field gradient coils - $\vec{G}(t)$

The magnetic field gradient coils are designed to add and subtract from the main magnetic field. In this way, they create a spatially varying magnetic field. Since the frequency of magnetic resonance is proportional to the magnetic field, this means that when the gradients are on the resonance frequency varies as function of position. This gives us a way to separate signals from different regions based on resonance frequency, and create images.

The gradient coil fields have a maximum amplitude of around 50 mT/m = 0.05 T/m, aligned with B_0 in the z-direction, and this field is switched on and off during MRI experiments. The gradients are typically varied every few ms, with minimum switching times around 0.1 ms. Mathematically, we can add this to our magnetic field vector:

$$\vec{B}(\vec{r},t) = \begin{bmatrix} 0 \ 0 \ B_0 + \vec{G}(t) \cdot \vec{r} \end{bmatrix}$$

Where the gradients creates linear changes in the z-component of the magnetic field, B_Z, across space.

5.6 Radiofrequency (RF) coils - $B_1^+(\vec{r},t), B_1^-(\vec{r},t)$

The purpose of the RF coils and RF systems are

1. Excitation – perturb the net magnetization from equilibrium by creating magnetic fields at the resonance frequency
2. Reception – after excitation, the precessing net magnetization creates magnetic fields at the resonance frequency

These are called RF coils since the resonance frequencies for MRI, typically around 100 MHz, is in the so-called radio-frequency (RF) range. All RF coils used for MRI must be tuned to the resonance frequency to effectively transmit and receive signals at this frequency.

Their amplitude is much smaller than the other magnetic fields, typically in the 10 μT = .00001 T range. The magnetic fields created by the RF coils are oriented perpedicularly to the main magnetic field, and thus are in the transverse (XY) plane. Mathematically, we add this final magnetic field as

$$\vec{B}(\vec{r},t) = \begin{bmatrix} B_{1,X}(\vec{r},t) \cos(2\pi\bar{\gamma} B_0 t) - B_{1,Y}(\vec{r},t) \sin(2\pi\bar{\gamma} B_0 t) \

- B_{1,X}(\vec{r},t) \sin(2\pi\bar{\gamma} B_0 t) - B_{1,Y}(\vec{r},t) \cos(2\pi\bar{\gamma} B_0 t) \ B_0 + \vec{G}(t) \cdot \vec{r} \end{bmatrix}$$

In this representation, the RF magnetic field, $B_{1,X}, B_{1,Y}$, is rotating at the Larmor frequency, $f_0 = \bar{\gamma} B_0$, in the transverse plane. In the rotating frame, this rotation is removed, and we are left with a transverse magnetic field with X and Y components.

The RF system is divided up into transmit and receive paths. All components are tuned to Larmor frequency, meaning they are most sensitive to transmitting or receiving signals at this frequency. Reactive components (capacitors and inductors) are used for tuning to the correct frequency.

5.6.1 Transmit (Tx) system for excitation - $B_1^+(\vec{r},t)$

- Transmit high power (10s of kW) RF signals
- Key requirement is homogeneity, i.e. same magnetic field that creates the same flip angle across imaging FOV
- Usually a single larger, coil surrounding entire FOV

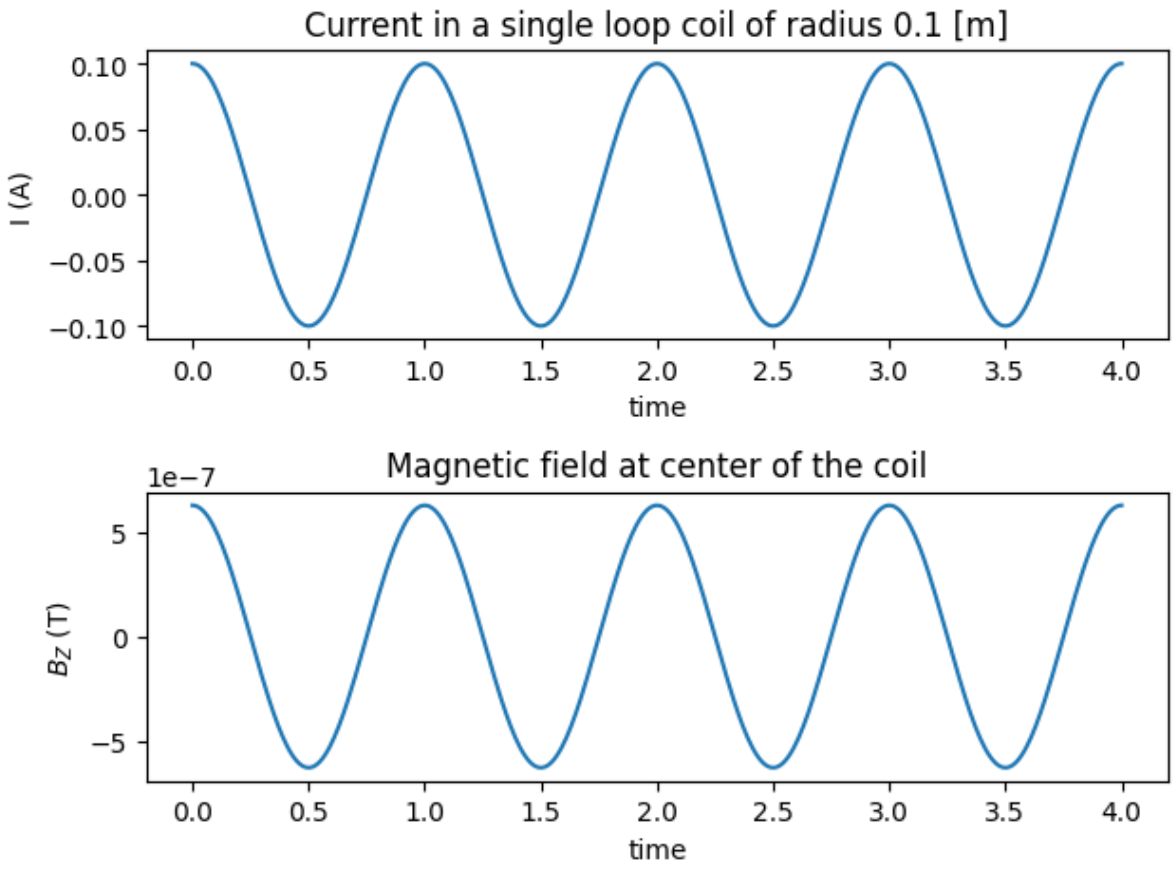
5.6.2 Receive (Rx) system for reception - $B_1^-(\vec{r},t)$

- Receive small signals (mV) arising from body
- Key requirement is high sensitivity to detect small signals with the least noise
- Usually an array of smaller coil elements (also known as channels), placed around the body

5.6.3 Creating Transmit RF fields

To create transmit RF fields, we simply put an oscillating electric current into a loop of wire, which is simulated below. When this is performed at the Larmor frequency, this creates RF excitation.

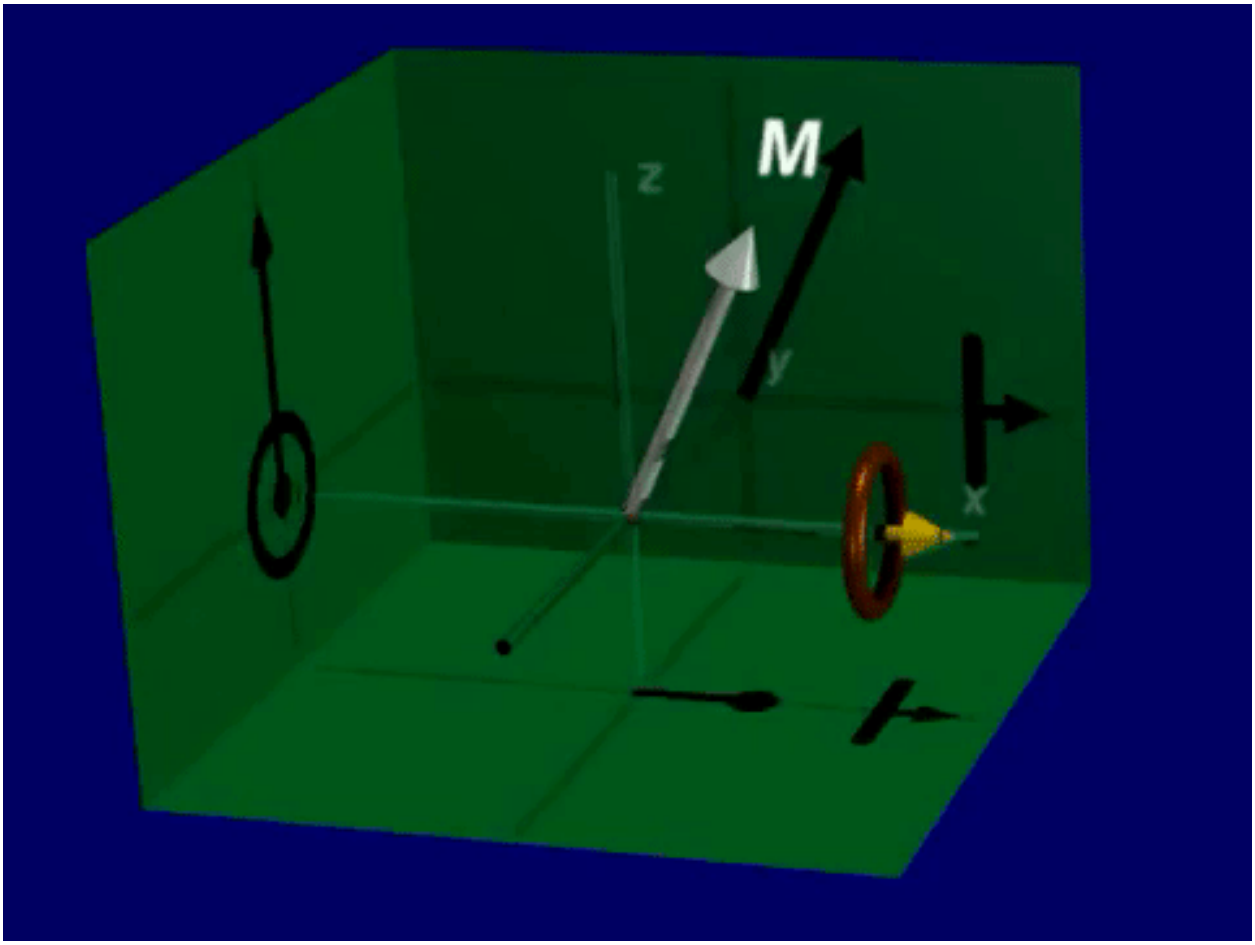
```
import numpy as np

import matplotlib.pyplot as plt

R = 0.1  # coil radius [m]
I0 = 0.1  # current amplitude [amps]
mu0 = 4 * np.pi * 1e-7  # T·m/A

t = np.linspace(0, 4, 401)
I = np.cos(2 * np.pi * t) * I0

BZ = mu0 * I / (2 * R)  # magnetic field at the center of a single loop

plt.subplot(211)
plt.plot(t, I)
plt.ylabel('I (A)')
plt.xlabel('time')
plt.title(f'Current in a single loop coil of radius {R} [m]')

plt.subplot(212)
plt.plot(t, BZ)
plt.ylabel('$B_Z$ (T)')
plt.xlabel('time')
plt.title('Magnetic field at center of the coil')

plt.tight_layout()
plt.show()
```



5.6.4 Receiving Magnetic Resonance Signal

Magnetic resonance signal is received by the following sequence of events

1. Spins and the net magnetization, $\vec{M}$, in a magnetic field will precess at the Larmor frequency, which depends on the gyromagnetic ratio and main magnetic field: $f_0 = \bar{\gamma} B_0$
2. This precession creates oscillating magnetic fields at this same (Larmor) frequency.
3. Oscillating magnetic fields will create oscillating current in loops of wire, which can be detected by amplifying and digitizing this signal



5.6.5 Reciprocity

The principle of reciprocity in electromagnetism tells us that the same relationships govern creating magnetic fields from oscillating currents and creating electric currents from oscillating magnetic fields. So far we have discussed that an oscillating current in a loop of wire will create an oscillating magnetic field, which describes the transmit RF process. So, reciprocity says that an oscillating magnetic field will create a current in a loop of wire. This describe the receive RF process, as shown above.

5.7 Transmit and Receive RF Coils

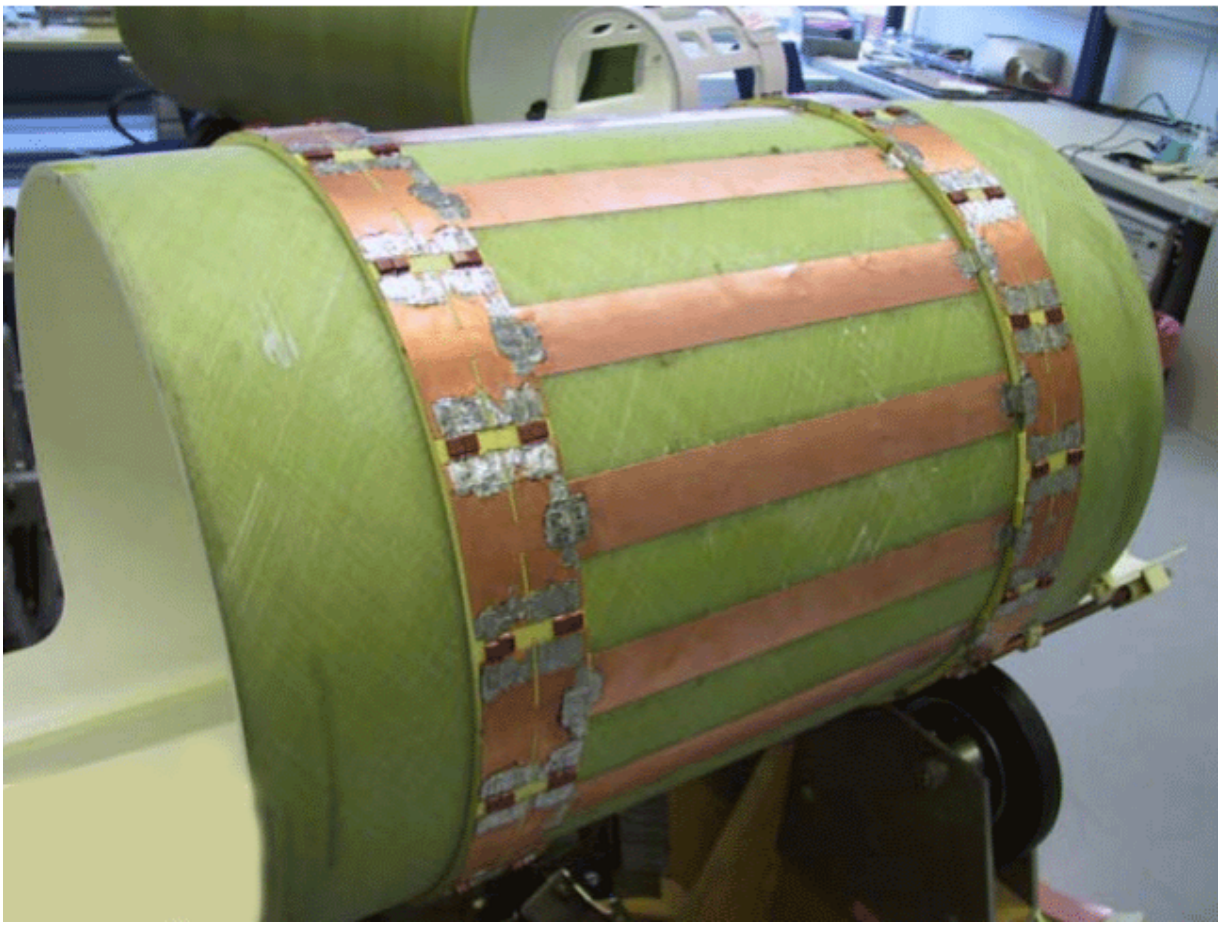
5.7.1 Transmit (Tx) RF coils

- Typically performed with “BODY” coil
- Birdcage design
- Provides homogeneous magnetic field amplitude
- Creates magnetic fields perpendicular to B0

In MRI we denote transmitted magnetic fields from RF energy as

$$B_1^+(\vec{r},t)$$

This field is typically in units of millitesla (mT) and can vary over space, $\vec{r}$ and time, t



5.7.2 Receive (Rx) RF coils

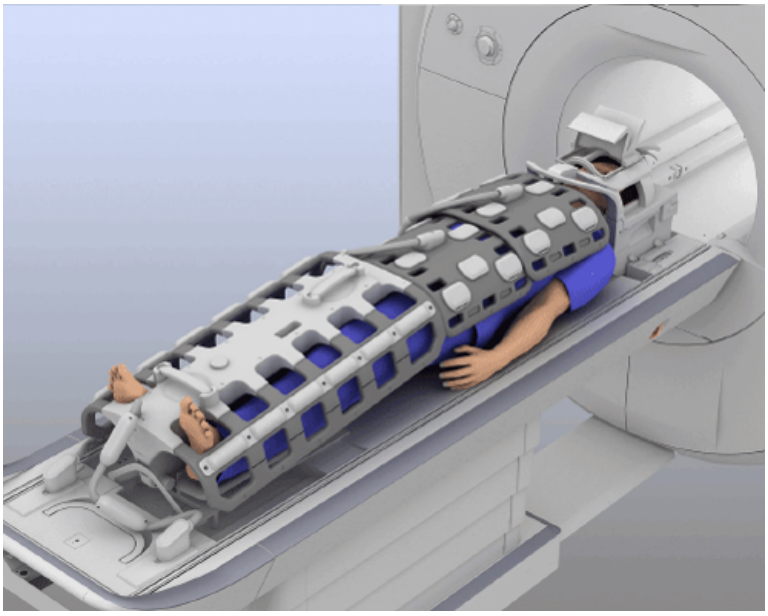
- Tailored to the anatomy of interest, e.g. knee coils, abdomen coils, head coils
- Configured in arrays (or "phased arrays")
- Between 8 to 64 individual coil elements (also known as channels) in a array coil is common
- Elements should be relatively isolated and decoupled to not interfere with each other
- Tuned to the Larmor frequency when receiving signals
- Detuned (turned off) when transmit RF is active so they don't interfere
- *Goal is to provide high sensitivity to receive MR signals by placing them as close as possible to the body*

In MRI we denote senitivity profile to varying magnetic fields of the RF receiver coils as

$$B_{1,n}^-(\vec{r},t)$$

This sensitivity is typically in units of millitesla/Amp (mT/A) and can vary over space, $\vec{r}$, time, t, and coil element, n, in the array coil.

Note that some RF coils are also often built into the table. In the picture below, array coils are placed on top of the subject and there are also coils in the table.



5.8 RF Coil Profiles

5.8.1 Receive elements

Coil elements are most sensitive to signals originating near the element, and the so-called sensitivity off as we move away from the coil. The following images show that each element is most sensitive, or has the brightest image, closest to that element:

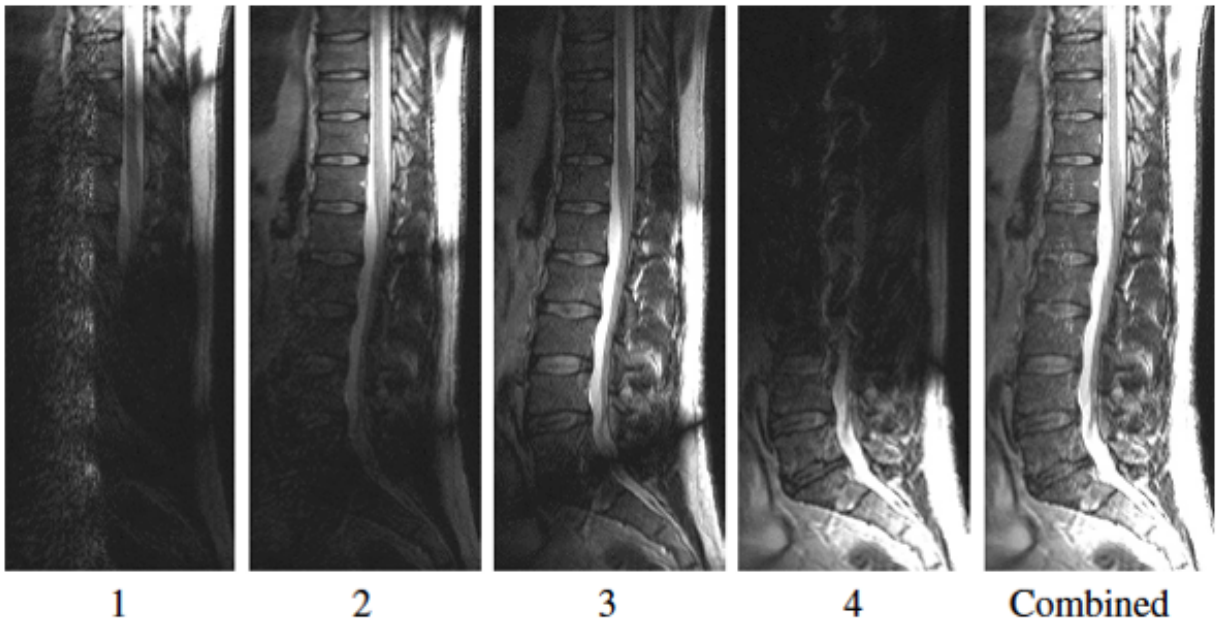


Figure 9.18 Separate images from each of the four channels of a lumbar spine phased array coil. The final combined image is also shown.

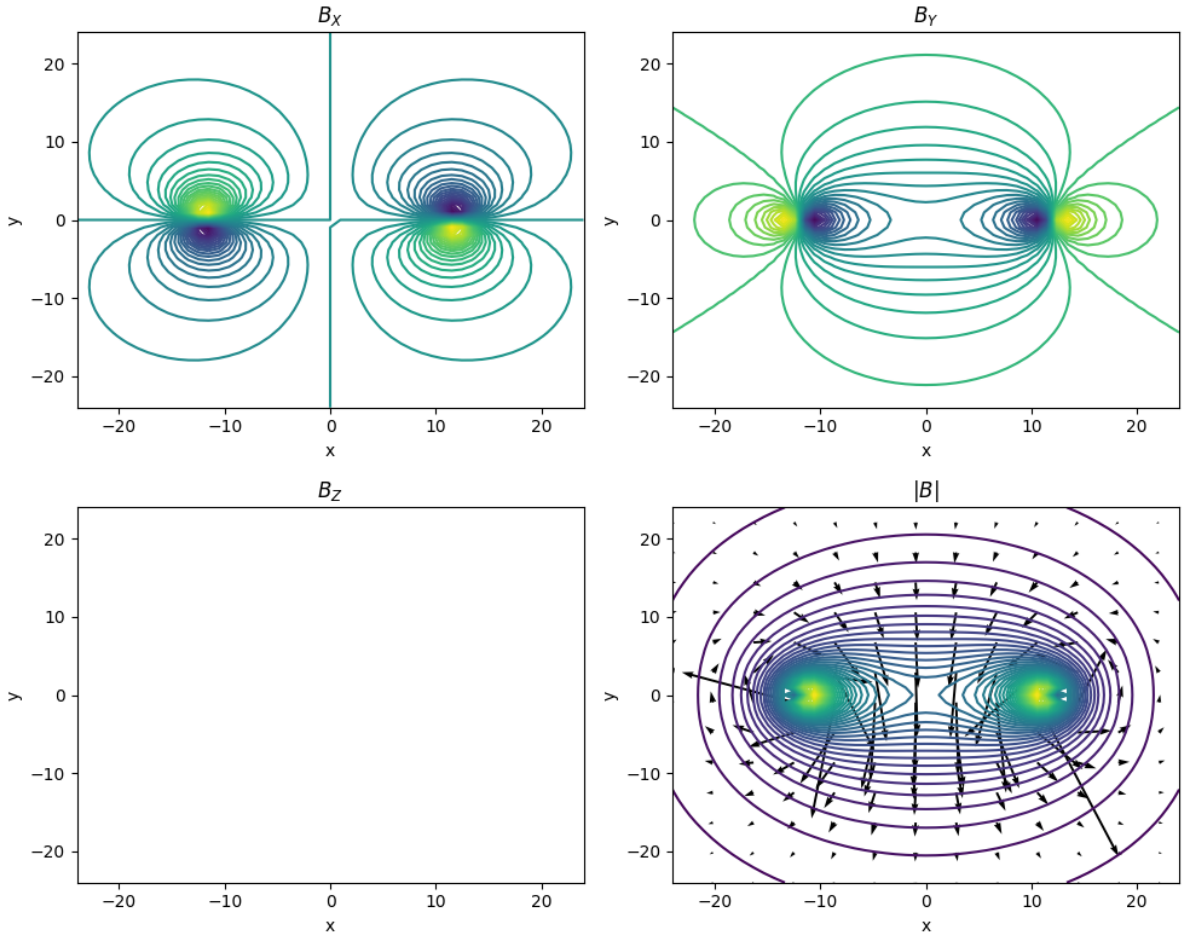
A combined image is created from multiple receive elements. This is typically done by taking the square root of the sum

of each image squared, or "sum-of-squares", although more SNR optimal combinations are possible if individual element sensitivities are known.

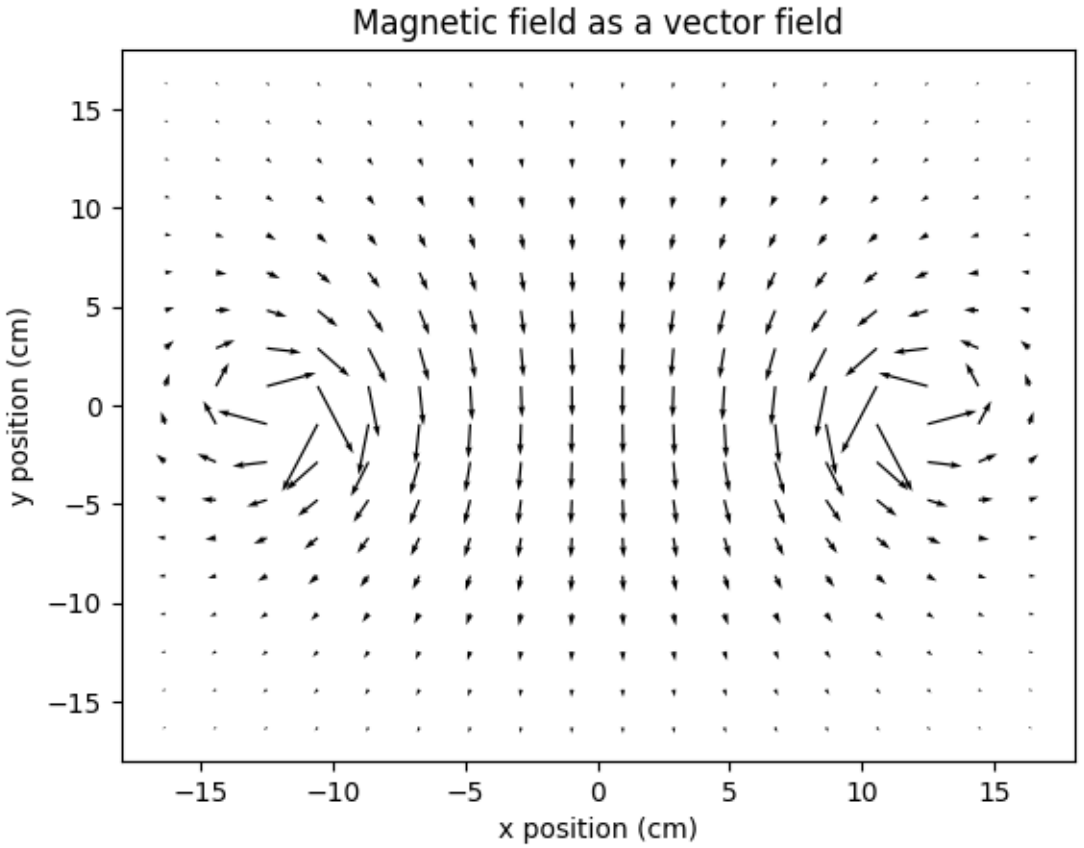
5.8.2 Sensitivity Simulation

The following simulates the sensitivity profile of a single loop RF coil varying over space, using the Biot-Savart law from electromagnetism. In the following plots, the magnetic field vector, $\vec{B}(\vec{r}) = [B_X(\vec{r}), B_Y(\vec{r}), B_Z(\vec{r})]$, that is created by the coil is plotted, as well as the magnetic field amplitude, $|\vec{B}(\vec{r})|$.

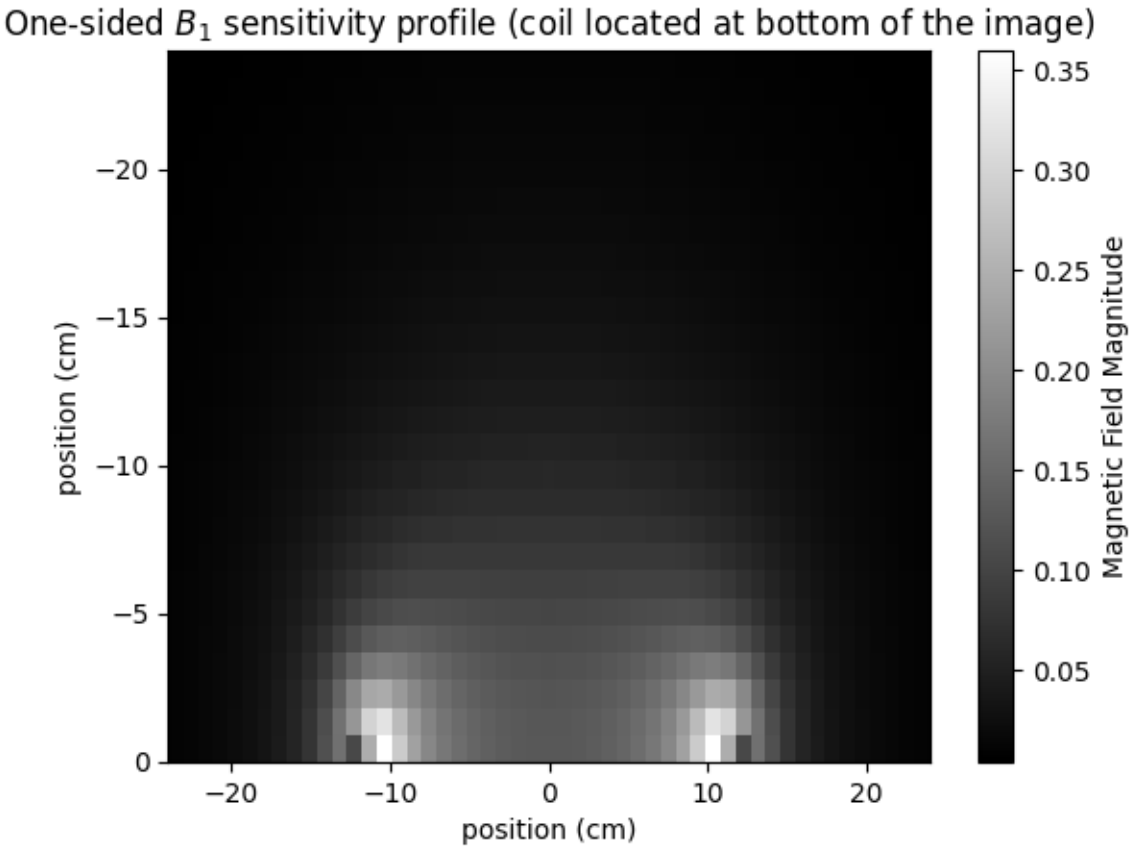
```
import numpy as np
import sys

sys.path.append('../Physics/')
# Import the script as a module
from loop_coil_field import loop_coil_field

R = 12  # coil radius, cm
I = 3  # current in the coil
flag_2d = 1  # just plot 2D profile


BX, BY, BZ, xp, yp, zp = loop_coil_field(R, I, flag_2d)
```



```
import matplotlib.pyplot as plt

skip_index = 2  # skip every 2nd point for plotting

plt.quiver(xp[::skip_index], yp[::skip_index], BX[::skip_index, ::skip_index, 0],␣
↪BY[::skip_index, ::skip_index, 0], scale=4)
plt.axis([-18, 18, -18, 18])
plt.xlabel('x position (cm)')
plt.ylabel('y position (cm)')
plt.title('Magnetic field as a vector field')
plt.show()
```




```
import numpy as np

import matplotlib.pyplot as plt

Bmag = np.sqrt(BX**2 + BY**2 + BZ**2)
xhalf = round(len(xp) / 2)

plt.imshow(Bmag[:xhalf, :], extent=[yp.min(), yp.max(), xp[:xhalf].max(), xp[:xhalf].
↪min()], aspect='auto', cmap='gray')
plt.xlabel('position (cm)')
plt.ylabel('position (cm)')
plt.axis('tight')
plt.colorbar(label='Magnetic Field Magnitude')
plt.title('One-sided $B_1$ sensitivity profile (coil located at bottom of the image)')
plt.show()
```


One-sided B_1 sensitivity profile (coil located at bottom of the image)
position (cm)
−20
−15
−10
−5
0
−20
−10
0
10
20
position (cm)
0.35
0.30
0.25
0.20
0.15
0.10
0.05
Magnetic Field Magnitude

Part III

MRI Experiment

CHAPTER

SIX

THE PULSE SEQUENCE

Magnetic resonance experiments are described by a “Pulse Sequence”, which is a timing diagram that shows how the different magnetic fields are manipulated. This chapter introduces the pulse sequence diagram and the sequence parameters of TE and TR.

6.1 Learning Goals

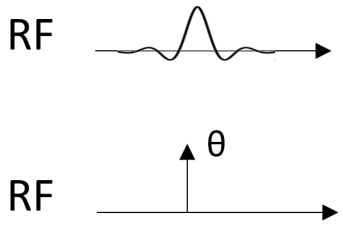
1. Understand what MRI is measuring
    - Identify when signals are measured during a MRI experiment
2. Describe how various types of MRI contrast are created
    - Understand what is shown in a pulse sequence diagram
3. Describe how images are formed
    - Understand that a pulse sequence must be repeated to form an image

6.2 Components of a Pulse Sequence

All MRI experiments can be defined by their pulse sequence. This describes all manipulations of the MRI scanner being done, including their relative timings and amplitudes.

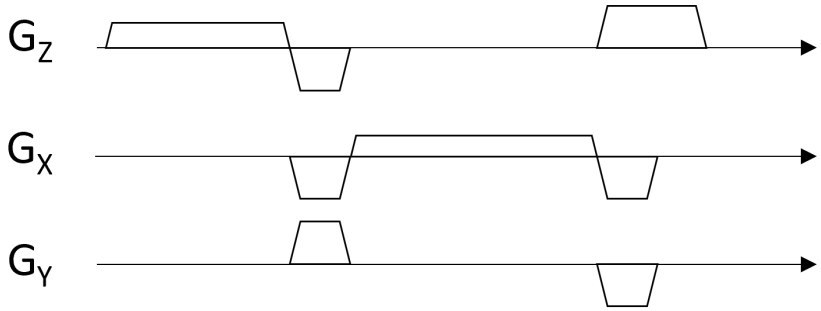
6.2.1 RF Excitation pulse (RF)

The RF excitation pulse is the first required component, as it rotates or flips the magnetization into the transverse plane, creating resonance of the net magnetization that can actually be measured.



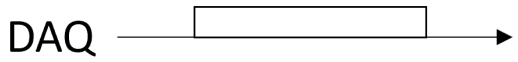
6.2.2 Magnetic field gradients for localization (G_X, G_Y, G_Z)

The magnetic field gradients are also used in a pulsed fashion, where they are switched on and off during the pulse sequence. This is done primarily for spatial localization, including RF pulse slice selection and spatial encoding, which are both described in later chapters. It is important to note here that multiple measurements with slightly different gradient pulses must typically be used to get enough data to form an image.



6.2.3 Data acquisition window (DAQ)

The data acquisition window indicates when signals are being measured and recorded off of the RF receive coils. This must be done after excitation, and is done in conjunction with magnetic field gradients to perform spatial encoding. It is important to remeber when looking at the data acquisition window that it is the **transverse** magnetization (M_{XY}) that is measured.

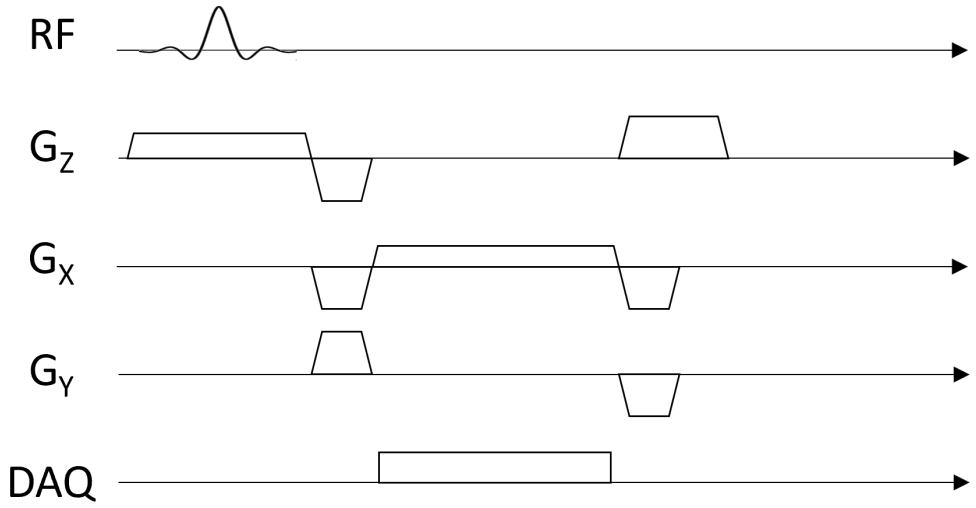


6.3 Pulse Sequence Diagrams

The pulse sequence diagram is an incredible tool for conveying a significant amount of information about a MRI experimental method. It can capture the expected image contrast, the type of spatial encoding, the overall imaging speed, and more.

6.3.1 Sample Pulse Sequence Diagram

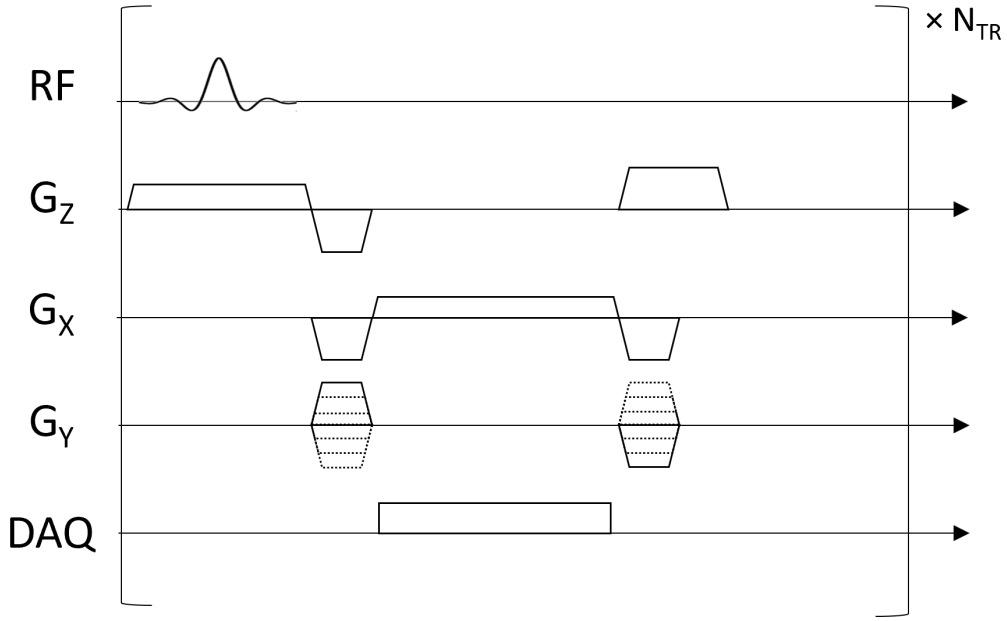
The pulse sequence diagram has lines for each component (RF pulses; magnetic field gradients in X, Y, and Z; data acquisition) to show their behavior over time. A single "block" of the sequence is usually shown, but this block is usually repeated many times (see below).



Note that the RF pulses as drawn to not show that these pulses are actually applied at very high frequencies near Larmor resonance frequency.

6.3.2 Annotations

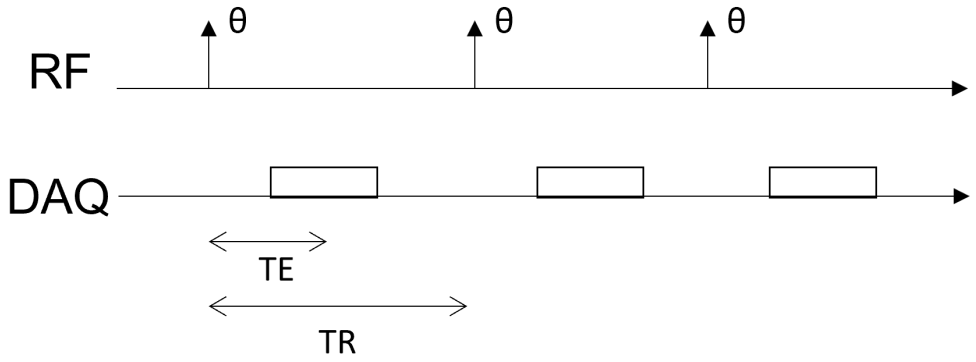
In most every MRI experiment, the basic pulse sequence block is repeated but with a different gradient pattern to perform the necessary spatial encoding. This is typically annotated in 2 ways. First, the changes in gradient patterns can be added through multiple lines on the gradient axes. Second, bracketing around the sequence is sometimes used along with annotation that shows how many times the pulse sequence block is repeated.



6.3.3 Simplified diagrams

For analyzing in particular image contrast and relaxation, it is often helpful to use a simplified pulse sequence diagram that only includes RF pulses and data acquisition windows. Sometimes, simlified representation of gradients are added as well (e.g. crushers to indicate purposeful destruction of transverse magnetization).

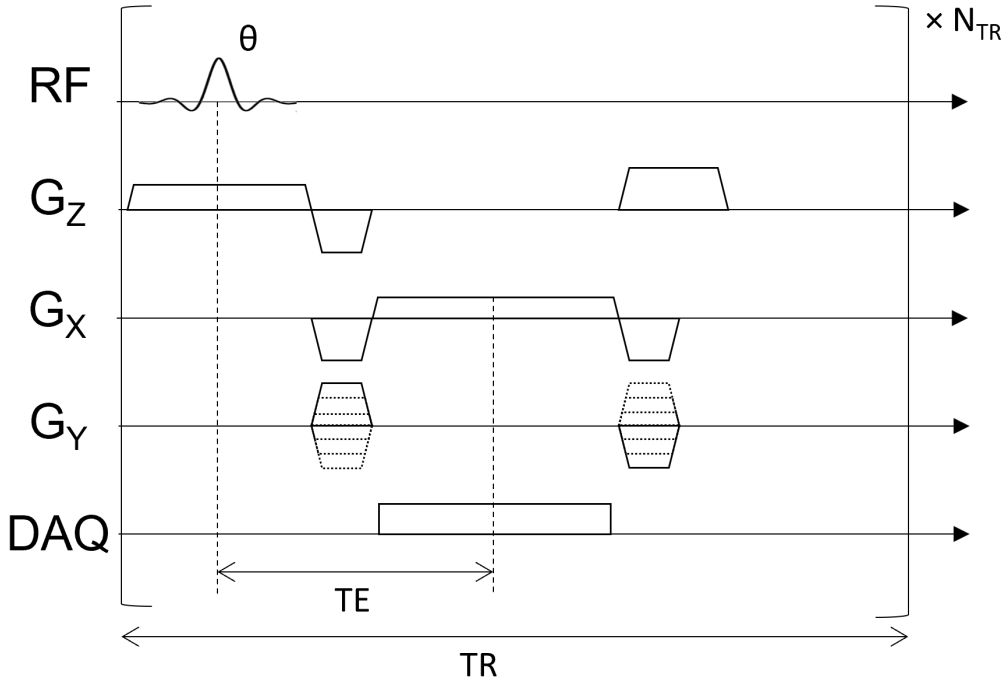
Also, pulse sequence diagrams are often drawn with idealized gradient shapes of rectangles, indicating the magnetic field gradients are instantaneously changed. In reality, there is some minimum time required for magnetic field gradients to change, defined by their "slew rate", which turns the rectangles into trapezoids.



6.4 Key Parameters

The following key parameters are typically defined on a pulse sequence diagram:

- Flip Angle (θ) - how much a RF pulse flips (i.e. rotates or tips) the net magnetization
- Echo Time (TE) - the time between excitation and data acquisition, typically measured between the center of the RF excitation pulse and the data acquisition window
- Repetition Time (TR) - the time between repetitions of the main pulse sequence block
- Number of Repetitions (N_TR) - how many times the pulse sequence block must be repeated
- Readout time (T_{read}) - the time the data acquisition window is open within a single repetition
- Active time (T_{active}) - the time required for all RF pulses and gradients to be applied within a single repetition. This determines the minimum TR.


θ
× N_{TR}
RF
G_Z
G_X
G_Y
DAQ
TE
TR

CHAPTER

SEVEN

THE MRI SIGNAL EQUATION

The MRI signal equation can be used to understand how images are formed as well as the influence of other factors on contrast, image quality and artifacts. It is derived from the Bloch equation and can incorporate the effects of RF coils and other in vivo spin physics. The signal equation can be simplified in many ways, which we will use throughout this book. Here we build up a comprehensive MRI signal equation.

7.1 Learning Goals

1. Understand what MRI is measuring
    - Describe what produces the MRI signal
    - Describe how gradients influence MRI signal
    - Describe how in vivo spin physics phenomena influence MRI signal

7.2 The Transverse Magnetization

The signal in MRI comes from the precession of the transverse component of the net magnetization, which creates a time-varying magnetic field that induces a voltage in the RF receive coil. The signal comes from the magnetization across a large volume:

$$s(t) = \int M_{XY}(\vec{r},t) \ d\vec{r}$$

For simplifaction, we use the notation

$$m(\vec{r}) = M_{XY}(\vec{r}, t=TE)$$

which indicates that our signal is primarily determined by the transverse magnetization at the echo time, TE, and assumes the transverse magnetization does not change during data acquisition (we will add this back in below as needed). Then we get a simple expression:

$$s(t) = \int m(\vec{r}) \ d\vec{r}$$

Here we also use the rotating frame.

7.3 Adding Relaxation

The transverse magnetization decays due to relaxation, which can be added to the signal equation:

$$s(t) = \int m(\vec{r})\ e^{-t/T_2(\vec{r})}\ d\vec{r}$$

7.4 Adding Off-resonance and Chemical Shift

Off-resonance and chemical shift lead to changes in the precession frequency of the magnetization. We can add this onto the idealized signal equation.

Here we separately define:

- $\Delta f_r(\vec{r}) = \bar \gamma \Delta B_0(\vec{r})$ - magnetic field inhomogeneities, including main field imperfections, and magnetic susceptibility effects
- Δf_{cs} - frequency changes due to chemical shift, so depends on the chemical species being examined. For example, for water, $\Delta f_{cs} =0$, while for fat, $\Delta f_{cs} = \Delta f_{fat}$.

For simplicity, combine all off-resonance and/or chemical shift into a single frequnecy shift term, $\Delta f(\vec{r}) = \Delta f_{cs} + \Delta f_r(\vec{r})$ as

$$s(t) = \int m(\vec{r})\ e^{-i 2 \pi \Delta f(\vec{r}) t} \ d\vec{r}$$

7.5 Adding Gradients

Magnetic field gradients also lead to changes in the precession frequency of the magnetization. Unlike off-resonance and chemical shift, which are static and fixed for a given experiment, the gradients can be controlled and can be varied over time. This leads to rotations in the transverse magnetization that depend on the gradient waveforms and position, and can be added to the signal equation as:

$$s(t) = \int m(\vec{r}) \ \exp\left(-i \gamma \int_0^t \vec{G}(\tau) \cdot \vec{r} \ d\tau \right) \ d\vec{r}$$

The effect of gradients depends on their cumulative area, as described by the integral in the exponential.

7.6 Adding RF Coil Profiles

The RF receive coils are not equally sensitive to the entire volume, but rather have a sensitivity profile that is spatially varying. For example, they are typically more sensitive to the precession of the transverse magnetization near the coil and less sensitive to the magnetization further away.

This sensitivity profile can be incorporated into the signal equation. The fact that MRI usually uses multiple receive elements is added as the received signal for coil element n, which has a coil sensitivity profile $C_n(\vec{r})$.

$$s_n(t) = \int m(\vec{r})\ C_n(\vec{r}) \ d\vec{r}$$

7.7 Complete Signal Equation

Finally, we can write the most intimidating version, a comprehensive version of signal equation, including relaxation, coil sensitivity, off-resonance and chemical shift, and gradients. The signal from an individual RF coil element n is

$$s_n(t) = \int m(\vec{r})\ C_n(\vec{r})\ e^{-t/T_2(\vec{r})}\ e^{-i 2 \pi \Delta f_{cs} t} e^{-i 2 \pi \Delta f_r(\vec{r}) t} \ \exp\left(-i \gamma \int_0^t \vec{G}(\tau) \cdot \vec{r} \ d\tau \right) \ d\vec{r}$$

7.8 Simplified Versions of the Signal Equation

For purposes including developing intuition, designing pulse sequences, formulating the image reconstruction, and understanding artifacts and imperfections, we use simplified versions of the signal equation. A few useful simplifications are described below.

7.8.1 Image Formation with only Gradients

An idealized signal equation only capturing the effects of gradients is commonly used, particularly for understanding image formation. It neglects relaxation, RF coil profile, and off-resonance

$$s(t) = \int m(\vec{r}) \ \exp\left(-i \gamma \int_0^t \vec{G}(\tau) \cdot \vec{r} \ d\tau \right) \ d\vec{r}$$

7.8.2 Relaxation, Off-resonance, and Chemical Shift in Images

Often the effects of the gradients and RF coil volume are neglected for understanding the effects of relaxation and off-resonance on the signal.

$$s(t) = m(\vec{r})\ e^{-t/T_2(\vec{r})}\ e^{-i 2 \pi \Delta f(\vec{r}) t} \ $$

When we perform proper spatial encoding and image reconstruction the effect of gradients and the integration over a volume are effectively removed, and the resulting image is proportional to this simplified version of the signal equation.

Part IV

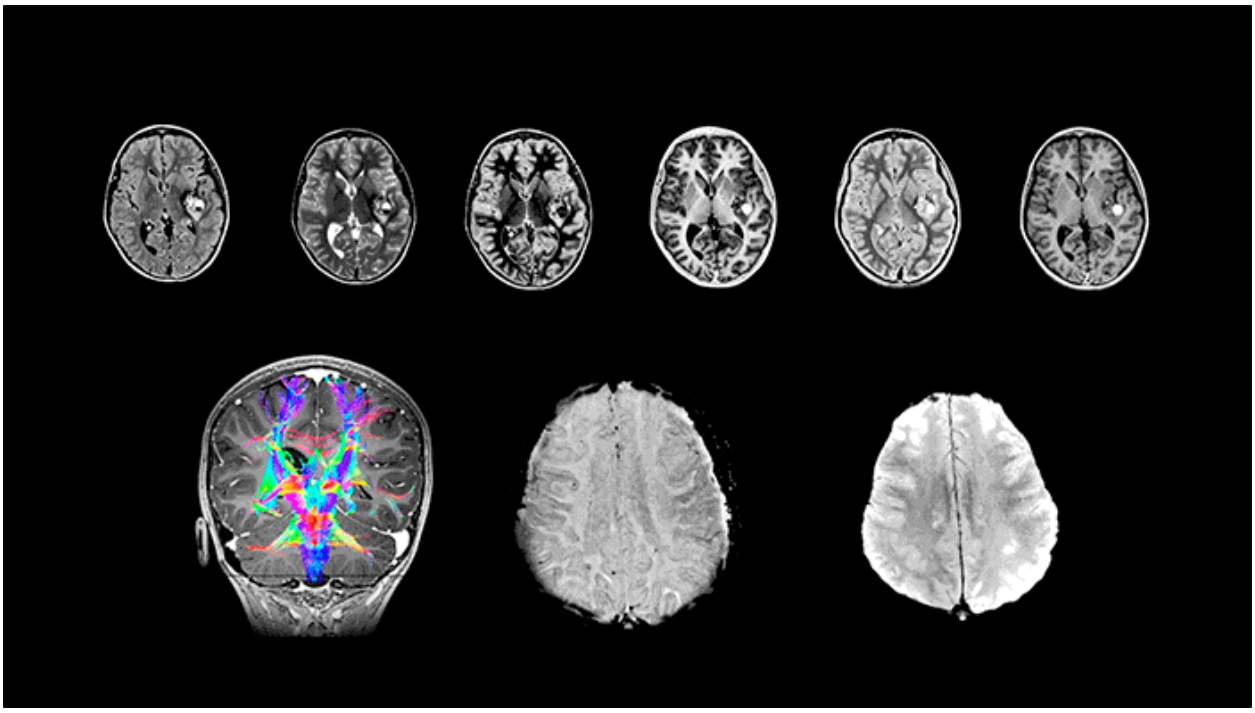
Contrast

CHAPTER

EIGHT

BASIC CONTRAST MECHANISMS

MRI contrast is a rich dimension of information and the variety of contrasts achieveable is arguably the main advantage of MRI. This section introduces the fundamental techniques of T_1, T_2, and proton density weighted contrasts, and magnetization preparation (e.g. inversion recovery).



8.1 Learning Goals

1. Describe how various types of MRI contrast are created
    - Describe how T_1, T_2, and proton density weighted images are formed
    - Describe how inversion recovery works
2. Manipulate MRI sequence parameters to improve performance
    - Choose flip angle, TE, TR, and TI for desired contrast

8.2 What determines T_1 and T_2 values?

T_1 and T_2 are intrinsic properties of materials that are determined by the molecular environment and structure, and provide a significant sources of tissue contrast. While a precise understanding of what determines T_1 and T_2 requires quantum mechanics, typical patterns in these values can be described based on typical types of tissues found in the body. human tissues can be approximately be divided into 4 classes of tissues that have similar T_1 and T_2 values:

1. Liquids/fluids
2. Solids (e.g. Cortical bone, tendons, fibrosis)
3. Fat
4. Proteinaceous materials (e.g. internal organs)

8.2.1 T_1

T_1 is also known as the spin-lattice relaxation time. T_1 is determined by the exchange of energy between spins and the surrounding "lattice" - the molecular structure around the spins. The lattice is in constant motion, and will have characteristic motional frequencies. The most efficient transfer of energy occurs when the natural motional frequencies are near the Larmor frequency ($f_0 = \bar{\gamma} B_0$), and natural motional frequencies depend on physical states of tissues:

1. Liquids/fluids: **Longer T_1** due to higher natural motional frequencies of protons
2. Solids (e.g. Cortical bone, tendons, fibrosis): **Intermediate T_1** due to lower natural motional frequencies of protons
3. Fat (lipids): **Shorter T_1** since natural motion frequencies are similar to f_0
4. Proteinaceous materials (e.g. internal organs): **Intermediate T_1** due to lower natural motional frequencies of protons
5. Proteinaceous fluids (e.g. Edema): **Shorter T_1** since water protons lose freedom of motion due to macromolecules, so natural motion frequencies are similar to f_0

8.2.2 T_2

T_2 is also known as the spin-spin relaxation time. T_2 is determined by coupling between adjacent spins. This also includes the transfer of energy to the lattice that determines T_1, meaning that T_2 is always shorter that T_1. Stronger coupling between spins leads to more microscopic dephasing and shorter T_2 values.

1. Liquids/fluids: **Longer T_2** due to sparsity between water molecules and limited interaction between protons within a water molecule
2. Solids (e.g. Cortical bone, tendons, fibrosis): **Shorter T_2** due to strong, fixed spin-spin coupling
3. Fat (lipids): **Intermediate T_2** due to intermediate coupling
4. Proteinaceous materials (e.g. internal organs): **Intermediate T_2** due to intermediate coupling

8.2.3 Typical T_1 and T_2 values

1.5 T

Tissue	T_1 (ms)	T_2 (ms)
Brain white matter	790	90
Brain gray matter	920	100
Cerebrospinal fluid (CSF)	>4000	~2000
Blood (arterial)	1200	50
Liver parenchyma	490	40
Lung	830	80
Myocardium	870	60
Skeletal muscle	870	50
Lipids	260	80

Source: Handbook of MRI Pulse Sequences

3 T

Tissue	T_1 [ms]	T_2 [ms]
White Matter	800	110
Gray Matter	1300	80
CSF	3700	1700
Muscle	1000	30
Fat	400	100
Liver	800	40
Cartilage	1200	40

Sources: MRI from Picture to Proton, MRI: The Basics

8.3 TE and T_2/T_2^*

The transverse magnetization decays with T_2 (spin-echoes) or $T_2^$ *(gradient-echoes) once it is flipped by an RF pulse away from the z-axis. Therefore, the T_2/T_2** contrast can be controlled by choosing the echo time (TE), which is the time between the center of the RF excitation pulse and the center of the data acquisition (when the center of k-space is acquired). The signal is then proportional to

$$S_{GE} \propto M_0 \exp(-TE/T_2^*)$$

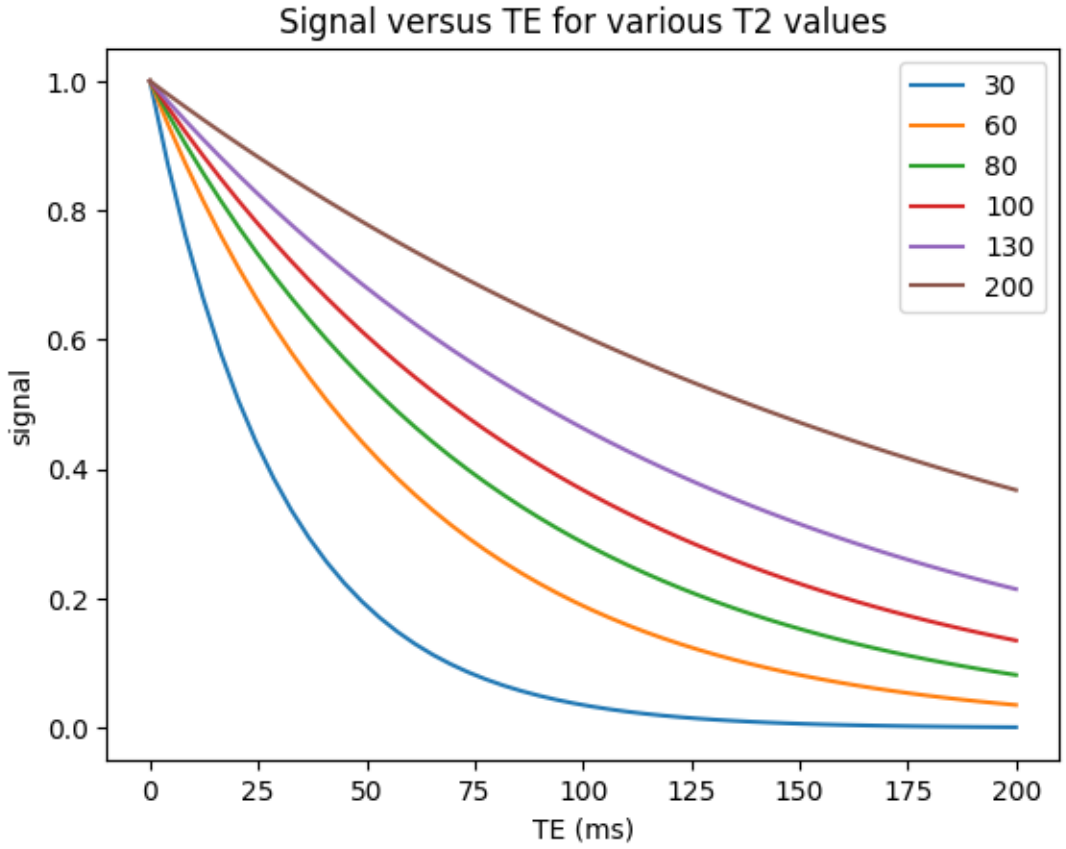
$$S_{SE} \propto M_0 \exp(-TE/T_2)$$

```
import numpy as np
import matplotlib.pyplot as plt

# TE contrast

TE = np.linspace(0, 200)  # ms
T2v = np.array([30, 60, 80, 100, 130, 200])
```

(continues on next page)

(continued from previous page)

```
T2, TE_grid = np.meshgrid(T2v, TE)
M0 = np.ones_like(T2)

S_TE = M0 * np.exp(-TE_grid / T2)

plt.figure()
plt.plot(TE, S_TE)
plt.xlabel('TE (ms)')
plt.ylabel('signal')
plt.legend([str(t2) for t2 in T2v])
plt.title('Signal versus TE for various T2 values')
plt.show()
```



8.4 TR and T_1

The longitudinal magnetization recovers with T_1 back to its equilibrium amplitude, M_0. However, we do not directly measure the longitudinal magnetization. In order to create T_1 contrast in the MR signal, repeated RF pulses are applied with a given repetition time (TR), such that there is incomplete recovery of the longitudinal magnetization. Then, after excitation, incomplete recovery appears in the transverse magnetization, creating T_1 contrast.

Note that the following signal equations assume perfect spoiling of transverse magnetization before the next RF pulse. This means that $M_{XY} = 0$ before each RF pulse. In practice this is done using an extra gradient at the end of the TR ("gradient spoiling") and/or modulating the phase of the RF excitation pulse ("RF spoiling"). See *Fast gradient-echo sequences* for more details.

8.4.1 90-degree flip angles

The simplest case is using 90-degree flip angles every TR, in which case

$$S \propto M_0 (1- \exp(-TR/T_1))$$

Illustrated in the first example below. This shows that the magnetization reaches steady state in the 2nd TR.

8.4.2 < 90-degree flip angles

For T_1-weighting, it is generally more efficient in terms of signal acqruied per time to use < 90-degree flip angles every TR, in which case

$$S \propto M_0 \sin(\theta) \frac{1- \exp(-TR/T_1)}{1- \cos(\theta) \exp(-TR/T_1)}$$

Illustrated in this second example below. This shows that the magnetization can take many TRs to reach steady state. These acquisitions are commonly done to maximize signal for a typical T_1 range. This is done by choosing using the so-called "Ernst angle", which is the flip angle that maximizes the signal for a given T_1 and TR:

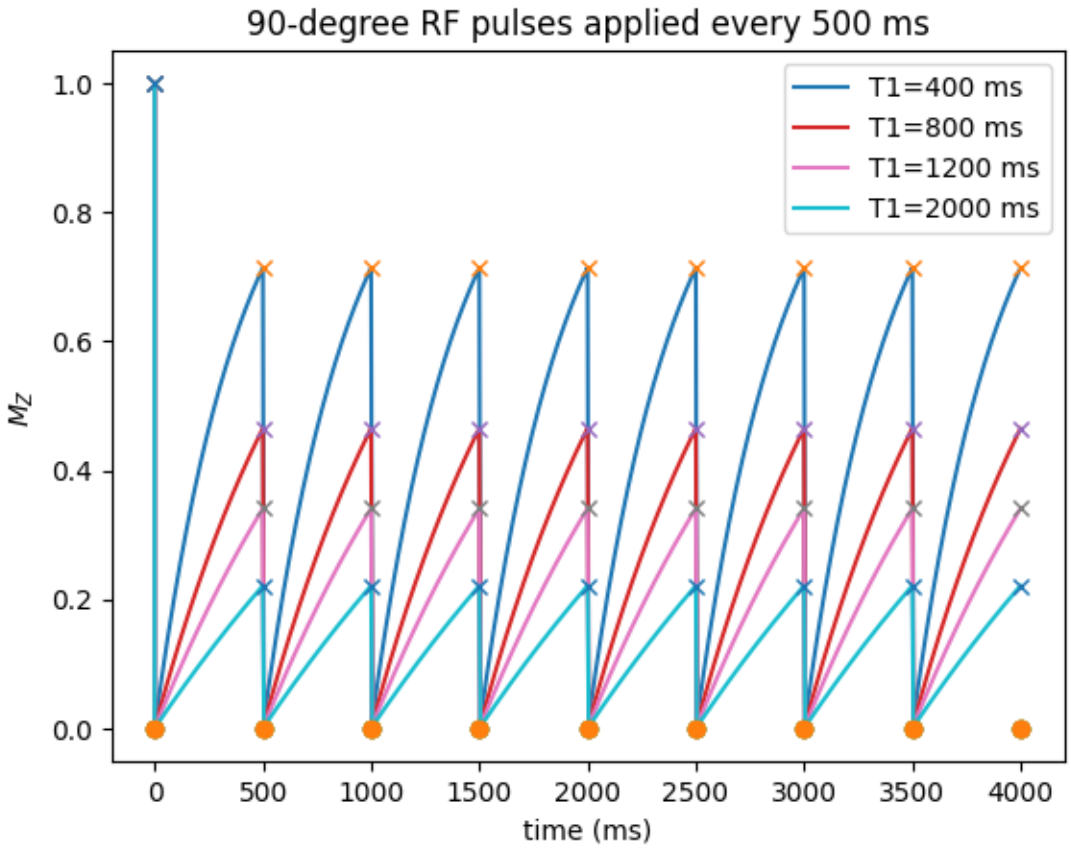
$$\theta_{optimal} = \cos^{-1}(\exp(-TR/T_1))$$

```
import numpy as np

import matplotlib.pyplot as plt

# Signal evolution between TRs with 90-degree pulses

M0 = np.ones(4)  # 4 tissue types
T1 = np.array([400, 800, 1200, 2000])  # ms

NTR = 8
flip = 90  # degrees
TR = 500  # ms
Nt_per_TR = 100
t_per_TR = np.arange(1, Nt_per_TR + 1) * TR / Nt_per_TR
t_minus = np.arange(0, NTR + 1) * TR
t_plus = t_minus + 1

# magnetization before each RF pulse
Mz_minus = np.zeros((len(T1), NTR + 1))
# magnetization after each RF pulse
Mz_plus = np.zeros((len(T1), NTR + 1))

# initial condition
Mz_minus[:, 0] = M0
Mz_plus[:, 0] = Mz_minus[:, 0] * np.cos(np.deg2rad(flip))
t = [0, np.finfo(float).eps]
Mz_all = [Mz_minus[:, 0], Mz_plus[:, 0]]

for I in range(NTR):
    t = np.concatenate([t, t_per_TR + I * TR])
    for It in range(Nt_per_TR):
        Mz_all.append(
            Mz_plus[:, I] * np.exp(-t_per_TR[It] / T1) + M0 * (1 - np.exp(-t_per_
↪TR[It] / T1))
        )
    Mz_minus[:, I + 1] = Mz_plus[:, I] * np.exp(-TR / T1) + M0 * (1 - np.exp(-TR /␣
↪T1))
```

(continues on next page)

(continued from previous page)

```
    # apply RF pulse - assumes spoiling of the transverse magnetization
    Mz_plus[:, I + 1] = Mz_minus[:, I + 1] * np.cos(np.deg2rad(flip))

Mz_all = np.column_stack(Mz_all)

plt.figure()
for i in range(len(T1)):
    plt.plot(t, Mz_all[i, :], label=f"T1={T1[i]} ms")
    plt.plot(t_minus, Mz_minus[i, :], 'x')
    plt.plot(t_plus, Mz_plus[i, :], 'o')
plt.legend()
plt.xlabel('time (ms)')
plt.ylabel(r'$M_Z$')
plt.title(f'{flip}-degree RF pulses applied every {TR} ms')
plt.show()
```

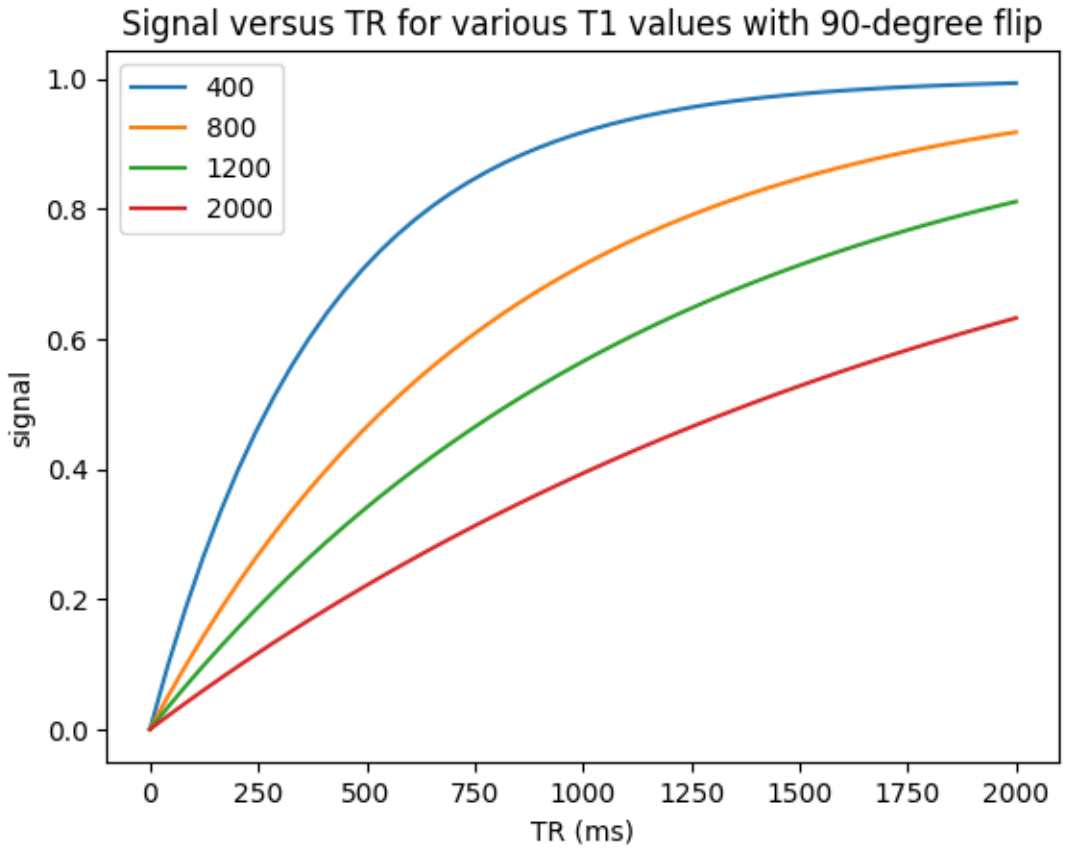



```
import numpy as np

import matplotlib.pyplot as plt

TR_vals = np.linspace(0, 2000)  # ms
T1v = np.array([400, 800, 1200, 2000])  # ms
T1, TR_grid = np.meshgrid(T1v, TR_vals)
M0 = np.ones_like(T1)

S_TR = M0 * (1 - np.exp(-TR_grid / T1))
```

(continues on next page)

(continued from previous page)

```
plt.figure()
plt.plot(TR_vals, S_TR)
plt.xlabel('TR (ms)')
plt.ylabel('signal')
plt.legend([str(t1) for t1 in T1v])
plt.title(f'Signal versus TR for various T1 values with {flip}-degree flip')
plt.show()
```

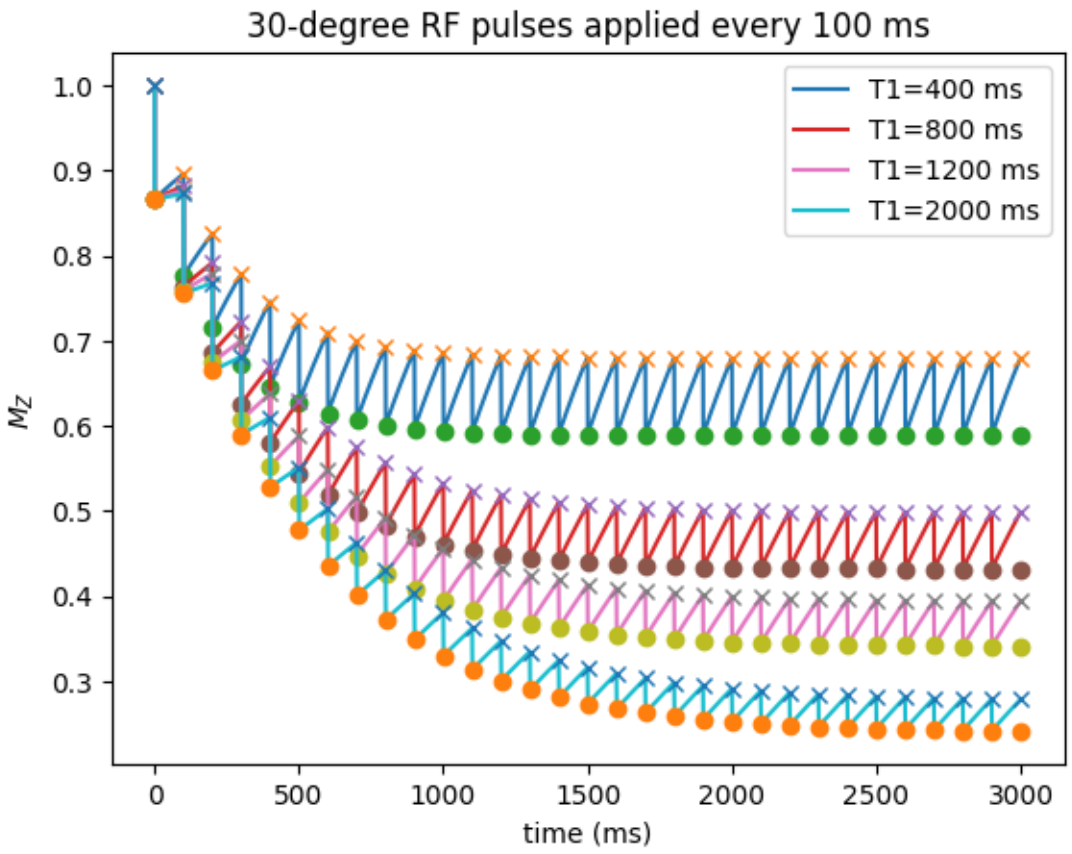



```
import numpy as np

import matplotlib.pyplot as plt

# Signal evolution between TRs with <90-degree pulses

M0 = np.ones(4)  # 4 tissue types
T1v = np.array([400, 800, 1200, 2000])  # ms

NTR = 30
flip = 30  # degrees
TR = 100  # ms
Nt_per_TR = 100
t_per_TR = np.arange(1, Nt_per_TR + 1) * TR / Nt_per_TR
t_minus = np.arange(0, NTR + 1) * TR
t_plus = t_minus + 1
```

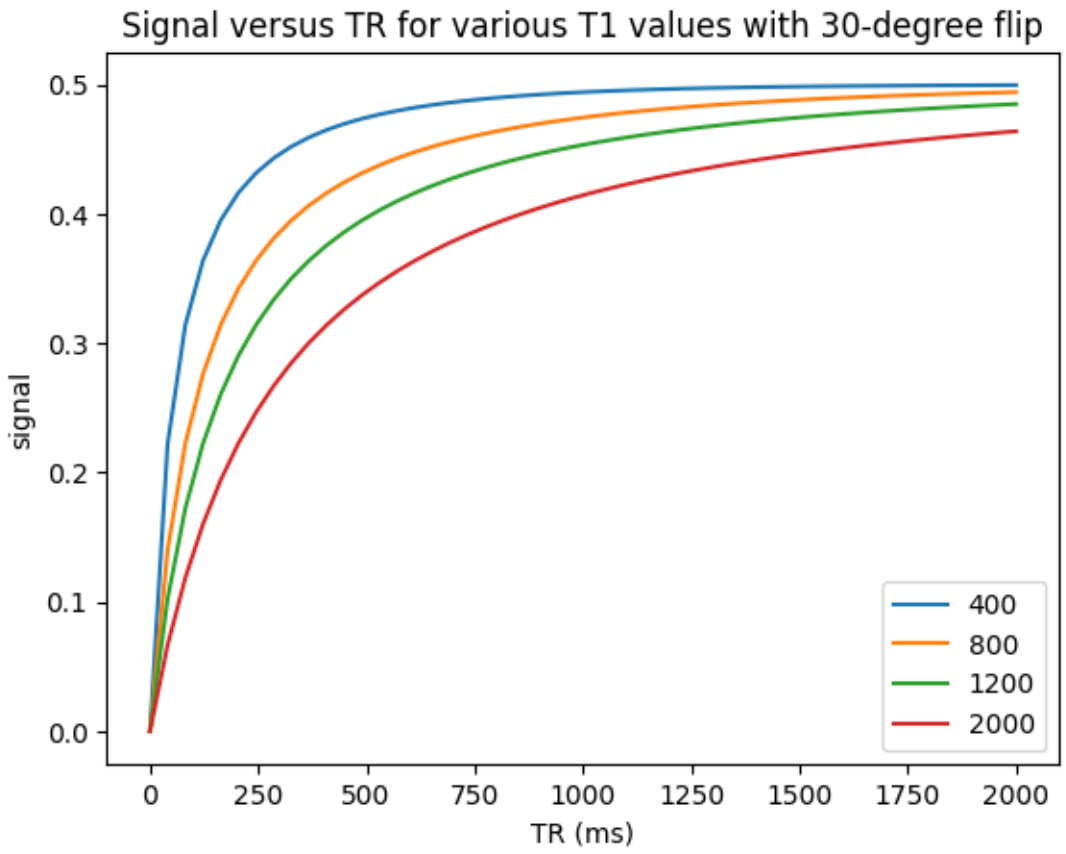
(continues on next page)

(continued from previous page)

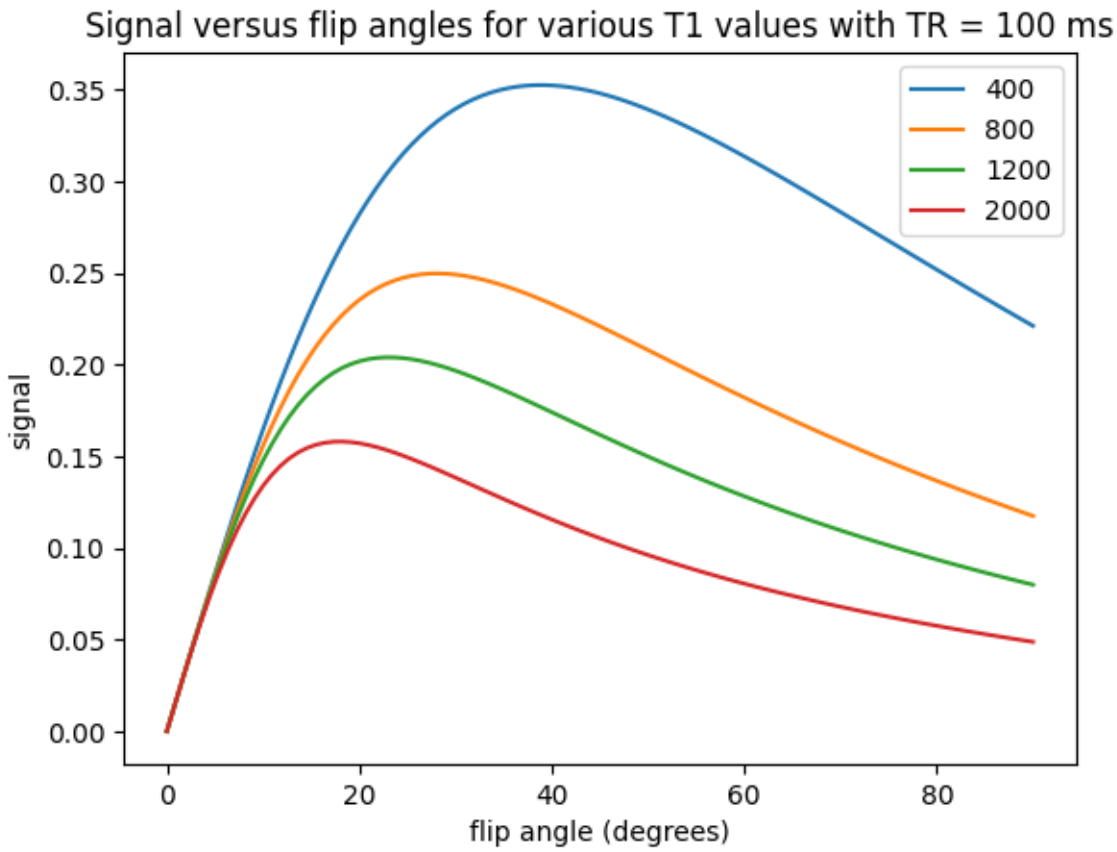
```
# magnetization before each RF pulse
Mz_minus = np.zeros((len(T1v), NTR + 1))
# magnetization after each RF pulse
Mz_plus = np.zeros((len(T1v), NTR + 1))

# initial condition
Mz_minus[:, 0] = M0
Mz_plus[:, 0] = Mz_minus[:, 0] * np.cos(np.deg2rad(flip))
t = [0, np.finfo(float).eps]
Mz_all = [Mz_minus[:, 0], Mz_plus[:, 0]]

for I in range(NTR):
    t = np.concatenate([t, t_per_TR + I * TR])
    for It in range(Nt_per_TR):
        Mz_all.append(
            Mz_plus[:, I] * np.exp(-t_per_TR[It] / T1v) + M0 * (1 - np.exp(-t_per_
 ↪TR[It] / T1v))
        )
    Mz_minus[:, I + 1] = Mz_plus[:, I] * np.exp(-TR / T1v) + M0 * (1 - np.exp(-TR /␣
 ↪T1v))
    Mz_plus[:, I + 1] = Mz_minus[:, I + 1] * np.cos(np.deg2rad(flip))

Mz_all = np.column_stack(Mz_all)

plt.figure()
for i in range(len(T1v)):
    plt.plot(t, Mz_all[i, :], label=f"T1={T1v[i]} ms")
    plt.plot(t_minus, Mz_minus[i, :], 'x')
    plt.plot(t_plus, Mz_plus[i, :], 'o')
plt.legend()
plt.xlabel('time (ms)')
plt.ylabel(r'$M_Z$')
plt.title(f'{flip}-degree RF pulses applied every {TR} ms')
plt.show()
```




```
import numpy as np

import matplotlib.pyplot as plt

flip_angle = 30  # degrees
TR_vals = np.linspace(0, 2000, 50)  # ms, 50 points for smoothness
T1v = np.array([400, 800, 1200, 2000])  # ms

T1, TR_grid = np.meshgrid(T1v, TR_vals)
M0 = np.ones_like(T1)

S_TR = np.sin(np.deg2rad(flip_angle)) * M0 * (1 - np.exp(-TR_grid / T1)) / (1 - np.
↪cos(np.deg2rad(flip_angle)) * np.exp(-TR_grid / T1))

plt.figure()
plt.plot(TR_vals, S_TR)
plt.xlabel('TR (ms)')
plt.ylabel('signal')
plt.legend([str(t1) for t1 in T1v])
plt.title(f'Signal versus TR for various T1 values with {flip_angle}-degree flip')
plt.show()
```




```
import numpy as np

import matplotlib.pyplot as plt

TR = 100
flip = np.linspace(0, 90, 100)  # degrees
T1v = np.array([400, 800, 1200, 2000])  # ms
T1, flip_grid = np.meshgrid(T1v, flip)
M0 = np.ones_like(T1)

S_flip = np.sin(np.deg2rad(flip_grid)) * M0 * (1 - np.exp(-TR / T1)) / (1 - np.cos(np.
 ↪deg2rad(flip_grid)) * np.exp(-TR / T1))

plt.figure()
plt.plot(flip, S_flip)
plt.xlabel('flip angle (degrees)')
plt.ylabel('signal')
plt.legend([str(t1) for t1 in T1v])
plt.title(f'Signal versus flip angles for various T1 values with TR = {TR} ms')
plt.show()
```



8.5 T_1, T_2 and Proton Density Weighting

By choosing the TE, TR and flip angle values in our pulse sequences and the relationships described above, we can create images that "weighted" by the MR properties of T_1, T_2, and proton density. This means that the resulting images will have contrast between tissues that have different T_1, T_2 or proton density.

	Short TE	Long TE
Short TR	T_1-weighted	Not used
Long TR	PD-weighted	T_2/T_2^*-weighted

8.5.1 T_2-weighting and T_2^*-weighting

To create T_2 weighting, the TE is chosen to be long enough such that the there are different amounts of transverse magnetization depending on the T_2 value. For example, at TE = 80 ms we can expect to see good contrast between white matter ($T_2 \approx 60$ ms), gray matter ($T_2 \approx 80$ ms), and cerebrospinal fluid ($T_2 \approx 1500$ ms) in the brain (see simulations above).

The TR is chosen to be long relative to T_1 values. This allows the net magnetization to fully recover each repetition, and thereby eliminating any T_1-weighting.

In T_2/$T_2^$ *weighting, longer T_2/T_2^** tissues have higher signal than shorter T_2/T_2^* tissues.

The difference between T_2 and $T_2^$ *weighting depends on the choice of pulse sequence: T_2-weighting comes from spin-echoes while T_2**-weighting comes from gradient-echoes. These are discussed later. The same principle applies of choosing a long enough TE, and a long TR.

8.5.2 T_1-weighting

To create T_1 weighting, the TR is chosen to be short enough such that there is incomplete recovery of the net magnetization each repetition. This means that the signal acquired will depend on T_1. For example. choosing a TR = 500 ms with a 90-degree flip angle we can expect to see contrast between fat ($T_1 \approx 400$ ms) and muscle ($T_1 \approx 1200$ ms) (see simulations above).

For T_1-weighting, we also can use smaller flip angles and smaller TRs to boost the overall signal. For example. choosing a TR = 200 ms with a 30-degree flip angle we can expect to see contrast between fat ($T_1 \approx 400$ ms) and muscle ($T_1 \approx 1200$ ms) but more overall signal (see simulations above).

The TE is chosen to be short relative to T_2 values. This reduces T_2-weighting.

In T_1 weighting, shorter T_1 tissues have higher signal than longer T_1 tissues.

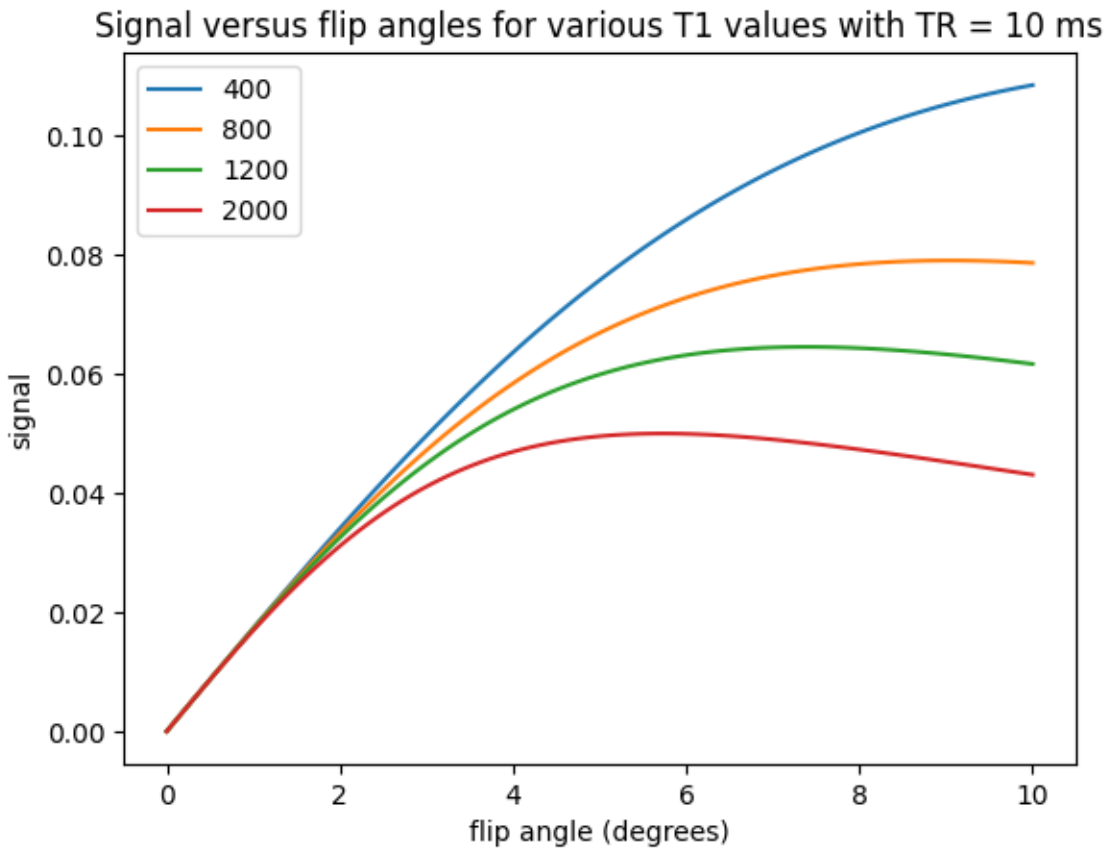
8.5.3 PD-weighting

To create proton density (PD) weighting, we choose a short TE to reduce T_2-weighting and a long TR to reduce T_1-weighting. This leaves the image contrast dependent on the proton density, which is captured in the equilibrium magnetization, M_0.

In PD weighting, higher PD tissues have higher signal than lower PD tissues.

Proton-density weighting can be achieved with short TRs when using very small flip angles. This weighting applies up to a certain TR/T_1 ratio. This can be seen in the following by zooming in on the low flip angle area from the plots above and noting that there is no difference between the signal for different T_1 values, meaning the T_1-weighting has been removed.

```
TR = 10
flip = np.linspace(0, 10, 100)  # degrees, 100 points for smoothness
T1v = np.array([400, 800, 1200, 2000])  # ms
T1, flip_grid = np.meshgrid(T1v, flip)
M0 = np.ones_like(T1)

S_flip = np.sin(np.deg2rad(flip_grid)) * M0 * (1 - np.exp(-TR / T1)) / (1 - np.cos(np.
 ↪deg2rad(flip_grid)) * np.exp(-TR / T1))

plt.figure()
plt.plot(flip, S_flip)
plt.xlabel('flip angle (degrees)')
plt.ylabel('signal')
plt.legend([str(t1) for t1 in T1v])
plt.title(f'Signal versus flip angles for various T1 values with TR = {TR} ms')
plt.show()
```



8.6 Inversion Recovery

Another way to create T_1 contrast are "Inversion Recovery" techniques. These use a 180-degree inversion pulse, following by an Inversion Time TI delay during which T_1 contrast is created. Then the magnetization is excited with a 90-degree pulse.

This strategy is commonly used to null tissue types. For example, so-called short-time inversion recovery (STIR) is used to null fat signals, while fluid attenuated inversion recovery (FLAIR) is used to null fluids.

$$S_{IR} \propto M_0 (1 - 2\exp(-TI/T_1) + \exp(-TR/T_1))$$

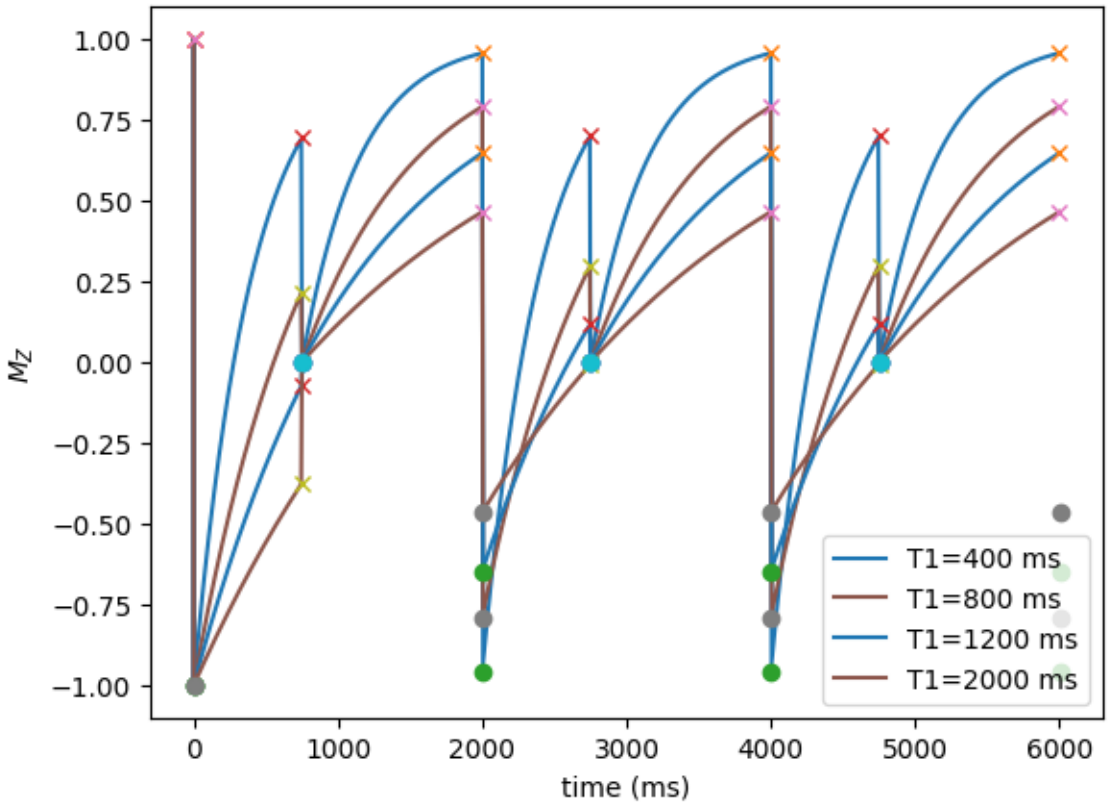
The illustration below shows that the magnetization reaches steady state in the 2nd TR of inversion recovery.

```
import numpy as np

import matplotlib.pyplot as plt

# Inversion Recovery

T1 = np.array([400, 800, 1200, 2000])  # ms
M0 = np.ones_like(T1)

NTR = 3
flip1 = 180  # degrees
flip2 = 90   # degrees
TR = 2000    # ms
```

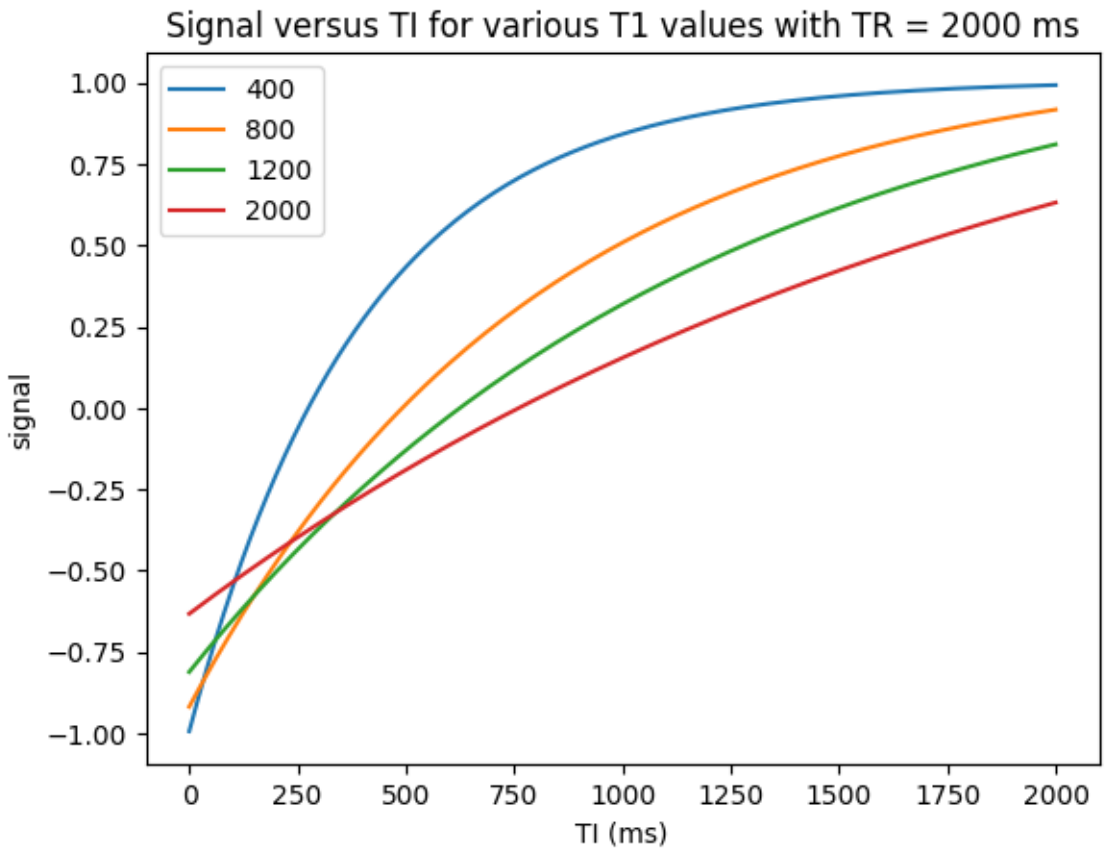
(continues on next page)

(continued from previous page)

```
TI = 750      # ms

dt = 5        # ms
t_per_TR = np.arange(dt, TR + dt, dt)

# magnetization before each RF pulse
Mz1_minus = np.zeros((len(T1), NTR + 1))
Mz2_minus = np.zeros((len(T1), NTR))

# magnetization after each RF pulse
Mz1_plus = np.zeros((len(T1), NTR + 1))
Mz2_plus = np.zeros((len(T1), NTR))

# time of RF pulses
t1_minus = np.arange(NTR + 1) * TR
t1_plus = t1_minus + 1
t2_minus = t1_minus[:-1] + TI
t2_plus = t1_plus[:-1] + TI

# initial condition
Mz1_minus[:, 0] = M0
Mz1_plus[:, 0] = Mz1_minus[:, 0] * np.cos(np.deg2rad(flip1))
Mz_all = [Mz1_minus[:, 0], Mz1_plus[:, 0]]
t = [0, np.finfo(float).eps]

for I in range(NTR):
    t = np.concatenate([t, t_per_TR + I * TR])
    # evolve for TI period after 180-pulse
    for It in np.where(t_per_TR < TI)[0]:
        Mz_all.append(Mz1_plus[:, I] * np.exp(-t_per_TR[It] / T1) + M0 * (1 - np.exp(-
 ↪t_per_TR[It] / T1)))
    Mz2_minus[:, I] = Mz1_plus[:, I] * np.exp(-TI / T1) + M0 * (1 - np.exp(-TI / T1))
    Mz2_plus[:, I] = Mz2_minus[:, I] * np.cos(np.deg2rad(flip2))
    # evolve for the rest of TR after 90-pulse
    for It in np.where(t_per_TR >= TI)[0]:
        Mz_all.append(Mz2_plus[:, I] * np.exp(-(t_per_TR[It] - TI) / T1) + M0 * (1 -␣
 ↪np.exp(-(t_per_TR[It] - TI) / T1)))
    Mz1_minus[:, I + 1] = Mz2_plus[:, I] * np.exp(-(TR - TI) / T1) + M0 * (1 - np.
 ↪exp(-(TR - TI) / T1))
    Mz1_plus[:, I + 1] = Mz1_minus[:, I + 1] * np.cos(np.deg2rad(flip1))

Mz_all = np.column_stack(Mz_all)

plt.figure()
for i in range(len(T1)):
    plt.plot(t, Mz_all[i, :], label=f"T1={T1[i]} ms")
    plt.plot(t1_minus, Mz1_minus[i, :], 'x')
    plt.plot(t1_plus, Mz1_plus[i, :], 'o')
    plt.plot(t2_minus, Mz2_minus[i, :], 'x')
    plt.plot(t2_plus, Mz2_plus[i, :], 'o')
plt.legend()
plt.xlabel('time (ms)')
plt.ylabel(r'$M_Z$')
plt.show()
```




```python
import numpy as np

import matplotlib.pyplot as plt

TR = 2000  # ms
TI = np.linspace(0, 2000, 200)  # ms, 200 points for smoothness
T1v = np.array([400, 800, 1200, 2000])  # ms
T1, TI_grid = np.meshgrid(T1v, TI)
M0 = np.ones_like(T1)

S_TI = M0 * (1 - 2 * np.exp(-TI_grid / T1) + np.exp(-TR / T1))

plt.figure()
plt.plot(TI, S_TI)
plt.xlabel('TI (ms)')
plt.ylabel('signal')
plt.legend([str(t1) for t1 in T1v])
plt.title(f'Signal versus TI for various T1 values with TR = {TR} ms')
plt.show()
```


Signal versus TI for various T1 values with TR = 2000 ms
400
800
1200
2000
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
signal
0
250
500
750
1000
1250
1500
1750
2000
TI (ms)

CHAPTER

NINE

IN VIVO CONTRAST MECHANISMS

In addition to the basic contrast mechanisms based on T1, T2 and proton density, in vivo tissues and interactions create additional sources of contrast. These include T2*, susceptibility, fat versus water, and admisistration of contrast agents. Additional important sources of contrast in vivo include flow and magnetization transfer, but these are not covered here.

9.1 Learning Goals

1. Describe how various types of MRI contrast are created
    - Describe the mechanisms that create susceptibility contrast
    - Describe the mechanisms that create T2* relaxation
    - Describe how fat/water imaging works
    - Describe how Gd-based contrast agents work

9.2 Off-resonance and Chemical Shift

9.2.1 Off-resonance

Ideally, the magnetic field is uniform in the absense of any applied gradients. However, in practice there are unavoidable variations in the magnetic field that lead to changes in resonance frequency. These magnetic field variations are referred to as “off-resonance”, and can be represented as

$$\Delta f_r(\vec{r}) = \bar \gamma \Delta B_0(\vec{r}) $$

These are due imperfections in the main B_0 magnet as well as subject-specific changes in the magnetic field that are largely due to magnetic susceptibility effects.

9.2.2 Chemical Shift

The resonance frequency can also change due the chemical environment of the nuclei. This is referred to as chemical shift, and is represented as

$$\Delta f_{cs} = \delta_{cs} \bar \gamma B_0$$

where δ_{cs} is the chemical shift in parts per million (ppm) and Δf_{cs} is the frequency shift.

9.3 Susceptibility Induced Contrasts

Differences in magnetic susceptibility between materials and tissues create differences in the local magnetic field. This leads to differences in the resonance frequency of the spins, which can be used to create contrast in MRI images. Local changes in resonance frequency can also lead to changes in measured transverse relaxation time, and this phenomenon is referred to as T_2^* relaxation.

9.3.1 Phase Contrast

The off-resonance causes the transverse magnetization to precess at a different rate. When imaging is performed, the transverse magnetization will have a different phase depending on this off-resonance. This difference in phase can be detected, and leads to phase differences in the resulting images. When trying to create susceptibility contrast, other sources of phase differences must be removed. These other sources of phase differences include imperfections in the main magnetic field, chemical shift effects, and RF coil induced phase.

9.3.2 T_2^*

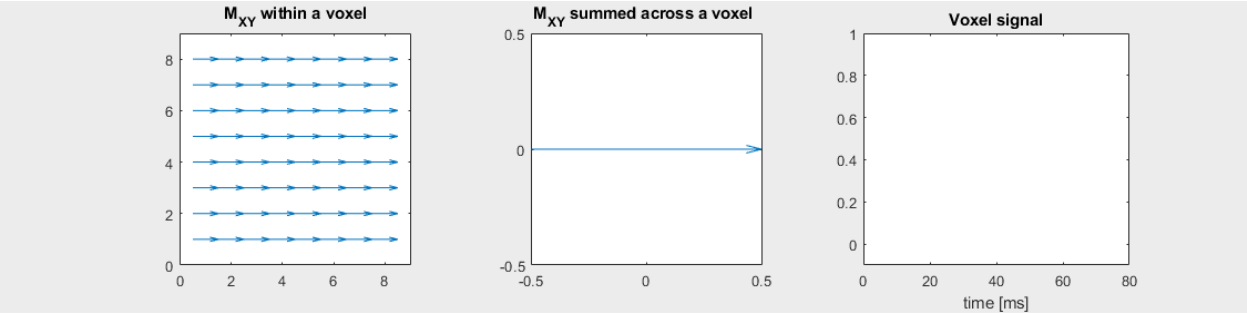
Ideally, the MRI signal decays with a time constant of T_2, the transverse decay rate. However, in practice every imaging voxel contains off-resonance that leads to changes in the signal decay. We characterize this change in signal decay with T_2^*.

The smallest size we can measure with MRI is an imaging voxel, so our image measurements are a sum of the transverse magnetizations across the voxel, including both T_2 and off-resonance:

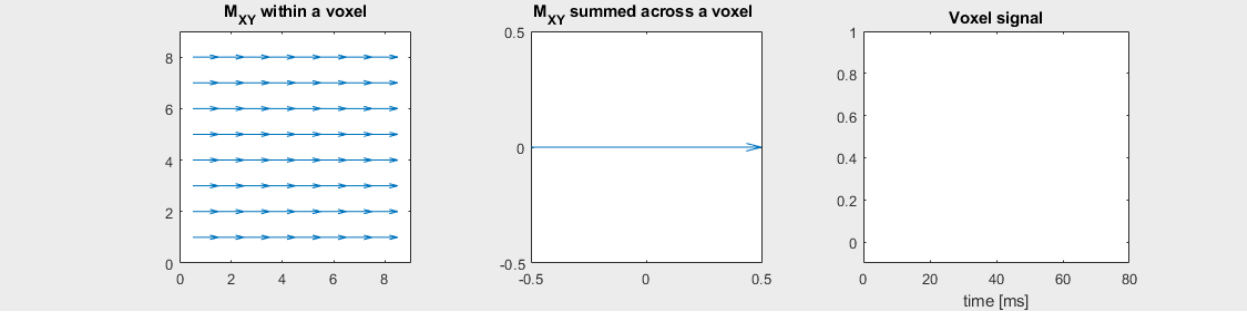
$$\begin{aligned} s_{\mathrm{voxel}}(t) & = \int_{\mathrm{voxel}} M_{XY}(\vec{r},0) \exp(-t/T_2(\vec{r})) \exp(-i 2 \pi \Delta f_r(\vec{r})) \ d\vec{r} \ & \approx M_{XY}(\vec{r},0) \exp(-t/T_2^*(\vec{r})) \end{aligned}$$

(Note that any effects of applied gradients are removed during the image reconstruction process.)

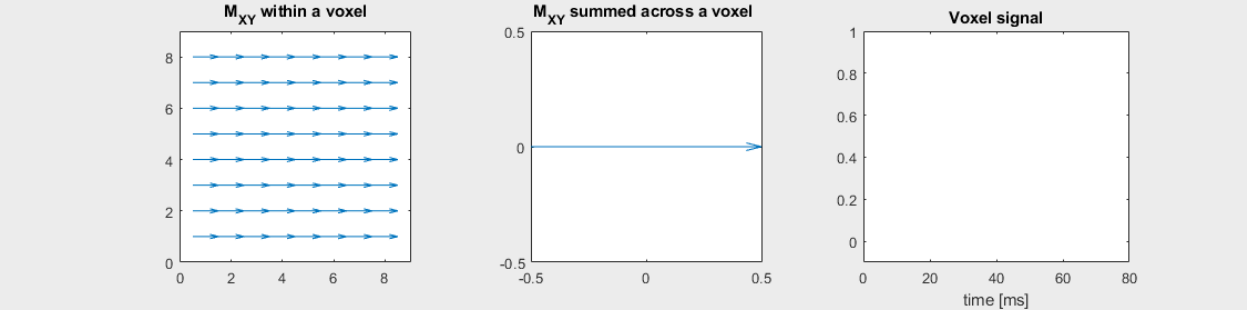
The mechanism of this decay is that, across an imaging voxel, the transverse magnetizations across the voxel precess at slightly different rates. This is called “de-phasing”, and means that when the transverse magnetization is averaged across the voxel it’s overall magnitude is reduced. This is illustrated in the examples below (generated by `t2star_spinecho_illustration.m`)



No off-resonance, $T_2 = 80$ ms



Mild off-resonance, $T_2 = 80$ ms



Severe off-resonance, $T_2 = 80$ ms

T_2^* is an approximation of these dephasing effects, and depends on the voxel size and location. It is often broken up as follows to separate out the additional dephasing rate, $1/T_2'$:

$$ \frac{1}{T_2^*} = \frac{1}{T_2}+\frac{1}{T_2'}$$

$$ T_2^* \leq T_2$$

9.3.3 T_2^* contrast

T_2^* includes T_2 as well as additional macroscopic dephasing due to local magnetic field inhomogeneities. The microscopic dephasing (T_2) cannot be refocused, but the macroscopic dephasing effects can be refocused by using a spin echo sequence. The macroscopic dephasing is caused by differences in magnetic susceptibility between materials and tissues.

While often the macroscopic dephasing effects creating $T_2^$ *are often undesirable and considered artifacts, $T_2^$* is also an important source of contrast to reflect differences in magnetic susceptibility:

1. Iron deposition: The presence of iron in a tissue will create shorter T_2^* due to the strong magnetic susceptibility of iron.
2. Calcifications: The presence of calcium in a tissue will create shorter T_2^* due to the strong magnetic susceptibility of calcium.
3. Oxygenated vs deoxygenated blood: Deoxygenated blood will have shorter T_2^* than oxygenated blood due to the magnetic susceptibility of deoxygenated hemoglobin. This can be used for selectively imaging veins, and is also the basis of blood oxygen level dependent (BOLD) functional MRI.

9.4 Fat/Water Imaging

The major sources of signal in MRI are hydrogen atoms in water molecules and hydrogen atoms in lipids. In lipids, the protons experience a different chemical shift, meaning they have a different resonance frequence from water protons. This can be exploited to create images that isolate fat and water signals.

While lipids have a relatively complex set of chemical shifts, in MRI a general "fat" signal is typically modeled as a single peak that has a -3.5 parts per million (ppm) shift from water. This leads to a resonance frequency shift Δf_{cs} of approximately 220 Hz at 1.5 T and 440 Hz at 3 T.

Separate fat and water images are created by so-called "Dixon" methods, which acquire images at multiple echo times and then use the phase differences that result from the different resonance frequencies to separate them. Ideally, the signal in these images can be represented as

$$s(TE) = m_{water}(\vec r) + m_{fat}(\vec r) \exp(-i 2 \pi \Delta f_{cs} TE)$$

Then fat and water images can be created from in-phase TE, $TE_{in-phase} = n / \Delta f_{cs}$ and out-of-phase TE, $TE_{out-of-phase} = (n+ 1/2) / \Delta f_{cs}$, images:

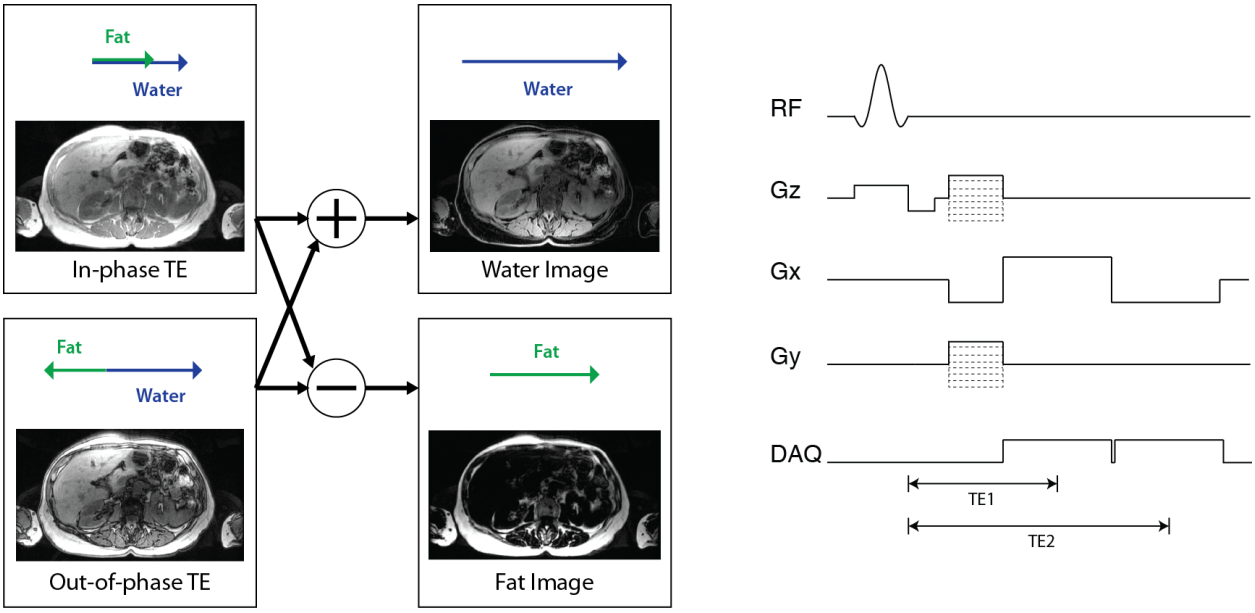
$$ s(TE_{in-phase}) = m_{water}(\vec r) + m_{fat}(\vec r)$$

$$ s(TE_{out-of-phase}) = m_{water}(\vec r) - m_{fat}(\vec r)$$

$$ \hat m_{water}(\vec r) = s(TE_{in-phase}) + s(TE_{out-of-phase})$$

$$ \hat m_{fat}(\vec r) = s(TE_{in-phase}) - s(TE_{out-of-phase})$$

In practice, additional corrections are required to account for off-resonance, $\Delta f_r(\vec r)$, and more advanced methods also use a multi-peak model accounting for the various chemical shifts in fatty tissue.



9.5 Contrast Agents

Contrast agents used in MRI, such as gadolinium chelates and superparamagnetic iron oxides, shorten the relaxation rates wherever they are present. This is quantified by their relaxivities, r_1, r_2, typically in units of [L / (mmol s)]:

$$\frac{1}{\hat{T}_1} = \frac{1}{T_1} + r_1 \cdot [CM]$$ $$\frac{1}{\hat{T}_2} = \frac{1}{T_2} + r_2 \cdot [CM]$$

Where $\hat{T}_1, \hat{T}_2$ are the relaxation time constants with the agent present, and $[CM]$ is the contrast agent concentration, typically in units of [mM = mmol/L].

CHAPTER

TEN

GRADIENT AND SPIN ECHO PULSE SEQUENCES

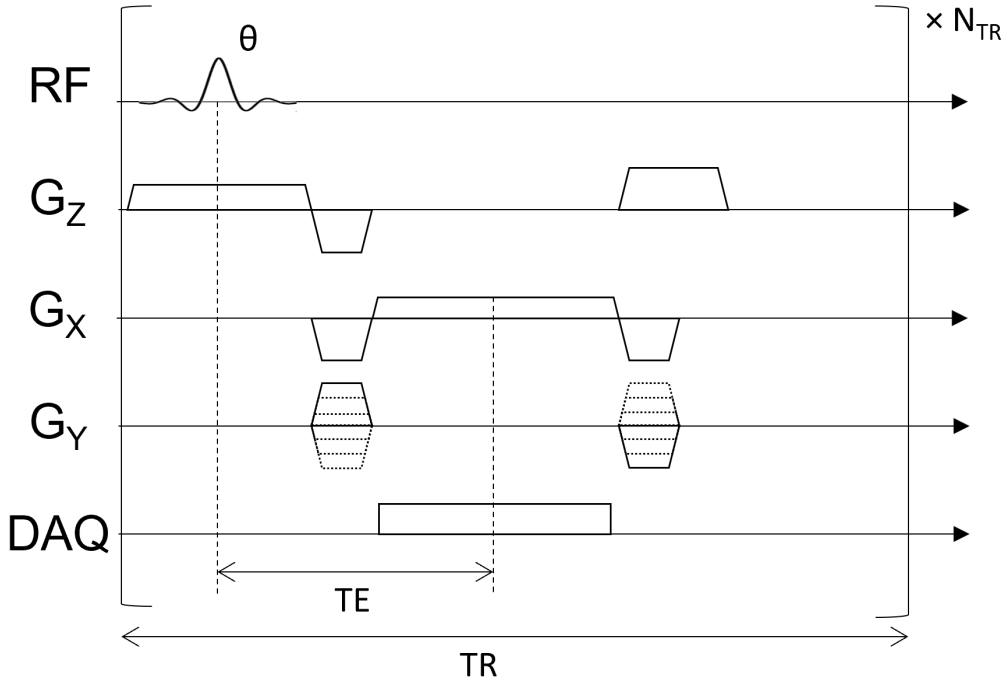
There are two broad classes of pulse sequences used: gradient-echo and spin-echo methods. Gradient-echo sequences use a simpler pulse and acquire acquisition, while spin-echo sequences use an additional RF refocusing pulse that eliminates the effects of off-resonance that are often undesirable.

10.1 Learning Goals

1. Describe how various types of MRI contrast are created
    - Describe the difference between gradient and spin echo contrast
2. Understand the most popular pulse sequences and their acronyms
    - Identify gradient echo and spin echo pulse sequences
3. Identify artifacts and how to mitigate them
    - Understand the effects of off-resonance and T_2^* on contrast

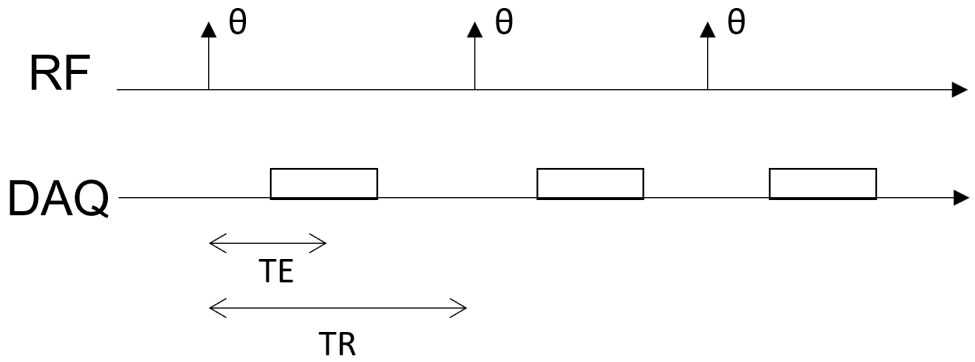
10.2 Gradient Echo Pulse Sequence (GE or GRE)

The building block of a gradient-echo (GE) pulse sequence includes an RF excitation pulse followed by imaging gradients. A complete 2D imaging sequence is



It is called a “gradient-echo (GE)” since the frequency encoding imaging gradients, shown on G_X, are refocused or canceled out at the echo time (more in Spatial Encoding section). Another name for these sequences is a “gradient-recalled echo (GRE)”.

And using a simplified diagram for the gradient echo sequence:

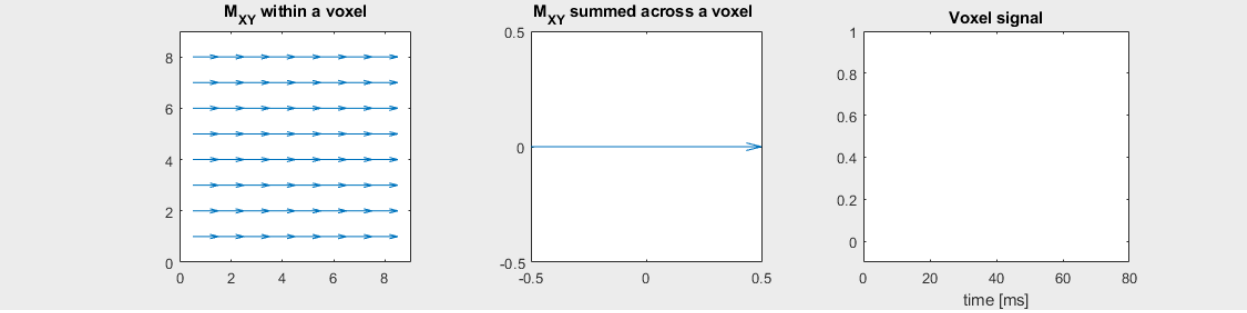


The gradient-echo pulse sequence exhibits T_2^* contrast.

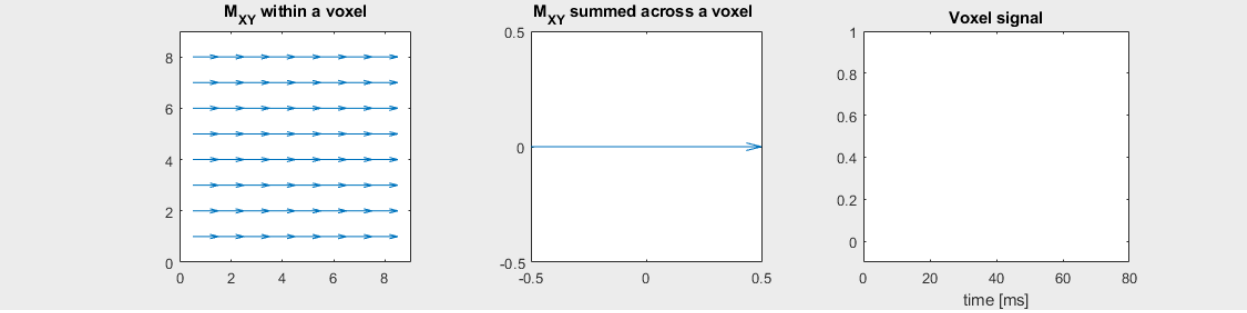
$$S_{GE} \propto M_0 \exp(-TE/T_2^*)$$

10.3 Refocusing off-resonance with a spin-echo

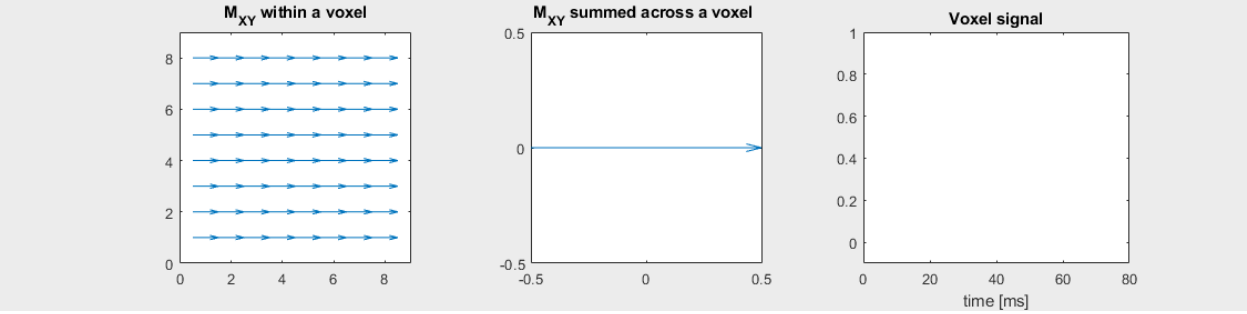
Gradient-echo sequences are sensitive to off-resonance effects and exhibit T_2^* contrast. However, the effects of off-resonance can be reversed by applying a 180-degree flip angle "refocusing" RF pulse some time after the initial excitation RF pulse. This has the effect of inverting the phase accumulation of off-resonance net magnetizations, after which the off-resonance phase begins to cancel out. This is most easily illustrated in the voxel decay illustrations below:



Spin-echo with No off-resonance, $T_2 = 80$ ms



Spin-echo with Mild off-resonance, $T_2 = 80$ ms



Spin-echo with Severe off-resonance $T_2 = 80$ ms

The spin-echo does not make any difference when there is no off-resonance.
Can you identify when the 180-degree refocusing pulse was applied? This is when the M_{XY} reverts its phase.

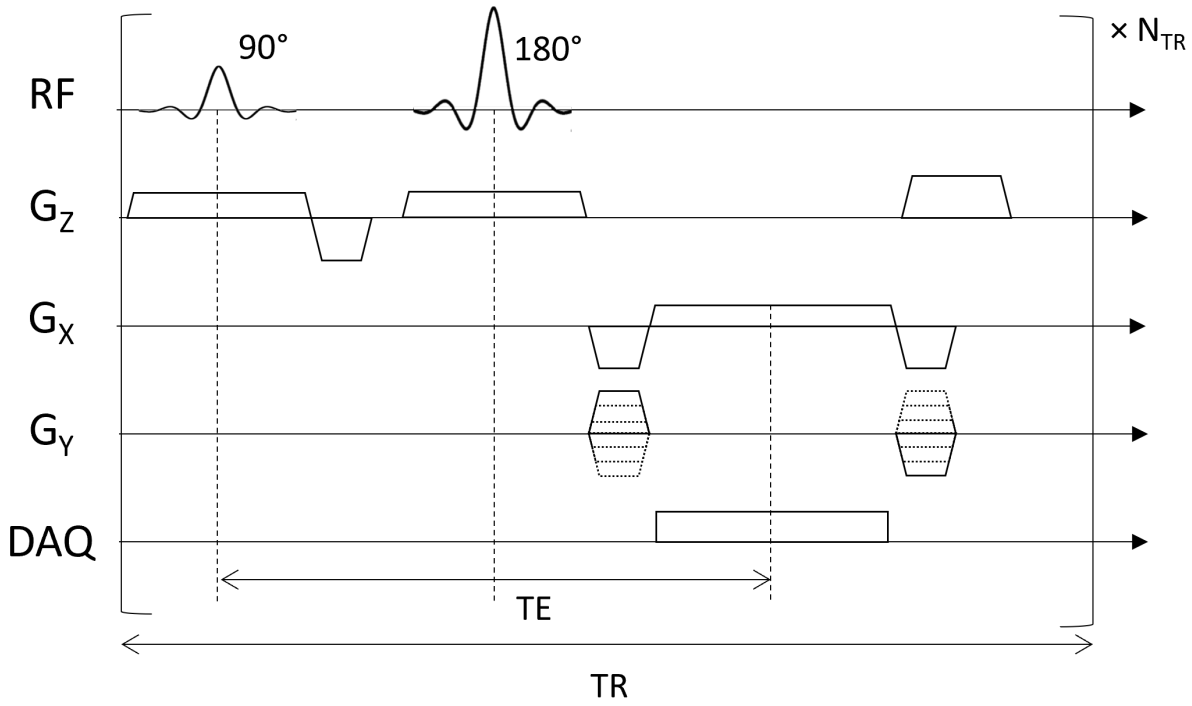
10.3.1 Simulation for Visualization of refocusing off-resonance

To simulate and visualize off-resonance, including various refocusing pulse flip angles, try

- Visit http://drcmr.dk/BlochSimulator/
- Scene: Inhomogeneity
- Apply 90x hard, then 180y hard - what happens to the dephasing spins?
- Apply 90x hard, then 180x hard - this applies the 180-degree pulse at a different angle. Is there still a spin-echo?
- Apply 90x hard, then repeated 180s - can multiple spin-echoes be created?

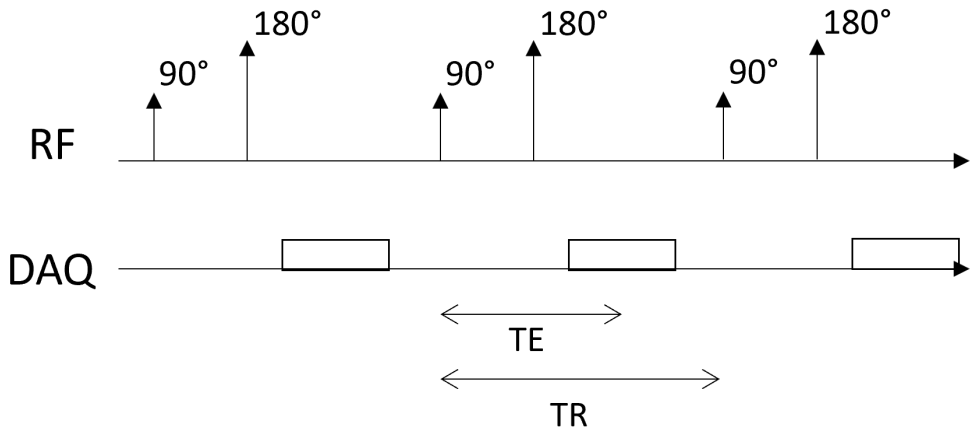
10.4 Spin Echo Pulse Sequence (SE)

The building block of a spin-echo (SE) pulse sequence has an additional 180-degree refocusing pulse between the excitation and data acquisition in order to refocus the effects of off-resonance and create pure T_2-weighting:



It is called a “gradient-echo (GE)” since the spins are all refocused at the echo time, TE.

This is a simplified diagram for the spin echo sequence:



Because of the refocusing, a spin-echo sequence gives pure T_2 contrast:

$$S_{SE} \propto M_0 \exp(-TE/T_2)$$

CHAPTER

ELEVEN

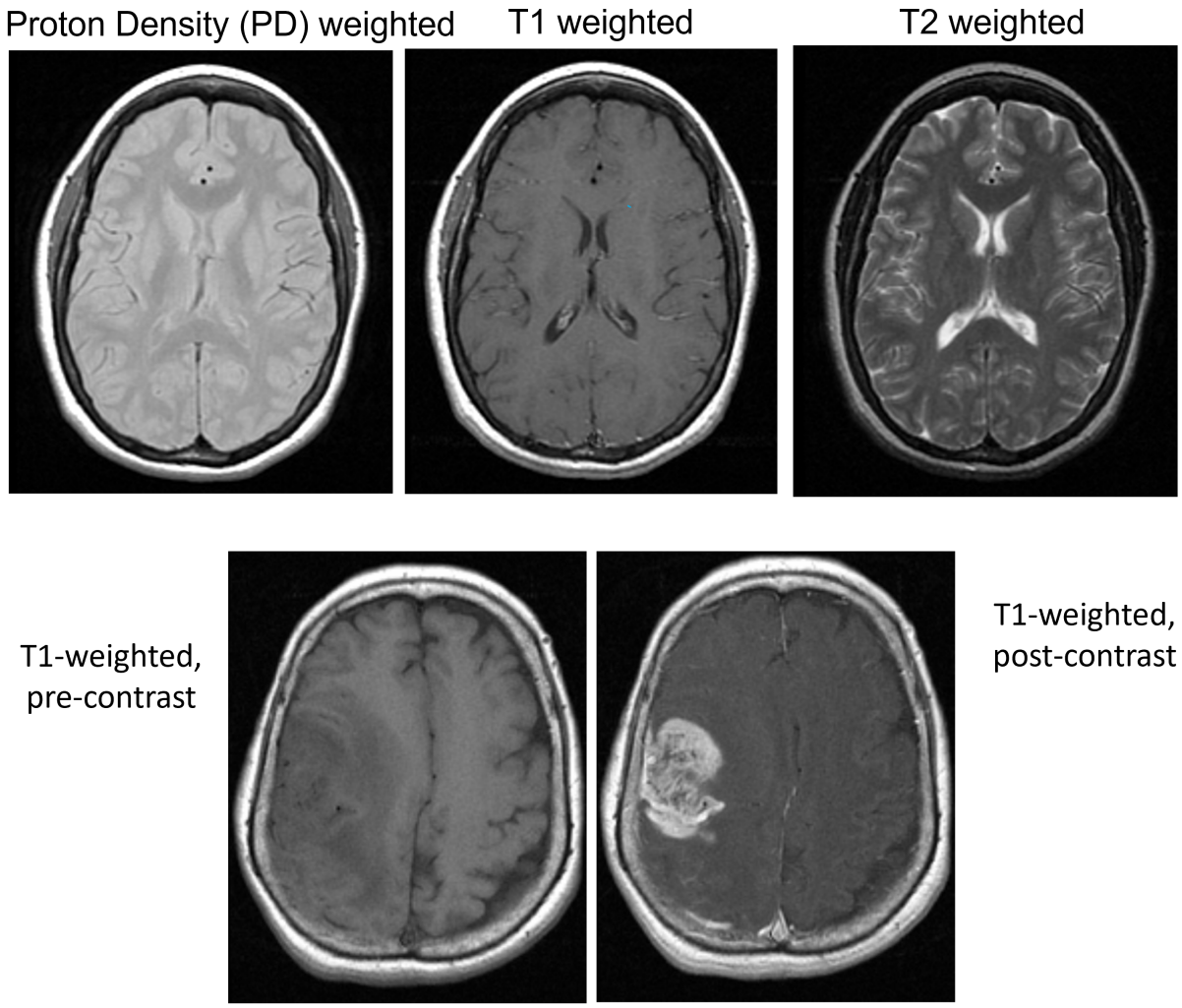
IMAGE CONTRAST EXAMPLES

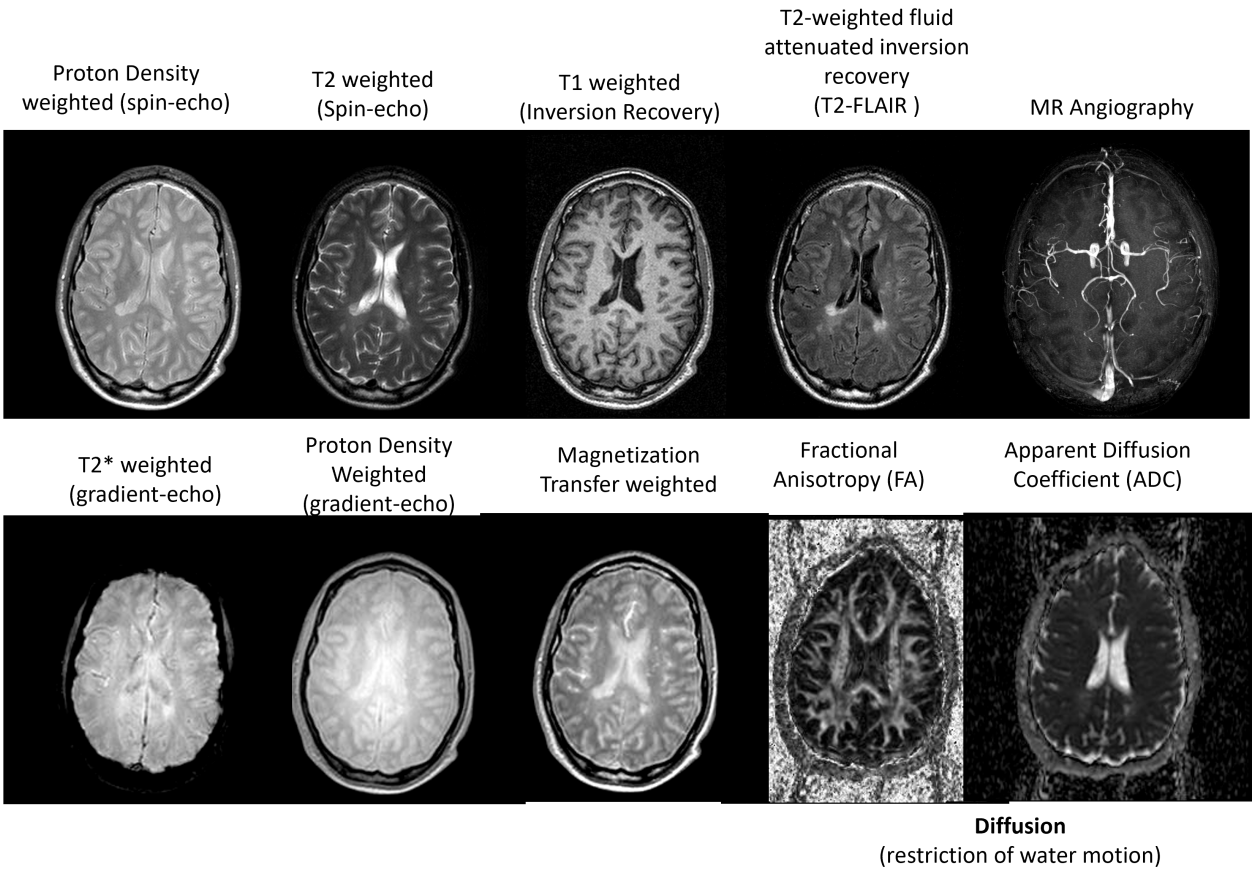
This page shows examples of different MRI contrasts, as well as examples of how to simulate MRI contrast.

11.1 Learning Goals

1. Describe how various types of MRI contrast are created
    - Identify various image contrasts

11.2 Image Contrast Examples




Proton Density weighted (spin-echo)
T2 weighted (Spin-echo)
T1 weighted (Inversion Recovery)
T2-weighted fluid attenuated inversion recovery (T2-FLAIR)
MR Angiography
T2* weighted (gradient-echo)
Proton Density Weighted (gradient-echo)
Magnetization Transfer weighted
Fractional Anisotropy (FA)
Apparent Diffusion Coefficient (ADC)
Diffusion
(restriction of water motion)

11.3 Simple Contrast Phantom

Below is a simple contrast phantom consisting of three circular objects, each with a different T_1 and T_2. The k-space data for these phantoms is created using a "jinc" function, which is the Fourier Transform of a circle:

$$\mathcal{F}{ \mathrm{circ}(r) } = \mathrm{jinc}(k_r)$$

The signal is simulated for varying sequence parameters (TE, TR) with a 90-degree flip angle in a gradient-echo scheme by using the helper function:

```
signal_gre = MRsignal_spoiled_gradient_echo(flip, TE, TR, M0, T1, T2)
```

11.3.1 Challenge Yourself!

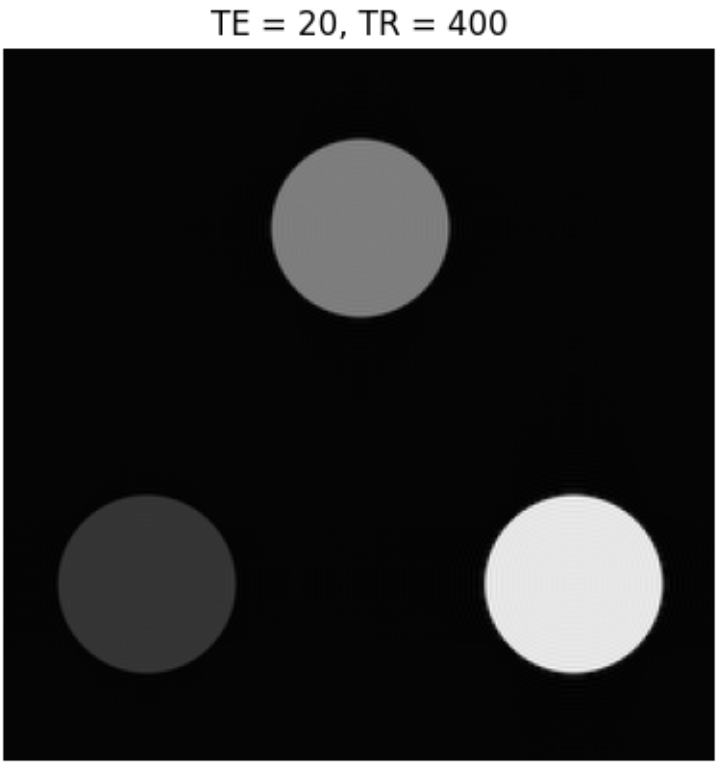
Based on the images and TE/TR parameters below, can you estimate T1 and T2 relaxation times of the 3 objects?

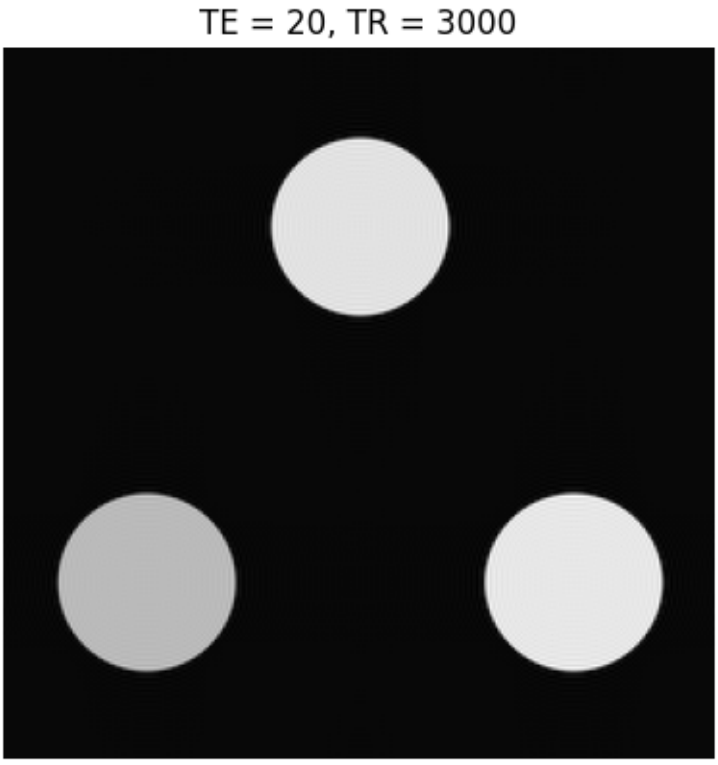
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.special import j1

# Helper functions (you must define these elsewhere in your notebook or import them)

def jinc(x):
```

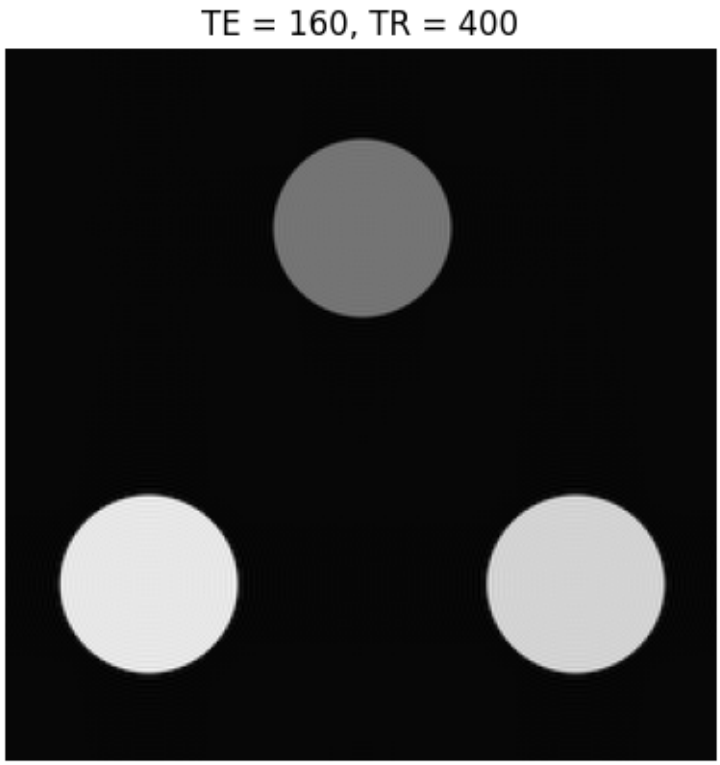
(continues on next page)

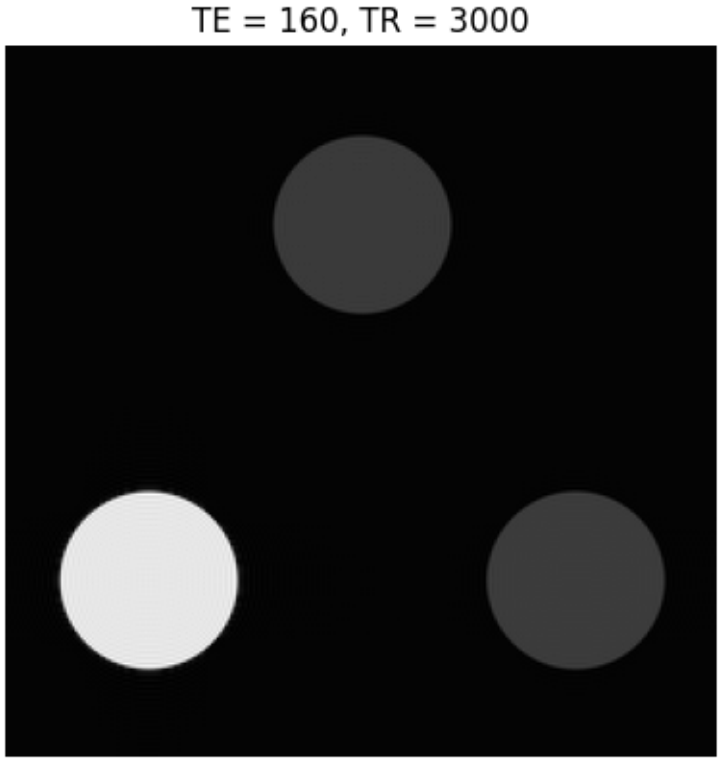
(continued from previous page)

```
    # Avoid division by zero
    result = np.ones_like(x)
    mask = x != 0
    result[mask] = j1(np.pi * x[mask]) / (2 * x[mask])
    return result

def MRsignal_spoiled_gradient_echo(flip, TE, TR, M0, T1, T2):
    # Calculate the MR signal for a spoiled gradient echo sequence
    E1 = np.exp(-TR / T1)
    E2 = np.exp(-TE / T2)
    return M0 * (1 - E1) * np.sin(flip) * E2 / (1 - E1 * np.cos(flip))

Nphan = 3
# phantom parameters
xc = np.array([-.3, 0, .3]) * 256  # x centers
yc = np.array([-.25, .25, -.25]) * 256  # y centers
M0 = np.array([1, 1, 1])  # relative proton densities
T1 = np.array([.3, .8, 3]) * 1e3  # T1 relaxation times (ms)
T2 = np.array([.06, .06, .2]) * 1e3  # T2 relaxation times (ms)
r = 1/64  # ball radius, where FOV = 1

flip = 90 * np.pi / 180
TEs = [20, 160]
TRs = [400, 3000]

N = 256

# matrices with kx, ky coordinates
kx, ky = np.meshgrid(np.arange(-N/2, N/2) / N, np.arange(-N/2, N/2) / N)

for TE in TEs:
    for TR in TRs:
        # initialize k-space data matrix
        M = np.zeros((N, N), dtype=complex)

        # Generate k-space data by adding together k-space data for each individual␣
↪phantom
        for n in range(Nphan):
            # Generates data using Fourier Transform of a circle (jinc) multiplied by␣
↪complex exponential to shift center location
            M = M + jinc(np.sqrt(kx**2 + ky**2) / r) * \
                np.exp(1j * 2 * np.pi * (kx * xc[n] + ky * yc[n])) * \
                MRsignal_spoiled_gradient_echo(flip, TE, TR, M0[n], T1[n], T2[n])

        # reconstruct and display ideal image
        m = np.fft.fftshift(np.fft.ifft2(np.fft.ifftshift(M)))
        plt.figure()
        plt.imshow(np.abs(m), cmap='gray', aspect='equal')
        plt.axis('off')
        plt.title(f'TE = {TE}, TR = {TR}')
        plt.show()
```


TE = 20, TR = 400


TE = 20, TR = 3000


TE = 160, TR = 400


TE = 160, TR = 3000

Part V

Radiofrequency Pulses

CHAPTER

TWELVE

RF PULSES

The purpose of appling RF magnetic fields is to excite the spins. Equivalently, this RF energy aims to flip the net magnetization. This RF energy is referred to as an RF pulse, since the RF energy is applied for a short period of time and then switched off.

12.1 Learning Goals

1. Describe how various types of MRI contrast are created
    - Understand how changes in flip angle are implemented
2. Manipulate MRI sequence parameters to improve performance
    - Understand how changes in flip angle are implemented
3. Describe how images are formed
    - Describe how RF pulses are frequency-selective

12.2 How RF Pulses are Used in MRI

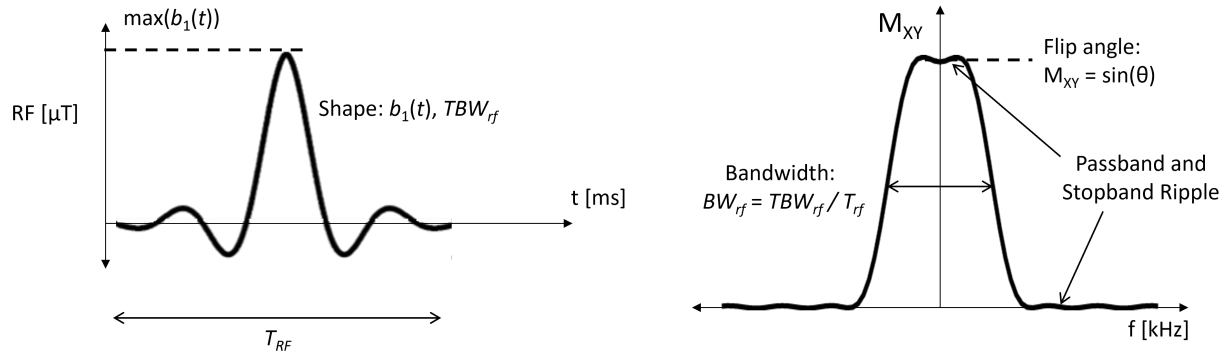
Fundamentally, RF pulses are used to flip the net magnetization. This creates signal and also manipultes contrast. Also, applying magnetic field gradients during an RF pulse will create slice selection.

12.3 Types of RF Pulses

Pulse Type	Purpose	Magnetization Effect	Flip Angle
Excitation	Create signal	M_Z to M_{XY}	up to 90-degrees
Inversion	Enhance contrast	M_Z to -M_Z	180-degrees
Refocusing	Create spin-echoes	M_{XY} to M_{XY}	180-degrees
Saturation	Suppress magnetization	M_Z to 0	90-degrees

12.4 RF Pulse Properties

Here is a summary of RF pulse properties and parameters that are important for MRI. These will be explained in detail later.



12.4.1 Flip Angle

The flip angle (on-resonance) is equal to the integral of the RF pulse shape, $b_1(t)$ (in units of magnetic field)):

$$\theta = \gamma \int_0^{T_{rf}} b_1(\tau) d\tau $$

12.4.2 Time-bandwidth Product

The time-bandwidth product (TBW) characterizes the relationship between pulse duration, T_{rf}, and bandwidth, BW_{rf} and is a fixed value for a given pulse shape. It is defined as

$$ TBW_{rf} = T_{rf} \cdot BW_{rf} $$

For a sinc or sinc-like pulse, the TBW product is equal to the number of zero-crossings in the pulse shape.

12.4.3 Specific Absorption Rate

The Specific Absorption Rate (SAR) measures the amount of energy absorbed by the body from a RF pulse. Since this absorption can cause tissue heating there are safety limits on SAR. It is proportional to the integrated total RF pulse power:

$$ SAR \propto \int_0^{T_{rf}} |b_1(\tau)|^2 d\tau $$

T_{rf} is the duration of the RF pulse. For for MRI, SAR can cause tissue heating and thus there are SAR safety limits to minimize this heating.

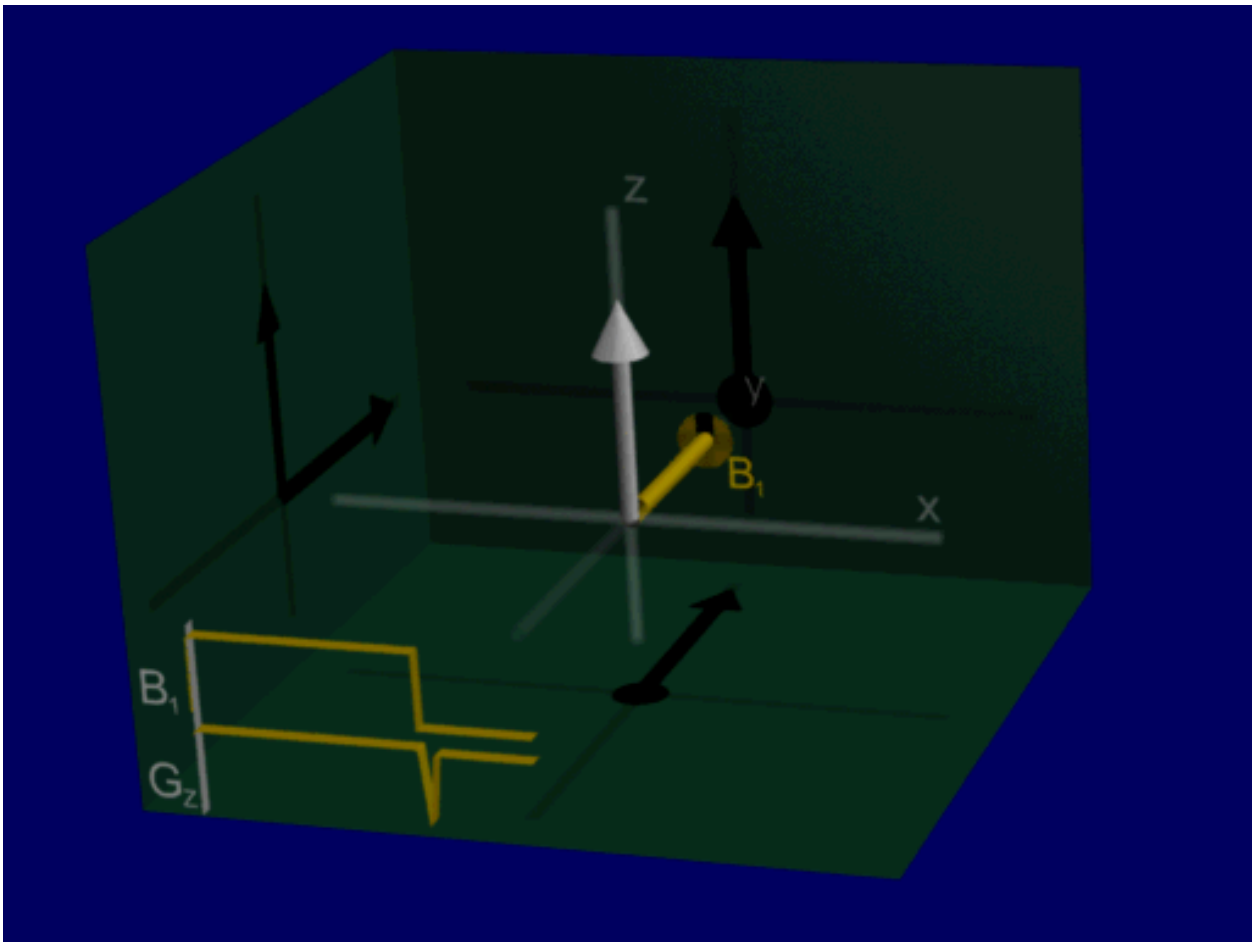
12.5 Hard pulse versus Shaped Pulse

The first RF pulse typically introduced in MRI education is the so-called “hard” pulse - a constant amplitude pulse applied for some period of time. As will be shown below, this pulse performs poorly for slice-selection, so shaped sinc-like pulses are used. Shaped pulses create frequency profiles that excite a range of frequencies approximately uniformly (with the same flip angle), while not exciting spins outside this frequency range.

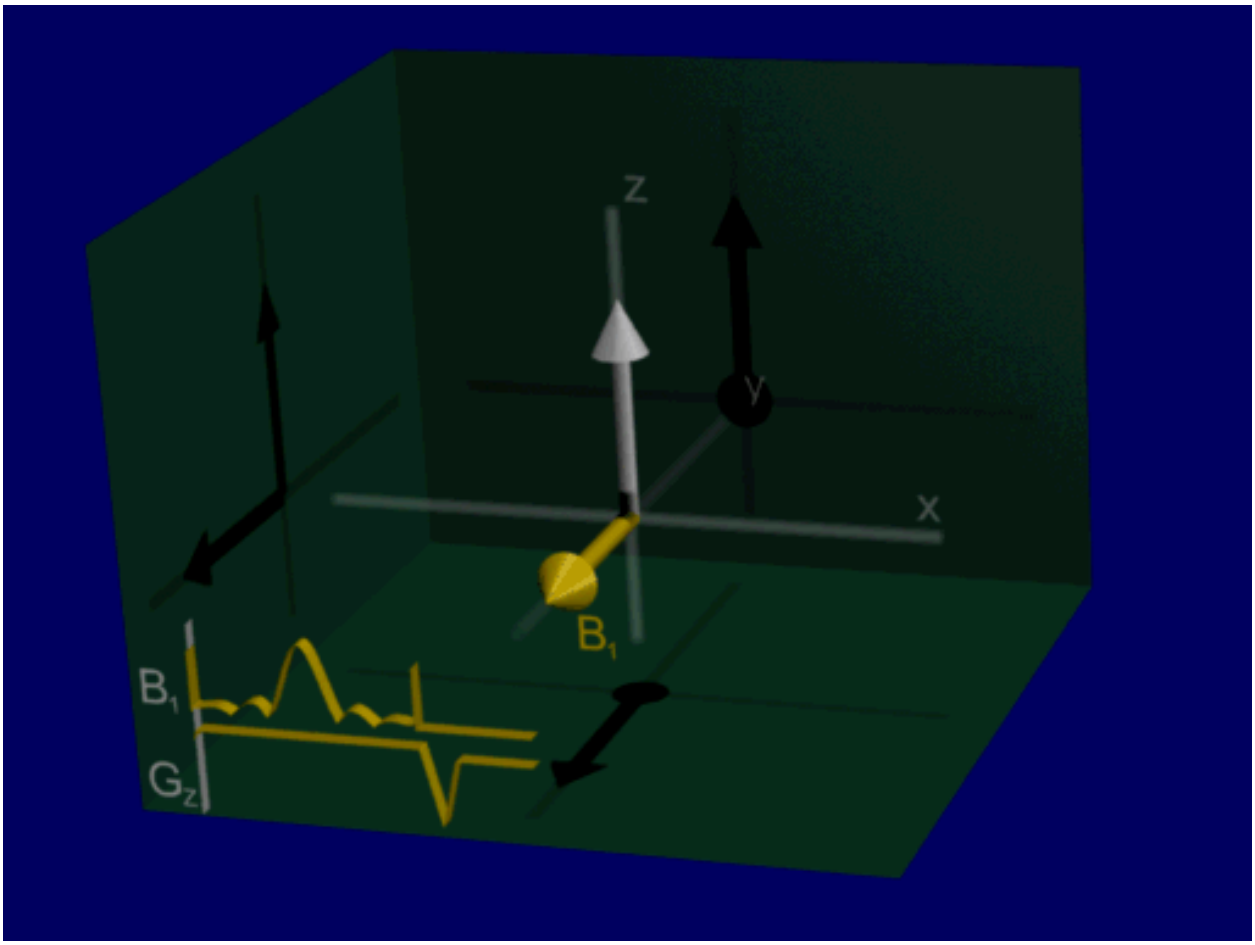
12.5.1 On-resonance Excitation

We can visualize the rotation of the net magnetization as before as a rotation about the RF pulse vector, for Hard and Shaped pulses:

hard pulse:



shaped pulse:



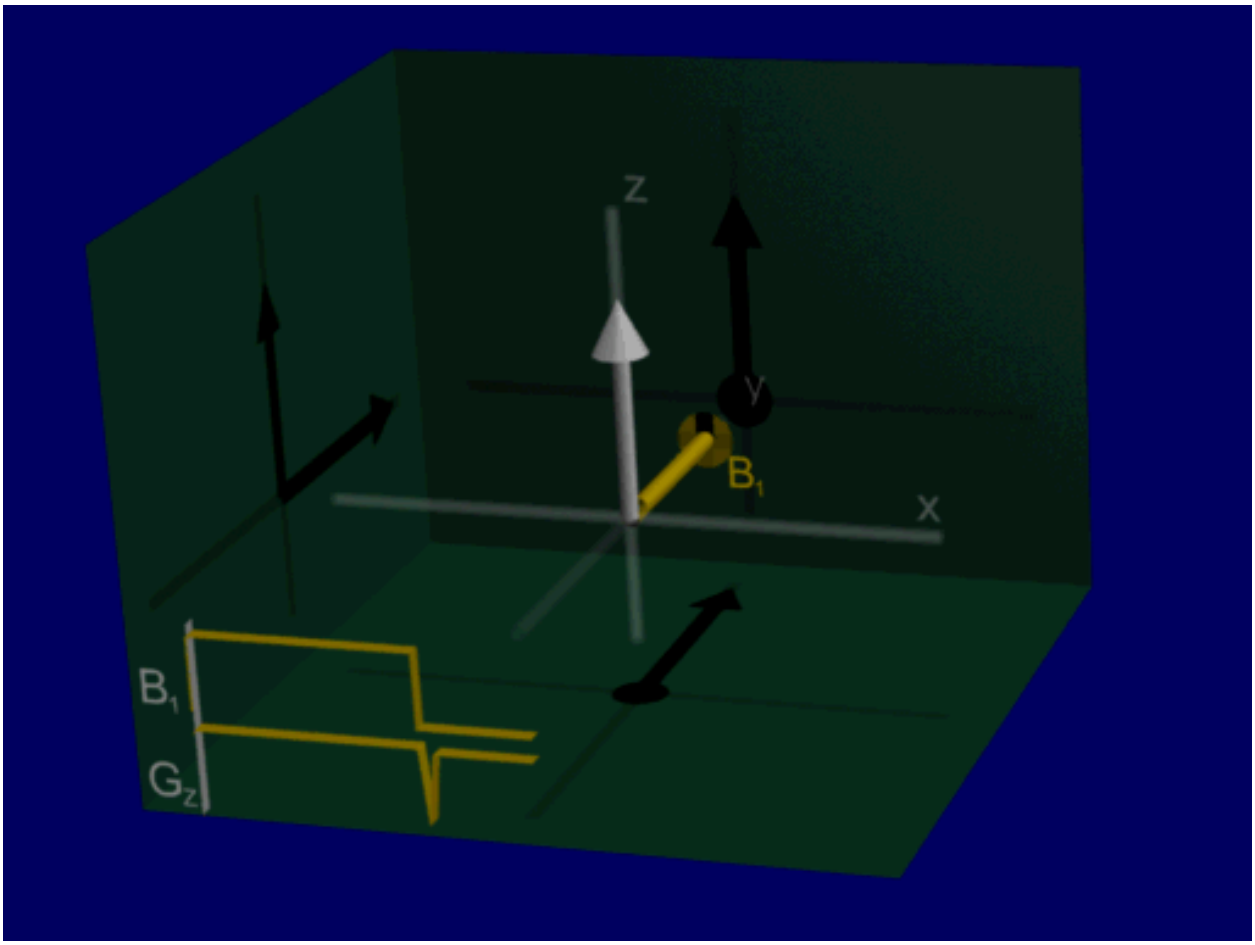
Looking at the on-resonance or center frequency response of a sinc-like shaped pulse, the flip angle is achieved slightly differently by tipping the net magnetization back and forth.

12.5.2 Off-resonance Excitation

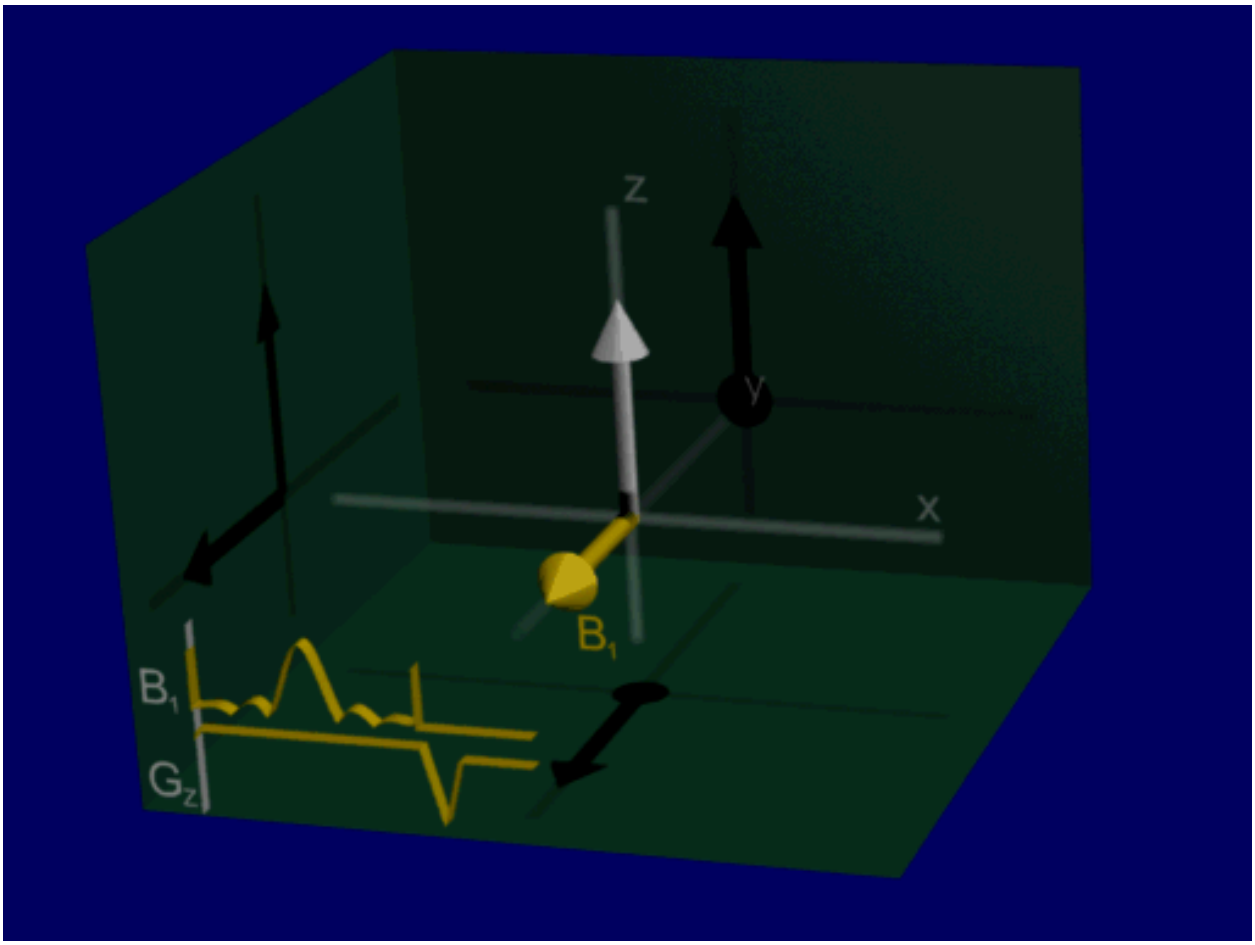
The flip angle provided by a hard pulse varies substantially as a function of off-resonance frequency (e.g. in the presence of a magnetic field gradient) so is not a good pulse for slice-selection. Meanwhile, the sinc-like pulse becomes advantageous when considering a range of resonance frequencies, where this pattern of back and forth tipping leads to net magnetizations either being flipped or not:

This is illustrated by the following animation, showing net magnetizations across a range of positions when the pulse is applied with a gradient:

hard pulse:



shaped pulse:



These simulations are for an identical range of positions, allowing for visualization the majority of net magnetizations in this case are either at an approximate 90-degree or 0-degree excitation, with very few intermediate flip angles. that the range of resulting magnetizations from the *hard pulse* is wide, representing a spread of different flip angles at different positions, whereas with the sinc-like pulse the majority of net magnetizations in this case are either at an approximate 90-degree or 0-degree excitation, with very few intermediate flip angles. (Also note the refocusing gradient at the end, which is necessary for corrected for additional phase during the gradient & RF.)

This behavior can be analyzed by simulation of the RF pulse profile as well, now using the Bloch equation for a more precise simulation as compared to using Fourier Transform approximation:

12.6 RF Pulse Profile

To precisely characterize the behavior of an RF pulse, a pulse profile can be computed.

In this section, we will characterize the behavior of RF pulses as a function of off-resonance, or frequency. In the next section, gradients will be added to perform slice selection and the profile will be computed as a function of space.

The RF pulse profile can plot several parameters, including the flip angle, the resulting transverse magnetization M_{XY}, and the resulting longitudinal magnetization M_Z.

The following simulations compute the magnetization profiles after hard and shaped RF pulses.

```
import numpy as np
import matplotlib.pyplot as plt
```

(continues on next page)

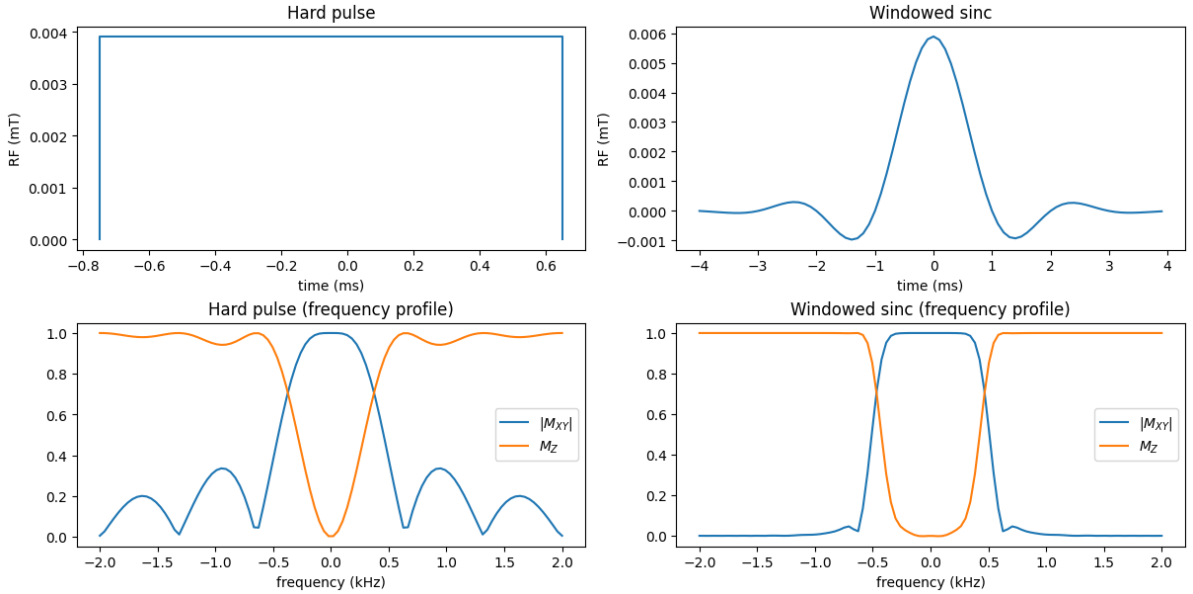
(continued from previous page)

```
import sys

sys.path.append('../Physics/')
from bloch_rotate import bloch_rotate

gammabar = 42.58  # kHz/mT

M0 = 1.0
M_equilibrium = np.array([0.0, 0.0, M0])
dt = 0.1  # ms

flip = 90.0
eps = 1e-9

# ---------- Hard Pulse ----------
tmax_hard = 1.5
N_hard = int(round(tmax_hard / dt))
t_hard = (np.arange(-N_hard/2, N_hard/2) * dt).astype(float)  # ms
RF_hard = np.ones(N_hard, dtype=float)
RF_hard = (flip * np.pi / 180.0) * RF_hard / RF_hard.sum() / (2 * np.pi * gammabar *␣
↪dt)

BW = 2.0  # kHz
df = np.linspace(-BW, BW, 100)

M_hard = np.tile(M_equilibrium.reshape(3, 1), (1, len(df)))
for n in range(len(t_hard)):
    for f_idx in range(len(df)):
        M_hard[:, f_idx] = bloch_rotate(M_hard[:, f_idx], dt,
                                        [float(np.real(RF_hard[n])), float(np.imag(RF_
↪hard[n])), float(df[f_idx] / gammabar)])

# ---------- Windowed Sinc Pulse ----------
tmax_sinc = 8.0
N_sinc = int(round(tmax_sinc / dt))
t_sinc = (np.arange(-N_sinc//2, N_sinc//2) * dt).astype(float)  # ms

RF_sinc = np.hamming(N_sinc) * np.sinc(t_sinc)
RF_sinc = (flip * np.pi / 180.0) * RF_sinc / RF_sinc.sum() / (2 * np.pi * gammabar *␣
↪dt)

M_sinc = np.tile(M_equilibrium.reshape(3, 1), (1, len(df)))
for n in range(len(t_sinc)):
    for f_idx in range(len(df)):
        M_sinc[:, f_idx] = bloch_rotate(M_sinc[:, f_idx], dt,
                                        [float(np.real(RF_sinc[n])), float(np.imag(RF_
↪sinc[n])), float(df[f_idx] / gammabar)])

# ---------- Plotting ----------
fig, axs = plt.subplots(2, 2, figsize=(12, 6), constrained_layout=True)

# Left column: Hard pulse
axs[0, 0].plot(np.hstack([t_hard[0] - eps, t_hard, t_hard[-1] + eps]), np.hstack([0.0,
↪ RF_hard, 0.0]))
axs[0, 0].set_xlabel('time (ms)')
axs[0, 0].set_ylabel('RF (mT)')
axs[0, 0].set_title('Hard pulse')
```

(continues on next page)

(continued from previous page)

```
axs[1, 0].plot(df, np.sqrt(M_hard[0, :]**2 + M_hard[1, :]**2), label=r'$|M_{XY}|$')
axs[1, 0].plot(df, M_hard[2, :], label=r'$M_{Z}$')
axs[1, 0].set_title('Hard pulse (frequency profile)')
axs[1, 0].set_xlabel('frequency (kHz)')
axs[1, 0].legend()

# Right column: Windowed sinc pulse
axs[0, 1].plot(t_sinc, RF_sinc)
axs[0, 1].set_xlabel('time (ms)')
axs[0, 1].set_ylabel('RF (mT)')
axs[0, 1].set_title('Windowed sinc')

axs[1, 1].plot(df, np.sqrt(M_sinc[0, :]**2 + M_sinc[1, :]**2), label=r'$|M_{XY}|$')
axs[1, 1].plot(df, M_sinc[2, :], label=r'$M_{Z}$')
axs[1, 1].set_title('Windowed sinc (frequency profile)')
axs[1, 1].set_xlabel('frequency (kHz)')
axs[1, 1].legend()

plt.show()
```



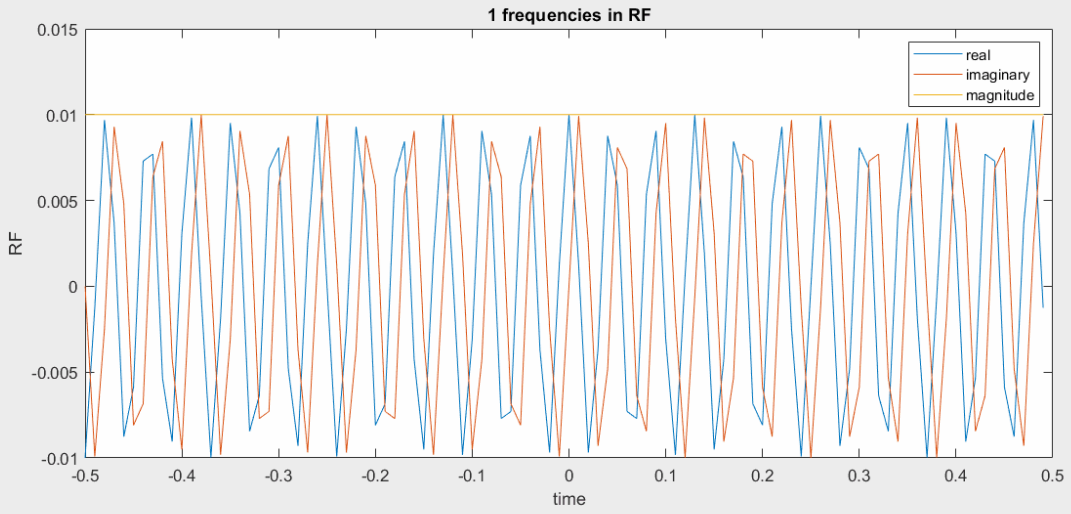
12.7 Frequency Content in RF Pulses

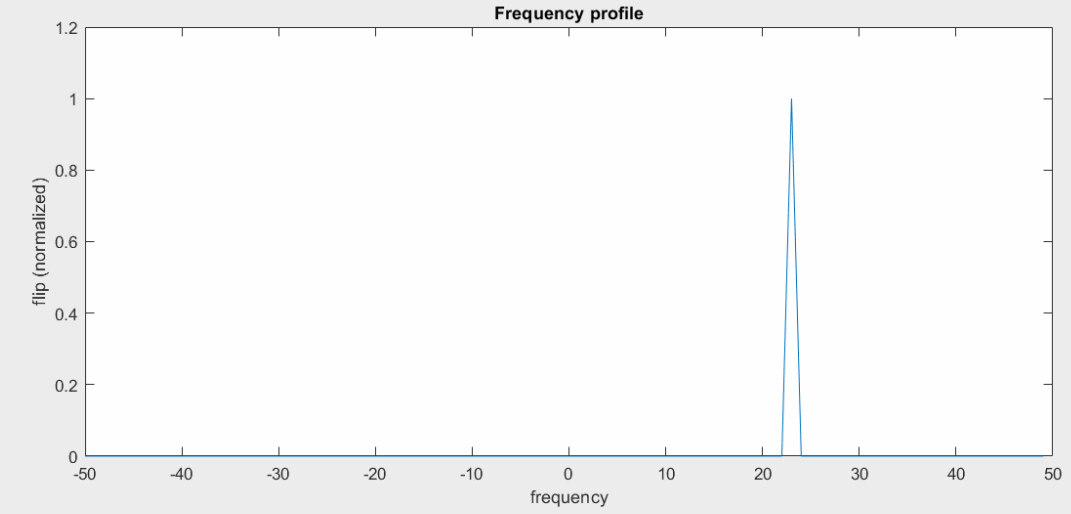
For RF pulses, we generally want to control the frequencies that are excited. Most commonly, we want to excite a range of frequencies that then will correspond to a range of positions to excite a slice.

12.7.1 Sum of Frequencies

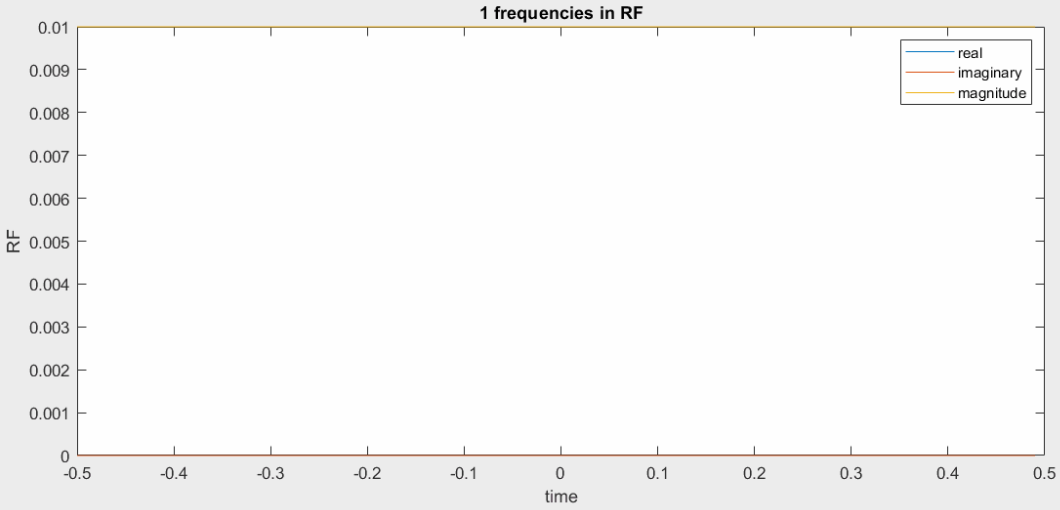
One way to think of the RF pulse design then, is that we can add a range of frequencies together in the RF pulse, which will then excite a range of off-resonance frequencies. This "sum of frequencies" illustration of RF pulses is shown in the following animations.

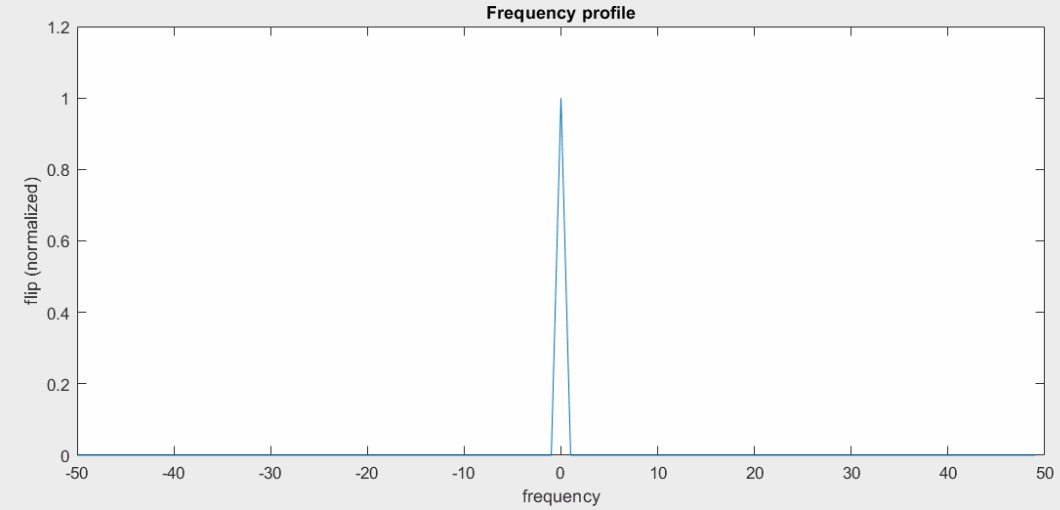
In the lab frame, this happens around the Larmor frequency:





In the rotating frame, the frequencies build up around frequency = 0





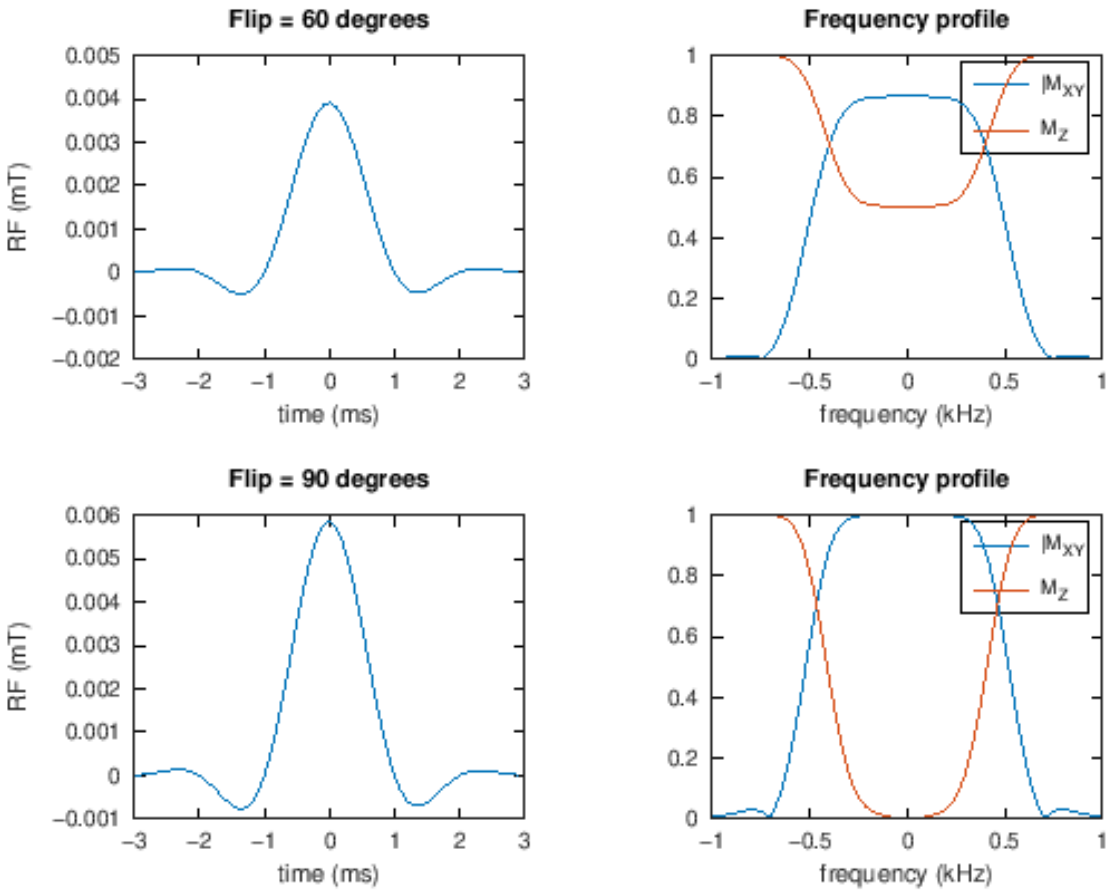
12.7.2 Fourier Transform Approximation

Based on the above illustration, we can see that the profile of an RF pulse is similar to a Fourier Transform - an operation that determines the frequency content of a signal.
In practice, the RF pulse profile is only approximately proportional to the Fourier Transform of the RF pulse shape, but this approximation is useful for understanding the frequency content of RF pulses.

$$\theta(f) \approx \gamma \mathcal{F}{b_1(t)}$$

where $b_1(t)$ is the RF pulse shape, and $\theta(f)$ is the flip angle at frequency f.

For this reason, our typical RF pulse shapes are sinc-like, which have a Fourier Transform that are rect-like. In practice, this approximation is only valid for small flip angles, typically up to 60 degrees. There are tools such as the SLR transform that can be used to design RF pulses with larger flip angles, which are described later.



12.8 Frequency-Selective Pulses for Fat Suppression

Fat is often a nuisance signal in MRI. We can use frequency-selective RF pulses for fat suppression by applying a saturation pulse that tips fat 90-degrees but tips water 0-degrees. This can be done based on the chemical shift between fat and water, and by using a frequency-selective RF pulse as described above. At this point, $M_{Z,fat} = 0$ but $M_{Z,water}$ is unaffected. This pulse is followed by a spoiler gradient which will dephase the transverse magnetization of fat, resulting in no signal from fat, $m_{fat} = 0$.

Frequency-selective inversion pulses can also be used for fat suppression along with inversion recovery. The inversion pulse tips fat 190-degrees but tips water 0-degrees, and is followed by an inversion time to null fat based on its T_1.

CHAPTER

THIRTEEN

SLICE SELECTION

An important tool in MRI for localization is to use slice selection. This is done by applying a magnetic field gradient during an RF pulse, which can be designed in order to only excite a limited amount of space.

13.1 Learning Goals

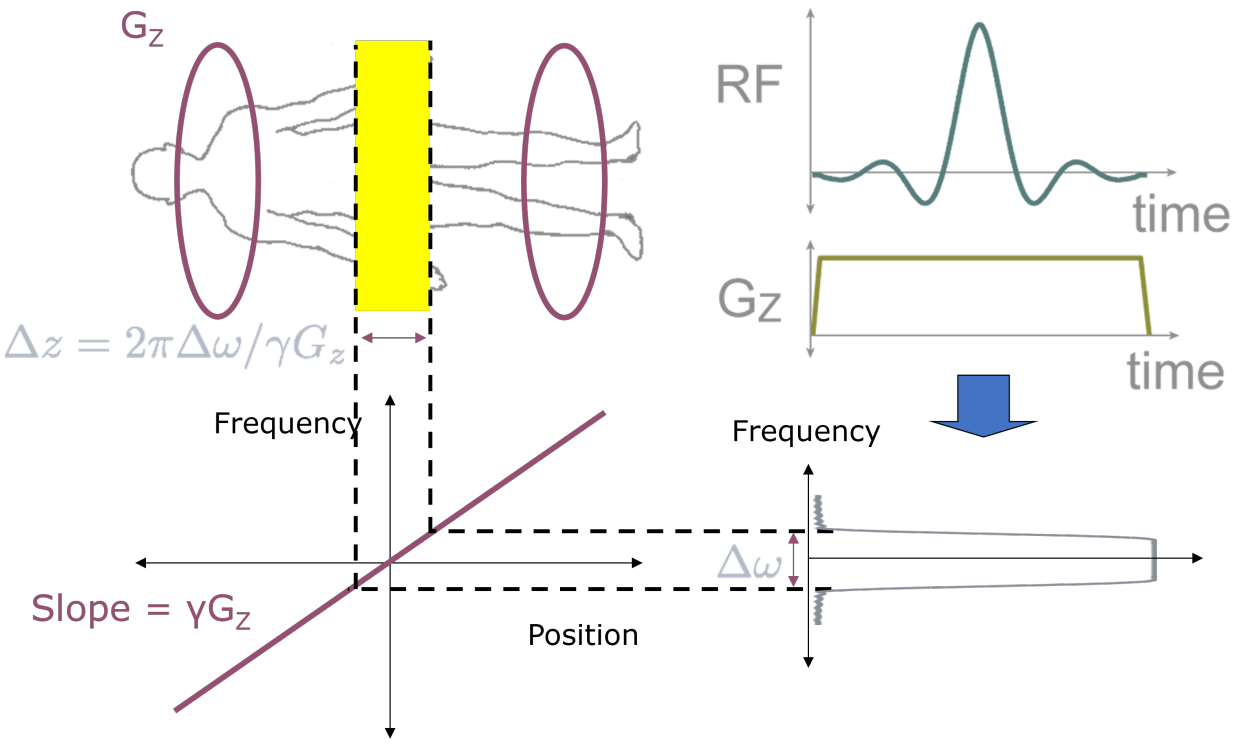
1. Describe how images are formed
    - Describe slice selection using RF Pulses

13.2 Spatial Selectivity

As described in the previous section, RF pulses can be designed to excite a limited range of frequencies, characterized by its frequency response profile. Magnetic field gradients create variations in the magnetic field across space, creating a mapping between frequency and space:

$$f(\vec r,t) = \bar \gamma (\vec G(t) \cdot \vec r)$$

So when a magnetic field gradient is turned on during a RF pulse, the frequency profile of the pulse is converted into a spatial profile. This is illustrated in the following diagram:



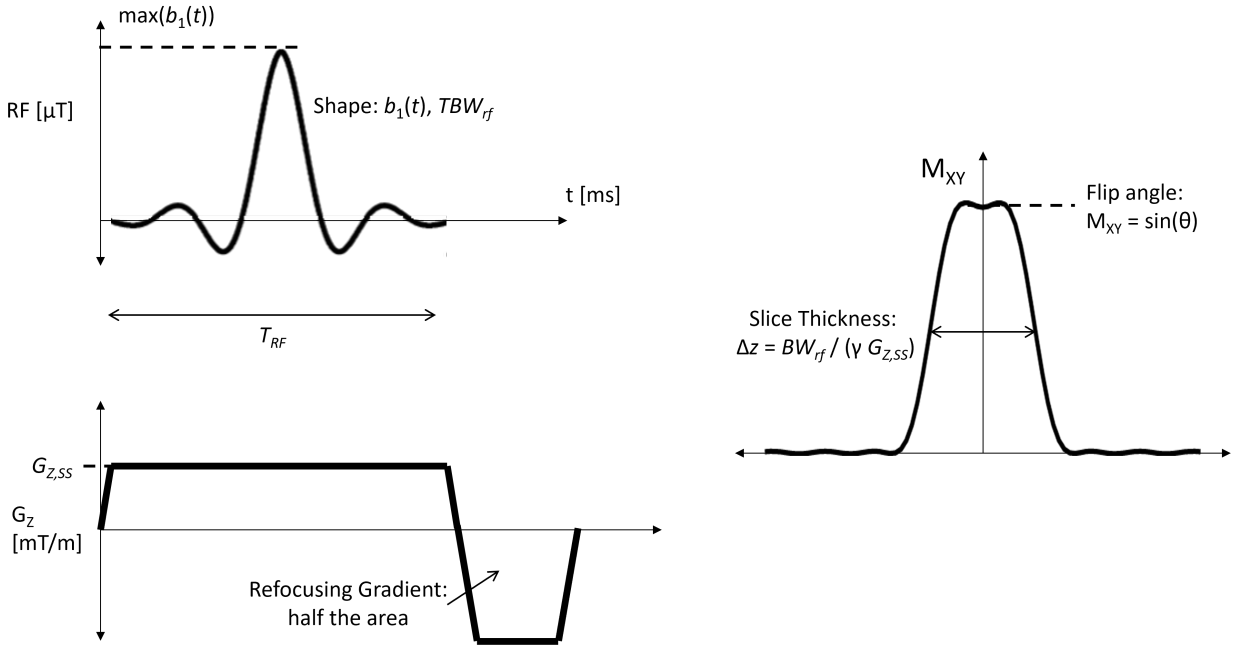
13.3 Slice Selective RF Pulses

The goal in slice selection is to excite a limited region in space, commonly refered to as a slice or slab. (The term "Slab selection" typically refers to exciting a larger slice in 3D imaging.)

13.3.1 Pulse Design

RF Pulse shape - Since the goal is to excite only a limited range of frequencies, a sinc-like pulse shape is used to create a rect-like frequency profile. See the previous section for details on the design.

Magnetic field gradient - A constant gradient is applied during the pulse. The gradient direction chosen corresponds to the direction of the slice selection, and the amplitude of the gradient determines the thickness of the slice. This is known as the slice-select gradient.



Conventionally, pulse sequence diagrams typically put the slice selection along the Z gradient direction, and the imaging gradients are on X and Y. In practice, the gradient directions used can be any combination of X, Y, and Z to create slice selection in any arbitrary direction.

13.3.2 Slice Thickness

The gradient amplitude, $G_{Z,SS}$, and RF pulse bandwidth determine the slice thickness:

$$ \Delta z = \frac{BW_{rf}}{\bar{\gamma} G_{Z,SS}} $$

13.3.3 Slice Shifting

The slice location can be shifted by a distance z_{off} through shifting the frequency of the RF pulse by f_{off}:

$$ f_{off} = \bar{\gamma} G_{Z,SS} \ z_{off} $$

13.3.4 Refocusing Gradient

At the end of the RF pulse in a slice-selective RF pulse there is unrefocused phase across the slice. This means that the transverse magnetization is pointing in various directions across the slice. If imaging occurs at this point, the transverse magnetization pointing in different directions can cancel out, leading to signal loss. Therefore, a refocusing gradient must be applied prior to imaging. This gradient has an area of half of the slice select gradient on during the RF pulse.

CHAPTER

FOURTEEN

RF PULSE DESIGN

RF pulse design in the small-tip regime (< 60-degrees) can utilize the Fourier Transform, while larger flip angle designs such as 90 and 180-degree pulses benefit from advanced techniques like the Shinnar-Le Roux (SLR) algorithm.

14.1 Learning Goals

1. Manipulate MRI sequence parameters to improve performance
    - Understand how to design RF pulses

14.2 General Considerations - Pulse Duration and Ripple

RF pulse designs should address the limited available pulse duration and, for large flip angle designs (e.g. 90 and 180 degrees) the non-linearities in the Bloch equation.

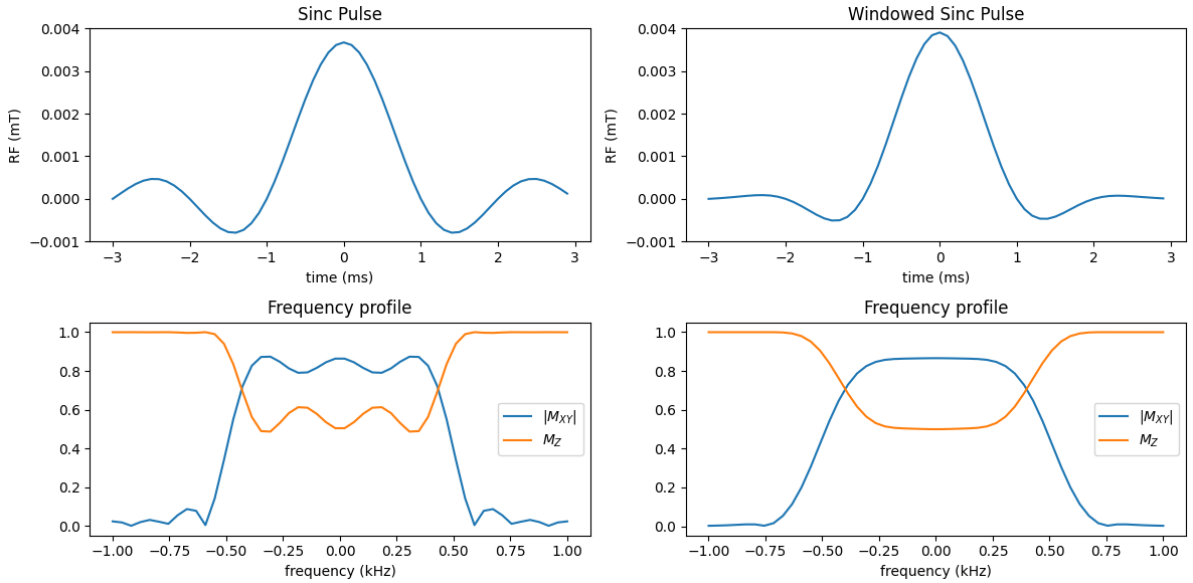
Limited available durations are addressed, in small flip angle pulses, by using windowing functions along with sinc pulses. The end result are windowed sinc pulses, which have less ringing in the coil profile as illustrated below.

```
import numpy as np
import matplotlib.pyplot as plt
import sys

sys.path.append('../Physics/')
from bloch_rotate import bloch_rotate

# parameters
gammabar = 42.58  # kHz/mT
M0 = 1.0
M_equilibrium = np.array([0.0, 0.0, M0])
dt = 0.1  # ms
flip = 60.0

tmax = 6.0
N = int(tmax / dt)
t = (np.arange(-N//2, N//2) * dt).astype(float)

BW = 1.0  # kHz
df = np.linspace(-BW, BW)

# side-by-side plots for Sinc and Windowed Sinc pulses
fig, axs = plt.subplots(2, 2, figsize=(12, 6))
```

(continues on next page)

(continued from previous page)

```
# Sinc pulse
RF_sinc = np.sinc(t)
RF_sinc = (flip * np.pi / 180.0) * RF_sinc / np.sum(RF_sinc) / (2 * np.pi * gammabar␣
↪* dt)

M_sinc = np.tile(M_equilibrium[:, None], (1, len(df)))
for n_idx in range(len(t)):
    for f_idx in range(len(df)):
        B = [np.real(RF_sinc[n_idx]), np.imag(RF_sinc[n_idx]), df[f_idx] / gammabar]
        M_sinc[:, f_idx] = bloch_rotate(M_sinc[:, f_idx], dt, B)

axs[0, 0].plot(t, RF_sinc)
axs[0, 0].set_xlabel('time (ms)')
axs[0, 0].set_ylabel('RF (mT)')
axs[0, 0].set_ylim(-0.001, 0.004)
axs[0, 0].set_title('Sinc Pulse')

axs[1, 0].plot(df, np.sqrt(M_sinc[0, :]**2 + M_sinc[1, :]**2), label=r'$|M_{XY}|$')
axs[1, 0].plot(df, M_sinc[2, :], label=r'$M_{Z}$')
axs[1, 0].set_title('Frequency profile')
axs[1, 0].set_xlabel('frequency (kHz)')
axs[1, 0].legend()

# Windowed Sinc Pulse
RF_win = np.hamming(N) * np.sinc(t)
RF_win = (flip * np.pi / 180.0) * RF_win / np.sum(RF_win) / (2 * np.pi * gammabar *␣
↪dt)

M_win = np.tile(M_equilibrium[:, None], (1, len(df)))
for n_idx in range(len(t)):
    for f_idx in range(len(df)):
        B = [np.real(RF_win[n_idx]), np.imag(RF_win[n_idx]), df[f_idx] / gammabar]
        M_win[:, f_idx] = bloch_rotate(M_win[:, f_idx], dt, B)

axs[0, 1].plot(t, RF_win)
axs[0, 1].set_xlabel('time (ms)')
axs[0, 1].set_ylabel('RF (mT)')
axs[0, 1].set_ylim(-0.001, 0.004)
axs[0, 1].set_title('Windowed Sinc Pulse')

axs[1, 1].plot(df, np.sqrt(M_win[0, :]**2 + M_win[1, :]**2), label=r'$|M_{XY}|$')
axs[1, 1].plot(df, M_win[2, :], label=r'$M_{Z}$')
axs[1, 1].set_title('Frequency profile')
axs[1, 1].set_xlabel('frequency (kHz)')
axs[1, 1].legend()

fig.tight_layout()
plt.show()
```



14.3 Time-bandwidth Product and Pulse Selectivity

Arguably the most important RF pulse design parameter is the "time-bandwidth" product. As the name implies, this is the pulse duration (time) times the pulse bandwidth

$$ TBW = T_{rf} \times BW_{rf} $$

A given RF pulse shape has an associated TBW, and this shape can be stretched or shrunk in time to modulate the resulting bandwidth.

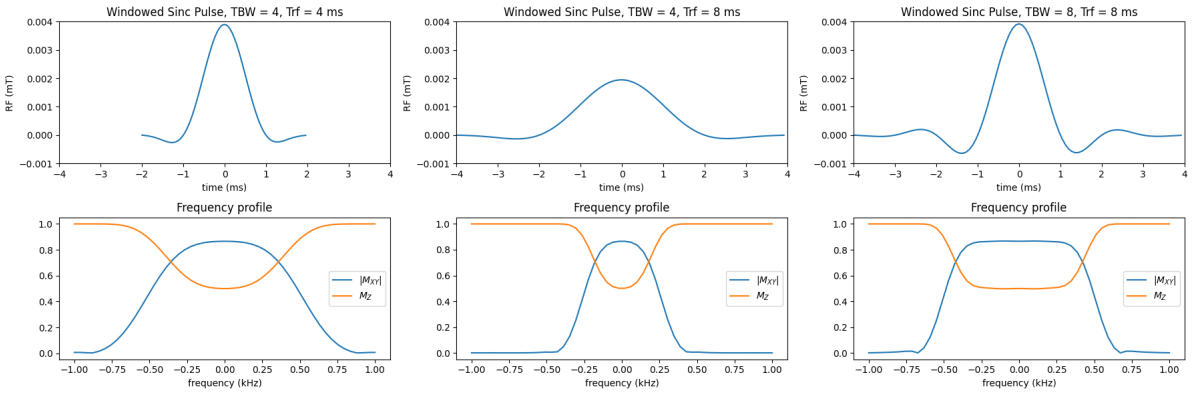
The key feature of the TBW is that it controls the *pulse selectivity*, that is how sharp the pulse profile is, or, in other words, how quickly the profile goes from the desired flip angle to zero flip angle.

```
# Convert MATLAB cell to Python and place the three pulse panels side-by-side
# Uses existing variables: gammabar, flip, df, M_equilibrium, bloch_rotate

cases = [
    (4, 4, 'TBW = 4, Trf = 4 ms'),
    (4, 8, 'TBW = 4, Trf = 8 ms'),
    (8, 8, 'TBW = 8, Trf = 8 ms'),
]

Np = 100  # sample points for pulse shape
fig, axs = plt.subplots(2, 3, figsize=(18, 6))

for i, (TBW, Trf, title) in enumerate(cases):
    IN = np.arange(-Np/2, Np/2) / Np
    RF_shape = np.hamming(Np) * np.sinc(IN * TBW)       # windowed sinc
    t_pulse = IN * Trf                                   # time axis (ms)
    dt_pulse = Trf / Np

    RF = (flip * np.pi / 180.0) * RF_shape / np.sum(RF_shape) / (2 * np.pi * gammabar␣
 ↪* dt_pulse)

    M = np.tile(M_equilibrium[:, None], (1, len(df)))
```

(continues on next page)

(continued from previous page)

```
    for n_idx in range(len(t_pulse)):
        for f_idx in range(len(df)):
            B_field = [np.real(RF[n_idx]), np.imag(RF[n_idx]), df[f_idx] / gammabar]
            M[:, f_idx] = bloch_rotate(M[:, f_idx], dt_pulse, B_field)

    axs[0, i].plot(t_pulse, RF)
    axs[0, i].set_xlim(-4, 4)
    axs[0, i].set_xlabel('time (ms)')
    axs[0, i].set_ylabel('RF (mT)')
    axs[0, i].set_ylim(-0.001, 0.004)
    axs[0, i].set_title(f'Windowed Sinc Pulse, {title}')

    axs[1, i].plot(df, np.sqrt(M[0, :]**2 + M[1, :]**2), label=r'$|M_{XY}|$')
    axs[1, i].plot(df, M[2, :], label=r'$M_{Z}$')
    axs[1, i].set_xlabel('frequency (kHz)')
    axs[1, i].set_title('Frequency profile')
    axs[1, i].legend()

fig.tight_layout()
plt.show()
```



Notice that the higher TBW pulse has much sharper frequency profile for the same pulse duration! (But also higher peak power.)

14.4 High flip angle pulse design

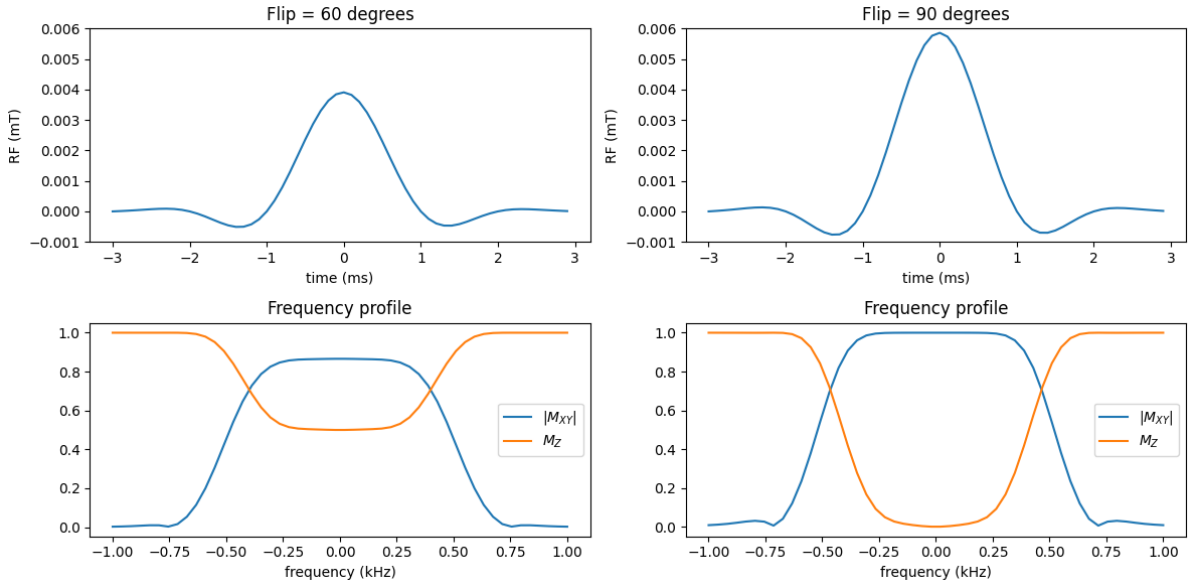
For small flip angles, the frequency profile of a RF pulse is approximately given by the Fourier Transform. However, non-linearities in the Bloch equation cause this approximation to break down for large flip angles, as shown below by the resulting "ripples" in the frequency profile when the flip angle of the windowed.

```
# small tip vs large tip (converted from MATLAB)
fig, axs = plt.subplots(2, 2, figsize=(12, 6))

# common RF shape (windowed sinc)
RF_shape = np.hamming(N) * np.sinc(t)

for col, flip_angle in enumerate([60, 90]):
    RF = (flip_angle * np.pi / 180.0) * RF_shape / np.sum(RF_shape) / (2 * np.pi *␣
 ↪gammabar * dt)
```

(continues on next page)

(continued from previous page)

```
    M = np.tile(M_equilibrium[:, None], (1, len(df)))
    for n_idx in range(len(t)):
        for f_idx in range(len(df)):
            B_field = [np.real(RF[n_idx]), np.imag(RF[n_idx]), df[f_idx] / gammabar]
            M[:, f_idx] = bloch_rotate(M[:, f_idx], dt, B_field)

    # top row: pulse in time
    axs[0, col].plot(t, RF)
    axs[0, col].set_xlabel('time (ms)')
    axs[0, col].set_ylim(-0.001, 0.006)
    axs[0, col].set_ylabel('RF (mT)')
    axs[0, col].set_title(f'Flip = {flip_angle} degrees')

    # bottom row: frequency profile
    axs[1, col].plot(df, np.sqrt(M[0, :]**2 + M[1, :]**2), label=r'$|M_{XY}|$')
    axs[1, col].plot(df, M[2, :], label=r'$M_{Z}$')
    axs[1, col].set_xlabel('frequency (kHz)')
    axs[1, col].set_title('Frequency profile')
    axs[1, col].legend()

fig.tight_layout()
plt.show()
```



Note that with Fourier Transform based design small ripples that appear outside the desired slice.

The solution is to use a more sophisticated pulse design that takes into account these non-linearities, such as the Shinnar-Le Roux (SLR) transform.

```
# Large-tip (90 deg) pulse comparison: Windowed Sinc vs SLR (converted from MATLAB)
from scipy.io import loadmat
import numpy as np
import matplotlib.pyplot as plt

# use existing variables: gammabar, M_equilibrium, dt, t, N, df, bloch_rotate
flip = 90.0
```

(continues on next page)

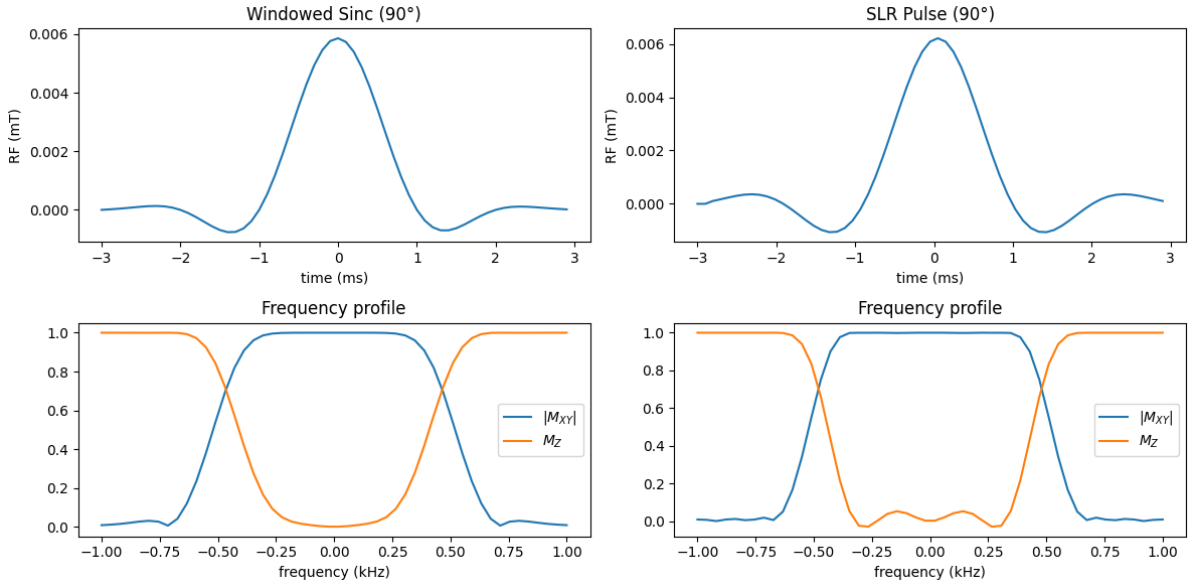
(continued from previous page)

```
fig, axs = plt.subplots(2, 2, figsize=(12, 6))

# --- Windowed sinc pulse (recreate the shape and scale for 90 deg) ---
RF_wsinc = np.hamming(N) * np.sinc(t)            # windowed sinc shape
RF_win90 = (flip * np.pi / 180.0) * RF_wsinc / np.sum(RF_wsinc) / (2 * np.pi *␣
↪gammabar * dt)

# simulate Bloch for windowed sinc
M_win = np.tile(M_equilibrium[:, None], (1, len(df)))
for n_idx in range(len(RF_win90)):
    for f_idx in range(len(df)):
        B_field = [np.real(RF_win90[n_idx]), np.imag(RF_win90[n_idx]), df[f_idx] /␣
↪gammabar]
        M_win[:, f_idx] = bloch_rotate(M_win[:, f_idx], dt, B_field)

# top-left: RF waveform, bottom-left: frequency profile
axs[0, 0].plot(t, RF_win90)
axs[0, 0].set_xlabel('time (ms)')
axs[0, 0].set_ylabel('RF (mT)')
axs[0, 0].set_title('Windowed Sinc (90°)')

axs[1, 0].plot(df, np.sqrt(M_win[0, :]**2 + M_win[1, :]**2), label=r'$|M_{XY}|$')
axs[1, 0].plot(df, M_win[2, :], label=r'$M_{Z}$')
axs[1, 0].set_xlabel('frequency (kHz)')
axs[1, 0].set_title('Frequency profile')
axs[1, 0].legend()

# --- SLR pulse: try to load .mat, fall back to existing RF variable if load fails ---
# SLR pulse created with dzrf() in MATLAB rf_tools using
#   RF = dzrf(N-1, tmax*1, 'ex');
RF_slr_raw = None
try:
    mat = loadmat('../RF Pulses/RFpulse_excitation_tbw6_SLR.mat')
    # pick the first ndarray-like variable that isn't metadata
    for k, v in mat.items():
        if not k.startswith('__') and isinstance(v, np.ndarray) and v.size > 0:
            RF_slr_raw = v
            break
    if RF_slr_raw is None:
        raise RuntimeError('no candidate RF array found in .mat')
    # flatten and ensure 1D
    RF_slr_raw = RF_slr_raw.flatten()
except Exception:
    # fallback: use existing RF variable in the notebook if available
    try:
        RF_slr_raw = RF.flatten()  # use preexisting RF if present
    except Exception:
        raise

# MATLAB code prepended a zero: [0, RF]
RF_slr90 = (flip * np.pi / 180.0) * np.concatenate(([0.0], RF_slr_raw)) / np.sum(np.
↪concatenate(([0.0], RF_slr_raw))) / (2 * np.pi * gammabar * dt)

# make time axis for SLR: length must match RF_slr90; reuse t if lengths match,␣
↪otherwise build symmetric axis
if len(RF_slr90) == len(t):
    t_slr = t
```

(continues on next page)

(continued from previous page)

```
else:
    t_slr = np.linspace(t[0], t[-1], len(RF_slr90))

# simulate Bloch for SLR
M_slr = np.tile(M_equilibrium[:, None], (1, len(df)))
for n_idx in range(len(RF_slr90)):
    for f_idx in range(len(df)):
        B_field = [np.real(RF_slr90[n_idx]), np.imag(RF_slr90[n_idx]), df[f_idx] /␣
↪gammabar]
        M_slr[:, f_idx] = bloch_rotate(M_slr[:, f_idx], dt, B_field)

# top-right: SLR RF, bottom-right: frequency profile
axs[0, 1].plot(t_slr, np.real(RF_slr90))
axs[0, 1].set_xlabel('time (ms)')
axs[0, 1].set_ylabel('RF (mT)')
axs[0, 1].set_title('SLR Pulse (90°)')

axs[1, 1].plot(df, np.sqrt(M_slr[0, :]**2 + M_slr[1, :]**2), label=r'$|M_{XY}|$')
axs[1, 1].plot(df, M_slr[2, :], label=r'$M_{Z}$')
axs[1, 1].set_xlabel('frequency (kHz)')
axs[1, 1].set_title('Frequency profile')
axs[1, 1].legend()

fig.tight_layout()
plt.show()
```



Note that now the ripples in the M_{XY} profile are now removed! (There is some increased ripple in M_Z, but this isn't an issue for excitation pulses). The RF pulses themselves are only subtly different.

The degradation of the slice (frequency) profile of sinc-pulses becomes substantial with a 180-degree flip angle inversion pulse:

```
# Inversion (large-tip) pulses (converted from MATLAB)
# uses existing: gammabar, M_equilibrium, dt, t, N, df, bloch_rotate

flip = 180.0
```

(continues on next page)

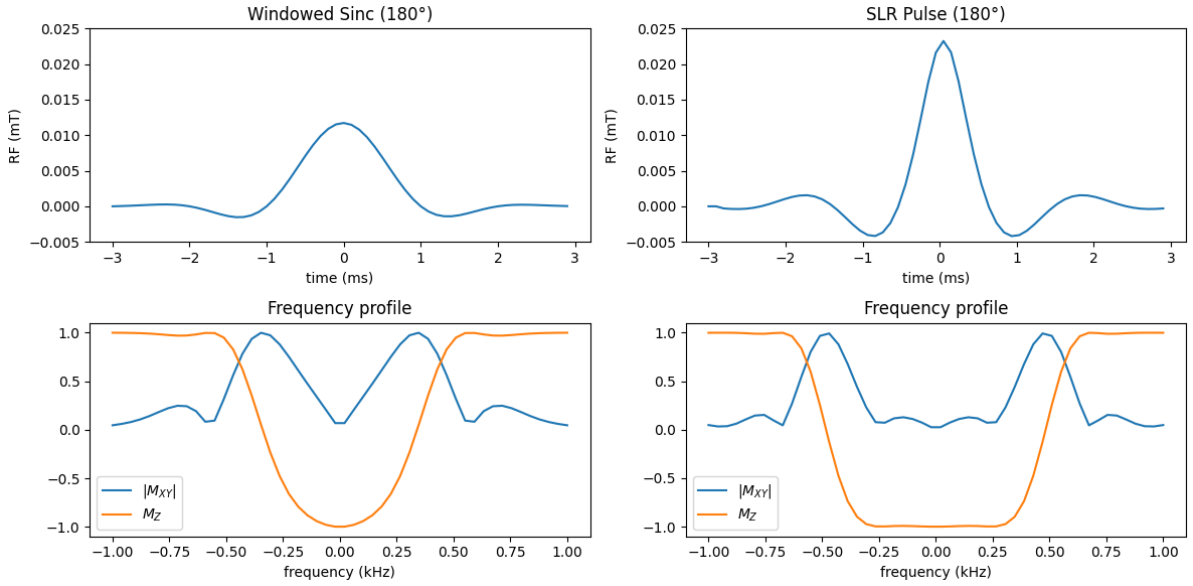
(continued from previous page)

```
fig, axs = plt.subplots(2, 2, figsize=(12, 6))

# -- Windowed Sinc Pulse --
RF_shape = np.hamming(N) * np.sinc(t)
RF_win_inv = (flip * np.pi / 180.0) * RF_shape / np.sum(RF_shape) / (2 * np.pi *␣
 ↪gammabar * dt)

M_win_inv = np.tile(M_equilibrium[:, None], (1, len(df)))
for n_idx in range(len(RF_win_inv)):
    for f_idx in range(len(df)):
        B_field = [np.real(RF_win_inv[n_idx]), np.imag(RF_win_inv[n_idx]), df[f_idx] /
 ↪ gammabar]
        M_win_inv[:, f_idx] = bloch_rotate(M_win_inv[:, f_idx], dt, B_field)

# top-left: RF, bottom-left: frequency profile
axs[0, 0].plot(t, RF_win_inv)
axs[0, 0].set_xlabel('time (ms)')
axs[0, 0].set_ylabel('RF (mT)')
axs[0, 0].set_ylim(-0.005, 0.025)
axs[0, 0].set_title('Windowed Sinc (180°)')

axs[1, 0].plot(df, np.sqrt(M_win_inv[0, :]**2 + M_win_inv[1, :]**2), label=r'$|M_{XY}|
 ↪$')
axs[1, 0].plot(df, M_win_inv[2, :], label=r'$M_{Z}$')
axs[1, 0].set_xlabel('frequency (kHz)')
axs[1, 0].set_title('Frequency profile')
axs[1, 0].legend()

# -- SLR Pulse: try to load .mat, fall back to existing RF variable if load fails --
# SLR pulse created with dzrf() in MATLAB rf_tools using
#   RF = dzrf(N-1, tmax*1, 'inv');
RF_slr_raw = None
try:
    mat = loadmat('../RF Pulses/RFpulse_inversion_tbw6_SLR.mat')
    for k, v in mat.items():
        if not k.startswith('__') and isinstance(v, np.ndarray) and v.size > 0:
            RF_slr_raw = v
            break
    if RF_slr_raw is None:
        raise RuntimeError('no candidate RF array found in .mat')
    RF_slr_raw = RF_slr_raw.flatten()
except Exception:
    # fallback: use existing RF (if present) / RF_slr_raw from workspace
    try:
        RF_slr_raw = RF.flatten()
    except Exception:
        raise

# MATLAB prepended zero in original script: [0, RF]
RF_slr_inv = (flip * np.pi / 180.0) * np.concatenate(([0.0], RF_slr_raw)) / np.sum(np.
 ↪concatenate(([0.0], RF_slr_raw))) / (2 * np.pi * gammabar * dt)

# time axis for SLR
if len(RF_slr_inv) == len(t):
    t_slr = t
else:
```

(continues on next page)

(continued from previous page)

```
    t_slr = np.linspace(t[0], t[-1], len(RF_slr_inv))

M_slr_inv = np.tile(M_equilibrium[:, None], (1, len(df)))
for n_idx in range(len(RF_slr_inv)):
    for f_idx in range(len(df)):
        B_field = [np.real(RF_slr_inv[n_idx]), np.imag(RF_slr_inv[n_idx]), df[f_idx] /
 ↪ gammabar]
        M_slr_inv[:, f_idx] = bloch_rotate(M_slr_inv[:, f_idx], dt, B_field)

# top-right: SLR RF, bottom-right: frequency profile
axs[0, 1].plot(t_slr, np.real(RF_slr_inv))
axs[0, 1].set_xlabel('time (ms)')
axs[0, 1].set_ylim(-0.005, 0.025)
axs[0, 1].set_ylabel('RF (mT)')
axs[0, 1].set_title('SLR Pulse (180°)')

axs[1, 1].plot(df, np.sqrt(M_slr_inv[0, :]**2 + M_slr_inv[1, :]**2), label=r'$|M_{XY}|
 ↪$')
axs[1, 1].plot(df, M_slr_inv[2, :], label=r'$M_{Z}$')
axs[1, 1].set_xlabel('frequency (kHz)')
axs[1, 1].set_title('Frequency profile')
axs[1, 1].legend()

fig.tight_layout()
plt.show()
```



Note that the inversion profile M_Z is much more selective, in other words more like a rectangular function, with the SLR pulse design. The SLR RF pulse is now noticeably different from a windowed sinc.

Part VI

Image Formation

CHAPTER

FIFTEEN

SPATIAL ENCODING

To create images of the net magnetization, magnetic field gradients are applied in order to modulate the resonance frequency as a function of position. This section describes the effect of magnetic field gradients during data acquisition to create spatial encoding, and introduces the intuition behind frequency and phase encoding.

15.1 Learning Goals

1. Describe how images are formed
    - Describe the effects of gradients on the signal
    - Understand how frequency encoding works
    - Understand how phase encoding works

15.2 Effects of Magnetic Field Gradients

Applied magnetic field gradients change the magnetic field linearly with position:

$$\Delta B_z = \vec{G}\cdot\vec{r}$$

which leads to linear variation in the resonance frequency as a function of position:

$$ f = \bar \gamma \ \vec{G}\cdot\vec{r} $$

This forms the basis for spatial encoding, where fundamentally position is encoded in the frequency of the net magnetization.

The received signal in MRI is an accumulation over space of all the net magnetizations, so without gradients there is no fine spatial encoding. (The RF coil sizes can provide some spatial encoding, but on the order of cm.)

Reconstructing images from MRI relies on the Fourier Transform in order to convert the spatial information encoded in frequency into space. The specifics of this are described later.

15.3 Frequency and Phase encoding, or FT Imaging

By far the most common way to perform spatial encoding is using so-called "frequency encoding" and "phase encoding". This is also known FT imaging, since we reconstruct images with a 2D Fourier Transform.

It consists of 2 types of encoding:

1. Frequency encoding - one dimension is encoded using a constant gradient applied during data acquisition
2. Phase encoding - additional dimensions are encoded using gradient applied before data acquisition. This gradient is incremented to provide complete spatial encoding. Phase encoding is applied in 1 dimension for 2D imaging, and 2 dimensions for 3D imaging

This approach is referred to as Cartesian k-space trajectory, which the name refers to the fact that data is sampled on a regularly spaced grid in k-space, which is introduced later.

15.3.1 Frequency Encoding

In this approach, one dimension of the object is encoded using "frequency encoding". This means that, after RF excitation, a magnetic field gradient is turned on and the signal is read out. The frequencies present in the signal correspond to given spatial locations. By convention, this is applied in the x-direction (but in practice can be rotated to any direction);

$$f = \bar \gamma G_{xr} x$$ where G_{xr} is the readout gradient amplitude. Therefore, the position is proportional to the frequency:

$$ x = \frac{f}{\bar \gamma G_{xr}}$$

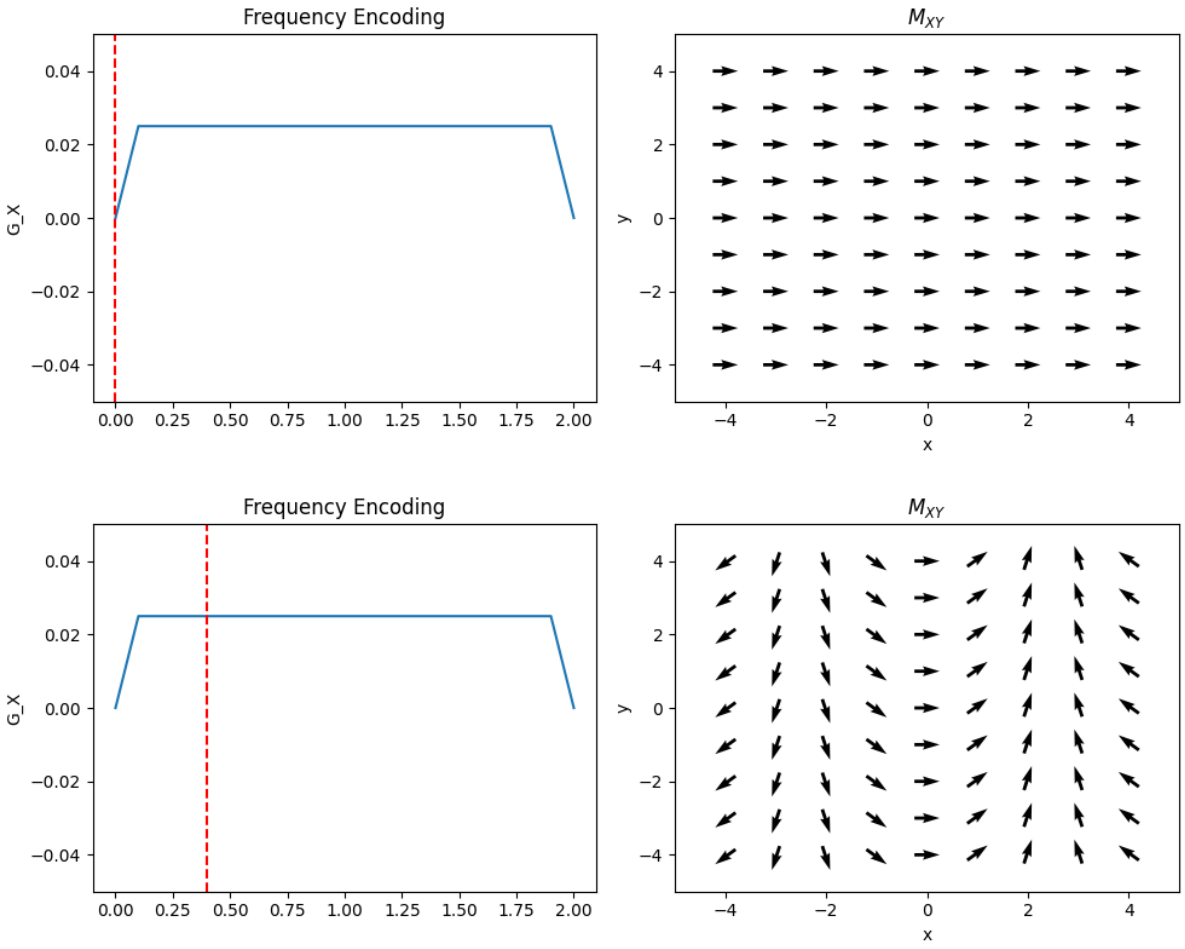
From frequency encoding data alone, a 1D image can be reconstructed with an inverse Fourier Transform.

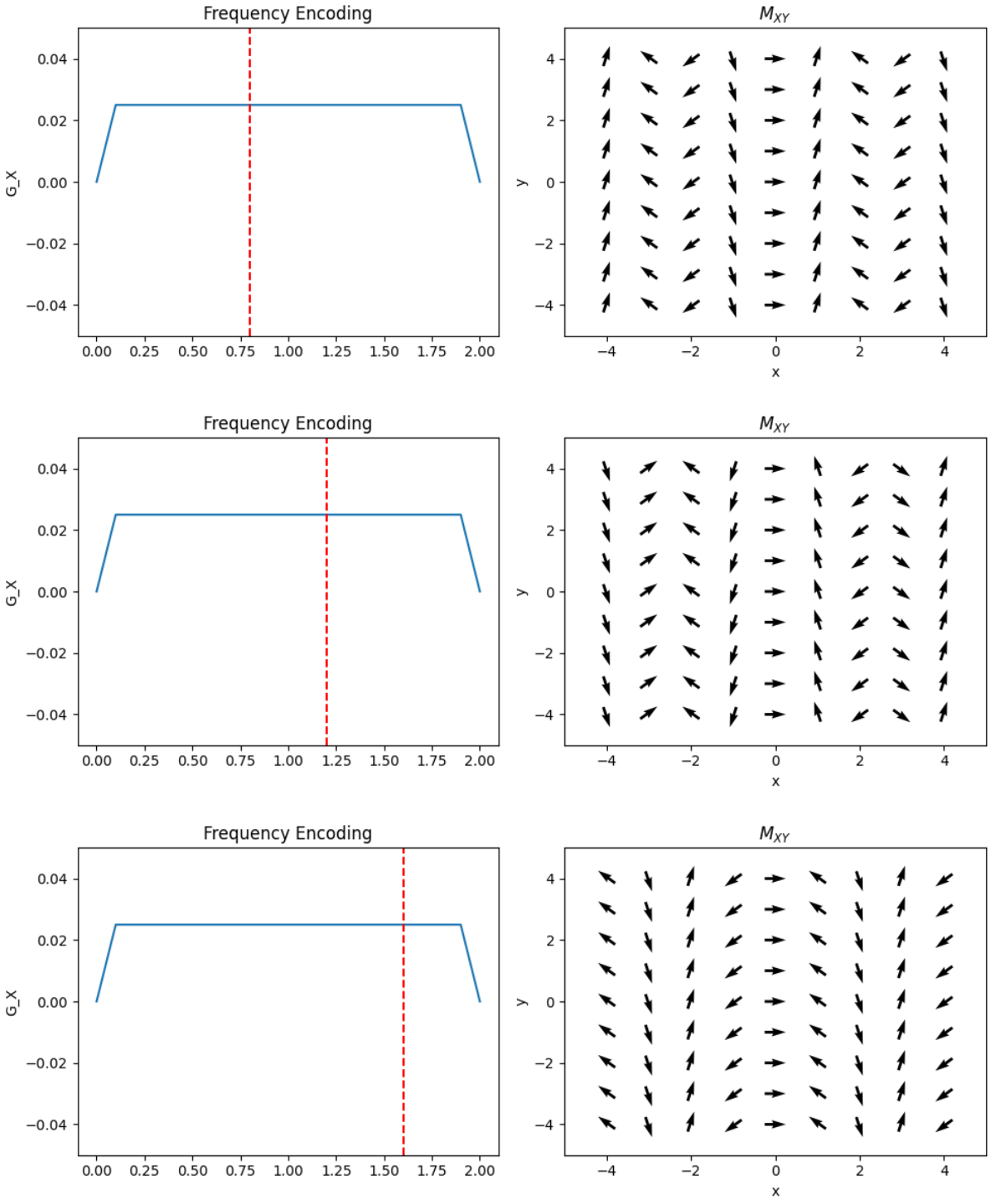
```
import numpy as np
import matplotlib.pyplot as plt

N = 8

# simulate frequency encoding
Mxy = np.ones((N+1, N+1), dtype=complex)

x = np.arange(-N/2, N/2+1)
x, y = np.meshgrid(x, x)
Splot = 0.5

kFE = 1/2
dt = 0.1
Tfe = 2

GAMMA = 42.58

for tfe in np.linspace(0, 1, 6) * Tfe:
    phase_x = 2 * np.pi * kFE * x * tfe / Tfe
    Mxy_FE = Mxy * np.exp(1j * phase_x)

    plt.figure(figsize=(10, 4))
    plt.subplot(1, 2, 1)
    t_axis = np.arange(0, Tfe+dt, dt)
    plt.plot(t_axis, np.concatenate([[0], np.ones(len(t_axis)-2)*kFE/(Tfe/dt), [0]]))
    plt.plot([tfe, tfe], [-0.05, 0.05], 'r--')
    plt.ylim([-0.05, 0.05])
    plt.ylabel('G_X')
```

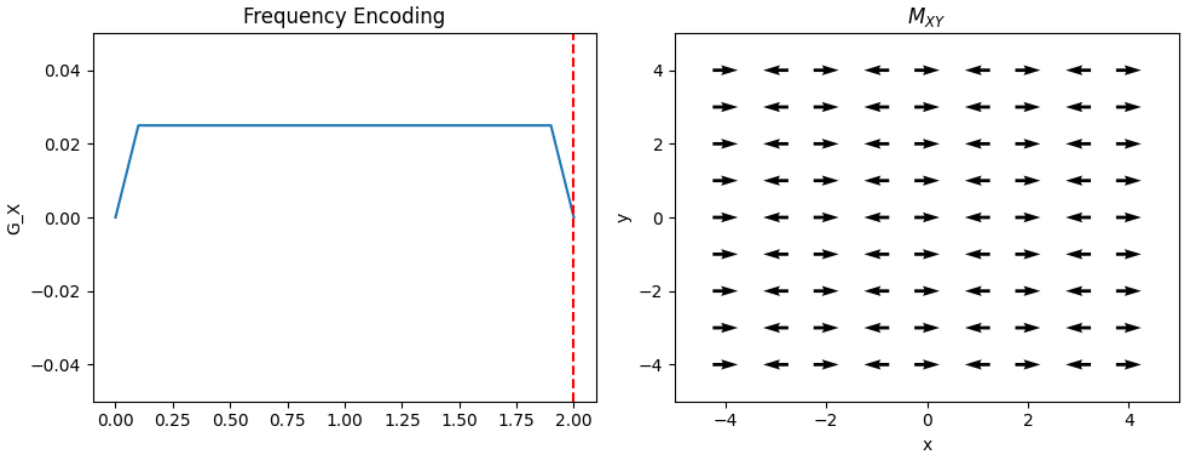
(continues on next page)

(continued from previous page)

```
    plt.title('Frequency Encoding')

    plt.subplot(1, 2, 2)
    plt.quiver(x - np.real(Mxy_FE)*Splot/2, y - np.imag(Mxy_FE)*Splot/2,
               np.real(Mxy_FE), np.imag(Mxy_FE), scale=10/Splot)
    plt.xlabel('x')
    plt.ylabel('y')
    plt.xlim([-N/2-1, N/2+1])
    plt.ylim([-N/2-1, N/2+1])
    plt.title(r'$M_{XY}$')
    plt.tight_layout()
    plt.show()
```




Frequency Encoding
G_X
0.04
0.02
0.00
−0.02
−0.04
0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00
M_{XY}
y
x
4
2
0
−2
−4
−4 −2 0 2 4
Frequency Encoding
G_X
0.04
0.02
0.00
−0.02
−0.04
0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00
M_{XY}
y
x
4
2
0
−2
−4
−4 −2 0 2 4
Frequency Encoding
G_X
0.04
0.02
0.00
−0.02
−0.04
0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00
M_{XY}
y
x
4
2
0
−2
−4
−4 −2 0 2 4



15.4 Phase Encoding

Typically the 2nd (and optionally 3rd) dimensions of the object are encoded using "phase encoding". This means that, after RF excitation but before the frequency encoding gradient, a pulsed gradient is applied such that the location is encoded in the phase of the next magnetization:

$$ \Phi(n) = \gamma (-G_{yp} + (n-1) G_{yi}) t_y y $$

This measurement is repeated for $n = 1, \ldots, N_{PE}$. G_{yp} is the maximum phase encoding gradient strength, G_{yi} is the phase encoding gradient amplitude increment, and t_y is the phase encoding gradient duration. Note that $2 G_{yp} = (N_{PE} - 1) G_{yi}$.

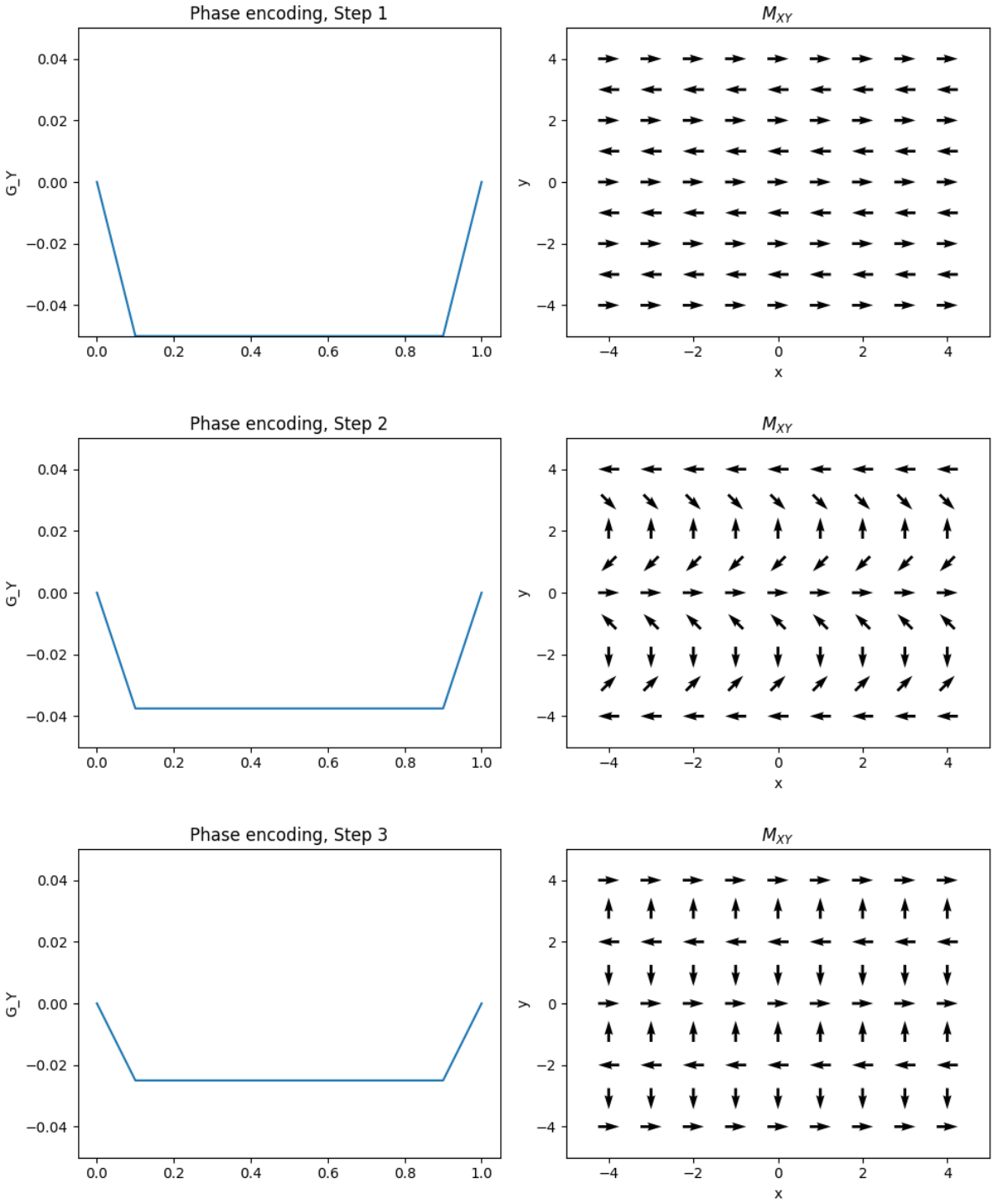
This is equivalent to taking different samples of a frequency encoding gradient.

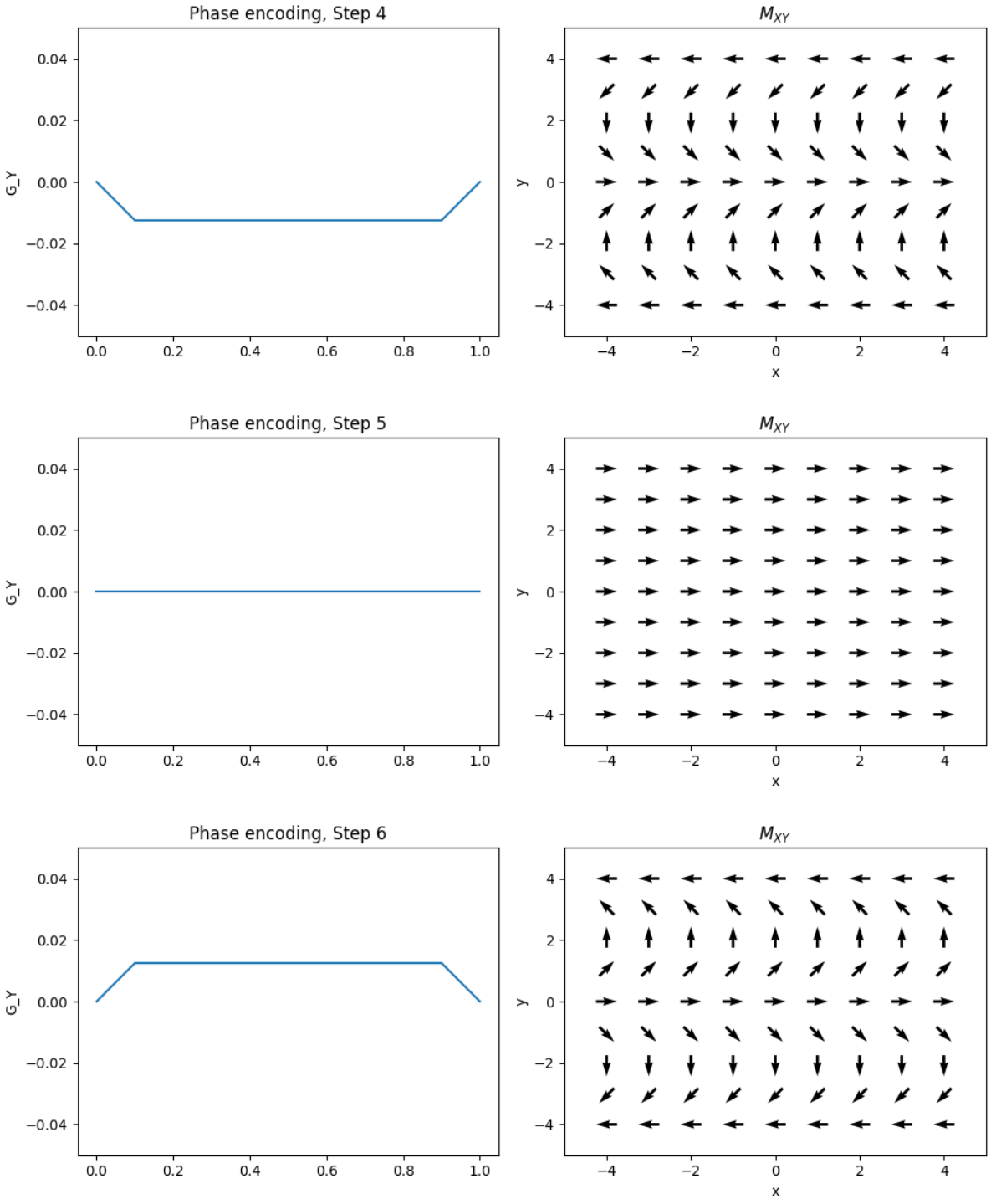
```
# 2D object: simulate frequencies + phase encoding

kPE = np.arange(-N/2, N/2+1) / N  # phase encoding steps
Tpe = 1

for Ipe, kpe in enumerate(kPE):
    tpe = Tpe  # only using tpe = Tpe as in the original code
    phase_y = 2 * np.pi * kpe * y * tpe / Tpe
    Mxy_PE = Mxy * np.exp(1j * phase_y)

    plt.figure(figsize=(10, 4))
    plt.subplot(1, 2, 1)
    t_axis = np.arange(0, Tpe + dt, dt)
    # Gradient waveform: 0, plateau, 0
    grad = np.concatenate([[0], np.ones(len(t_axis)-2) * kpe / (Tpe/dt), [0]])
    plt.plot(t_axis, grad)
    plt.ylim([-0.05, 0.05])
    plt.ylabel('G_Y')
    plt.title(f'Phase encoding, Step {Ipe+1}')

    plt.subplot(1, 2, 2)
    plt.quiver(x - np.real(Mxy_PE)*Splot/2, y - np.imag(Mxy_PE)*Splot/2,
               np.real(Mxy_PE), np.imag(Mxy_PE), scale=10/Splot)
    plt.xlabel('x')
    plt.ylabel('y')
```

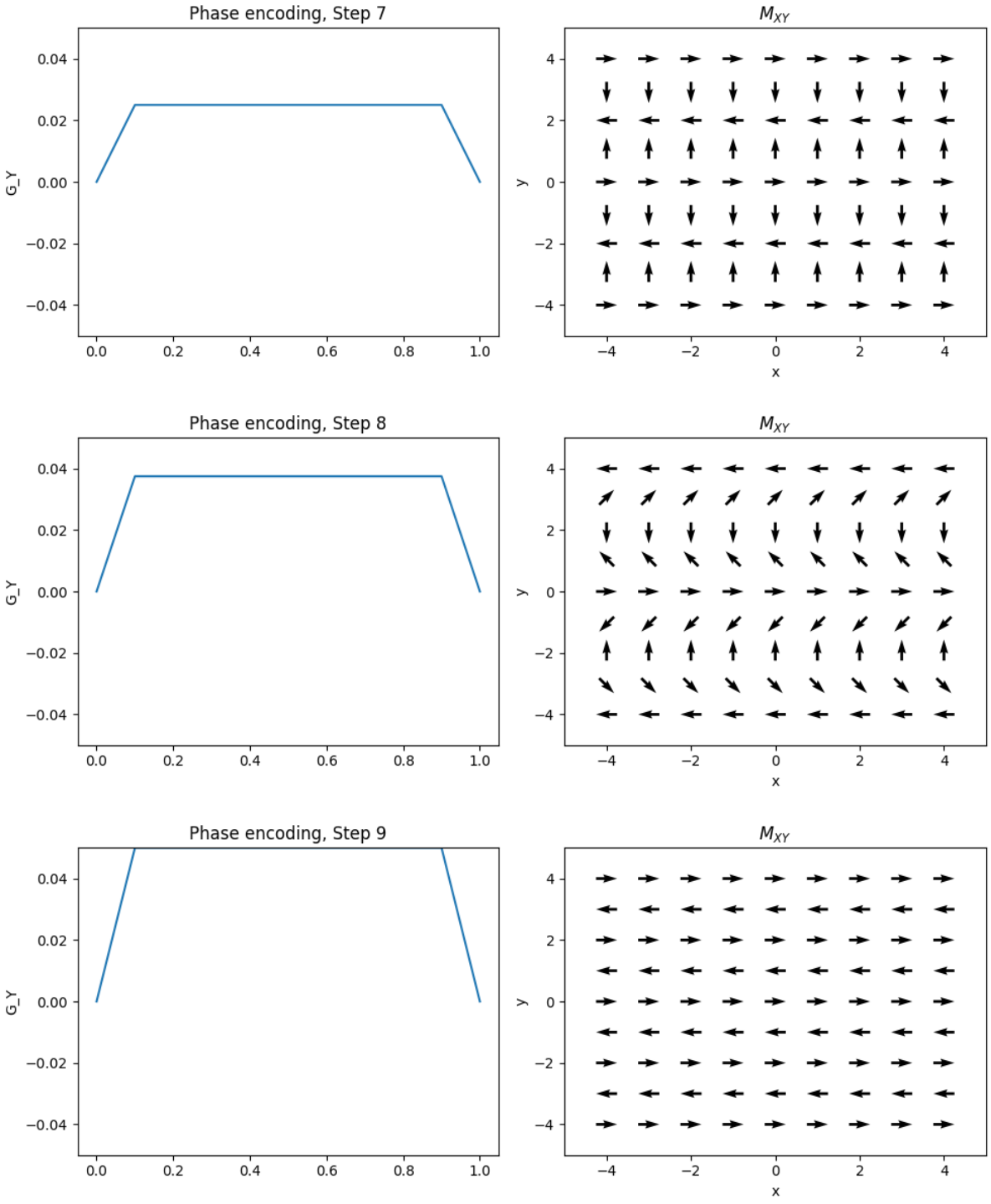
(continues on next page)

(continued from previous page)

```
    plt.xlim([-N/2-1, N/2+1])
    plt.ylim([-N/2-1, N/2+1])
    plt.title(r'$M_{XY}$')
    plt.tight_layout()
    plt.show()
```

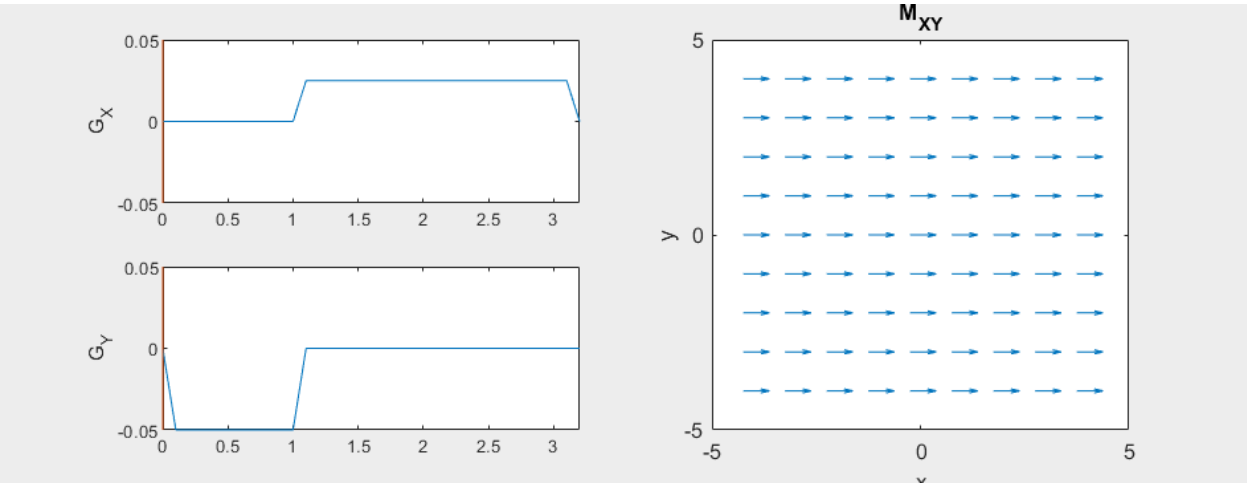



Phase encoding, Step 4
M_{XY}
G_Y
y
x
Phase encoding, Step 5
M_{XY}
G_Y
y
x
Phase encoding, Step 6
M_{XY}
G_Y
y
x

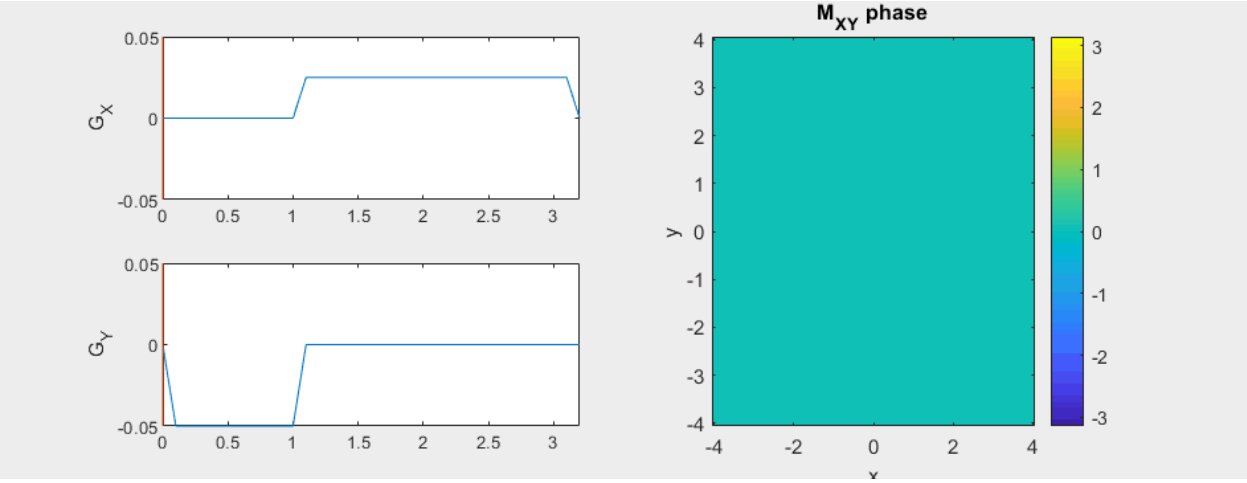

Phase encoding, Step 7
M_{XY}
Phase encoding, Step 8
M_{XY}
Phase encoding, Step 9
M_{XY}
G_Y
y
x

15.5 Frequency and Phase Encoding

The following simulation of the net magnetizations shows how first the phase encoding gradient (G_Y) creates some phase variation in y, and then during the frequency encoding gradient (G_X) the net magnetizations rotate at varying frequencies depending on their x position:



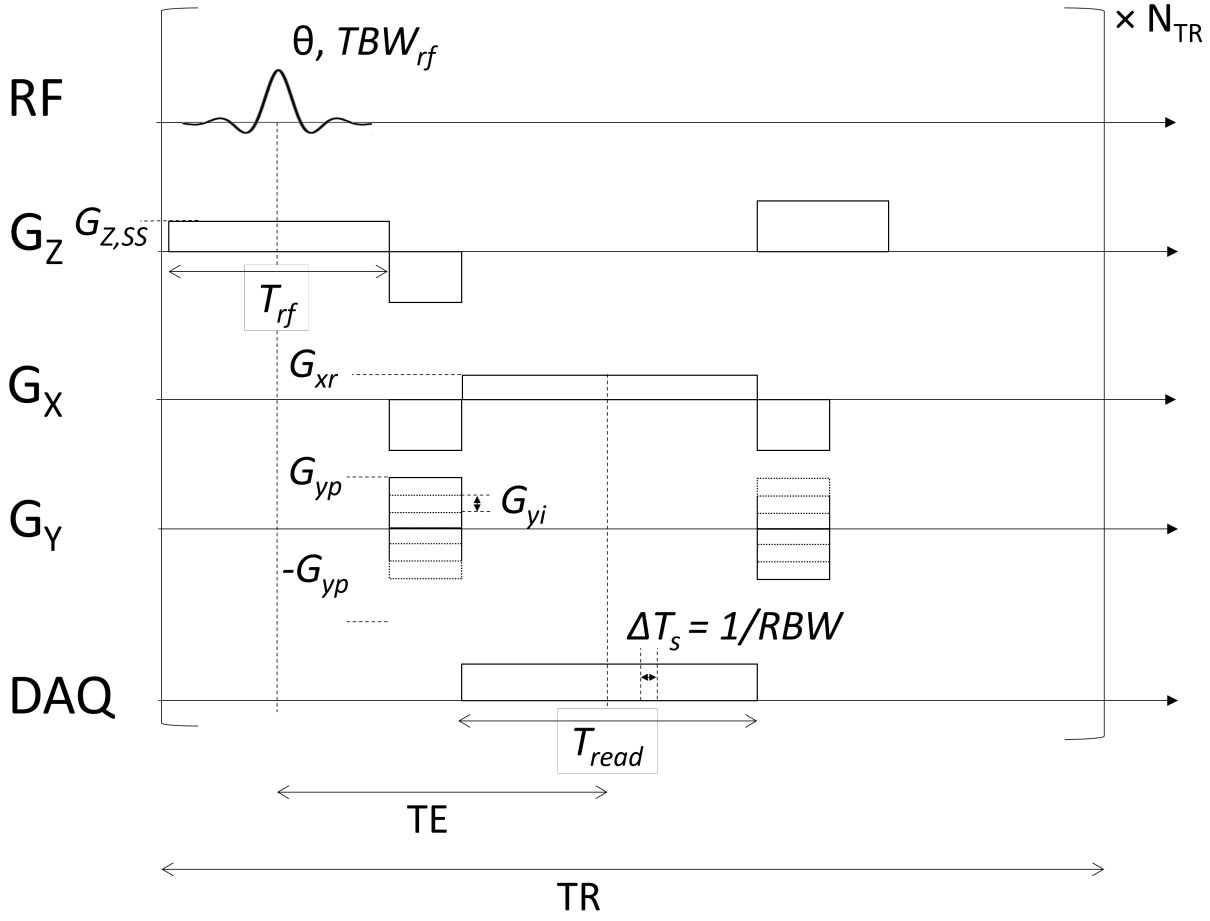
Instead of viewing the net magnetizations, we can also visualize this encoding as a map of the phase of the transverse magnetization:



(See `spatial_encoding_Mxy_illustration.m` for code generating this movie)

15.6 2D FT Imaging Pulse Sequence

The following pulse sequence diagram shows a Cartesian 2D FT sequence with frequency and phase encoding including relevant pulse amplitudes and durations. A gradient-echo sequence is shown, and the principles can simply be extended to a spin-echo sequence. Idealized rectangular gradients are shown for simplicity.


× N_TR
θ, TBW_rf
RF
G_Z
G_Z,SS
T_rf
G_X
G_xr
G_Y
G_yp
G_yi
-G_yp
ΔT_s = 1/RBW
DAQ
T_read
TE
TR

CHAPTER

SIXTEEN

K-SPACE

MRI is a so-called Fourier imaging technique, where data is acquired in the Fourier Transform domain. The Fourier Transform domain is referred to as "k-space". This section describes the how to characterize MRI spatial encoding using k-space and introduces the concept of k-space trajectories for sampling data.

16.1 Learning Goals

1. Describe how images are formed
    - Understand why MRI is a Fourier imaging method
    - Describe how MRI data is accumulated and sorted
    - Describe what a k-space trajectory is

16.2 Generalized Spatial Encoding

In general, the application of magnetic field gradients will change the phase of the transverse magnetization with dependence on the gradient waveforms used and position:

$$ M_{XY}(\vec{r}, t) = M_{XY}(\vec{r}, 0) \exp\left(-i \gamma \int_0^t \vec{G}(\tau) \cdot \vec{r} \ d\tau \right) $$

Using the notation $m(\vec{r}) = M_{XY}(\vec{r}, t=0)$, this can be written as

$$M_{XY}(\vec{r}, t) = m(\vec{r}) \exp\left(-i \gamma \int_0^t \vec{G}(\tau) \cdot \vec{r} \ d\tau \right)$$

A key fact here is that the phase accumulation depends on the cumulative gradient area, or the integral of the gradient. This means we can turn the gradients on and off to start and stop the phase accumulation, and also reverse the gradient polarity to undo any prior phase accumulation.

16.2.1 Definition of k-space

Here, we introduce the concept of k-space, which is a simplified representation of the phase accumulation due to magnetic field gradients. It is defined as

$$\vec{k}(t) = [k_X(t), k_Y(t), k_Z(t)]^T = \frac{\gamma}{2\pi} \int_0^t \vec{G}(\tau) \ d\tau$$

The effect of gradients on the transverse magnetization then becomes

$$M_{XY}(\vec{r}, t) = m(\vec{r}) e^{ -i 2 \pi \vec{k}(t) \cdot \vec{r} }$$

The specific motivation for k-space will become apparent soon, when we formulate the signal equation and see that there is a Fourier Transform.

16.2.2 MRI signal in the Fourier domain

The MRI signal comes from the precession of the transverse magnetization, and thus is proportional to the transverse magnetization. The MRI signal also comes from any precessing magnetization within the sensitive volume of the RF receive coils. In other words, it is also not localized to a single location, but rather is summed over a volume:

$$\begin{aligned} s(t) & = \int M_{XY}(\vec{r},t) \ d\vec{r} \ & = \int m(\vec{r}) \exp(-i2\pi \vec{k}(t) \cdot \vec{r}) \ d\vec{r} \end{aligned}$$

The amazing result here is that this describes our MRI signal in the form of a Fourier Transform:

$$\mathcal{F} { f(\vec{r}) } = F(\vec{k}) = \int_{-\infty}^\infty f(\vec{r}) \exp(-i 2 \pi \vec{k} \cdot \vec{r}) \ d\vec{r}$$

This is the power of the k-space representation, that it describes how MRI is sampling data in the Fourier Transform domain, or the spatial frequency domain, of the object net magnetization. In other words, MRI signals are a measure of the spatial frequencies of our object.

This result means that, to reconstruct an image we need to put our MRI signals into their k-space location based on the applied gradients, and then use an inverse Fourier Transform. Our signal is the Fourier Transform of the initial transverse magnetization, evaluated at a spatial frequnecy, $\vec{k}$, that is determined by the k-space trajectory, $\vec{k}(t)$:

$$ s(t) = \mathcal{F} { M_{XY}(\vec{r},0) } |_{\vec{k} = \vec{k}(t)} $$

Defining

$$\mathcal{F}{ m(\vec{r}) } = M(\vec{k})$$

then we get

$$ s(t) = M(\vec{k}(t)) $$

This key result says that the received signal is proportional to the Fourier Transform of the transverse magnetization, evaluated at the k-space location determined by the k-space trajectory.

16.3 From MRI data to images

The flow of the experiment and data is as follows:

1. RF excitation to create transverse magnetization, $M_{XY}(\vec{r},0)$
2. Gradients applied as $\vec{G}(t)$ and data is acquired
3. k-space locations, $\vec{k}(t)$, are determined based on the applied gradients
4. MR signal acquired represents the Fourier Transform of the transverse magnetization at the k-space location: $ s(t) = M(\vec{k}(t))$
5. MR signal over time is stored in a data matrix with known k-space locations to create $M(\vec{k})$
6. Inverse Fourier Transform is applied to reconstruct an image of the transverse magnetization $\mathcal{F}^{-1}{ M(\vec{k})} = m(\vec{r})$

Key Result: *We have now have the incredible result that we used magnetic field gradients, the k-space framework, and appropriate acquisition and ordering of the MR signal to create an* **IMAGE**!!

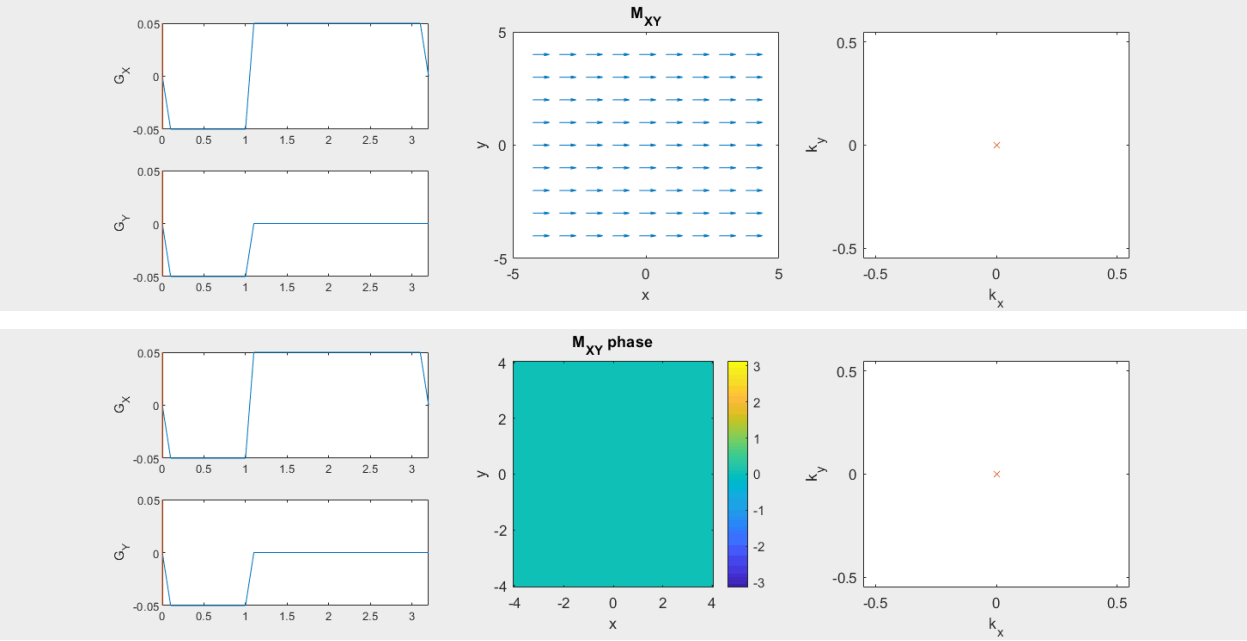
16.4 K-space Trajectories

K-space is a very general method for capturing the effect of spatial encoding gradients, and the "k-space trajectory" is defined as the pattern created over time by the gradients:

$$\vec{k}(t) = \frac{\gamma}{2\pi} \int_0^t \vec{G}(\tau) d\tau$$

Note that k-space trajectories always start at the center of k-space, $\vec{k}(0) = 0$ and are only defined once there is transverse magnetization (e.g. after an RF pulse).

The following simulation of the net magnetizations shows how rotations and k-space trajectory during a typical Cartesian (or 2D FT) gradient pulse sequence, which is differs from the simulation above in that an initial dephasing gradient in the frequency encoding direction is applied to sample both positive and negative spatial frequencies in k-space:



(See `spatial_encoding_Mxy_illustration.m` for code generating this movie)

16.4.1 Mapping Gradients to K-space trajectory

To determine the k-space trajectory, the integral or cumulative sum of the gradients after excitation is computed based on the formula above. Once the trajectory, $\vec{k}(t)$, is known, this can be mapped onto k-space to show the k-space coverage of the chosen gradients.

To determine the gradients for a given k-space trajectory, the derivative of the k-space trajectory can be used.

```
import numpy as np
import matplotlib.pyplot as plt

# Cartesian
N_PE = 8
kPE = np.linspace(-N_PE/2, N_PE/2, N_PE+1) / N_PE
dt = 0.1
Tpe = 1.0

kFE = 1/2
```

(continues on next page)

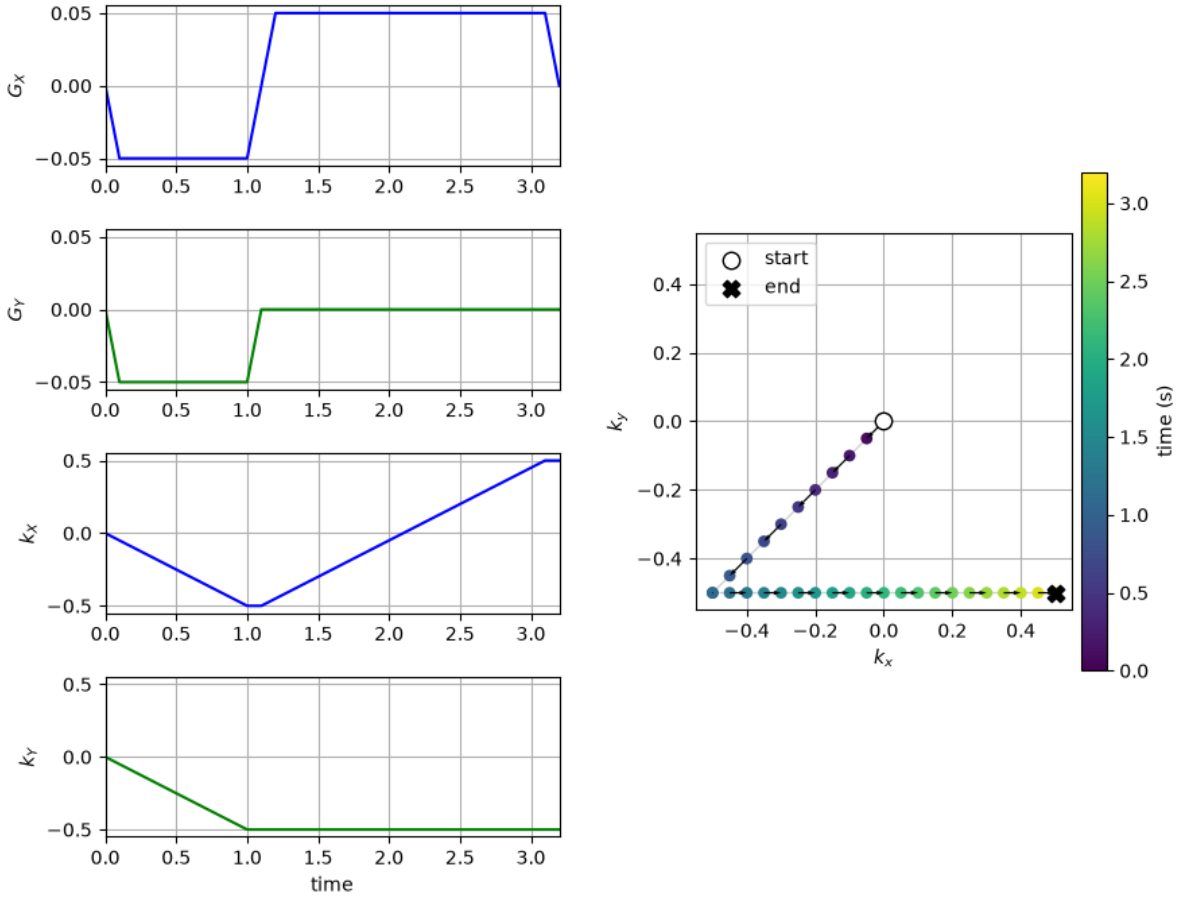
(continued from previous page)

```
Tfe = 2.0

GAMMA = 42.58

Tall = np.arange(0, Tpe + Tfe + 2*dt + dt/2, dt)

imall = []
map = []

n_tpe = int(round(Tpe / dt))
n_tfe = int(round(Tfe / dt))

# Change this to view trajectory for different phase-encode lines
i_pe = 0
GY = np.zeros_like(Tall)
GX = np.zeros_like(Tall)

GY[1 : n_tpe + 1] = kPE[i_pe] / (Tpe / dt)
GX[1 : n_tpe + 1] = -kFE / (Tpe / dt)

GX[n_tpe + 2 : len(Tall) - 1] = 2 * kFE / (Tfe / dt)

kX = np.cumsum(GX)
kY = np.cumsum(GY)

import matplotlib.gridspec as gridspec

# create a 4x2 grid where the right column spans all 4 rows
fig = plt.figure(figsize=(10, 8))
gs = gridspec.GridSpec(4, 2, width_ratios=[1, 1], wspace=0.3, hspace=0.4)

# left column: 4 rows for Gx, Gy, kx, ky
ax_gx = fig.add_subplot(gs[0, 0])
ax_gy = fig.add_subplot(gs[1, 0])
ax_kx = fig.add_subplot(gs[2, 0])
ax_ky = fig.add_subplot(gs[3, 0])

# right column: single plot spanning all rows for kx-ky trajectory
ax_kxy = fig.add_subplot(gs[:, 1])

# Plot signals vs time
ax_gx.plot(Tall, GX, color='blue')
ax_gx.set_ylabel(r'$G_X$')
ax_gx.set_xlim([0, Tall[-1]])
ax_gx.set_ylim([-0.055, 0.055])
ax_gx.grid(True)

ax_gy.plot(Tall, GY, color='green')
ax_gy.set_ylabel(r'$G_Y$')
ax_gy.set_xlim([0, Tall[-1]])
ax_gy.set_ylim([-0.055, 0.055])
ax_gy.grid(True)

ax_kx.plot(Tall, kX, color='blue')
ax_kx.set_ylabel(r'$k_X$')
ax_kx.set_xlim([0, Tall[-1]])
ax_kx.set_ylim([-0.55, 0.55])
```

(continues on next page)

(continued from previous page)

```
ax_kx.grid(True)

ax_ky.plot(Tall, kY, color='green')
ax_ky.set_ylabel(r'$k_Y$')
ax_ky.set_xlabel('time')
ax_ky.set_xlim([0, Tall[-1]])
ax_ky.set_ylim([-0.55, 0.55])
ax_ky.grid(True)

# Plot k-space trajectory on the right
# faint base line for context
ax_kxy.plot(kX, kY, color='lightgray', linewidth=1)

# color points by time to show evolution
sc = ax_kxy.scatter(kX, kY, c=Tall, cmap='viridis', s=40, edgecolors='none', zorder=3)

# add directional arrows sampled along the trajectory
step = max(1, len(kX)//12)
idx = np.arange(0, len(kX)-1, step)
dx = kX[idx+1] - kX[idx]
dy = kY[idx+1] - kY[idx]
ax_kxy.quiver(kX[idx], kY[idx], dx, dy, angles='xy', scale_units='xy', scale=1,␣
 ↪width=0.004, color='k', zorder=4)

# mark start and end points
ax_kxy.scatter(kX[0], kY[0], marker='o', facecolor='white', edgecolor='black', s=80,␣
 ↪zorder=5, label='start')
ax_kxy.scatter(kX[-1], kY[-1], marker='X', color='black', s=80, zorder=5, label='end')
ax_kxy.legend(loc='upper left')
ax_kxy.set_xlabel(r'$k_x$')
ax_kxy.set_ylabel(r'$k_y$')
ax_kxy.set_xlim([-.55, .55])
ax_kxy.set_ylim([-.55, .55])
ax_kxy.grid(True)
ax_kxy.set_aspect('equal', adjustable='box')
fig.colorbar(sc, ax=ax_kxy, label='time (s)', shrink=0.6, pad=0.02)

plt.show()
```



16.4.2 Spin-echo Effect on K-space

The refocusing 180-degree pulses use in creating a spin-echo invert the phase of the transverse magnetization. This is essential for forming a spin-echo. Since k-space is encoded in the phase of the transverse magnetization, this also inverts the location in k-space.

If the k-space location prior to the 180° pulse is $\vec{k}^-(t_{180})$, then after the pulse it becomes

$$\vec{k}^+(t_{180}) = -\vec{k}^-(t_{180})$$

16.4.3 Example K-space Trajectories

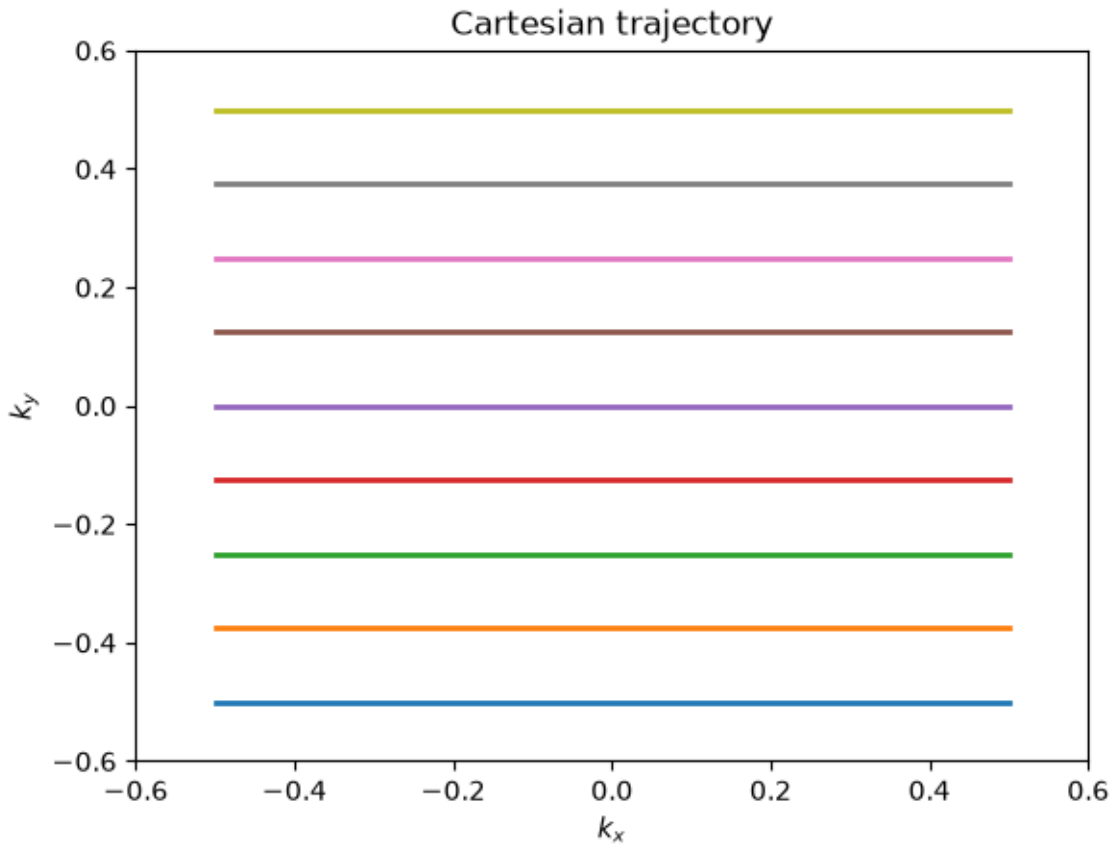
The majority of MRI uses Cartesian trajectories, in which parallel lines of k-space are covered to sample a 2D (or 3D) grid. This is the trajectory for frequency encoding and phase encoding sampling for FT imaging. K-space trajectories with other patterns, such as radial lines, spirals, rastered lines (echo-planar trajectories), or blades can also be used. The main requirement is that a sufficient region in k-space is measured. The specific requirements for defining this region depend on the desired image resolution and field-of-view, and are also affected by any accelerated imaging method being used.

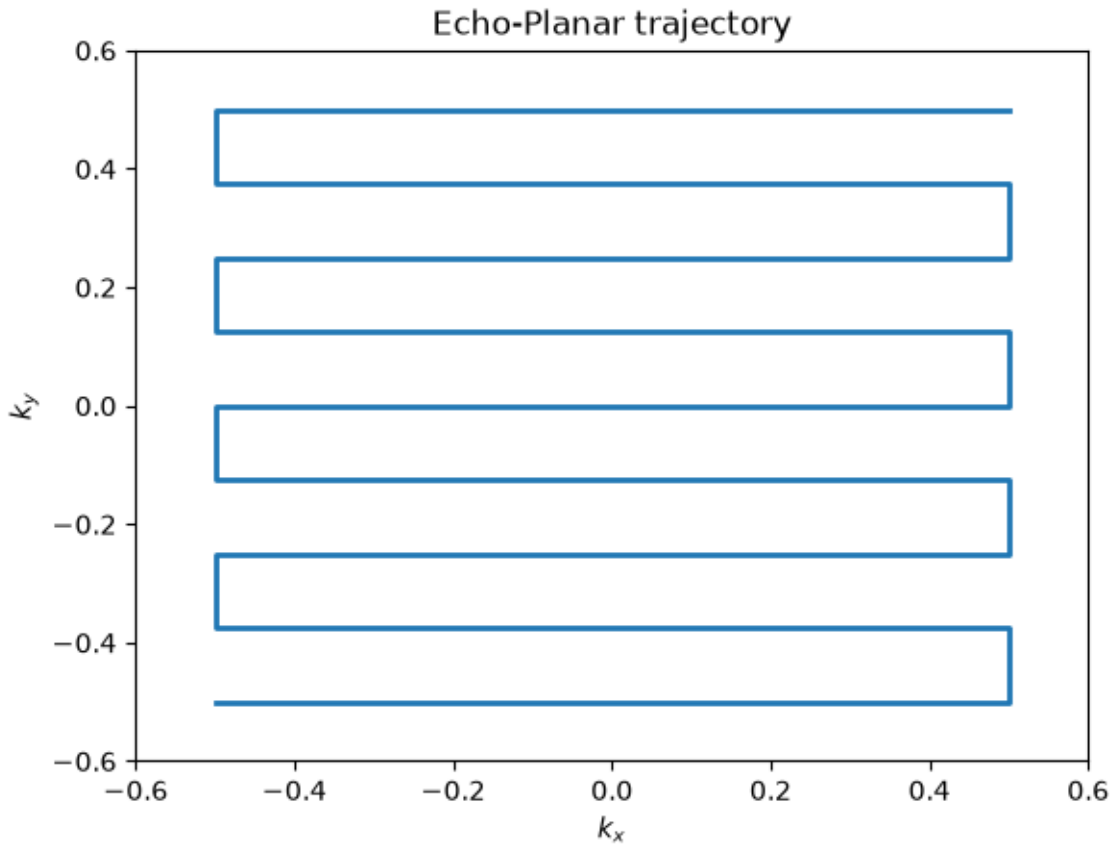
```
import numpy as np
import matplotlib.pyplot as plt

# Cartesian
```

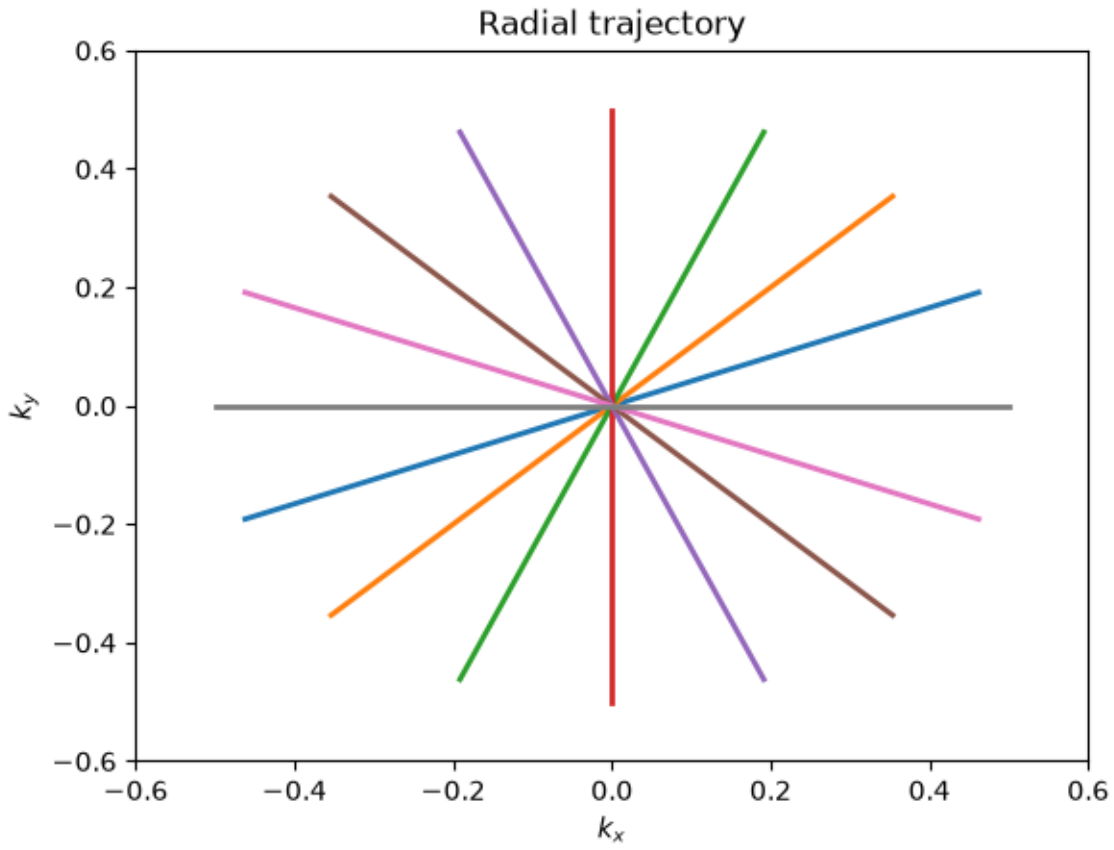
(continues on next page)

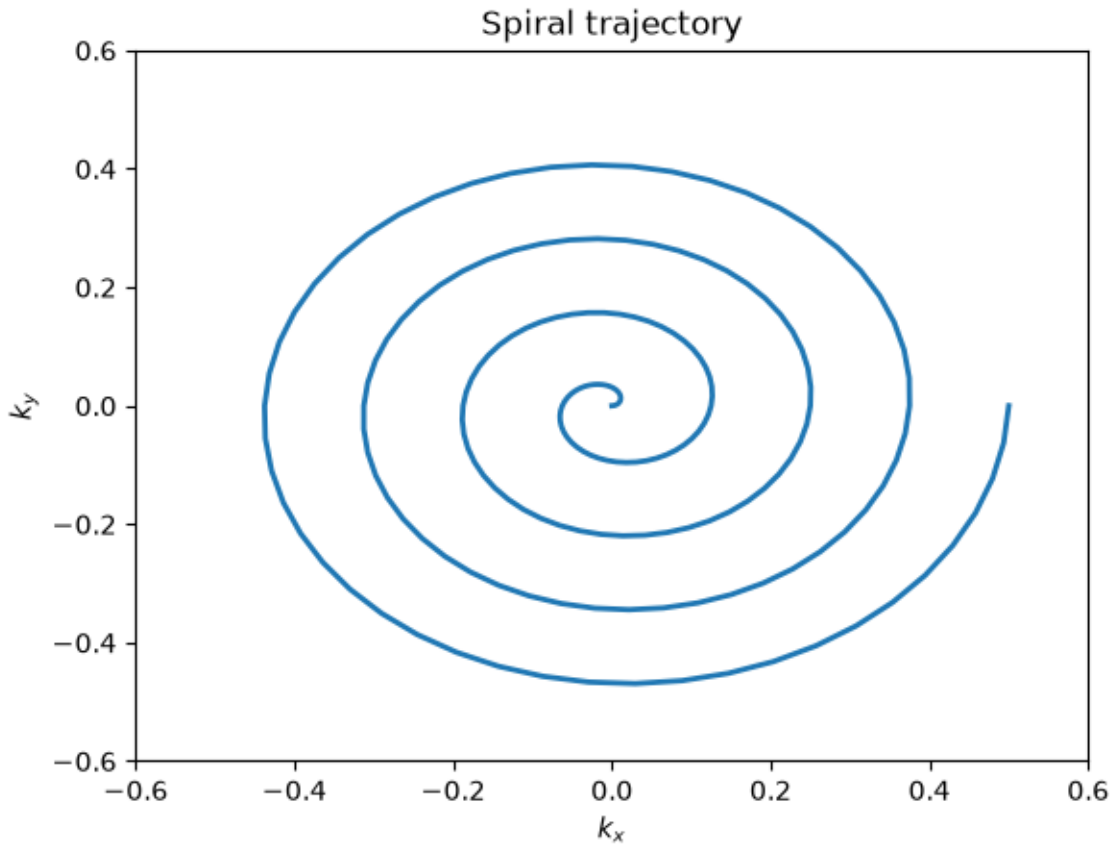
(continued from previous page)

```
N = 8
k = np.linspace(-N/2, N/2, N+1) / N
ky, kx = np.meshgrid(k, k)

plt.figure()
plt.plot(kx, ky, linewidth=2)
plt.xlim([-.6, .6])
plt.ylim([-.6, .6])
plt.xlabel('$k_x$')
plt.ylabel('$k_y$')
plt.title('Cartesian trajectory')

# Echo-Planar
kx_ep = ky.copy()
kx_ep[1::2,:] = kx_ep[1::2,::-1] # Reverse every other row
ky_ep = kx.copy()
kx_ep = kx_ep.flatten()
ky_ep = ky_ep.flatten()

plt.figure()
plt.plot(kx_ep, ky_ep, linewidth=2)
plt.xlim([-.6, .6])
plt.ylim([-.6, .6])
plt.xlabel('$k_x$')
plt.ylabel('$k_y$')
plt.title('Echo-Planar trajectory')

# Radial
theta = 2 * np.pi * np.arange(1, N+1) / (2*N)
k_theta = np.exp(1j * theta)
k_radial = np.outer(k, k_theta)

plt.figure()
for i in range(N):
    plt.plot(np.real(k_radial[:, i]), np.imag(k_radial[:, i]), linewidth=2)
plt.xlim([-.6, .6])
plt.ylim([-.6, .6])
plt.xlabel('$k_x$')
plt.ylabel('$k_y$')
plt.title('Radial trajectory')

# Spiral
n = np.linspace(0, 1, 201)
Nturns = N / 2
k_spiral = 0.5 * n * np.exp(1j * 2 * np.pi * Nturns * n)

plt.figure()
plt.plot(np.real(k_spiral), np.imag(k_spiral), linewidth=2)
plt.xlim([-.6, .6])
plt.ylim([-.6, .6])
plt.xlabel('$k_x$')
plt.ylabel('$k_y$')
plt.title('Spiral trajectory')

plt.show()
```


Cartesian trajectory
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
k_y
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
k_x


Echo-Planar trajectory
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
k_y
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
k_x


Radial trajectory
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
k_y
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
k_x

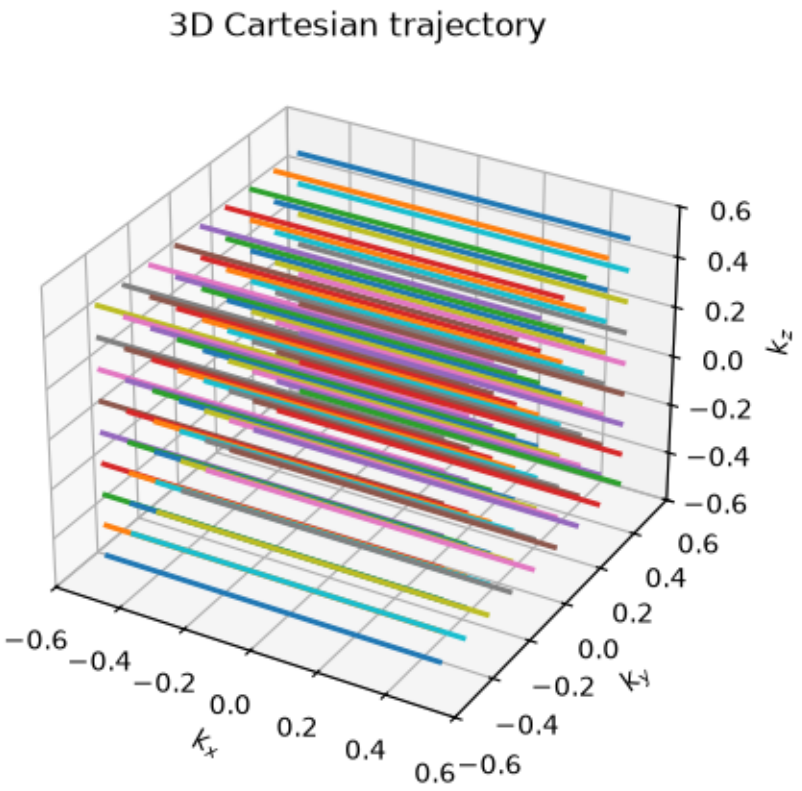


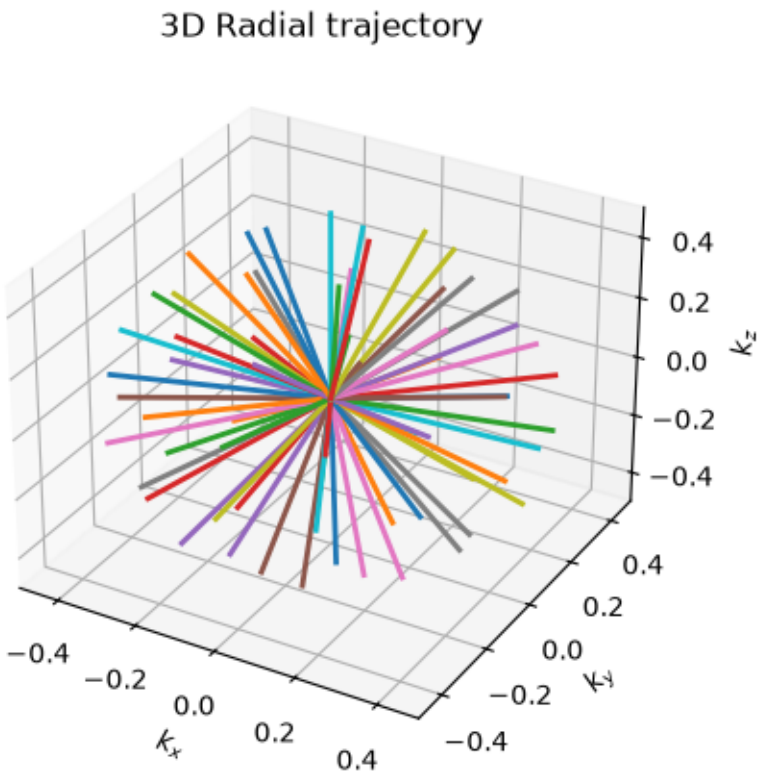

```
import numpy as np
import matplotlib.pyplot as plt

N=8

# Cartesian
k = np.linspace(-N/2, N/2, N+1) / N
kx, ky, kz = np.meshgrid(k, k, k)

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
# ax.scatter(kx, ky, kz)
# Plot all points in the 3D Cartesian k-space grid
for i in range(kx.shape[0]):
    for j in range(kx.shape[1]):
        ax.plot(kz[i,j,:], ky[i,j,:], kx[i,j,:], linewidth=2)
ax.set_xlim([-.6, .6])
ax.set_ylim([-.6, .6])
ax.set_zlim([-.6, .6])
ax.set_xlabel('$k_x$')
ax.set_ylabel('$k_y$')
ax.set_zlabel('$k_z$')
ax.set_title('3D Cartesian trajectory')

# Radial
N_radial = N**2
n = np.arange(1, N_radial+1)
```

(continues on next page)

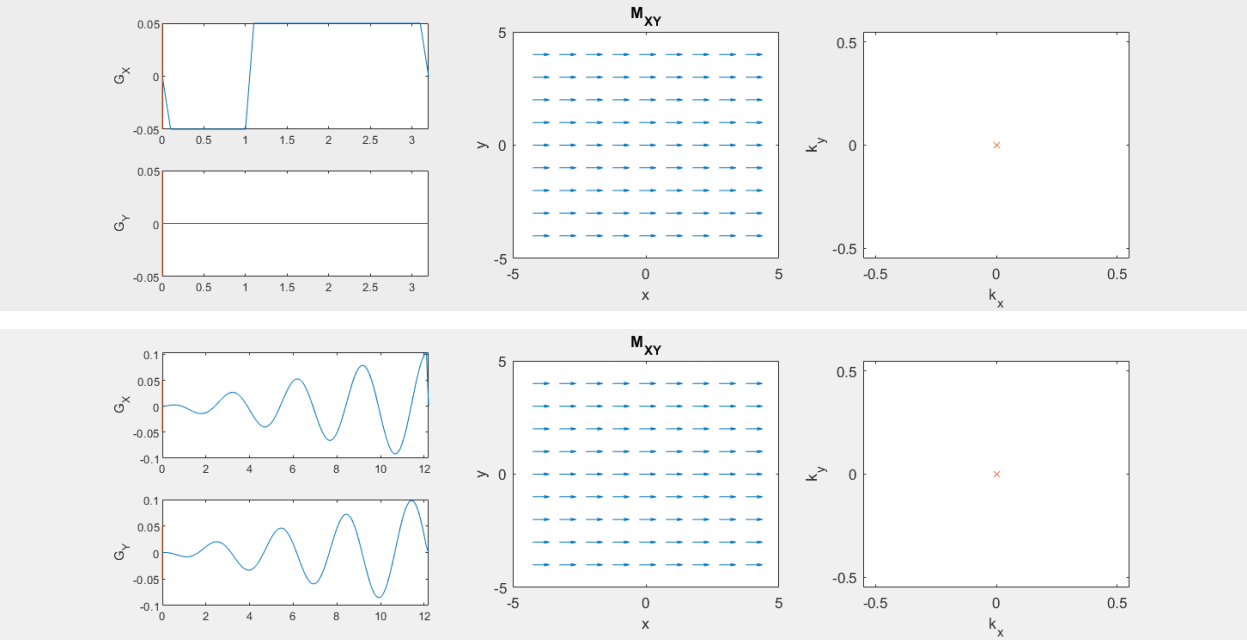
(continued from previous page)

```
kz_r = (2*n - N_radial - 1) / N_radial
phi = np.sqrt(N_radial * np.pi) * np.arcsin(kz_r)
kx_r = np.cos(phi) * np.sqrt(1 - kz_r**2)
ky_r = np.sin(phi) * np.sqrt(1 - kz_r**2)

kx_r = kx_r/2
ky_r = ky_r/2
kz_r = kz_r/2

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
for i in range(N_radial):
    # Plot radial lines from the origin to each point
    ax.plot([0, kx_r[i]], [0, ky_r[i]], [0, kz_r[i]], linewidth=2)
ax.set_xlim([-.5, .5])
ax.set_ylim([-.5, .5])
ax.set_zlim([-.5, .5])
ax.set_xlabel('$k_x$')
ax.set_ylabel('$k_y$')
ax.set_zlabel('$k_z$')
ax.set_title('3D Radial trajectory')

plt.show()
```





These movies illustrate the phase accumulation during non-Cartesian trajectories



(See `spatial_encoding_Mxy_illustration.m` for code generating this movie)

CHAPTER

SEVENTEEN

IMAGE RECONSTRUCTION

Image reconstruction in MRI means converting the raw signal collected in k-space into images. For a fully-sampled acquisition using Cartesian sampling (e.g. on a grid), this can be done with a discrete Fourier Transform. In anticipation of advanced reconstruction methods, image reconstruction for MRI is formulated as a linear system.

17.1 Learning Goals

1. Describe how images are formed
    - Describe how MRI raw data is reconstructed into an image
2. Manipulate and analyze MRI data
    - Reconstruct an image from raw data

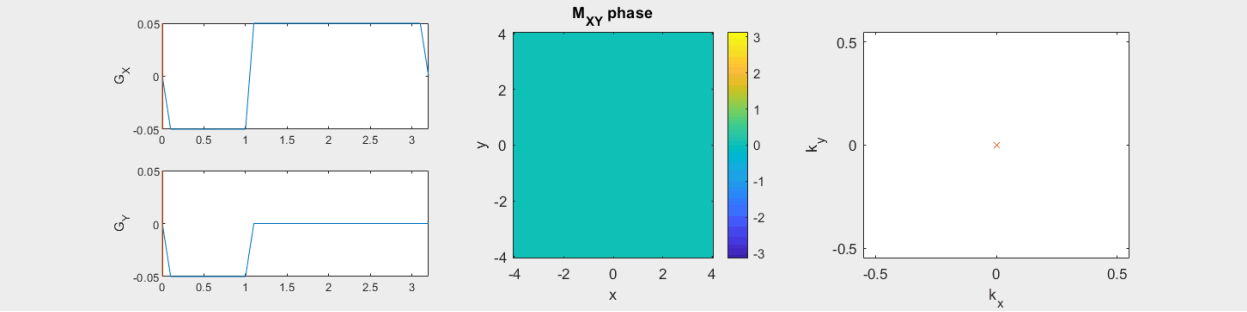
17.2 Sorting the MRI Data into k-space

Recall that the MRI signal is proportional to the Fourier Transform of the net magnetization of our object, evaluated at the k-space location defined by the gradients:

$$ s_i(t) = \mathcal{F} { M_{XY}(\vec{r},0) } |_{\vec{k} = \vec{k}_i(t)} $$

Note that this includes a dimension for the readout, t, as well as different readouts for each TR, denoted by the subscript for the i^th TR.

The data acquired during the readout will be sorted into data based on the k-space trajectory. The following example shows the Cartesian k-space trajectory, which acquired data in a raster fashion in k-space:



From this knowledge, the MR signal over time is stored in a data matrix with known k-space locations to create $M(\vec{k})$.

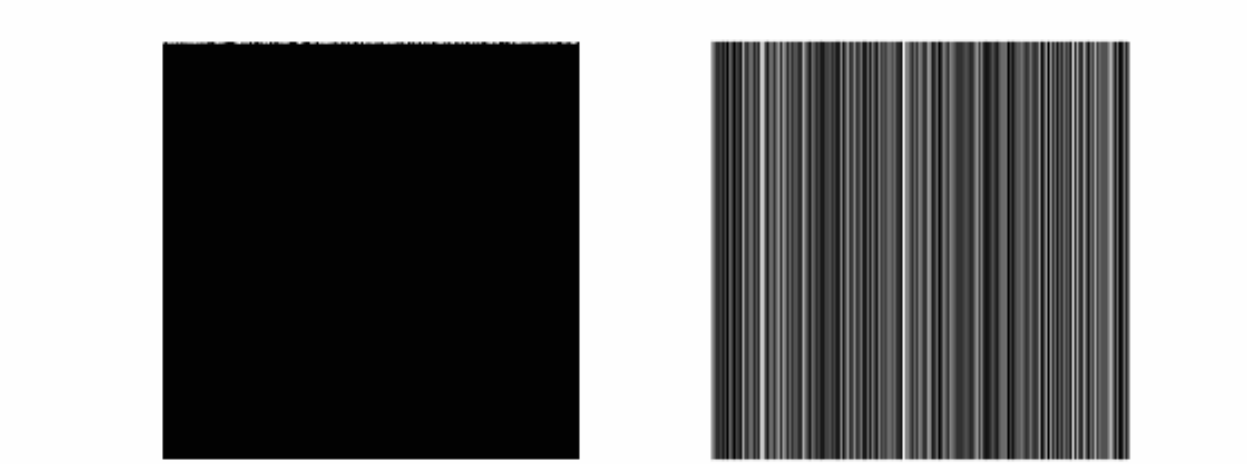
17.3 Fourier Transform Reconstruction

Once the data has been sorted into the corresponding lines in k-space, an inverse Fourier Transform is applied to reconstruct an image of the transverse magnetization

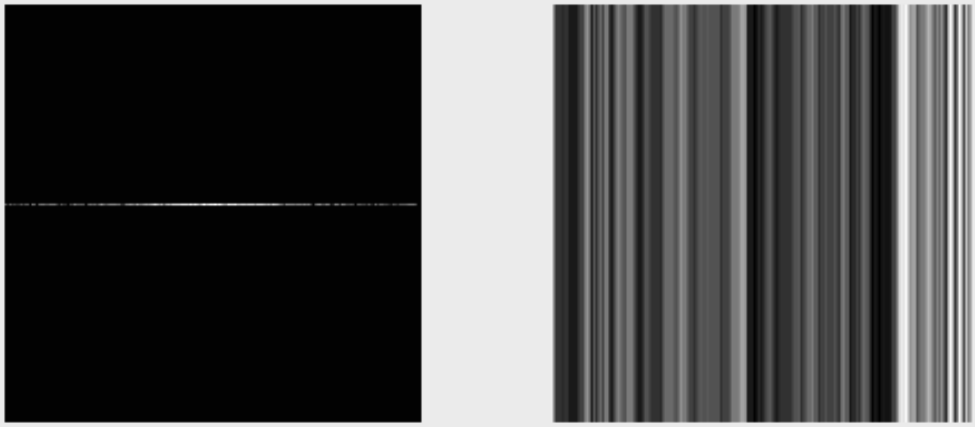
$$\mathcal{F}^{-1}{ M(\vec{k})} = m(\vec{r})$$

For typical Cartesian sampling, this can be done very simply with the 2D or 3D Fast Fourier Transform (FFT) algorithm.

The following animation illustrates typical k-space data as it would be acquired for different k-space lines (left) and the resulting image as more and more lines of k-space are accumulated



We can also acquire our k-space lines in a “center-out” or random ordering, shown below



17.4 Linear System Formulation

Alternatively, the reconstruction can be formulated as a linear system, using a discrete Fourier Transform matrix, $\mathbf{F}$. For this matrix, the entries are defined by k-space location being sampled, $\vec{k}_i$ and the spatial locations, $\vec{r}_j$:

$$ \mathbf{f}_{i,j} = \exp(-i 2 \pi \vec{k}_i \cdot \vec{r}_j)$$

In this case, the measurement is the discrete Fourier Transform of the image

$$ y = \mathbf{F}x + n $$

and image reconstruction is performed by inverse discrete Fourier Transform, which for fully sampled 2D FT imaging is well defined by matrix inversion:

$$ \hat{x} = \mathbf{F^{H}} y $$

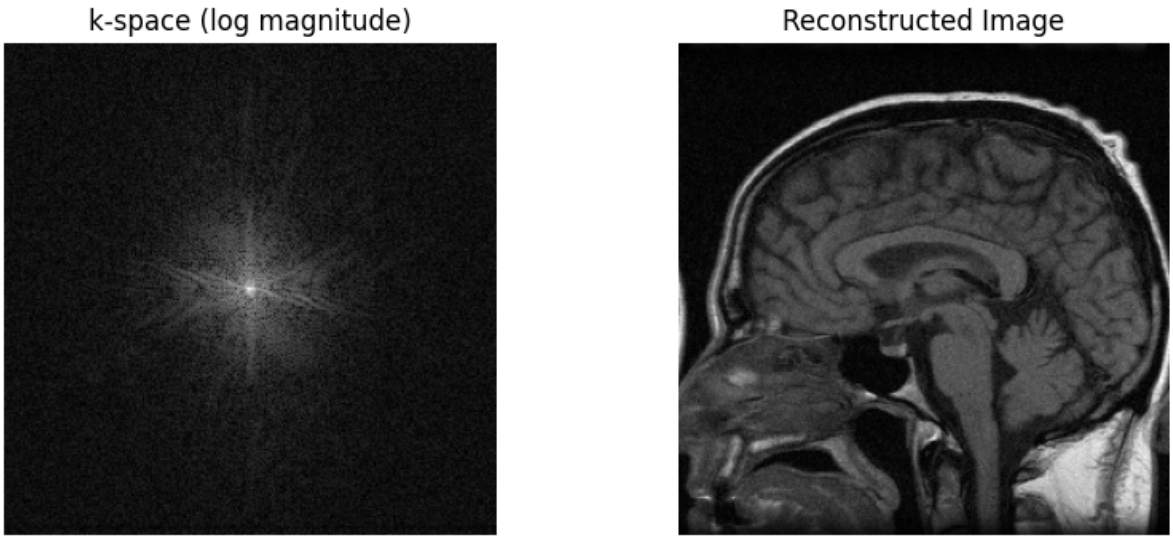
```
# Sample Image Reconstruction

import numpy as np
from scipy.io import loadmat
import matplotlib.pyplot as plt

# Load k-space data from .mat file
mat = loadmat('../Data/se_t1_sag_data.mat')
kspace = mat['data']

# Reconstruct the image by inverse FFT
recon_image = np.fft.fftshift(np.fft.ifft2(np.fft.ifftshift(kspace)))

# Plot k-space data and reconstructed images
fig, axs = plt.subplots(1, 2, figsize=(10, 4))

axs[0].imshow(np.log(np.abs(kspace) + 1e1), cmap='gray')
axs[0].set_title('k-space (log magnitude)')
axs[0].axis('off')
```

(continues on next page)

(continued from previous page)

```
axs[1].imshow(np.abs(recon_image), cmap='gray')
axs[1].set_title('Reconstructed Image')
axs[1].axis('off')
```

```
(-0.5, 255.5, 255.5, -0.5)
```



Part VII

Image Characteristics

CHAPTER

EIGHTEEN

FIELD OF VIEW (FOV) AND RESOLUTION

The image we create is fundamentally defined by the field-of-view (FOV) and spatial resolution captured. These are determined by our k-space sampling pattern.

18.1 Learning Goals

1. Describe how images are formed
    - Describe what determines the image FOV and resolution
2. Manipulate MRI sequence parameters to improve performance
    - Manipulate the gradients and k-space sampling to change the FOV and resolution

18.2 Field-of-View (FOV)

The FOV is determined by the sample spacing in k-space, with the simple relationship based on the sample spacing, Δk as:

$$ FOV = \frac{1}{\Delta k}$$

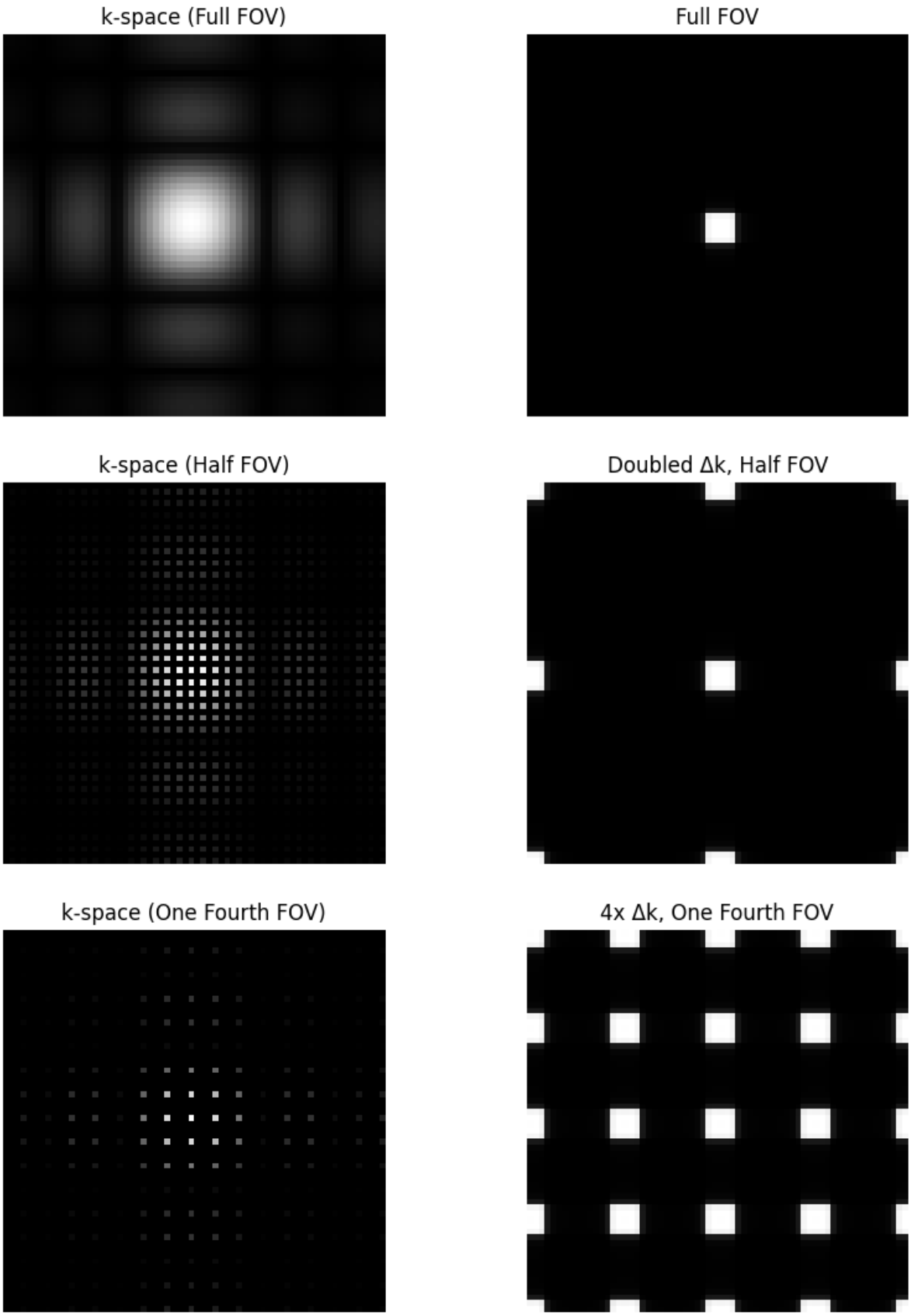
For example, $FOV_{x} = \frac{1}{\Delta k_x}$

```
import numpy as np

import matplotlib.pyplot as plt

def sinc(x):
    return np.sinc(x / np.pi)

def ifft2c(x):
    # Centered inverse 2D FFT
    return np.fft.ifftshift(np.fft.ifft2(np.fft.fftshift(x)))

# rectangular object to demonstrate FOV

N = 64
kx = np.arange(-N/2+1, N/2+1) / N
N_rect = N // 4
kdata = np.outer(sinc(kx * N_rect), sinc(kx * N_rect))

rect_recon = ifft2c(kdata)
```

(continues on next page)

(continued from previous page)

```
plt.figure(figsize=(10,4))
plt.subplot(1,2,1)
plt.imshow(np.log(np.abs(kdata)+1e1), cmap='gray', aspect='equal')
plt.axis('off')
plt.title('k-space (Full FOV)')
plt.subplot(1,2,2)
plt.imshow(np.abs(rect_recon), cmap='gray', aspect='equal')
plt.axis('off')
plt.title('Full FOV')
plt.show()

kdata2 = kdata.copy()
kdata2[::2, :] = 0
kdata2[:, ::2] = 0
rect_recon2 = ifft2c(kdata2)

plt.figure(figsize=(10,4))
plt.subplot(1,2,1)
plt.imshow(np.log(np.abs(kdata2) + 1e1), cmap='gray', aspect='equal')
plt.axis('off')
plt.title('k-space (Half FOV)')
plt.subplot(1,2,2)
plt.imshow(np.abs(rect_recon2), cmap='gray', aspect='equal')
plt.axis('off')
plt.title('Doubled Δk, Half FOV')
plt.show()

kdata3 = kdata.copy()
kdata3[::4, :] = 0
kdata3[1::4, :] = 0
kdata3[2::4, :] = 0
kdata3[:, ::4] = 0
kdata3[:, 1::4] = 0
kdata3[:, 2::4] = 0
rect_recon3 = ifft2c(kdata3)

plt.figure(figsize=(10,4))
plt.subplot(1,2,1)
plt.imshow(np.log(np.abs(kdata3) + 1e1), cmap='gray', aspect='equal')
plt.axis('off')
plt.title('k-space (One Fourth FOV)')
plt.subplot(1,2,2)
plt.imshow(np.abs(rect_recon3), cmap='gray', aspect='equal')
plt.axis('off')
plt.title('4x Δk, One Fourth FOV')
plt.show()
```


k-space (Full FOV)
Full FOV
k-space (Half FOV)
Doubled Δk, Half FOV
k-space (One Fourth FOV)
4x Δk, One Fourth FOV

18.3 Spatial Resolution

The spatial resolution is determined by the maximum sample extent in k-space, with the simple relationship based on the maximum k-space sample locations, k_{max} as:

$$ \delta = \frac{1}{2 k_{max}}$$

For example, $\delta_x = \frac{1}{2 k_{x,max}}$.

For symmetric sampling in k-space, this can also be defined based on the width of the k-space sampling, $W_k = 2 k_{max}$:

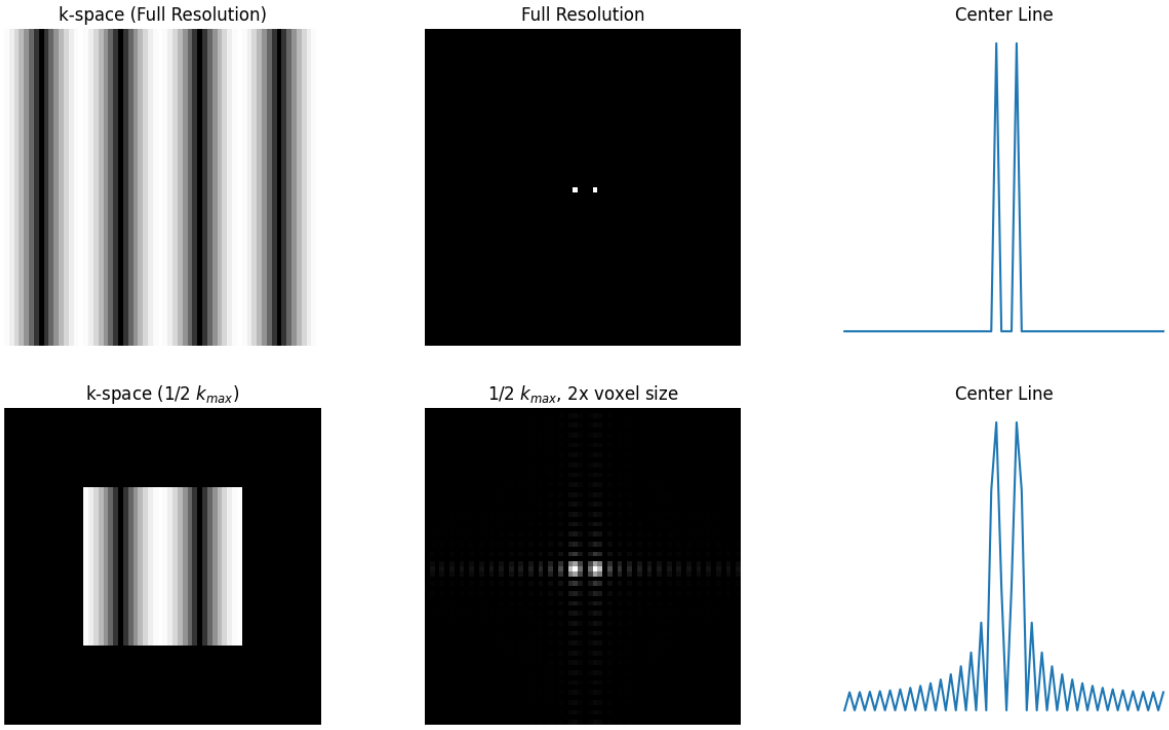
$$ \delta = \frac{1}{W_k}$$

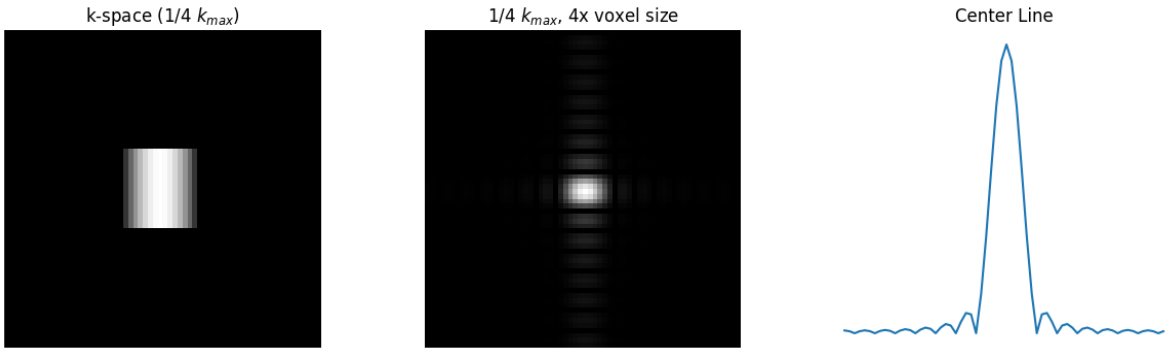
```
# rectangular object to demonstrate resolution

def plot_center_line(recon, fig_num, title):
    center = recon.shape[0] // 2
    plt.subplot(1, 3, 3)
    plt.plot(np.abs(recon[center, :]))
    plt.title(title)
    plt.axis('off')


# Create k-space for two point objects separated by 1 pixel
kdata = np.outer(np.ones(N), np.exp(1j * np.pi * kx *4) + np.exp(-1j * np.pi * kx * 4␣
 ↪))
rect_recon = ifft2c(kdata)

# Full resolution (already defined as kdata and rect_recon)
plt.figure(figsize=(15, 4))
plt.subplot(1, 3, 1)
plt.imshow(np.log(np.abs(kdata) + 1e1), cmap='gray', aspect='equal')
plt.axis('off')
plt.title('k-space (Full Resolution)')
plt.subplot(1, 3, 2)
plt.imshow(np.abs(rect_recon), cmap='gray', aspect='equal')
plt.axis('off')
plt.title('Full Resolution')
plot_center_line(rect_recon, 1, 'Center Line')
plt.show()

# 1/2 k_max, 2x voxel size
kdata2 = kdata.copy()
kdata2[:int(1*N/4), :] = 0
kdata2[int(3*N/4):, :] = 0
kdata2[:, :int(1*N/4)] = 0
kdata2[:, int(3*N/4):] = 0
rect_recon2 = ifft2c(kdata2)

# 1/2 k_max, 2x voxel size
plt.figure(figsize=(15, 4))
plt.subplot(1, 3, 1)
plt.imshow(np.log(np.abs(kdata2) + 1e1), cmap='gray', aspect='equal')
plt.axis('off')
plt.title('k-space (1/2 $k_{max}$)')
plt.subplot(1, 3, 2)
plt.imshow(np.abs(rect_recon2), cmap='gray', aspect='equal')
```

(continues on next page)

(continued from previous page)

```
plt.axis('off')
plt.title('1/2 $k_{max}$, 2x voxel size')
plot_center_line(rect_recon2, 2, 'Center Line')
plt.show()

# 1/4 k_max, 4x voxel size
kdata4 = kdata.copy()
N = kdata.shape[0]
kdata4[:int(3*N/8), :] = 0
kdata4[int(5*N/8):, :] = 0
kdata4[:, :int(3*N/8)] = 0
kdata4[:, int(5*N/8):] = 0
rect_recon4 = ifft2c(kdata4)


# 1/4 k_max, 4x voxel size
plt.figure(figsize=(15, 4))
plt.subplot(1, 3, 1)
plt.imshow(np.log(np.abs(kdata4) + 1e1), cmap='gray', aspect='equal')
plt.axis('off')
plt.title('k-space (1/4 $k_{max}$)')
plt.subplot(1, 3, 2)
plt.imshow(np.abs(rect_recon4), cmap='gray', aspect='equal')
plt.axis('off')
plt.title('1/4 $k_{max}$, 4x voxel size')
plot_center_line(rect_recon4, 3, 'Center Line')
plt.show()
```




k-space (1/4 k_{max})
1/4 k_{max}, 4x voxel size
Center Line

CHAPTER

NINETEEN

SCAN TIME AND SIGNAL TO NOISE RATIO (SNR)

The scan time and signal to noise ratio (SNR) are critical characteristics that impacts image quality and what is practical in MRI. For MRI, there is a strong theoretical foundation for the SNR dependency based on the MRI physics and pulse sequence characteristics.

19.1 Learning Goals

1. Understand the constraints and tradeoffs in MRI
    - Determine how long a scan takes
    - Describe the factors that influence image SNR
2. Manipulate MRI sequence parameters to improve performance
    - Understand how modifications in sequence parameters will affect the total scan time and expected SNR

19.2 Scan Time

The total scan time, T_{scan}, depends on the length of each component in the pulse sequence as well as how many times these components are repeated.

For sequences with a single component of length TR, this can be expressed as

$$T_{scan} = TR \ N_{TR}$$

where N_{TR} is the total number of TRs which can depend on the number of phase encodes, preparation or dummy cycles to get into the steady-state, acceleration factor, and number of averages.

19.3 SNR Dependencies

The SNR in factors we control in an MRI experiment can be decomposed into dependencies on three factors:

1. Voxel volume
2. Data acquisition time
3. Transverse magnetization

There are other system factors that influence SNR such as the magnetic field strength and the RF coils.

19.3.1 Voxel Volume

The MRI signal is an integral over a volume of the transverse magnetization. Therefore there is a linear dependency on the MRI signal and the volume of magnetization being examined. So we have

$$SNR \propto \mathrm{Voxel\ Volume} = \delta_x\ \delta_y\ \delta_z$$

where $\delta_x, \delta_y, \delta_z$ are the voxel dimensions in x, y, and z.

19.3.2 Data Acquisition Time

Every additional measurement allows for signal averaging since the noise varies at each measurement. Note that MRI noise is zero-mean with a Gaussian distribution. Additional data acquisition time increases signal linearly, while the total noise increases as the square root. So

$$SNR \propto \frac{S}{\sigma} = \frac{T_{meas}}{\sqrt{T_{meas}}} = \sqrt{T_{meas}}$$

where T_{meas} is the *total data acquisition/measurement time* including all TRs and signal averaging, but not including the sequence downtime. This can be defined in terms of the readout duration per TR, T_{read}, and the total number of TRs, N_{TR}, which for example includes number of ,phase encoding steps and signal averages, as:

$$T_{meas} = N_{TR} T_{read}$$

19.3.3 Transverse Magnetization

The MRI signal is also proportional to the transverse magnetization, including modulation that occurs due to pulse sequence parameters such as TE, TR, flip angle, spin-echo versus gradient-echo, etc. This is then also varying depending on the MR properties such as proton density, relaxation rates, diffusion coefficients, etc. Since this varies widely by pulse sequence, we summarize this as

$$SNR \propto f_{seq}$$

For example, in a simple spoiled gradient-echo pulse sequence,

$$f_{seq,GRE} = M_0 \sin(\theta) \frac{1- \exp(-TR/T_1)}{1- \cos(\theta) \exp(-TR/T_1)}$$

19.4 SNR Equations

Putting it all together,

$$SNR \propto f_{seq}\ \mathrm{Voxel\ Volume}\ \sqrt{T_{meas}}$$

This simple equation describes the majority of SNR effects in MRI, and is extremely powerful when designing and improving MRI scans!

19.4.1 Alternative formulations of SNR

MRI scanner interfaces often do not provide direct control of the resolutions or measurement times, but instead parameters such as matrix sizes, field-of-views (FOVs) and readout bandwidths are used. The following relationships can generally be used to convert these parameters into the SNR equation above.

$$ FOV = \delta \ N_{matrix} $$

$$ T_{read} = \frac{N_{readout}}{BW} = \frac{N_{readout}}{2 BW_{one-sided}} $$

If acceleration methods such as partial Fourier, parallel imaging, or compressed sensing are NOT used, the following relationships also apply

$$ T_{read} = \frac{N_{FE}}{BW} = \frac{1}{BW/pixel} $$

$$ N_{TR} = N_{FE} \ N_{PE} \ N_{averages} $$

However, partial Fourier methods can potentially be used to reduce the readout time or number of encoding steps, and parallel imaging/compressed sensing can be used to reduce the number of encoding steps.

19.5 SNR Efficiency

The SNR efficiency for MRI provides a measure of overall efficiency within a MRI scan, which is the SNR normalized by the *total scan time*, T_{scan}

$$SNR_{efficiency} = \frac{SNR}{\sqrt{T_{scan}}} \propto f_{seq}\ \mathrm{Voxel\ Volume}\ \sqrt{\frac{T_{meas}}{T_{scan}}}$$

19.6 Resolution and SNR in Post-Processing

The image resolution is often changed in post-processing, and can be either upsampled to finer ("higher") spatial resolution or downsampled to coarser ("lower") spatial resolution. When using the SNR equation above, the acquired resolution should be considered.

19.6.1 Upsampling

Upsampling is typically done with some type of interpolation. In MRI, this can be easily implemented in k-space through zero-padding, which is equivalent to sinc interpolation. Upsampling with a reasonable method does not affect SNR.

19.6.2 Downsampling

Downsampling is done to increase SNR by changing the image to a coarser resolution. However, downsampling is effectively averaging signals from nearby voxels which also averages out the noise. Therefore, *downsampling scales SNR as the square root of the voxel size, similarly to the measurement time.*

This is a very important distinction from changed the acquired resolution, where the SNR scales linearly with voxel size. Therefore, the SNR is higher when acquiring at the desired spatial resolution compared to acquiring at a finer resolution and then downsampling. Care should be taken to choose the acquired resolution to match the desired resolution.

CHAPTER

TWENTY

CARTESIAN SAMPLING IMAGE CHARACTERISTICS

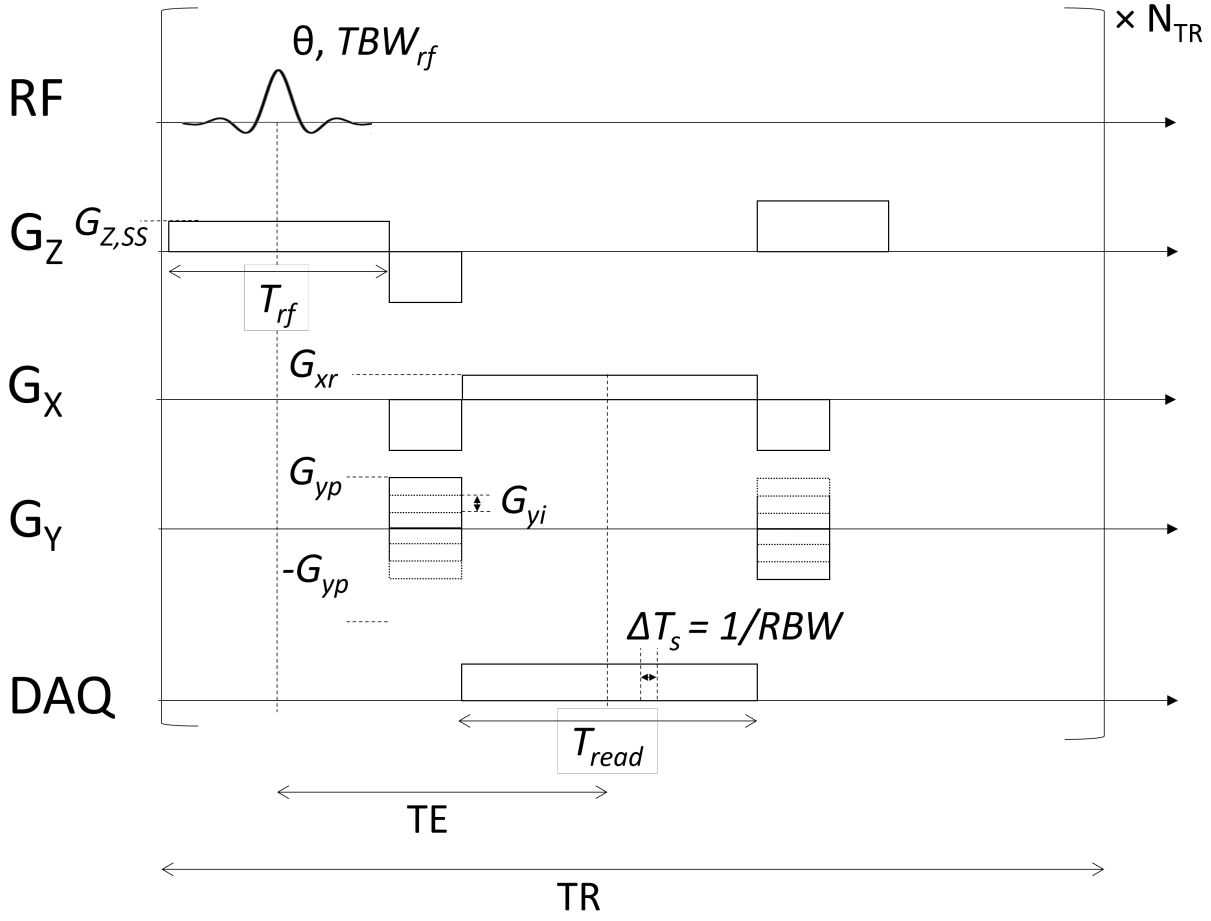
The most common type of pulse sequence uses Cartesian sampling on a k-space grid using frequency and phase encoding. Here we explicitly define the image characteristics of FOV, resolution, scan time, and SNR for this sequence.

20.1 Learning Goals

1. Describe how images are formed
    - Determine the image FOV and resolution when using Cartesian sampling
    - Determine the scan time when using Cartesian sampling
2. Manipulate MRI sequence parameters to improve performance
    - Determine how Cartesian sequence parameters will change FOV and resolution
    - Determine how changes in sequence parameters will affect the scan time and SNR

20.2 Cartesian Pulse Sequence

The following pulse sequence diagram shows a Cartesian 2D FT sequence including relevant pulse amplitudes and durations needed to determine the image characteristics. A gradient-echo sequence is shown, and the principles can simply be extended to a spin-echo sequence.



Note that idealized rectangular gradients are shown for simplicity.

20.3 Spatial Resolution

The spatial resolution is determined by the maximum sample extent in k-space. In the frequency encoding, this can be calculated from the total gradient area during the readout:

$$ \delta_x = \frac{1}{\bar \gamma G_{xr} T_{read}}$$

In the phase encoding, we can calculate the maximum k-space extent bassed on the largest phase encoding gradient area:

$$k_{y,max} = \bar \gamma G_{yp} t_{y}$$

$$ \delta_y = \frac{1}{2 k_{y,max}} = \frac{1}{ \bar \gamma 2 G_{yp} t_{y}}$$

In the slice select dimension, the spatial resolution is determined by the slice thickness:

$$ \delta_z = \Delta z = \frac{TBW_{rf} / T_{rf}}{\bar{\gamma} G_{Z,SS}}$$

20.4 Field-of-View (FOV)

The FOV is determined by the sample spacing in k-space. In the frequency encoding, this is determined by the sampling rate or the readout bandwidth, $\Delta T_s = 1/RBW$ and the gradient area during that time:

$$ FOV_x = \frac{1}{\bar \gamma G_{xr} \Delta T_s}$$

In the phase encoding, this is determined by the difference in gradient area between adjacent phase encoding steps:

$$ FOV_y = \frac{1}{\bar \gamma G_{yi} \Delta t_y}$$

All of these relationships can also be rewritten using the relationship between FOV, resolution, and number of voxels.

$$ FOV_x = N_{FE} \ \delta_x$$

$$ FOV_y = N_{PE} \ \delta_y$$

where N_{FE} is the number of samples in the frequency encoding direction and N_{PE} is the number of samples in the phase encoding direction.

20.5 Scan Time

For this sequence, the total scan time is determined by the TR and the number of TRs, N_{TR}:

$$T_{scan} = N_{TR} \ TR$$

Assuming no acceleration, the number of TRs is equal to the number of phase encoding steps, N_{PE}, times the number of averages, N_{avg}:

$$T_{scan} = N_{PE} \ N_{avg} \ TR$$

20.6 SNR

The SNR can be written for this sequence in numerous ways using the above relationships. This is left as an exercise to the reader.

Part VIII

Fast Imaging Methods

CHAPTER

TWENTYONE

VOLUMETRIC IMAGING

Volumetric imaging refers to the ability to acquire images across a volume. So far we have discussed methods for a single slice. This chapter will cover the techniques of 2D multi-slice imaging and 3D imaging.

21.1 Learning Goals

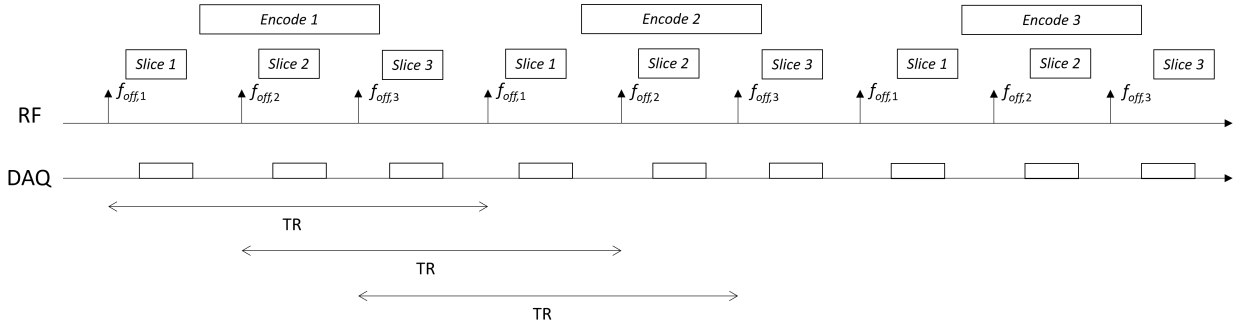
1. Describe how images are formed
    - Describe how 2D-multislice imaging works
    - Describe how 3D imaging works
2. Understand the constraints and tradeoffs in MRI
    - Determine what type of contrast benefits from 2D-multislice vs 3D imaging
    - Determine how SNR changes with 2D-multislice vs 3D imaging
    - Understand when either 2D-multislice or 3D imaging is preferred

21.2 2D Multi-slice Imaging

A single slice sequence can be easily extended to acquire multiple slices. The identical sequence can be used, and only the center frequency of the RF pulse needs to be changed to move the slice location.

21.2.1 Interleaving

2D multi-slice is typically done in an interleaved fashion, meaning that the acquisitions for different slices will be going on at the same time. This is done because there is often extra time within each TR, which can be used to acquire other slices. An example of interleaving is shown below:



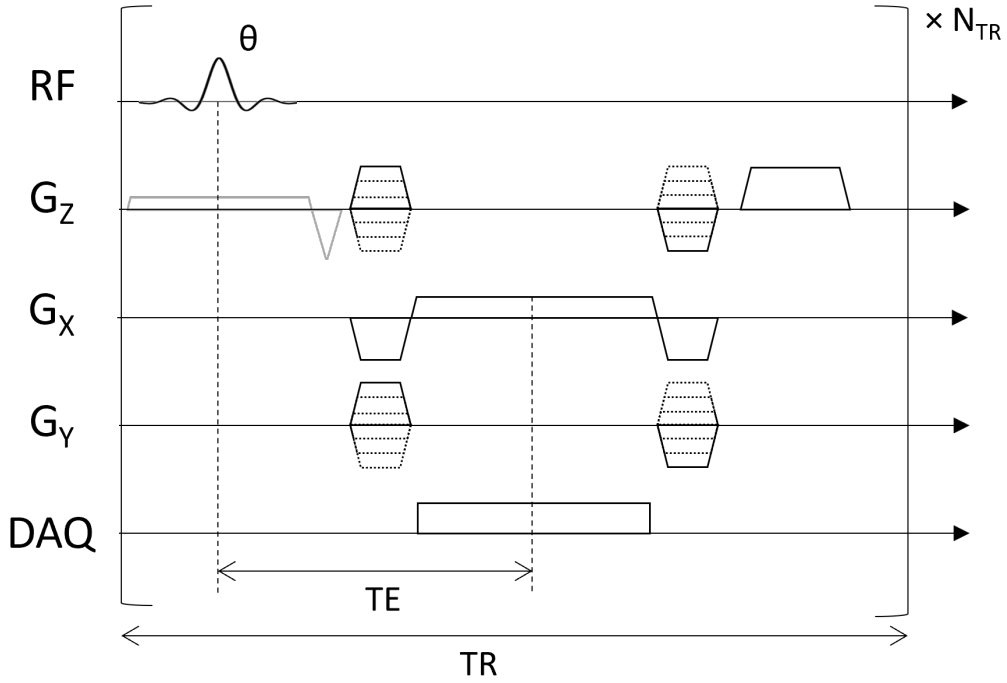
Note that the TR for each slice is defined when each individual slice is excited, meaning this time defines the T1 recovery and contrast.

21.2.2 Cross-talk

Cross-talk occurs due to the imperfect slice profiles of RF pulses. If slices are placed immediately adjacent to each other, then there is some excitation by the adjacent slice RF pulses. This can be mitigated by using higher TBW RF pulses for sharper slice profiles, or by placing gaps between slices.

21.3 3D Imaging

True 3D imaging resolves voxels in 3 dimensions with spatial encoding gradients. In Cartesian encoding, this is done by adding a phase encoding gradient in what is normally the slice dimensions. Then, a 3D area in k-space is sampled. A slice selective RF pulse can still be used, but with a bigger slice thickness, and are sometimes referred to as "slab-selective". In some cases, non-slice selective RF pulses are used, instead relying on the limited extent of the RF receive coil profiles to limit the FOV. A 3D FT pulse sequence is shown below:



Note that now the number of TRs required is the product of the number of phase encodes in two phase encoding directions.

21.4 2D Multi-slice vs 3D Imaging

This section outlines the tradeoffs between 2D multi-slice and 3D imaging.

21.4.1 Coverage

2D multi-slice imaging can have gaps in coverage if slices are not contiguous.

3D imaging provides continuous coverages without gaps, allow for more seamless evaluation of structures.

21.4.2 Contrast

2D multi-slice imaging methods with interleaving can efficiently support longer TRs for PD and T2-weighted imaging, as any extra time in the TR can be used to acquire other slices.

3D imaging methods require a larger N_{TR} to complete two dimensions of phase encoding. For this reason, they are much better suited for T1-weighted imaging which can use shorter TRs.

21.4.3 Scan Time

In 2D multi-slice imaging, the scan time tradeoff depends on the number of slices that can be interleaved in a single TR. The maximum number of slices is determined by the active time:

$$N_{slices,max} = \frac{TR}{T_{active}}$$

Provided the number of slices is limited by this maximum, the scan time for fully sampled Cartesian encoding is:

$$T_{scan, 2D} = TR \ N_{PE} \ N_{avg}$$

In 3D imaging, the scan time depends on the size of both phase encoding dimensions:

$$T_{scan, 3D} = TR \ N_{PE,1} \ N_{PE,2} \ N_{avg}$$

21.4.4 SNR

Here are relationships if fully-sampled Cartesian encoding is used:

$$SNR_{2D} \propto f_{seq,2D}\ \mathrm{Voxel\ Volume}\ \sqrt{T_{read} N_{PE} N_{avg}}$$

$$SNR_{3D} \propto f_{seq,3D}\ \mathrm{Voxel\ Volume}\ \sqrt{T_{read} N_{PE,1} N_{PE,2} N_{avg}}$$

3D imaging has a measurement time advantage over 2D multi-slice imaging, since each acquisition acquires signal from the entire volume. However, some of this advantage is mitigated by greater T1 recovery in 2D multi-slice imaging, which is captured by f_{seq}.

CHAPTER

TWENTYTWO

FAST IMAGING PULSE SEQUENCES

Modern MRI scanning relies heavily fast imaging pulse sequences, primarily echo-planar imaging (EPI) and multiple Spin-echo (RARE/FSE/TSE) methods. These allow for multiple k-space lines to be acquired within a single TR.

22.1 Learning Goals

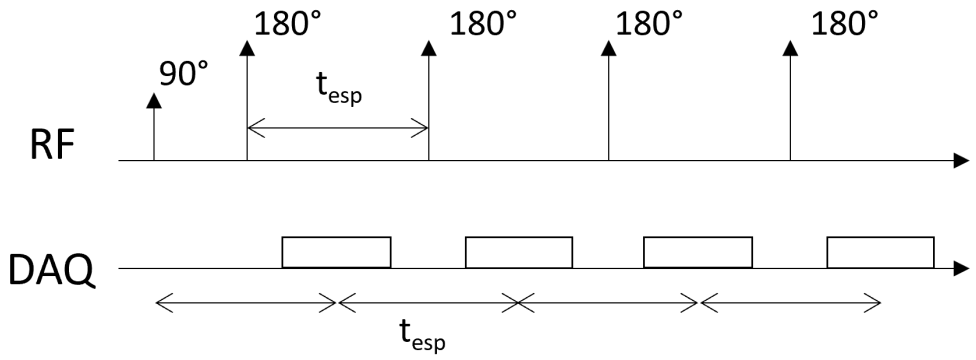
1. Describe how images are formed
    - Describe how data is acquired in FSE/TSE sequences
    - Describe how EPI works
    - Understand what “spoiling” means
2. Understand the most popular pulse sequences and their acronyms
    - Describe how FSE/TSE sequences work
    - Describe how EPI works
    - Describe how fast gradient-echo sequences work
3. Manipulate MRI sequence parameters to improve performance
    - Understand how and when to accelerate with FSE/TSE
    - Understand how and when to accelerate with EPI
    - Understand how contrast changes in fast gradient-echo sequences
4. Identify artifacts and how to mitigate them
    - Identify FSE/TSE artifacts include T2 blurring
    - Identify EPI artifacts including distortion and T2*
    - Identify fast gradient-echo artifacts such as banding in bSSFP

22.2 Multiple Spin-echo Sequences (RARE/FSE/TSE)

These pulse sequences use multiple spin-echo refocusing pulses after a single exictation pulse to acquire multiple k-space lines to be acquired during the multiple spin-echoes that are formed. This technique was originally called Rapid Acquisition with Relaxation Enhancement (RARE), and is known on various MRI scanners as fast spin-echo (FSE, GE Healthcare), turbo spin-echo (TSE, Siemens Healthineers), or turbo field-echo (TFE, Philips Healthcare).

This is the most common pulse sequence used in clinical practice. This is because it is highly SNR efficient and can also be used to generate multiple contrasts. They are particularly effective for T2 and PD-weighted imaging.

22.2.1 Simplified Pulse Sequence Diagram



22.2.2 Sequence parameters

- Echo spacing (t_{esp}) - time between each spin-echo
- Echo train length (ETL) - number of spin-echoes with a repetition
- Effective TE (TE_{eff}) - the echo time when the data closest to the center of k-space is acquired.

22.2.3 Tradeoffs

Pros

- Fast
- SNR efficient
- Supports multiple contrasts by manipulating TE_{eff}

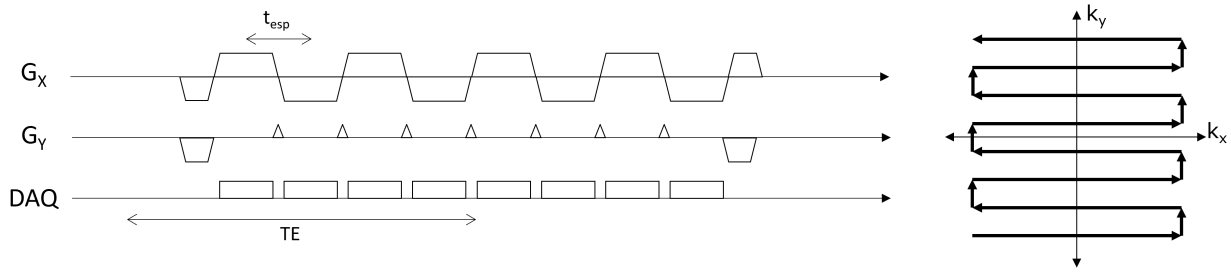
Cons

- Long echo train lengths can lead to T2 blurring or other distortions
- High SAR from repeated refocusing pulses

22.3 Echo-planar Imaging (EPI)

These pulse sequences readout multiple k-space lines sequentially for faster imaging. It is commonly used for diffusion-weighted imaging and fMRI.

22.3.1 Pulse Sequence and K-space Diagram



22.3.2 Sequence parameters

- Echo spacing (t_{esp}) - time between each readout and gradient-echo
- Echo train length (ETL) - number of readouts or k-space lines acquired within a repetition
- TE - the time when the data closest to the center of k-space is acquired.

22.3.3 Tradeoffs

Pros

- Extremely Fast, supporting single-shot imaging

Cons

- Susceptible to chemical shift and susceptibility displacement artifacts
- Sensitive to gradient fidelity artifacts (e.g. Nyquist ghosting)
- Long echo train lengths can lead to T2* blurring

22.4 Fast gradient-echo sequences

The basic gradient-echo sequence is typically a spoiled gradient-echo sequence. The sequence is called spoiled because the transverse magnetization is spoiled by a spoiler gradient before the next RF pulse.

22.4.1 Spoiler or Crusher gradients

A large, unbalanced gradient will create dephasing of the transverse magnetization across the imaging voxels. This effectively eliminates the signal. However, the net magnetization is not truly eliminated, and this can be refocused by gradients.

22.4.2 RF spoiling

The RF axis of rotation is changed every TR. This reduces the chances of magnetization from previous TRs to become coherently excited. Specific RF spoiling schemes, such as quadratic phase incrementation, is required for this to be effective.

22.4.3 Types of fast gradient-echo sequences

Spoiled gradient-echo (SPGR)/fast low-angle shot (FLASH)

- Aims to full spoil transverse magnetization every TR
- Both gradient and RF spoiling are used
- Pure T1 weighted contrast

$$S_{SPGR} \propto M_0 \sin \theta \frac{1 - \exp(-TR/T_1)}{1 - \cos \theta \exp(-TR/T_1)}$$

Gradient-recalled acquisition in the steady state (GRASS)/fast imaging with steady-state precession (FISP)

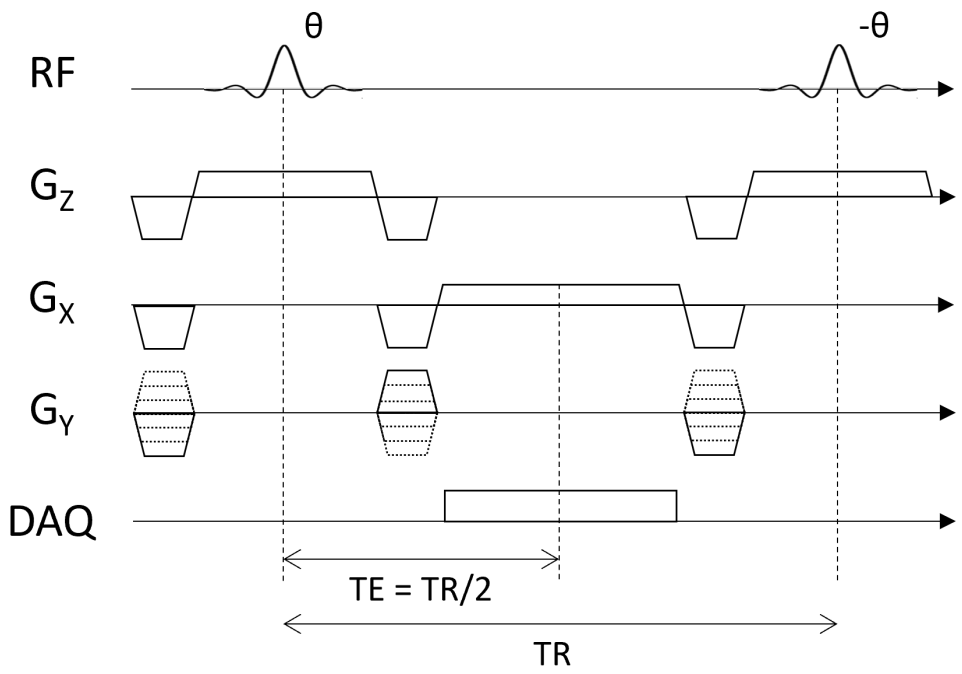
- Allows for residual transverse magnetization to be used in the next TR
- Refouces frequency and phase encoding gradients
- RF spoiling used
- Increases T2* contrast

Steady-state free precession (SSFP)/time-reversed fast imaging with steady-state precession (PSIF)

- Use gradients to refocus signals from previous TRs
- Gradient spoiling but refocused in later TRs
- Creates T2/T1 contrast

Balanced SSFP/TrueFISP

- Balanced gradients every TR
- No spoiling, all magnetization preserved each TR
- Creates T2/T1 contrast
- High SNR efficiency

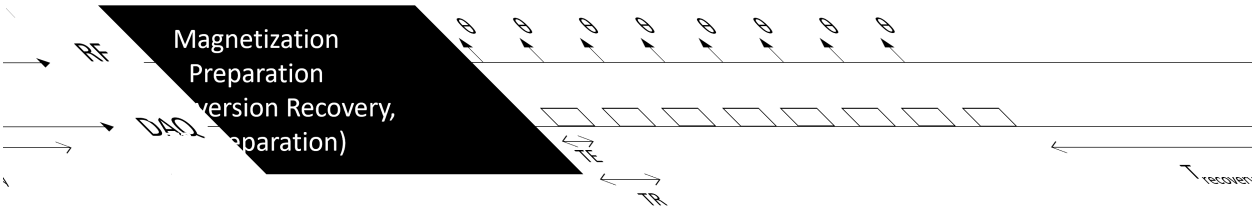


$$S_{bSSFP} \propto \frac{M_0}{2} \sqrt{\frac{T_2}{T_1}}$$

22.4.4 Magnetization Prepared Fast Gradient-echo Sequences

Magnetization preparation schemes, such as inversion recovery, T2-preparation, and magnetization transfer, encode contrast in the longitudinal magnetization. This contrast can be efficiently imaged with multiple gradient-echo readouts after a single magnetization preparation module.

The most common of this type of sequence is the Magnetization Prepared Rapid Gradient Echo (MPRAGE) sequence for brain imaging, which specifically uses inversion recovery magnetization preparation to create T1-weighted contrast, followed by multiple spoiled gradient-echo readouts to retain only T1-weighted contrast.



This sequence includes several additional parameters, including the recovery time, $T_{recovery}$, and number of readouts per preparation, $N_{segments}$, as well as the specific paramters of the magnetization preparation.

```
# Simulate MPRAGE MRI Pulse Sequence, which typically uses inversion preparation

import numpy as np
import matplotlib.pyplot as plt

flip_angle = 10 * np.pi / 180  # Flip angle in radians
TE = 1  # Echo time in ms
TR = 5 # Repetition time in ms
N_segments = 100  # Number of pulses
```

(continues on next page)

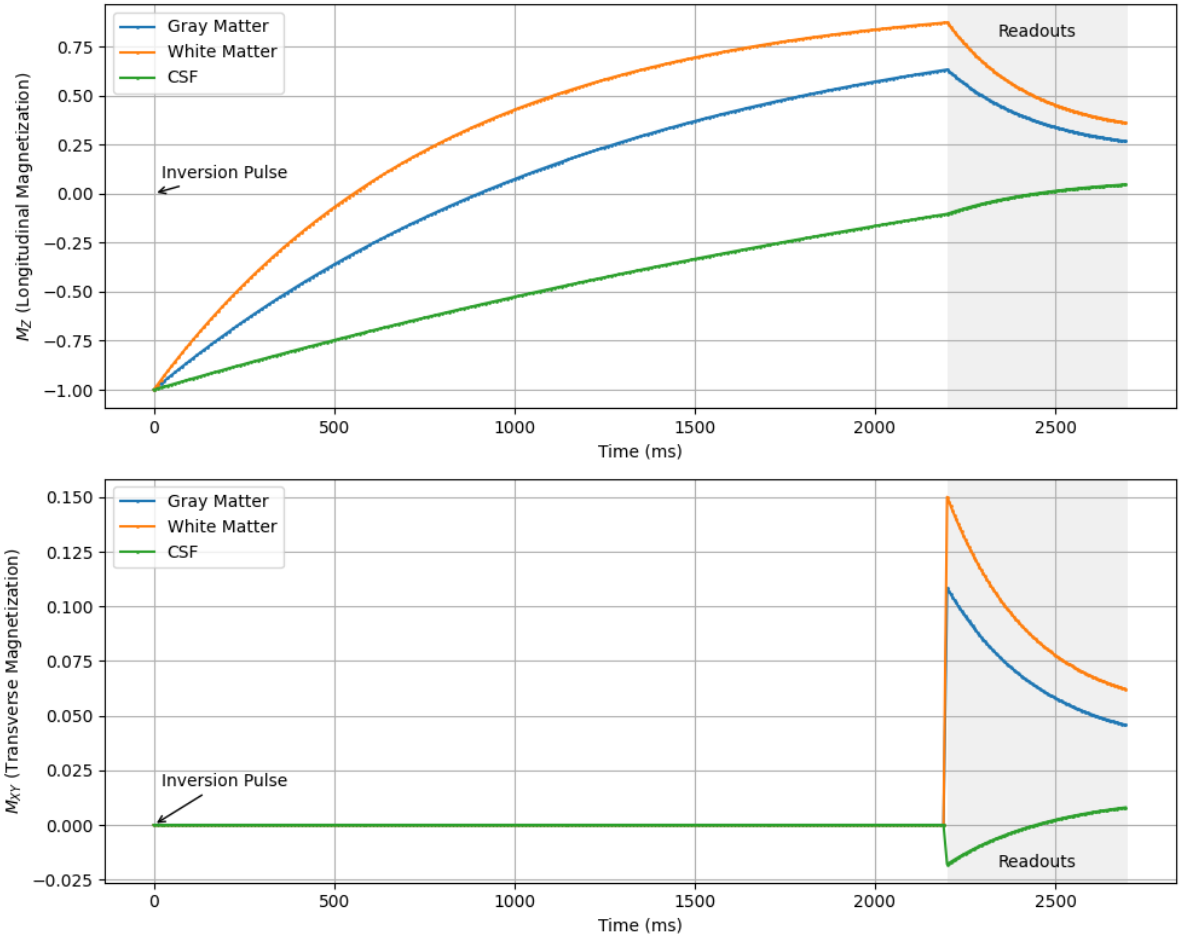
(continued from previous page)

```
T_recovery = 2200  # Delay after inversion pulse in ms

MZ_start = 1.0  # Initial longitudinal magnetization
MZ_inversion = -MZ_start  # After inversion pulse

t_recovery = np.arange(0, T_recovery, 10)  # Time array for recovery period
t_segments = np.arange(0, N_segments * TR, TR) + T_recovery  # Time array for pulse␣
↪sequence

# Define T1/T2 values for different tissue types
tissues = {
    'Gray Matter': {'T1': 1300, 'T2': 80},
    'White Matter': {'T1': 800, 'T2': 110},
    'CSF': {'T1': 3700, 'T2': 1700}
}

fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(10, 8))

for tissue_name, params in tissues.items():
    T1, T2 = params['T1'], params['T2']

    # Vectorized longitudinal recovery during the inversion recovery period
    MZ_recovery = MZ_inversion * np.exp(-t_recovery / T1) + (1 - np.exp(-t_recovery /␣
↪T1))

    MZ_segments = np.zeros(N_segments)
    MXY_segments = np.zeros(N_segments)

    # Longitudinal magnetization at the start of the segmented readouts
    MZ_segments[0] = MZ_inversion * np.exp(-T_recovery / T1) + (1 - np.exp(-T_
↪recovery / T1))
    for n in range(1, N_segments):
        MZ_segments[n] = MZ_segments[n-1] * np.cos(flip_angle) * np.exp(-TR / T1) +␣
↪(1 - np.exp(-TR / T1))

    for n in range(0, N_segments):
        MXY_segments[n] = MZ_segments[n] * np.sin(flip_angle) * np.exp(-TE / T2)

    ax1.plot(np.concatenate((t_recovery, t_segments)), np.concatenate((MZ_recovery,␣
↪MZ_segments)), label=tissue_name, marker='o', markersize=1)
    ax2.plot(np.concatenate((t_recovery, t_segments)), np.concatenate((np.zeros_
↪like(MZ_recovery), MXY_segments)), label=tissue_name, marker='o', markersize=1)

ax1.set_xlabel('Time (ms)')
ax1.set_ylabel('$M_Z$ (Longitudinal Magnetization)')
ax1.legend()
ax1.grid(True)

ax2.set_xlabel('Time (ms)')
ax2.set_ylabel('$M_{XY}$ (Transverse Magnetization)')
ax2.legend()
ax2.grid(True)

# Annotations
start_readouts = T_recovery
end_readouts = T_recovery + N_segments * TR
mid_readouts = (start_readouts + end_readouts) / 2
```

(continues on next page)

(continued from previous page)

```
y1_min = ax1.get_ylim()[0]
y2_min = ax2.get_ylim()[0]
y1_max = ax1.get_ylim()[1]
y2_max = ax2.get_ylim()[1]

ax1.annotate('Inversion Pulse', xy=(0, 0), xytext=(20, y1_max * 0.15),
             arrowprops=dict(arrowstyle='->'), ha='left', va='top')
ax2.annotate('Inversion Pulse', xy=(0, 0), xytext=(20, y2_max * 0.15),
             arrowprops=dict(arrowstyle='->'), ha='left', va='top')

ax1.text(mid_readouts, y1_max * 0.9, 'Readouts', ha='center', va='top')
ax2.text(mid_readouts, y2_min * 0.5, 'Readouts', ha='center', va='top')
ax1.axvspan(start_readouts, end_readouts, color='0.9', alpha=0.6, linewidth=0)
ax2.axvspan(start_readouts, end_readouts, color='0.9', alpha=0.6, linewidth=0)

plt.tight_layout()
plt.show()
```



CHAPTER

TWENTYTHREE

ACCELERATED IMAGING METHODS

MRI scans can be accelerated by applying methods that reduce the number of k-space samples required, also known as undersampling. These require a corresponding advanced image reconstruction method to allow for the reduced sampling. The techniques covered here include partial Fourier reconstruction, parallel imaging, compressed sensing, and deep learning methods.

23.1 Learning Goals

1. Describe how images are formed
    - Describe how accelerated imaging methods work
2. Understand the constraints and tradeoffs in MRI
    - Identify accelerated imaging methods
    - Understand when accelerated imaging methods can be applied
3. Manipulate MRI sequence parameters to improve performance
    - Define which parameters are modified when accelerated imaging methods are applied
4. Manipulate and analyze MRI data
    - Reconstruct an image from undersampled raw data

23.2 General Principles

Accelerated Imaging in MRI are methods that reduce the number of k-space samples required. Since this violates our sampling criteria (Nyquist) this is often referred to as "undersampling." This allows for shorter scan times, which can be used to incease throughput, reduce artifacts, acquire more contrasts, and/or improve spatial resolution.

All of these methods require additional steps in image reconstruction beyond using a Fourier Transform. These steps are based on the assumptions and a model for the accelerated imaging method being used.

23.3 Methods Inventory and Comparison

Method	Assumptions	Methods	Pros	Cons
Partial Fourier	Transverse magnetization phase is slowly varying across the image	Skip almost one side of k-space; Estimate phase with center of k-space	Can be used in many circumstances; Reduce phase encodes or frequency encoding readout length	Should not be used when phase contrast is important
Parallel Imaging	RF receive coil arrays provide spatial encoding information	Skip phase encoding lines and/or simultaneous multi-slice excitation; Measure or model RF coil sensitivities and incorporate in reconstruction	Can be used with all pulse sequences	Geometry "g-factor" SNR losses
Compressed Sensing	Data is compressible and has a sparse representation	Pseudo-random skip of phase encoding lines; Solve optimization problem in reconstruction	Can be used with all pulse sequences	Non-linear reconstruction artifacts
Deep Learning Methods	Typical patterns in data can be learned	Skip phase encoding lines; Train a model based on existing data	Can be used with all pulse sequences	Overfitting reconstruction artifacts that are not visible

23.4 Partial Fourier Imaging

Partial Fourier imaging, also known as fractional NEX or partial k-space, utilizes the conjugate symmetry property of the Fourier Transform. This property states that the Fourier Transform of a real-valued function has conjugate symmetry. If our image, $m(\vec{r})$, is real-valued, then the k-space data satisfies

$$M(\vec{k}) = M^*(-\vec{k}), \ \mathrm{if} \ \mathcal{Im}(m(\vec{r})) = 0$$

Where $\mathcal{Im}(m(\vec{r}))$ is the imaginary part. Note that this can also be used if a constant phase offset is present, and this can be estimated and removed such that $\mathcal{Im}(m(\vec{r}) e^{i \phi}) = 0$.

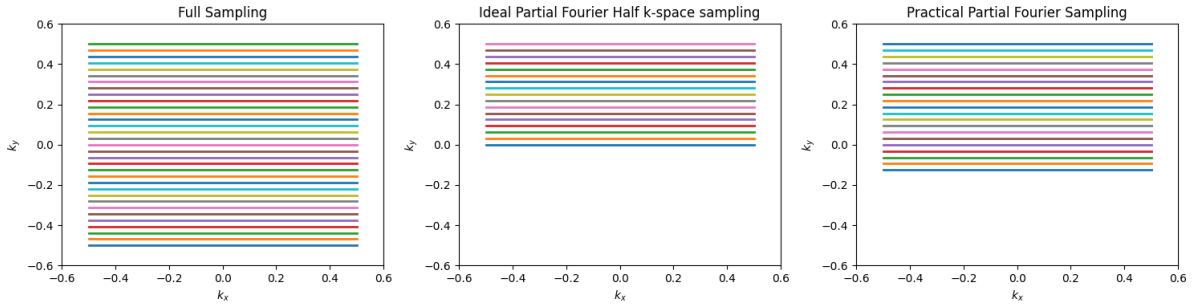
Our object is the transverse magnetization, so this conidition is satisfied if our transverse magnetization is aligned across the entire image. With a strict definition of $m(\vec{r}) = M_{XY}(\vec{r},0) = M_{X}(\vec{r},0) + i M_{Y}(\vec{r},0)$, this would mean all transverse magnetization is along M_X. In practice a global phase offset can be estimated so this relationship holds as long as the transverse magnetizations are all aligned.

In this case, only half of k-space is required, and the other half can be filled in by this property. This means 2 times faster scanning!

In practice, there are sources of phase that result in violation of the real-valued object condition. First, there can be a global phase offset (e.g. transverse magnetization aligned along M_Y), but this is not a problem since a global phase correction can be applied to correct for this. More problematic are other sources of phase, which come from the RF coil profiles, off-resonance, and chemical shift.

However, much of the additional phase is typically contained in the low spatial frequencies, especially from the RF coils and main magnetic field inhomogeneities, $\Delta B_0(\vec r)$, which are captured in central k-space values. Thus, practical Partial Fourier methods acquire slightly more than half of k-space, and then use specialized algorithms such as projection onto convex sets (POCS) or homodyne detection reconstructions that include correction of the low spatial frequency phase.

```
# Sample k-space sampling patterns for Partial Fourier Imaging

import numpy as np
import matplotlib.pyplot as plt

# Cartesian
N = 32
k = np.linspace(-N/2, N/2, N+1) / N
ky, kx = np.meshgrid(k, k)

fig, axes = plt.subplots(1, 3, figsize=(15, 4))

# Full Sampling
axes[0].plot(kx, ky, linewidth=2)
axes[0].set_xlim([-.6, .6])
axes[0].set_ylim([-.6, .6])
axes[0].set_xlabel('$k_x$')
axes[0].set_ylabel('$k_y$')
axes[0].set_title('Full Sampling')

# Ideal Partial Fourier Half k-space sampling
# Reshape to have one dimension of size N (here N+1 points exist, reshape accordingly)
mask = ky >= 0
kx_half = kx[mask].reshape(N + 1, -1)
ky_half = ky[mask].reshape(N + 1, -1)

axes[1].plot(kx_half, ky_half, linewidth=2)
axes[1].set_xlim([-.6, .6])
axes[1].set_ylim([-.6, .6])
axes[1].set_xlabel('$k_x$')
axes[1].set_ylabel('$k_y$')
axes[1].set_title('Ideal Partial Fourier Half k-space sampling')

# Practical Partial Fourier Sampling with over-sampling the center of k-space
pf_factor = 0.75

mask_pf = ky >= -(pf_factor-0.5)/2
kx_pf = kx[mask_pf].reshape(N + 1, -1)
ky_pf = ky[mask_pf].reshape(N + 1, -1)

axes[2].plot(kx_pf, ky_pf, linewidth=2)
axes[2].set_xlim([-.6, .6])
axes[2].set_ylim([-.6, .6])
axes[2].set_xlabel('$k_x$')
axes[2].set_ylabel('$k_y$')
axes[2].set_title('Practical Partial Fourier Sampling')

plt.tight_layout()
```



23.5 Parallel Imaging

Parallel Imaging uses the spatial information provided by individual RF receive coil elements to accelerate image acquisition. With multiple RF receive coil elements, they can provide additional coarse scale localization information. The primary parallel imaging approach is to undersample k-space by skipping phase encoding lines. Another approach is to simultaneously excite and acquire multiple slices, known as simultaneous multi-slice (SMS) imaging.

23.5.1 Undersampled k-space Methods

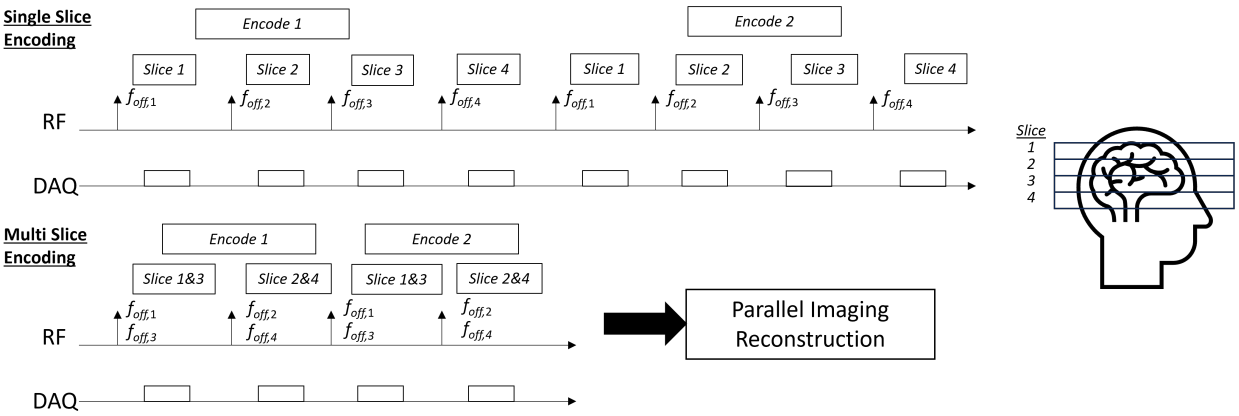
In these approaches, phase encoding lines are skipped in k-sapce. With direct Fourier Transform reconstruction, this results in aliasing artifacts due to violation of the sampling requirements. Parallel imaging reconstruction methods use the spatial information from the multiple RF receive coil elements to resolve this aliasing.

All of these methods require incorporation of the coil sensitivity profiles, either through measurement, estimation, or incorporation into a model. Coil sensitivity profiles can be measured with a coarse resolution (i.e. fast) pre-scan, and signal from the relatively uniform Body RF coil can be used as a reference to estimate the local receive coil element sensitivities. This is incoporated into the popular "SENSE" method. This approach will suffer from residual aliasing artifacts if there are mismatches between the measured coil sensitivities and the actual coil sensitivities during the accelerated scan, for example due to motion.

Methods that estimate or incorporate into the model the coil sensitivies typically require a fully sampled calibration region in the center of k-space. This increases the scan time, but are more robust since they estimate the coil sensitivities directly from the data being reconstructed. This includes the popular "GRAPPA" method.

23.5.2 Simultaenous Multi-Slice (SMS) Methods

Simultaneous Multi-Slice (SMS) methods, also known as multi-band imaging, excite and acquire multiple slices simultaneously, then using the coil sensitivity profiles in the reconstruction to separate the signal from each slice. The RF excitation pulses must excite multiple slice locations, which naively is done by summing multiple frequency-shifted RF pulses, each corresponding to a different slice location. In practice, more complex designs are used to mitigate peak power and improve robustness. The resulting k-space data contains the superposition of the signals from the multiple slices.



These methods also often use specialized RF phase modulation schemes, such as CAIPIRINHA, to improve the conditioning of the reconstruction problem by shifting the simultaneously excited slices in the phase encoding direction.

23.5.3 SNR in Parallel Imaging

There is an SNR penalty when using these methods that varies in severity depending on the conditioning of the undersampled reconstruction matrix. Variation in RF coil element sensitivity profiles in the direction of the undersampling leads to a more well-conditioned reconstruction and lower SNR penalty. Conversely, less RF coil element sensitivity profile variation in the direction of the undersampling leads to a more ill-conditioned reconstruction matrix and higher SNR penatly. Also, regions will little difference betweeen RF coils (typically in the center of the body), also tend to have larger SNR penalties.

This is characterized by the geometry factor, or "g-factor", where $g(\vec{r}) \geq 1$ characterizes a spatially-varying SNR loss that is dependent on the coil loading and geometry, k-space sampling pattern, and parallel imaging reconstruction method. When appyling an acceleration factor of R, the total readout time reduces the SNR by $\sqrt{R}$ as well, leading to the SNR relationship:

$$SNR_{PI} = \frac{SNR_{full}}{g(\vec{r})\sqrt{R}}$$

23.5.4 Artifacts with Parallel Imaging

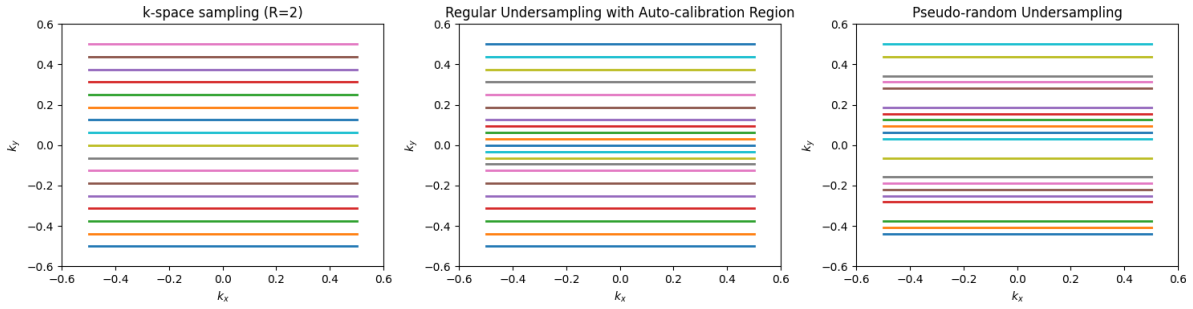
Parallel Imaging artifacts typically appear as aliasing. The undersampled data will have aliasing when not using a parallel imaging reconstruction method, and sometimes the aliasing is not completely removed. There are also aliasing like artifacts when there is a mismatch between the data and measured coil sensitivity profiles.

```
# Sample k-space sampling patterns for parallel imaging and compressed sensing

import numpy as np
import matplotlib.pyplot as plt

# Cartesian
N = 32
k = np.linspace(-N/2, N/2, N+1) / N
ky, kx = np.meshgrid(k, k)

fig, axes = plt.subplots(1, 3, figsize=(15, 4))

# Regular undersampling pattern
R = 2  # Acceleration factor
mask = (np.arange(N + 1) % R) == 0  # undersample only along ky (row) dimension
```

(continues on next page)

(continued from previous page)

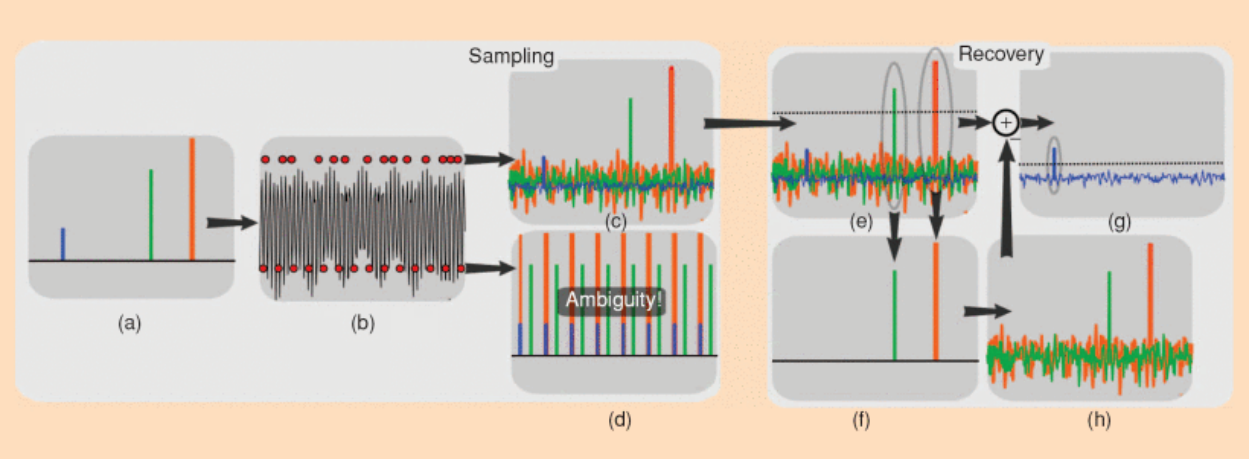
```
kx_regular = kx[:,mask]             # keep all kx samples for selected ky lines
ky_regular = ky[:,mask]

axes[0].plot(kx_regular, ky_regular, linewidth=2)
axes[0].set_xlim([-.6, .6])
axes[0].set_ylim([-.6, .6])
axes[0].set_xlabel('$k_x$')
axes[0].set_ylabel('$k_y$')
axes[0].set_title('k-space sampling (R=2)')

# Regular undersampling, with autocalibration region
acs_size = 8  # Size of autocalibration region
mask_acs = ((np.arange(N + 1) % R) == 0) | (np.abs(ky) < (acs_size / (N + 1) / 2))

kx_acs = kx[mask_acs].reshape(N + 1, -1)
ky_acs = ky[mask_acs].reshape(N + 1, -1)

axes[1].plot(kx_acs, ky_acs, linewidth=2)
axes[1].set_xlim([-.6, .6])
axes[1].set_ylim([-.6, .6])
axes[1].set_xlabel('$k_x$')
axes[1].set_ylabel('$k_y$')
axes[1].set_title('Regular Undersampling with Auto-calibration Region')

# Pseudo-random undersampling
mask_random = np.random.rand(N + 1) < (1 / R)
kx_random = kx[:,mask_random]
ky_random = ky[:,mask_random]

axes[2].plot(kx_random, ky_random, linewidth=2)
axes[2].set_xlim([-.6, .6])
axes[2].set_ylim([-.6, .6])
axes[2].set_xlabel('$k_x$')
axes[2].set_ylabel('$k_y$')
axes[2].set_title('Pseudo-random Undersampling')

plt.tight_layout()
```



23.6 Compressed Sensing Methods

Compressed Sensing (CS) theory says that an image that is compressible in some domain can then be reconstructed from a subset of data samples. MRI images are in fact compressible, so we can further accelerate our data acquisition with CS.

Compressed Sensing for MRI is achieved by using k-space sampling patterns with pseudo-random undersampling, which is illustratid by the diagram below of the undersampled reconstruction procedure. This results in incoherent artifacts that appear noise-like, which can then be removed by constraining the reconstruction to a sparse solution in some transform domain.



Caption (Figure from https://doi.org/10.1109/MSP.2007.914728): Heuristic procedure for reconstruction from undersampled data. A sparse signal (a) is 8-fold undersampled in its 1-D k -space domain (b). Equispaced undersampling results in signal aliasing (d) preventing recovery. Pseudo-random undersampling results in incoherent interference (c). Some strong signal components stick above the interference level, are detected and recovered by thresholding (e) and (f). The interference of these components is computed (g) and subtracted (h), thus lowering the total interference level and enabling recovery of weaker components.

Popular sparsifying transforms include total variation (TV), total generalized variation (TGV), and wavelets, where MRI images are typically sparse. The pseudo-random undersampling typically uses a variable density that preferentially has more samples near the center of k-space.

23.6.1 SNR in Compressed Sensing

It is difficult to define SNR when using compressed sensing reconstruction methods, as they inherently perform some denoising when constraining the reconstruction based on sparsity. The apparent SNR can also vary significantly depending on the choice of the regularization factor.

23.6.2 Artifacts with Compressed Sensing

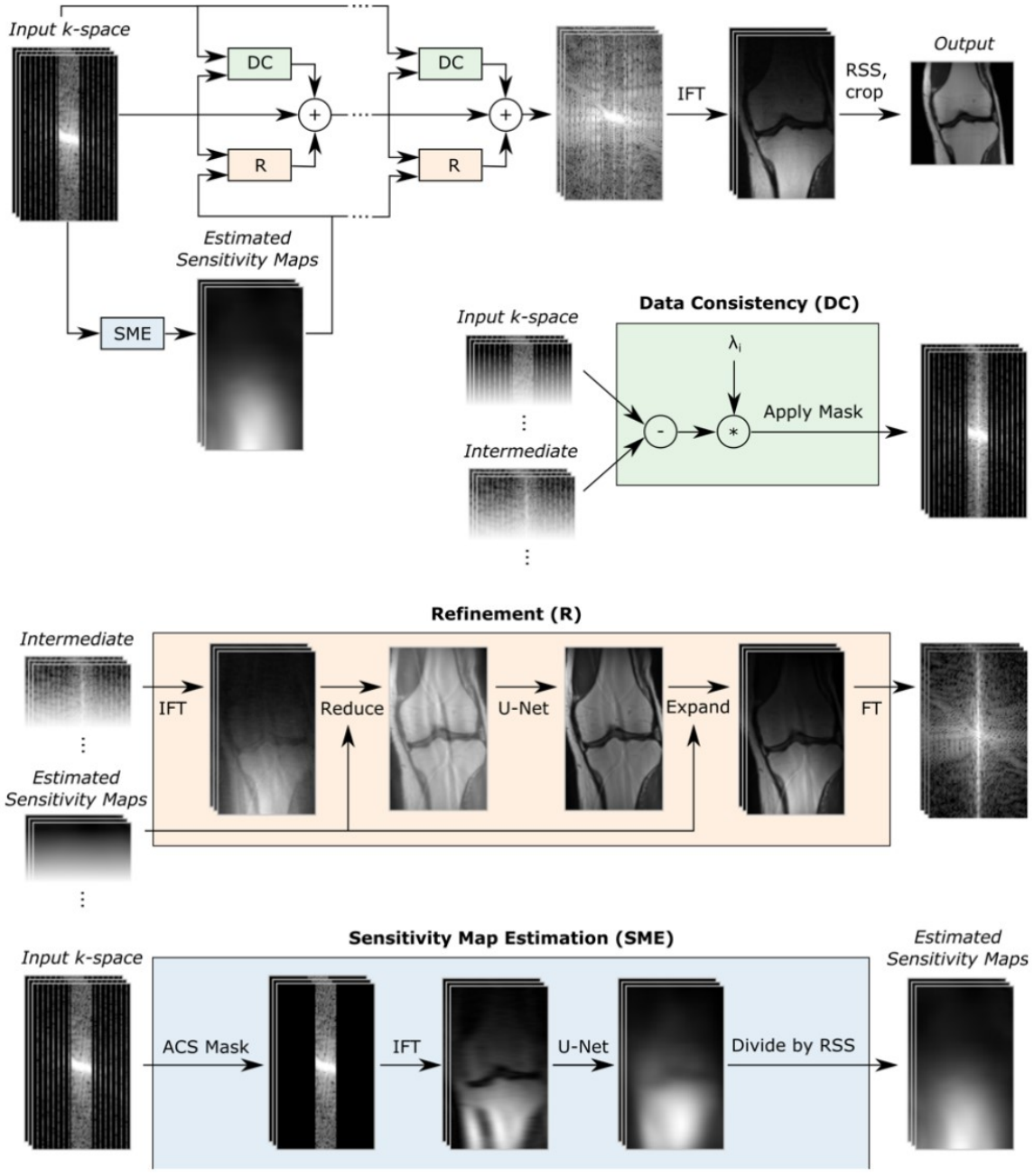
The most common artifacts from compressed sensing come from overfitting to the sparsity penalty. This result in either an over-smoothed, unnatural appearance or, in the case of a wavelet sparsifying transform, blocking artifacts.

23.7 Deep Learning Reconstructions

Deep learning (DL) based methods can be used to reconstructed undersampled k-space into image data from a subset of data samples as well. Conceptually, these methods can learn how to incorporate coil sensitivity information like parallel imaging and typical image sparsity patterns like compressed sensing. Since they learn from real-world data, they can learn information that is the most relevant to MRI data and thus have been shown to support higher acceleration factors. For example, they can learn what patterns within the image and/or k-space domain and most relevant and most common in MRI.

These methods require learning from large amounts (e.g. thousands or more) of training data samples, including fully-sampled MRI data as the ground truth. Fortunately, efforts such as the fastMRI project have provided large, high-quality datasets for training and evaluating these methods.

So-called un-rolled networks have become more popular for MRI reconstruction, as they can better incorporate parallel imaging information, data consistency, and learning common patterns in the data. They are designed to mimic iterations of classical algorithms to solve optimization problems, and each iteration has been "un-rolled" into a series of cascaded neural networks. Some popular methods include MoDL and Variational Networks.



Caption (from https://arxiv.org/abs/2004.06688): The E2E-VN model takes under-sampled k-space as input and applies several cascades, followed by IFT and RSS. Bottom: The DC module brings intermediate k-space closer to measured values, the Refinement module maps multi-coil k-space to one image, applies a U-Net and maps back to k-space, and the SME estimates sensitivity maps used in the refinement step.

23.7.1 SNR in DL Reconstructions

It is difficult to define SNR when using machine learning reconstruction methods, as they inherently perform some denoising when constraining the reconstruction.

23.7.2 Artifacts with DL Reconstructions

A challenge with machine learning reconstructions is that artifacts can be difficult to identify. They can result in an oversmoothed appearance, but often a well-trained network will retain realistic texture and noise appearance features. These methods can also overfit to the learned anatomy and data, meaning features might be erased or hallucinated, although using physics-based techniques provide some assurances of data-consistency.

Part IX

Imperfections and Artifacts

CHAPTER

TWENTYFOUR

POINT SPREAD FUNCTION ANALYSIS

In this section, we will show how to use point spread function analysis and the MRI signal equation along to understand the effects of phenomena such as relaxation and off-resonance/chemical shift on the resulting images, including resulting artifacts created.

24.1 Learning Goals

1. Describe how images are formed
    - What is the impact of relaxation and off-resonance/chemical shift on the signal
2. Understand the constraints and tradeoffs in MRI
    - How do relaxation and off-resonance/chemical shift limit the readout duration
3. Manipulate MRI sequence parameters to improve performance
    - How does the readout duration affect artifacts due to relaxation and off-resonnace/chemical shift
4. Identify artifacts and how to mitigate them
    - What is the effect of relaxation and off-resonnace/chemical shift during the readout on the resulting images

24.2 MRI Signal Equation

To assess the effects of imperfections on resulting images, we start with a comprehensive version of the MRI signal equation, including relaxation, coil sensitivity, off-resonance and chemical shift, and gradients. The signal from an individual RF coil element n is

$$s_n(t) = \int m(\vec{r})\ C_n(\vec{r})\ e^{-t/T_2^*(\vec{r})}\ e^{-i 2 \pi \Delta f_{cs} t} e^{-i 2 \pi \Delta f_r(\vec{r}) t} \ \exp\left(-i \gamma \int_0^t \vec{G}(\tau) \cdot \vec{r} \ d\tau \right) \ d\vec{r}$$

which is typically simplified for understanding image formation using k-space:

$$s_n(t) = \int m(\vec{r})\ C_n(\vec{r})\ e^{-t/T_2^*(\vec{r})}\ e^{-i 2 \pi \Delta f_{cs} t} e^{-i 2 \pi \Delta f_r(\vec{r}) t} \ e^{-i 2 \pi \vec{k}(t) \cdot \vec{r}} \ d\vec{r}$$

Note that this describes the signal behavior between RF pulses. Incorporating RF pulses can be done either through rotation matrices or some solution of the Bloch equation such as numerical solvers or phase graphs.

24.3 K-space Data Weighting

A powerful framework for characterizing the effects of imperfections is to define a k-space weighting that determines the signal modulation. Expressing this generally, we define a weighting function $W(\vec{k})$ that can be complex-valued and capture amplitude and phase modulations. Then,

$$\hat{M}(\vec{k}) = M(\vec{k}) W(\vec{k}) $$

$$\hat{m}(\vec{r}) = m(\vec{r}) * \mathcal{F}^{-1} { W(\vec{k}) } = m(\vec{r}) * w(\vec{r})$$

Where $w(\vec{r})$ is the inverse Fourier Transform of the weighting function and $*$ is the convolution operation. $w(\vec{r})$ is also known as a point spread function (PSF) or impulse response function. Thus the effects can be boiled down to a convolution operation in the resulting reconstructed image.

When the frequency shift and relaxation rates vary across space, it is helpful to decompose the image into components, m_i that experience the same weighting (e.g. same frequency shift, or same T_2^*) W_i.

$$ m(\vec{r}) = \sum_i m_i(\vec{r}) $$

Since the Fourier Transform is linear,

$$\hat{M}(\vec{k}) = \sum_i M_i(\vec{k}) W_i(\vec{k}) $$

$$\hat{m}(\vec{r}) =\sum_i m_i(\vec{r}) * w_i(\vec{r})$$

So each component is convolved with its own convolution kernel. For example, to analyze chemical shift we can divide our image up into water, $m_1(\vec{r})$, and fat, $m_2(\vec{r})$, components, then $W_1(\vec{k}) = 1$, $W_2(\vec{k}) = e^{-i 2 \pi\ \upsilon(\vec{k}) \Delta f_{fat}}$, and the resulting image will be

$$\hat{m}(\vec{r}) = m_1(\vec{r}) + m_2(\vec{r}) * w_2(\vec{r})$$

providing an accurate image of the water component but some corrupted fat image (typically shifted in space for Cartesian trajectories).

The k-space data weighting concept is very powerful. It can include frequency shifts. It can include weighting from all types of relaxation (T_1, T_2, T_2^*) from more complex acquisition strategies such as SSFP and fast spin-echos. It can include diffusion-weighting, flow effects, and more.

24.4 Relaxation during signal acquisition

To examine the effects of relaxation, we will neglect coil sensitivity, off-resonance, and chemical shift. Relaxation modulates the receieved signal over time. This will create some type of weighting in k-space, which leads to image filtering (often blurring). This weighting is spatially varying, depending on relaxation of different tissues.

$$s(t) = \int m(\vec{r})\ e^{-t/T_2^*(\vec{r})}\ e^{-i 2 \pi \vec{k}(t) \cdot \vec{r}} \ d\vec{r}$$

To interpret the effect of relaxation, consider a single value of $T_2^*(\vec{r}) = T_2^*$, in which case this can be removed from the integral and

$$s(t) = e^{-t/T_2^*} \int m(\vec{r})\ e^{-i 2 \pi \vec{k}(t) \cdot \vec{r}} \ d\vec{r} \ = M(\vec{k}(t))\ e^{-t/T_2^*} $$

From this, we can see that relaxation leads to signal amplitude modulation as we acquire data across k-space. To see the effect on image formation, we determine the function $\upsilon (\vec{k})$ which captures across k-space trajectory in time and across all TRs to characterize this modulation:

$$\hat{M}(\vec{k}) = M(\vec{k}) e^{- \upsilon(\vec{k}) / T_2^*}$$

Examples of $\upsilon(\vec{k})$ are shown below. In the PSF framework, the k-space weighting function is

$$W(\vec{k}) = e^{- \upsilon(\vec{k}) / T_2^*}$$

Thus the reconstructed image will corrupted by a convolution (denoted by $*$) based on the k-space amplitude weighting

$$\hat{m}(\vec{r}) = \mathcal{F}^{-1}\{ \hat{M}(\vec{k})\} = m(\vec{r}) * \mathcal{F}^{-1}\{ e^{-\upsilon(\vec{k})/ T_2^*} \}$$

And in the framework introduced above,

$$w(\vec{r}) = \mathcal{F}^{-1}\{ e^{-\upsilon(\vec{k})/ T_2^*} \}$$

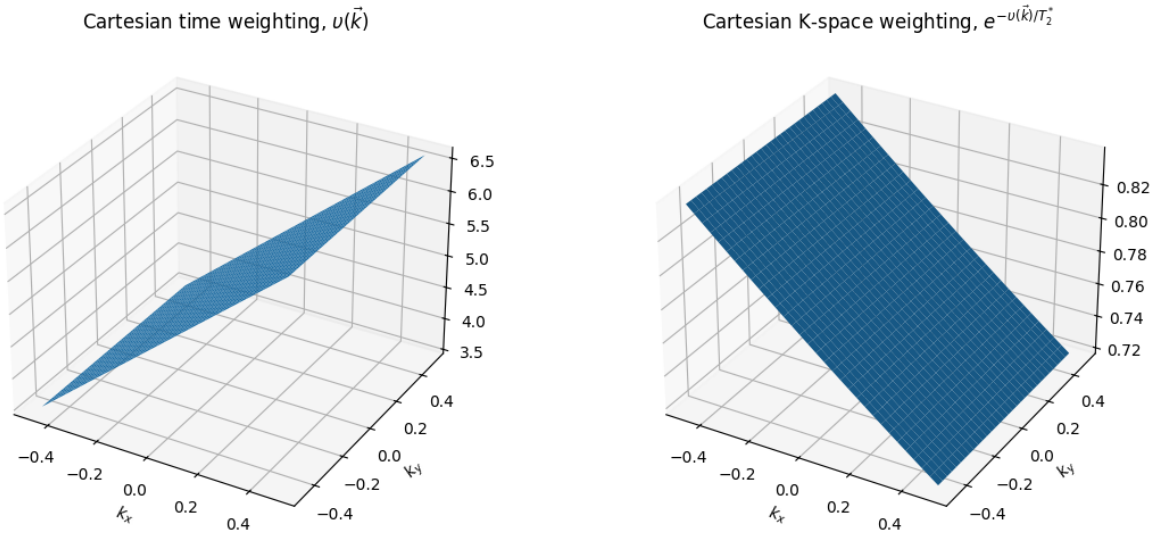
This convolution can result in blurring (low-pass filtering) for center-out type trajectories, or high-pass filtering (e.g. edge enhancement but overall SNR loss) when starting at the edge of k-space.

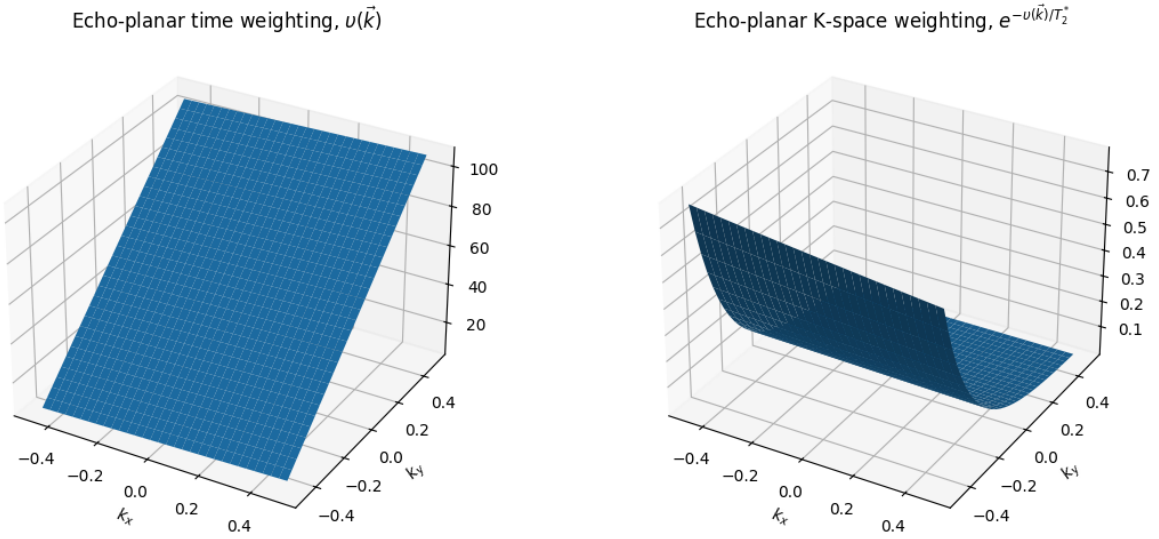
```
import numpy as np
import matplotlib.pyplot as plt

# Example of k-space trajectory time weighting functions, upsilon, and resulting k-
 ↪space weighting
T2star = 20 # ms

# Cartesian
gamma = 42.58  # kHz/mT
N = 32
dt = 0.1  # ms
TE = 5  # ms
t = np.arange(-N/2+1, N/2+1) * dt

Gxr = 1 / (N * dt * gamma)

kx0 = gamma * Gxr * t
kx, ky = np.meshgrid(kx0, kx0)

upsilon_Cartesian = kx / (gamma * Gxr) + TE

fig = plt.figure(figsize=(12, 5))

# First subplot: Cartesian time weighting
ax1 = fig.add_subplot(1, 2, 1, projection='3d')
ax1.plot_surface(kx, ky, upsilon_Cartesian)
ax1.set_xlabel('$k_x$')
ax1.set_ylabel('$k_y$')
ax1.set_title(r'Cartesian time weighting, $\upsilon(\vec{k})$')

# Second subplot: Cartesian K-space weighting
ax2 = fig.add_subplot(1, 2, 2, projection='3d')
ax2.plot_surface(kx, ky, np.exp(-upsilon_Cartesian / T2star))
ax2.set_xlabel('$k_x$')
ax2.set_ylabel('$k_y$')
ax2.set_title(r'Cartesian K-space weighting, $e^{-\upsilon(\vec{k})/  T_2^*}$')

plt.tight_layout()
plt.show()


# EPI
tesp = N * dt  # simplified, assuming no deadtime between ky lines
TE_epi = 5 + tesp * N / 2
tall = np.arange(-N**2/2+1, N**2/2+1) * dt + TE_epi

upsilon_EPI = np.reshape(tall, (N, N))
```

(continues on next page)

(continued from previous page)

```
fig = plt.figure(figsize=(12, 5))

# First subplot: Echo-planar time weighting
ax1 = fig.add_subplot(1, 2, 1, projection='3d')
ax1.plot_surface(kx, ky, upsilon_EPI)
ax1.set_xlabel('$k_x$')
ax1.set_ylabel('$k_y$')
ax1.set_title(r'Echo-planar time weighting, $\upsilon(\vec{k})$')

# Second subplot: Echo-planar K-space weighting
ax2 = fig.add_subplot(1, 2, 2, projection='3d')
ax2.plot_surface(kx, ky, np.exp(-upsilon_EPI / T2star))
ax2.set_xlabel('$k_x$')
ax2.set_ylabel('$k_y$')
ax2.set_title(r'Echo-planar K-space weighting, $e^{-\upsilon(\vec{k})/  T_2^*}$')

plt.tight_layout()
plt.show()
```





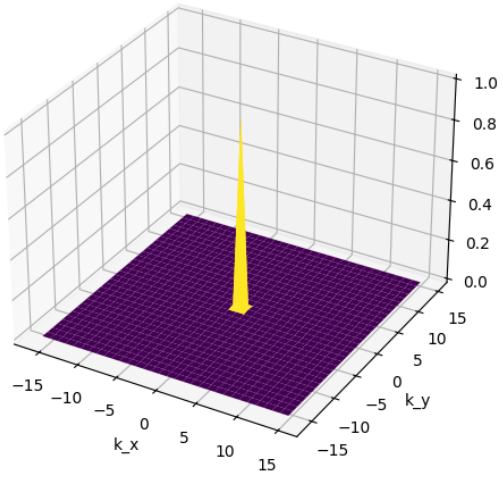

```
import numpy as np

import matplotlib.pyplot as plt

def ifft2c(x):
    # Centered 2D inverse FFT
    return np.fft.ifftshift(np.fft.ifft2(np.fft.fftshift(x)))

# Convolution kernels for T2* weighting

T2star = 100  # ms

W_Cartesian = ifft2c(np.exp(-upsilon_Cartesian / T2star))
W_EPI = ifft2c(np.exp(-upsilon_EPI / T2star))

fig = plt.figure(figsize=(12, 5))
ax1 = fig.add_subplot(1, 2, 1, projection='3d')
ax1.plot_surface(x, y, np.abs(W_Cartesian), cmap='viridis')
ax1.set_xlabel('k_x')
ax1.set_ylabel('k_y')
ax1.set_title(r'Cartesian convolution kernel, $T_2^* = {}$ ms'.format(T2star))
ax1.set_zlim(0, 1)

ax2 = fig.add_subplot(1, 2, 2, projection='3d')
ax2.plot_surface(x, y, np.abs(W_EPI), cmap='viridis')
ax2.set_xlabel('k_x')
ax2.set_ylabel('k_y')
ax2.set_title(r'EPI convolution kernel, $T_2^* = {}$ ms'.format(T2star))
ax2.set_zlim(0, 1)

plt.tight_layout()
plt.show()

T2star = 10  # ms

W_Cartesian = ifft2c(np.exp(-upsilon_Cartesian / T2star))
W_EPI = ifft2c(np.exp(-upsilon_EPI / T2star))
```

(continues on next page)

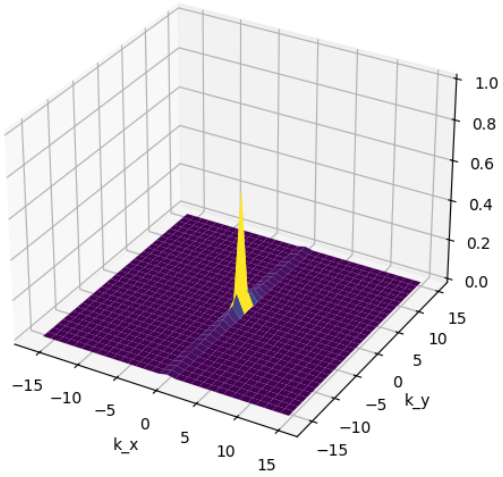
(continued from previous page)

```
fig = plt.figure(figsize=(12, 5))
ax1 = fig.add_subplot(1, 2, 1, projection='3d')
ax1.plot_surface(x, y, np.abs(W_Cartesian), cmap='viridis')
ax1.set_xlabel('x')
ax1.set_ylabel('y')
ax1.set_title(r'Cartesian convolution kernel, $T_2^* = {}$ ms'.format(T2star))
ax1.set_zlim(0, 1)

ax2 = fig.add_subplot(1, 2, 2, projection='3d')
ax2.plot_surface(x, y, np.abs(W_EPI), cmap='viridis')
ax2.set_xlabel('x')
ax2.set_ylabel('y')
ax2.set_title(r'EPI convolution kernel, $T_2^* = {}$ ms'.format(T2star))
ax2.set_zlim(0, 1)

plt.tight_layout()
plt.show()
```

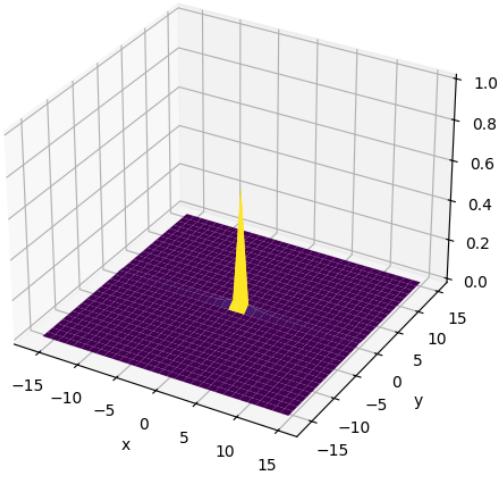
Cartesian convolution kernel, T_2^* = 100 ms



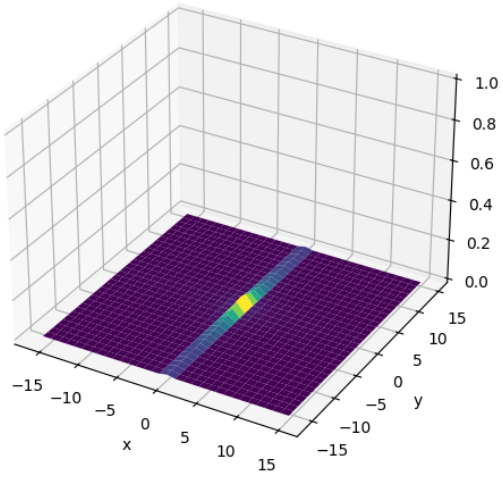
EPI convolution kernel, T_2^* = 100 ms



Cartesian convolution kernel, T_2^* = 10 ms



EPI convolution kernel, T_2^* = 10 ms



In the above plots, the height of the main peak in the center represents the expected SNR, including losses due to blurring, while the signal amplitude outside of the main peak represents blurring that will occur. These show that the blurring and signal loss from T_2^* gets worse as the relaxation time is shorter, the blurring it is much worse for EPI (in phase encoding direction) versus Cartesian trajectories.

24.5 Off-resonance and Chemical Shift

Off-resonance and chemical shift lead to changes in the precession frequency of the magnetization.
For simplicity, combine all off-resonance and/or chemical shift into a single frequnecy shift term, $\Delta f(\vec{r}) = \Delta f_{cs} + \Delta f_r(\vec{r})$ as

$$s(t) = \int m(\vec{r})\ e^{-i 2 \pi \Delta f(\vec{r}) t} e^{-i 2 \pi \vec{k}(t) \cdot \vec{r}} \ d\vec{r}$$

Solving for the effect of frequency shifts then requires specific knowledge of the k-space trajectory, and the interaction of a spatially varying field further complicates this effect. However, some intuition can be gained by assuming a single frequency shift, $\Delta f(\vec{r}) = \Delta f_0$, which can then be removed from the integral as

$$s(t) = e^{-i 2 \pi \Delta f_0 t} \int m(\vec{r})\ e^{-i 2 \pi \vec{k}(t) \cdot \vec{r}} \ d\vec{r} \ = M(\vec{k}(t))\ e^{-i 2 \pi \Delta f_0 t} $$

From this, we can see that the frequency shift leads to a phase accumulation as we acquire data across k-space. To see the effect on image formation, we use the function $\upsilon (\vec{k})$ which captures across k-space trajectory in time (also used in relaxation effects above) and across all TRs to characterize this phase accumulation:

$$\hat{M}(\vec{k}) = M(\vec{k}) e^{-i 2 \pi\ \upsilon(\vec{k}) \Delta f_0}$$

$$W(\vec{k}) = e^{-i 2 \pi\ \upsilon(\vec{k}) \Delta f_0}$$

Thus the reconstructed image will corrupted by a convolution based on the k-space phase accumulation

$$\hat{m}(\vec{r}) = m(\vec{r}) * \mathcal{F}^{-1} { e^{-i 2 \pi\ \upsilon(\vec{k}) \Delta f_0} }$$

$$w(\vec{r}) = \mathcal{F}^{-1} { e^{-i 2 \pi\ \upsilon(\vec{k}) \Delta f_0} }$$

24.5.1 Displacement Artifacts

The result of the above convolution results in a shift or displacement for Cartesian and echo-planar sampling, which is illustrated below. This is a result of a linear phase accumulation across k-space, which leads to a shift in image space.

Cartesian Sampling

In Cartesian sampling, the phase accumulation occurs during the readout, which is by convention defined as k_x. For 2D imaging, we use the relationship that $k_x(t) = \bar\gamma G_{xr} t = \frac{RBW}{FOV_x} t$ (neglecting an constant offset that leads to a constant phase offset), then

$$\upsilon(k_x,k_y) = t = \frac{FOV_x}{RBW} k_x$$

$$W(k_x,k_y) = e^{-i 2 \pi\ k_x \frac{FOV_x}{RBW} \Delta f_0}$$

$$w(k_x, k_y) = \delta(k_x - \frac{\Delta f_0}{RBW} FOV_x,k_y)$$

Convolution with a delta function is equivalent to a shift in the image in k_x:

$$\hat{m}(x,y) = m(x,y) * \delta(k_x - \frac{\Delta f_0}{RBW} FOV_x,k_y) = m(k_x- \frac{\Delta f_0}{RBW} FOV_x, k_y)$$

EPI Sampling

In EPI sampling, the phase accumulation occurs primarily during phase encoding, which is by convention defined as k_y. Assume that the phase during frequency encoding is negligible, and use the relationship that $k_y(t) = \Delta k_y t / t_{esp} = \frac{BW_{PE}}{FOV_y} t$ (again, neglecting an constant offset that leads to a constant phase offset). Here, a phase encoding bandwidth is now defined as the inverse of the echo spcaing $BW_{PE} = 1 / t_{esp}$.

$$\upsilon(k_x,k_y) = \frac{FOV_y}{BW_{PE}} k_y$$

$$W(k_x,k_y) = e^{-i 2 \pi\ k_y \frac{FOV_y}{BW_{PE}} \Delta f_0}$$

$$w(k_x, k_y) = \delta(k_x,k_y - \frac{\Delta f_0}{BW_{PE}} FOV_y)$$

Convolution with a delta function is equivalent to a shift in the image in k_y:

$$\hat{m}(x,y) = m(k_x, k_y- \frac{\Delta f_0}{BW_{PE}} FOV_y)$$

Since the overall readout time is typically much longer in EPI than Cartesian sampling, this leads to typically larger phase accumulations and larger shifts in the image. Equivalently, the phase encoding bandwidths are typically lower than the readout bandwidths.

Other Trajectories

In other trajectories, off-resonance and chemical shift do not appear as simple shifts. For example, they appear as a blurring and ringing like artifact for spiral and radial sampling.

24.5.2 Chemical Shift Displacement Artifact

For the specific case of fat and water, we use the decomposition above to separate the two components, $m_1(\vec{r}) = m_{water}(\vec{r})$ for water and $m_2(\vec{r}) = m_{fat}(\vec{r})$ for fat, with their own frequency shifts, $\Delta f_{water} = 0$ and Δf_{fat}, respectively. This leads The weighting functions are then

$$W_{water}(\vec{k}) = 1$$

$$W_{fat}(\vec{k}) = e^{-i 2 \pi\ \upsilon(\vec{k}) \Delta f_{fat}}$$

And the resulting image will be

$$\hat{m}(\vec{r}) = m_{water}(\vec{r}) + m_{fat}(\vec{r}) * w_{fat}(\vec{r})$$

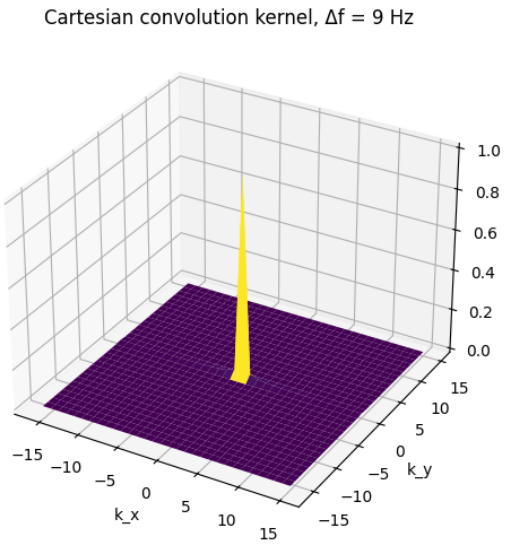
In the case of Cartsian or echo-planar sampling, the water component will be accurately represented, while the fat component will be shifted in space.

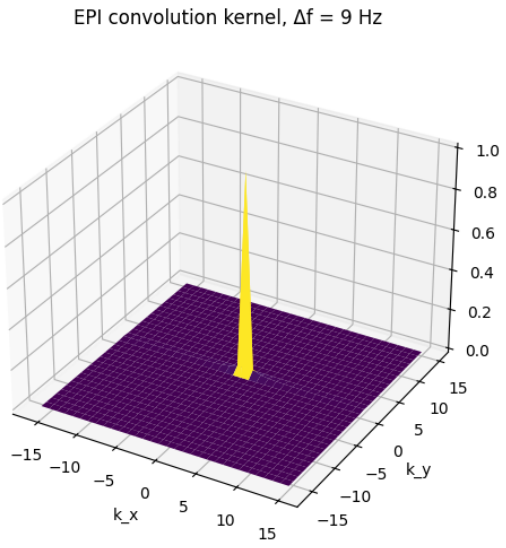
```
# Convolution kernels for off-resonance weighting

def ifft2c(x):
    # Centered 2D inverse FFT
    return np.fft.ifftshift(np.fft.ifft2(np.fft.fftshift(x)))

df = 1 / (N * tesp)  # kHz

W_Cartesian = ifft2c(np.exp(-1j * 2 * np.pi * df * upsilon_Cartesian))
W_EPI = ifft2c(np.exp(-1j * 2 * np.pi * df * upsilon_EPI))

fig = plt.figure(figsize=(12, 5))
ax1 = fig.add_subplot(1, 2, 1, projection='3d')
ax1.plot_surface(x, y, np.abs(W_Cartesian), cmap='viridis')
ax1.set_xlabel('k_x')
```

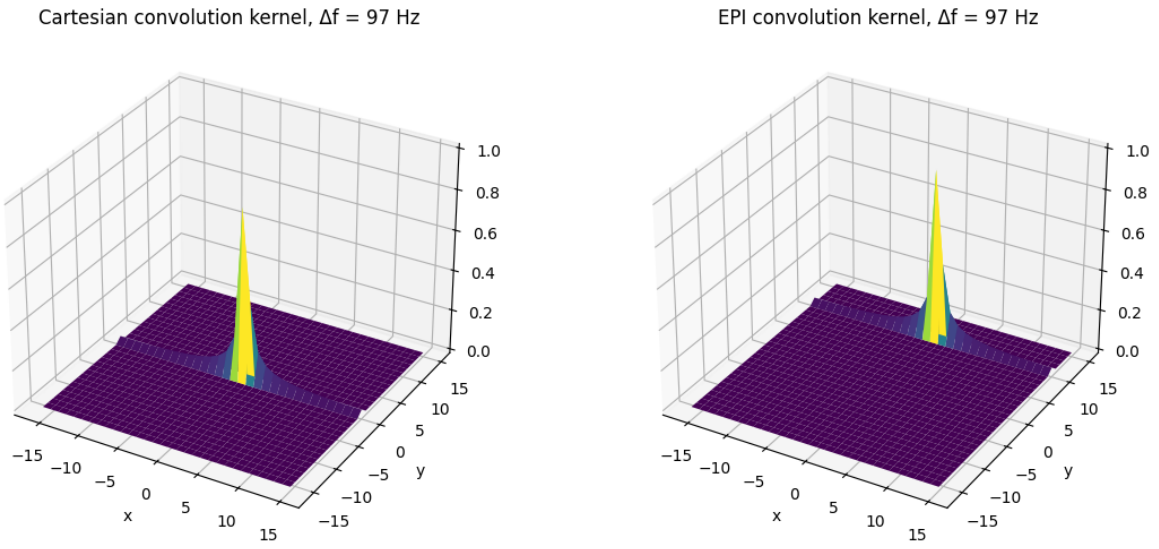
(continues on next page)

(continued from previous page)

```
ax1.set_ylabel('k_y')
ax1.set_title(f'Cartesian convolution kernel, Δf = {int(df*1e3)} Hz')
ax1.set_zlim(0, 1)

ax2 = fig.add_subplot(1, 2, 2, projection='3d')
ax2.plot_surface(x, y, np.abs(W_EPI), cmap='viridis')
ax2.set_xlabel('k_x')
ax2.set_ylabel('k_y')
ax2.set_title(f'EPI convolution kernel, Δf = {int(df*1e3)} Hz')
ax2.set_zlim(0, 1)

plt.tight_layout()
plt.show()

df = 10 / (N * tesp)  # kHz

W_Cartesian = ifft2c(np.exp(-1j * 2 * np.pi * df * upsilon_Cartesian))
W_EPI = ifft2c(np.exp(-1j * 2 * np.pi * df * upsilon_EPI))

fig = plt.figure(figsize=(12, 5))
ax1 = fig.add_subplot(1, 2, 1, projection='3d')
ax1.plot_surface(x, y, np.abs(W_Cartesian), cmap='viridis')
ax1.set_xlabel('x')
ax1.set_ylabel('y')
ax1.set_title(f'Cartesian convolution kernel, Δf = {int(df*1e3)} Hz')
ax1.set_zlim(0, 1)

ax2 = fig.add_subplot(1, 2, 2, projection='3d')
ax2.plot_surface(x, y, np.abs(W_EPI), cmap='viridis')
ax2.set_xlabel('x')
ax2.set_ylabel('y')
ax2.set_title(f'EPI convolution kernel, Δf = {int(df*1e3)} Hz')
ax2.set_zlim(0, 1)

plt.tight_layout()
plt.show()
```







For frequency shift, the main peak of the convolution kernels is shifted frfom the origin. This will result in a shift in the reconstructed image. The shift is much larger for EPI and is in the phase encoding instead of the frequency encoding direction. (The residual side lobes are due to sinc interpolation effects, similar to Gibbs ringing.)

CHAPTER

TWENTYFIVE

ARTIFACTS

This notebook includes a simulation of various MRI artifacts, as well as a high-level Artifact Comparison below. The wikipedia entry https://en.wikipedia.org/wiki/MRI_artifact is also very comprehensive.

25.1 Learning Goals

1. Identify artifacts and how to mitigate them

25.2 Introduction

Many of the artifacts that occur in MRI can be understood and analyzed using the k-space perspective. In particular, they can be understood as the k-space data being modified by some function. This is described mathematically in *MRI Signal Equation*, and expanded upon in the *Point Spread Function Analysis* section.

25.3 Artifact Comparison

MRI artifacts can generally be categorized as originating from the sample, the sequence, or the system (Source in table below). The table below provides a high-level comparison of these artifacts.

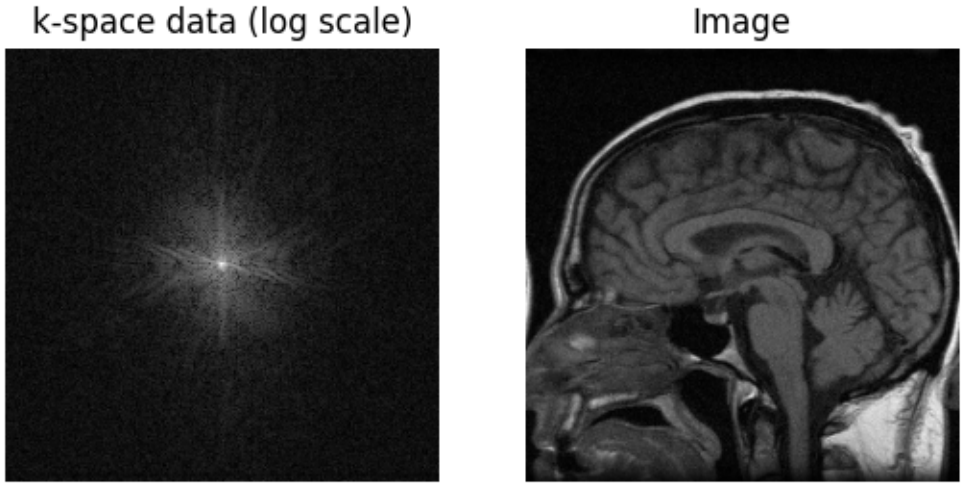
Artifact Name	Source	Appearance	What to do?	Frequency or Phase encoding
Aliasing	Sequence - FOV too small, or parallel imaging failed	Signal folds across image	Increase FOV, swap PE/FE, reacquire sensitivity maps or reduce acceleration factor (parallel imaging)	Phase encoding
Gibbs Ringing/Truncat	Sequence - due to Fourier encoding of edges	Ripples/ringing at sharp edges	Filtering of data, decrease voxel size	N/A
Boundary Artifacts/Partia Volume	Sequence and Sample - opposing phase of M_{XY} within a voxel due to inversion or chemical shift	Artificial dark lines at tissue boundaries	fat suppresion, fat/water imaging, change inversion time	N/A
Fat Displacement	Sample - chemical shift (fat)	Shifts of fat signals in image	Increase bandwidths, fat suppression, fat/water imaging, swap PE/FE	Frequency encoding (2DFT), Phase encoding (EPI)
Off-resonance Distortion	Sample - off-resonance (e.g. magnetic susceptibility differences)	Stretch and Compression in image	Increase bandwidths, shimming, swap PE/FE	Frequency encoding (2DFT), Phase encoding (EPI)
Slice Displacement	Sample - chemical shift (fat) and off-resonance (e.g. magnetic susceptibility differences)	Slice location shifts	Increase RF bandwidth, shimming, use 3D scanning	N/A
T2*	Sample - magnetic suscpetibility differences (e.g. implants, air)	Signal loss	Use spin-echoes, shorten TE, shimming	N/A
Motion/Flow	Sample - signal modulated during spatial encoding	Copies or "Ghosts" of image regions that are changing due to motion or flow	Breath-hold, gating, triggering, flow compensation, spatial saturation, swap PE/FE	Phase encoding
Spike Noise	System - intermittent unwanted signal	Specific spatial frequency on top of entire image	check RF hardware (loose connections, broken components)	N/A
RF interference/Zippe	System - unwanted frequencies received in data	Line of noisy signal on top of image	RF shielding (e.g. close door), check RF hardware, check room lights	At a specific frequency encoding location
Dielectric Shading	System - RF propogates unevenly, creating B1 inhomogeneity	Shading across entire image, particularly with large FOVs and higher B0	B1-insensitive pulse sequences (e.g. adiabatic pulses), image bias field corrections	N/A
Gradient nonlinearity	System - magnetic field gradients are not linear, especially away from isocenter	Distortion of subject, particularly near edges of large FOV	Gradient warping correction	N/A

```
# Sample k-space dat for simulated artifacts
```

(continues on next page)

(continued from previous page)

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import loadmat

# load k-space data
dataname = '../Data/se_t1_sag_data.mat'
mat = loadmat(dataname)
kdata = mat['data']
S = kdata.shape

def ifft2c(kspace):
    # Centered inverse 2D FFT
    return np.fft.ifftshift(np.fft.ifft2(np.fft.fftshift(kspace)))

im_original = ifft2c(kdata)

plt.subplot(1,2,1)
plt.imshow(np.log(np.abs(kdata) + 1e1), cmap='gray', aspect='equal')
plt.axis('off')
plt.title('k-space data (log scale)')
plt.subplot(1,2,2)
plt.imshow(np.abs(im_original), cmap='gray', aspect='equal')
plt.axis('off')
plt.title('Image')
plt.show()
```



25.4 Sampling Characteristics Artifacts - Aliasing and Gibbs Ringing

25.4.1 Aliasing

Aliasing occurs when the FOV, determined by the sample spacing chosen in k-space, is too small. (See “FOV and Resolution” section.)

Note that in MRI, aliasing only occurs in the phase encoding direction. This is because during frequency encoding a low-pass anti-aliasing filter can be applied to the analog signal prior to digitization. This truncates the bandwidth of the signal in the frequency encode direction, which is equivalent to truncating the image.

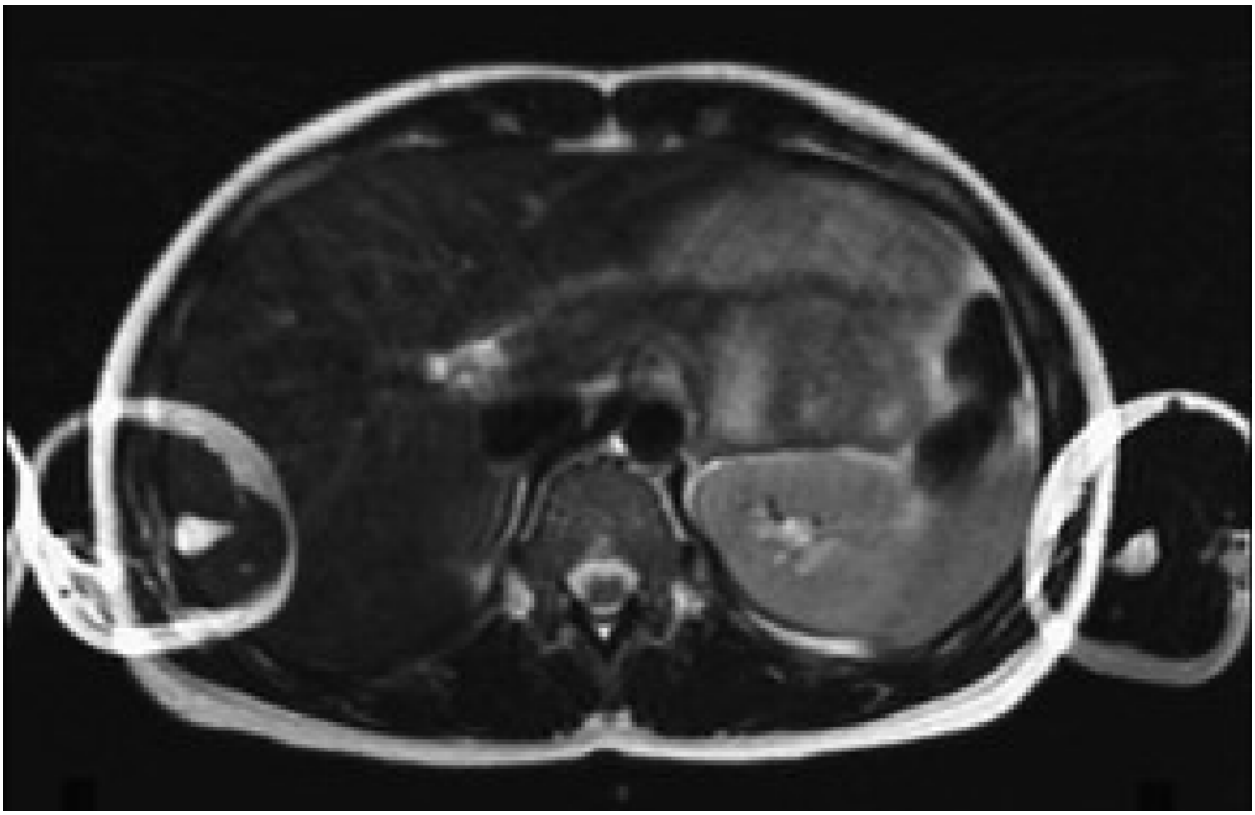
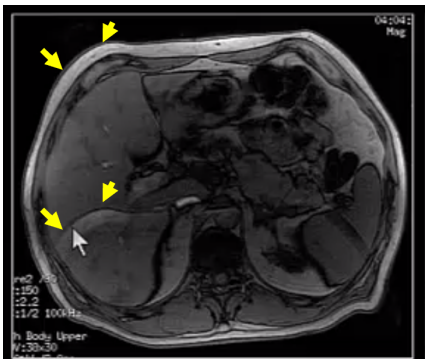


Fig. 25.1: Axial abdomen image showing aliasing artifact in the phase encoding direction (left/right)

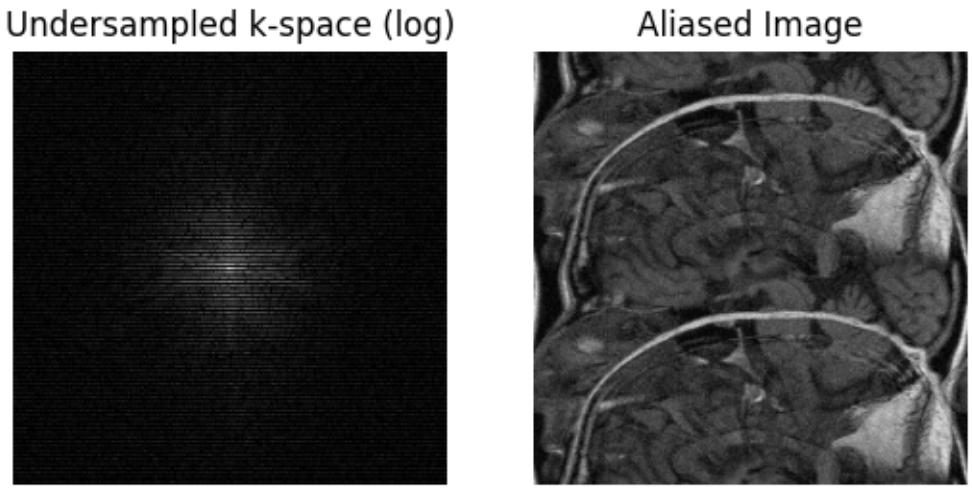
25.4.2 Aliasing in Parallel Imaging

In parallel imaging, k-space is purposefully undersampled, resulting in aliasing artifacts that are removed by using the coil sensitivities in the image reconstruction (See "Accelerated Imaging Methods" section). However, failures can lead to residual aliasing artifacts. These can be the result of inconsistencies in the coil sensitivity maps, motion, or setting the acceleration factor too high.



Axial abdomen image showing an aliasing artifact (yellow arrows) due to a parallel imaging failure

```
# Aliasing simulation
N_undersamp = 2  # >= 1
data_undersamp = np.zeros_like(kdata)
Iundersamp = np.arange(0, S[0], N_undersamp)
data_undersamp[Iundersamp, :] = kdata[Iundersamp, :]
im_undersamp = ifft2c(data_undersamp)

plt.subplot(1, 2, 1)
plt.imshow(np.log(np.abs(data_undersamp) + 1e1), cmap='gray', aspect='equal')
plt.axis('off')
plt.title('Undersampled k-space (log)')
plt.subplot(1, 2, 2)
plt.imshow(np.abs(im_undersamp), cmap='gray', aspect='equal')
plt.axis('off')
plt.title('Aliased Image')
plt.show()
```



25.4.3 Gibbs Ringing/Truncation

The so-called Gibbs ringing or truncation artifact is inherent when using Fourier encoding. These occur at sharp edges, which have an infinite content in the k-space spatial frequency domain. Since only a limited region in k-space can be acquired, i.e. k-space is truncated, the sharp edge becomes slightly blurred out and also has ripples or ringing that occur on either side of the edge.

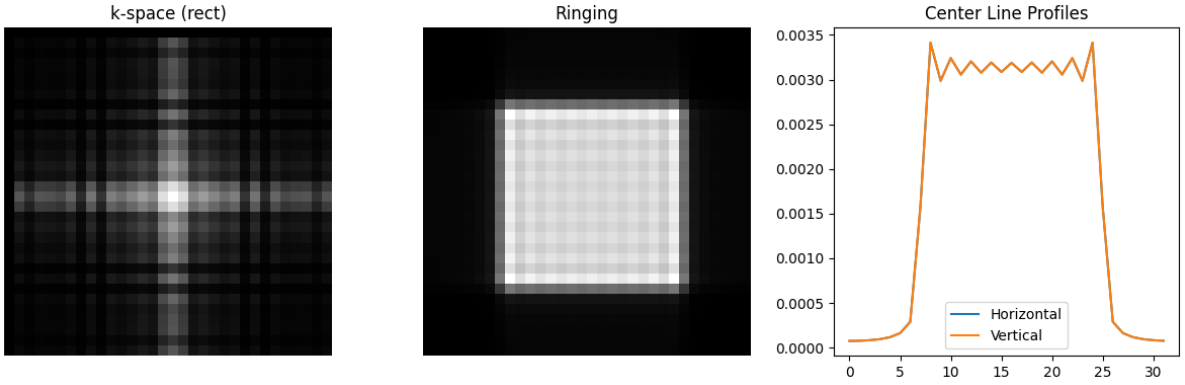
To alleviate this artifact, the k-space data can be weighted with a so-called windowing function, which applies a filter in image space. These windows smoothly decay at the edge of k-space, and have the effects of reducing ringing, but also lead to a slight loss in spatial resolution.

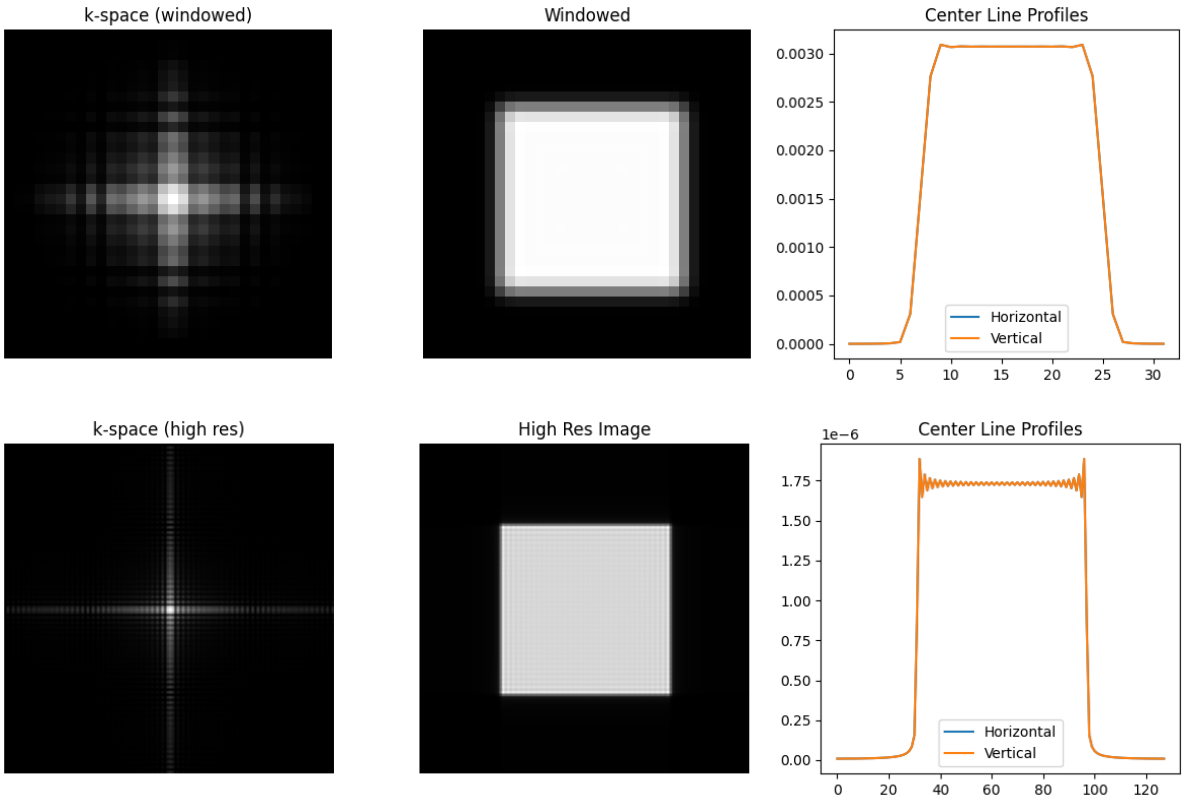
```
N = 32
kx = np.arange(-N/2, N/2) / N
N_rect = N // 2 + 2
kdata_rect = np.outer(np.sinc(kx * N_rect), np.sinc(kx * N_rect))

rect_ringing = ifft2c(kdata_rect)

plt.figure(figsize=(12, 4))
plt.subplot(1, 3, 1)
plt.imshow(np.log(np.abs(kdata_rect)+1e-2), cmap='gray', aspect='equal')
plt.axis('off')
plt.title('k-space (rect)')

plt.subplot(1, 3, 2)
plt.imshow(np.abs(rect_ringing), vmin=0, vmax=np.max(np.abs(rect_ringing)), cmap='gray
 ↪', aspect='equal')
plt.axis('off')
plt.title('Ringing')

plt.subplot(1, 3, 3)
center = N // 2
plt.plot(np.abs(rect_ringing[center, :]), label='Horizontal')
plt.plot(np.abs(rect_ringing[:, center]), label='Vertical')
plt.title('Center Line Profiles')
plt.legend()
plt.tight_layout()
plt.show()


# Apply a 2D Hanning window to k-space to reduce ringing
window = np.outer(np.hanning(N), np.hanning(N))
kdata_windowed = kdata_rect * window
rect_windowed = ifft2c(kdata_windowed)

plt.figure(figsize=(12, 4))
plt.subplot(1, 3, 1)
plt.imshow(np.log(np.abs(kdata_windowed)+1e-2), cmap='gray', aspect='equal')
plt.axis('off')
plt.title('k-space (windowed)')

plt.subplot(1, 3, 2)
plt.imshow(np.abs(rect_windowed), vmin=0, vmax=np.max(np.abs(rect_windowed)), cmap=
 ↪'gray', aspect='equal')
plt.axis('off')
plt.title('Windowed')

plt.subplot(1, 3, 3)
```

(continues on next page)

(continued from previous page)

```
plt.plot(np.abs(rect_windowed[center, :]), label='Horizontal')
plt.plot(np.abs(rect_windowed[:, center]), label='Vertical')
plt.title('Center Line Profiles')
plt.legend()
plt.tight_layout()
plt.show()

# Increase the resolution (N) to reduce ringing
N_high_res = 128
kx_high_res = np.arange(-N_high_res/2, N_high_res/2) / N_high_res
N_rect_high_res = N_high_res // 2 + 2
kdata_high_res = np.outer(np.sinc(kx_high_res * N_rect_high_res), np.sinc(kx_high_res␣
↪* N_rect_high_res))

rect_high_res = ifft2c(kdata_high_res)

plt.figure(figsize=(12, 4))
plt.subplot(1, 3, 1)
plt.imshow(np.log(np.abs(kdata_high_res) + 1e-2), cmap='gray', aspect='equal')
plt.axis('off')
plt.title('k-space (high res)')
plt.subplot(1, 3, 2)
plt.imshow(np.abs(rect_high_res), vmin=0, vmax=np.max(np.abs(rect_high_res)), cmap=
↪'gray', aspect='equal')
plt.axis('off')
plt.title('High Res Image')
plt.subplot(1, 3, 3)
plt.plot(np.abs(rect_high_res[center, :]), label='Horizontal')
plt.plot(np.abs(rect_high_res[:, center]), label='Vertical')
plt.title('Center Line Profiles')
plt.legend()
plt.tight_layout()
plt.show()
```





25.5 Displacement and Distortion Artifacts due to Changes in Frequency

Since MRI relies on frequency to encode spatial information, unknown frequency shifts can lead to disortions of the image. These frequency shifts are due to chemical shift (e.g. fat) and/or off-resonance (e.g. main field inhomogeneities, susceptibility differences).

25.5.1 Shifts in the Image

There will be a shift during frequency encoding of

$$\Delta_{FE} = \frac{\Delta f}{RBW} FOV_{FE}$$

where Δf is the frequency shift, RBW is the receiver bandwidth, and FOV_{FE} is the field of view in the frequency encoding direction.

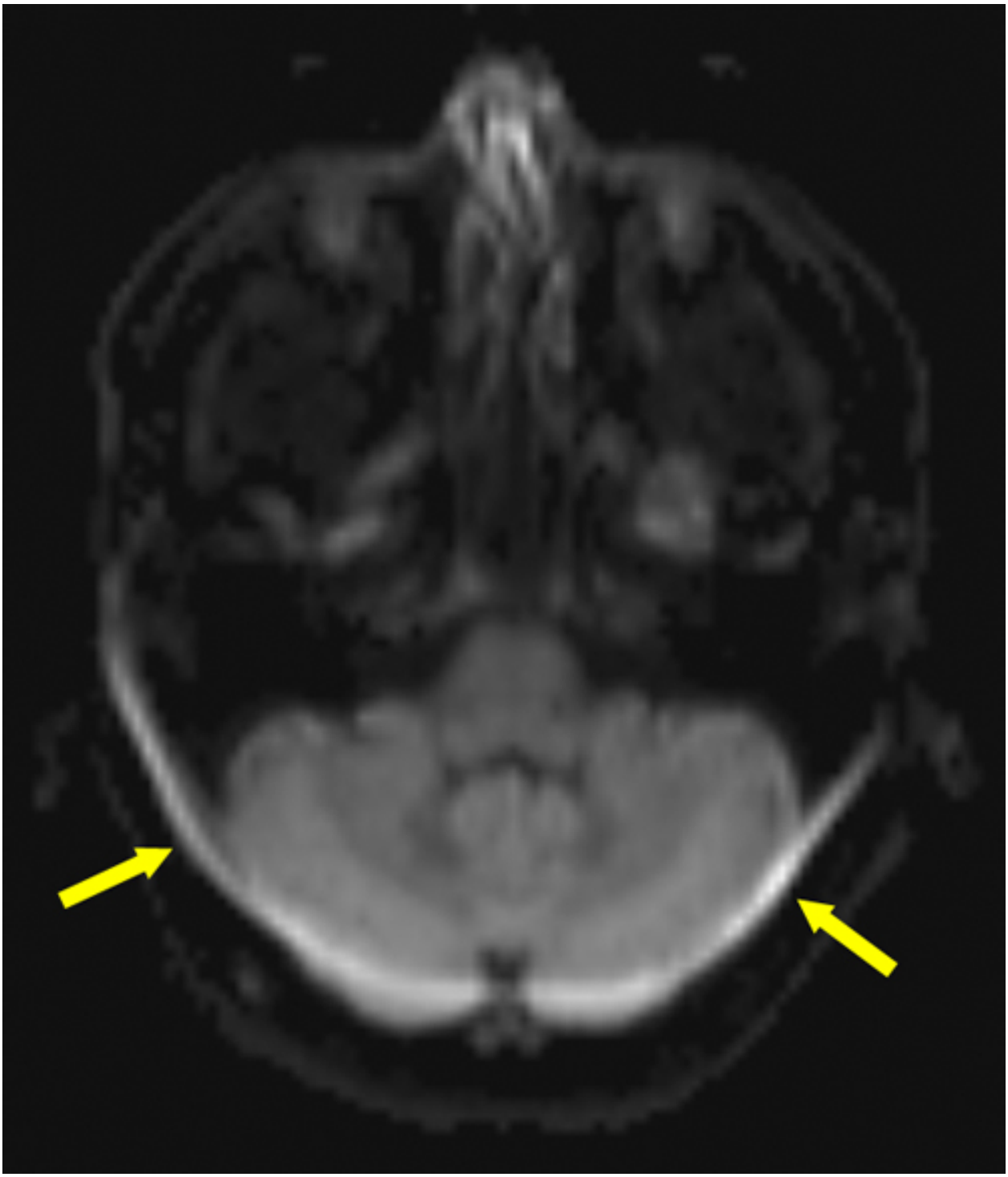
During EPI, there is typically a much larger shift in the phase encoding direction that depends on how much phase accumulates across k-space. This depends on the echo spacing, t_{esp}, and well as how many k-space lines are covered in adjacent echoes, defined here as N_{steps}. To characterize this, we can define a "phase encoding bandwidth", $BW_{PE} = N_{steps}/t_{esp}$, and the displacement will be

$$\Delta_{PE,EPI} = \frac{\Delta f}{BW_{PE}} FOV_{PE}$$

N_{steps} will depend on whether there is any interleaving, and whether parallel imaging is used to skip lines. For example, EPI without acceleration in a single-shot is $N_{steps} = 1$, 2 interleaves would have $N_{steps} = 2$, while single-shot with $R=2$ parallel imaging acceleration would have $N_{steps} = 2$, and 2 interleaves with $R=2$ parallel imaging would have $N_{steps} = 4$.

25.5.2 Fat Chemical Shift Artifacts - Displacement

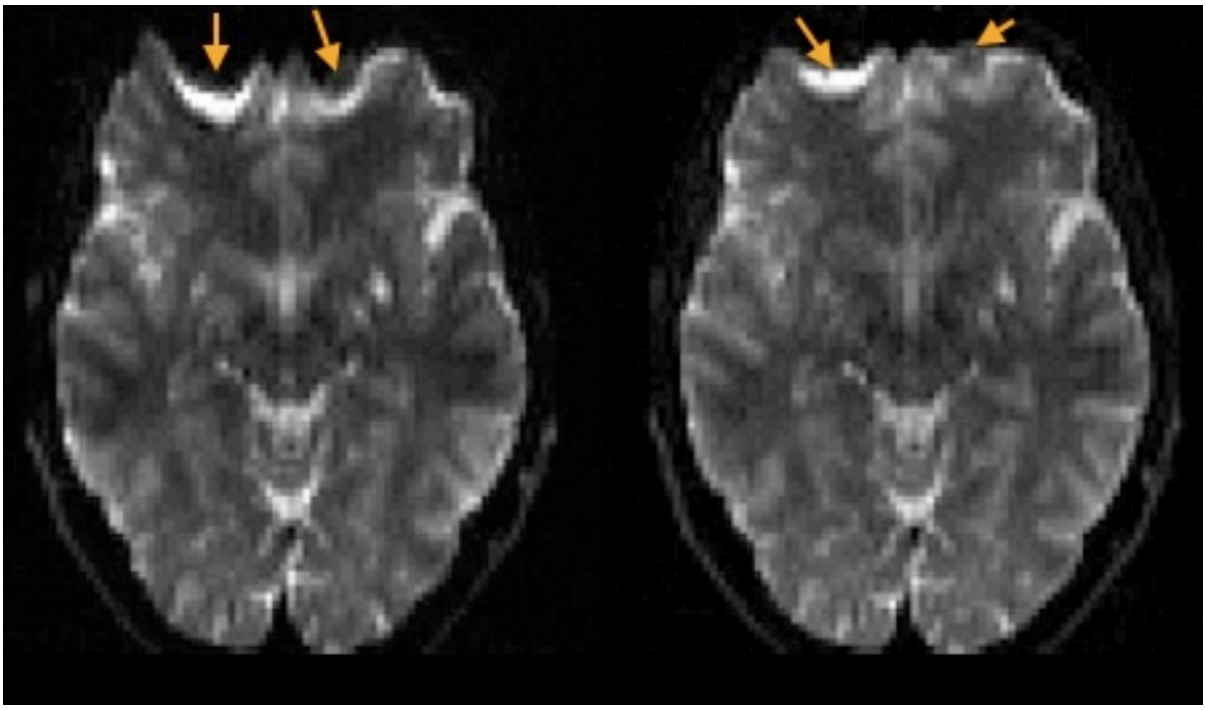
There is a constant frequency difference between fat and water. This leads to a **constant displacement** of the fat signal relative to the water signal.



Axial brain EPI image showing a fat displacement artifact (arrows) of the lipids in the skull

25.5.3 Off-resonance Artifacts - Distortion

Off-resonance, which most significantly arises from differences in magnetic susceptibility, is smoothly varying and increases at the boundaries between materials with different susceptibilities. This leads variable shifts in the image, which are the largest at these material boundaries. This is typically referred to as a **distortion** artifact, and has characteristic stretching or compression ("pile-up") of the image.



Axial brain EPI images showing distortion artifacts (arrows) near the frontal sinus where there are air-tissue interfaces. The right image used an reduced echo spacing, leading to increased phase encoding bandwidth that reduces the severity of this artifact.

25.5.4 Shifts in Slice Selection

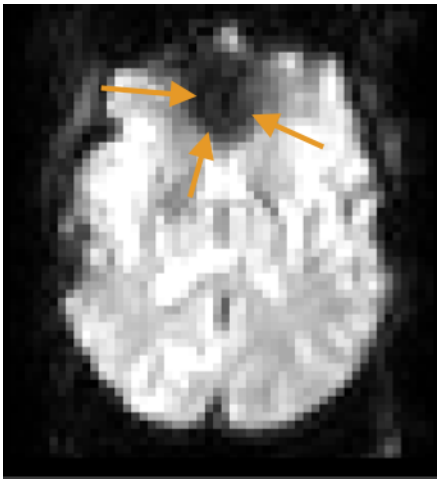
Finally, there will also be a displacement of the slice selection, and this will be

$$\Delta_{SS} = \frac{\Delta f}{BW_{rf}} \Delta z$$

where BW_{rf} is the slice select pulse bandwidth. This effect is not as obvious in images, but important to be aware of in pulse sequence design.

25.6 T2*

T2* can be a source of artifact and contrast. Macroscopic, inter-voxel, effects, such as arise from boundaries between materials with different magnetic susceptibilities, lead to shorter T2* and is typically considered an artifact. This can be removed by using spin-echo sequences.



Axial brain EPI image showing a T2 artifact (arrows) in the frontal lobe, which is due to magnetic susceptibility differences between the air in the frontal sinuses and the brain tissue.*

25.7 Motion and Flow Artifacts

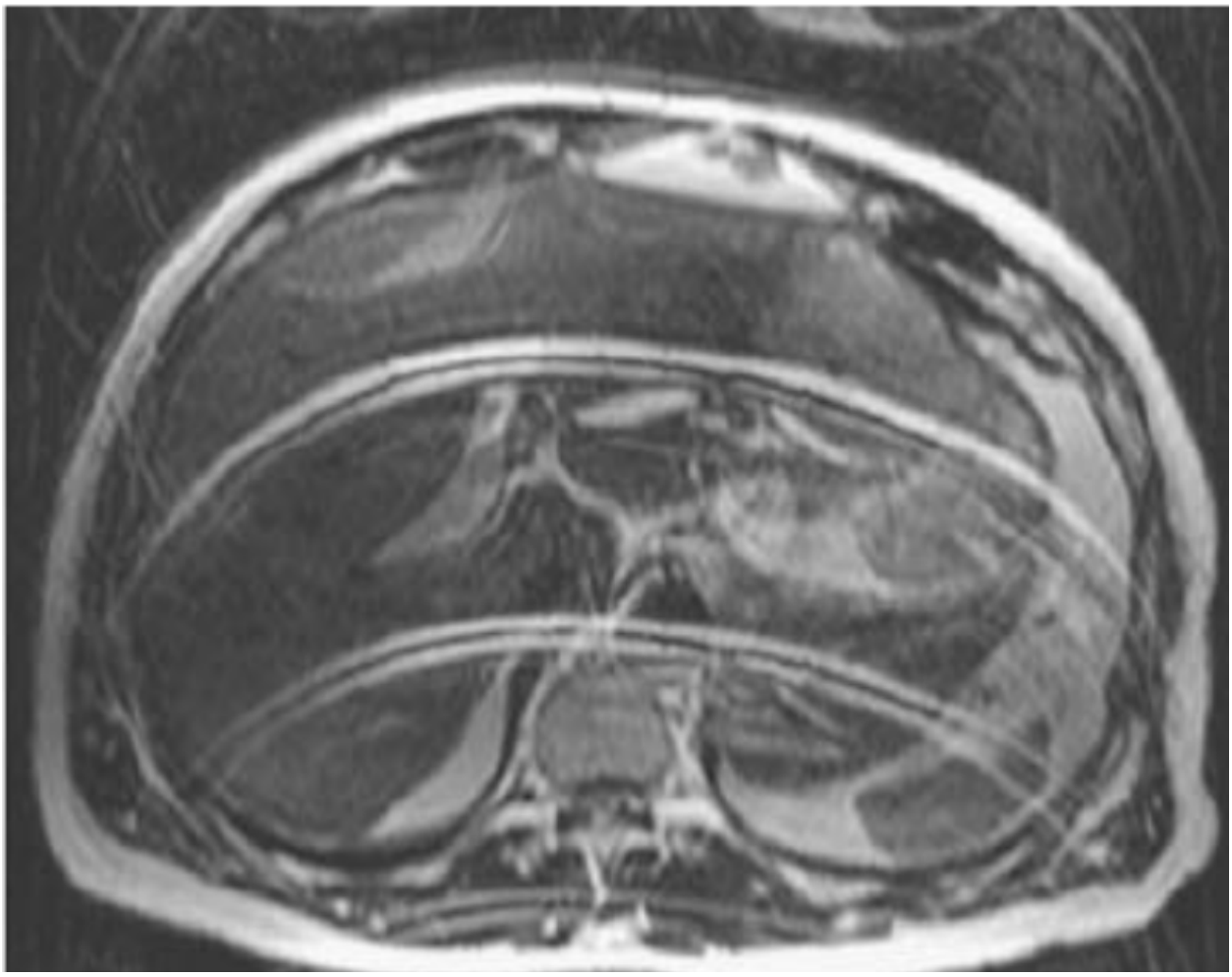
Motion, including flow, can result in artifacts in MRI if it leads to inconsistency in the data.

25.7.1 Pediodic Motion - Ghosting

Periodic motion such as breathing, heart beating, and pulsatile flow will lead to distinct ghosting artifacts in the phase encoding direction. The location of ghosting artifacts in sequential phase encoding without parallel imaging will be at predictable intervals in the phase encoding direction:

$$\Delta_{PE} = \frac{TR}{T_{motion}} FOV_{PE}$$

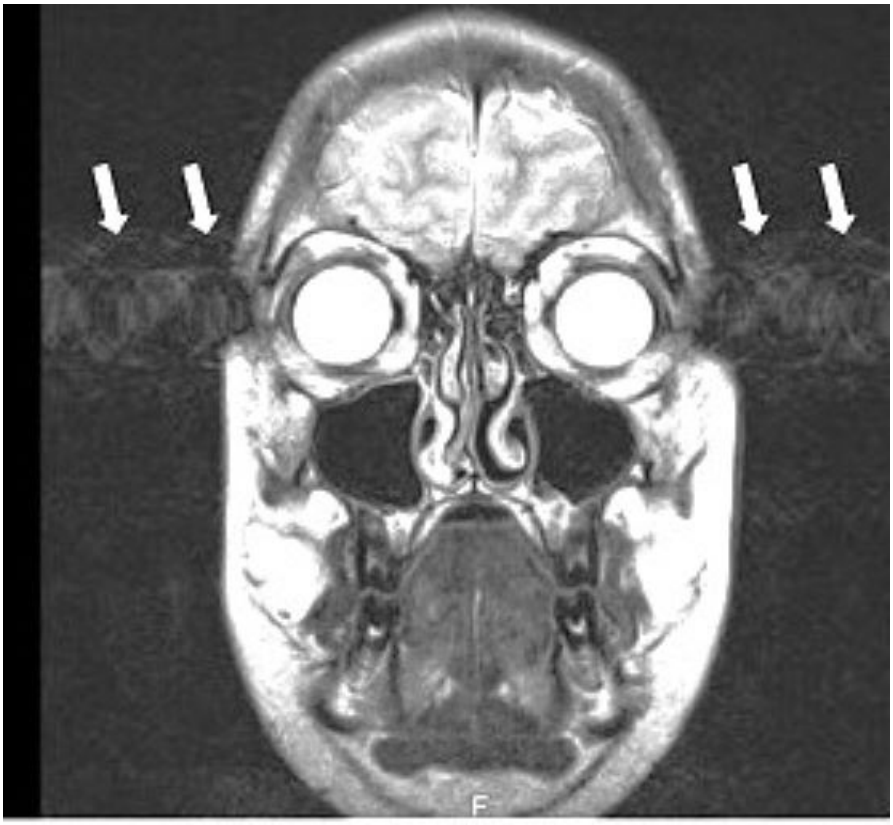
where T_{motion} is the period of the motion (e.g. $T_{motion} = 1 s$ for a heart rate of 60 beats per minute).



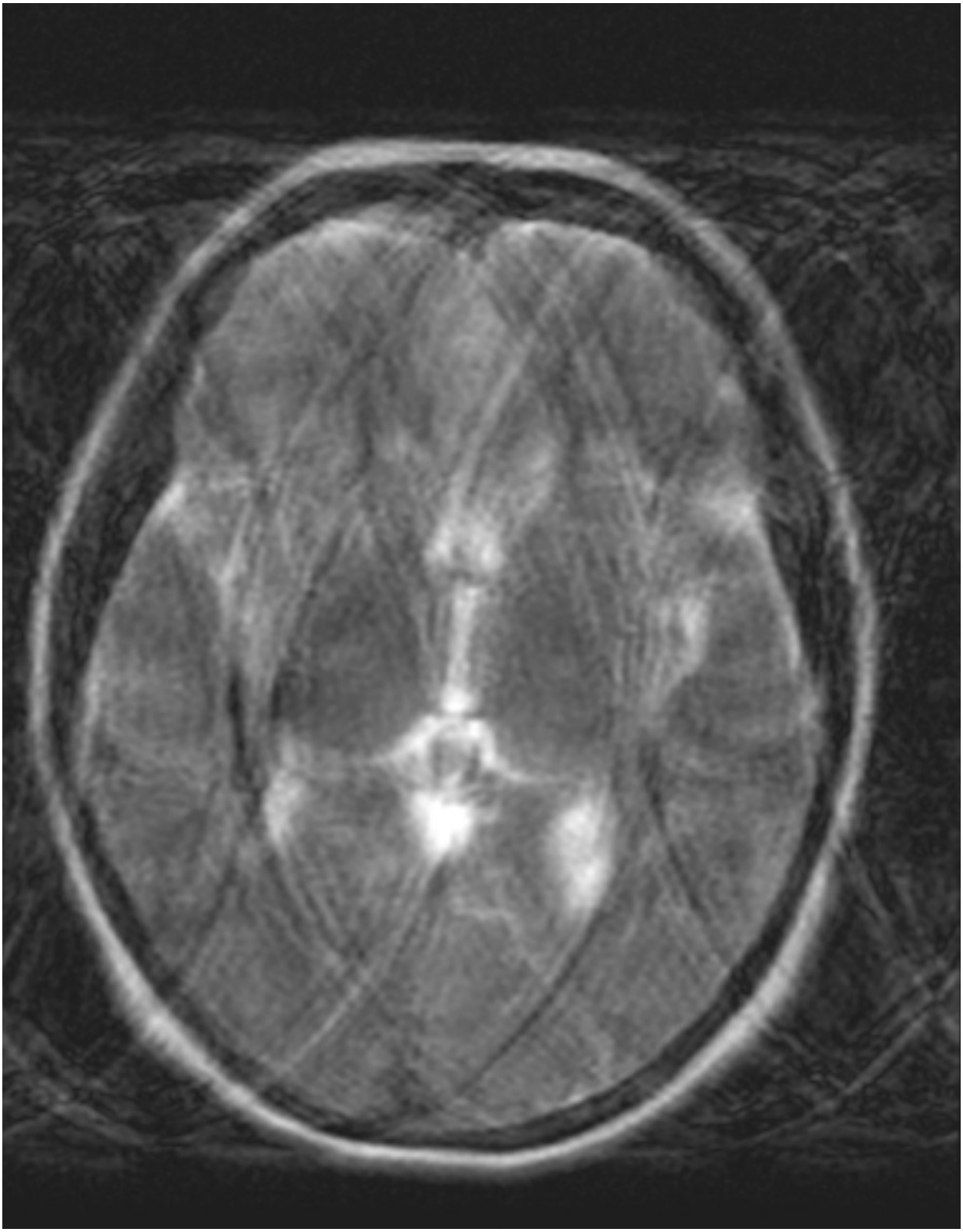
Axial abdomen image with artifacts from respiratory motion in the phase encoding direction. There are multiple distinct ghosts of the anterior abomdinal wall, which is what is moving during breathing

25.7.2 Incoherent Motion - Diffuse Ghosting

Incoherent or more random motion such as irregular breathing, arrhytthmias, coughing, bulk motion will lead to more diffuse ghosting artifacts. These have an inconsistent pattern of k-space distortion in the phase encoding direction, leading to a more diffuse appearance of the ghosting artifacts.



Coronal head image with motion artifacts (arrows) in the phase encoding direction that are the result of eye motion.



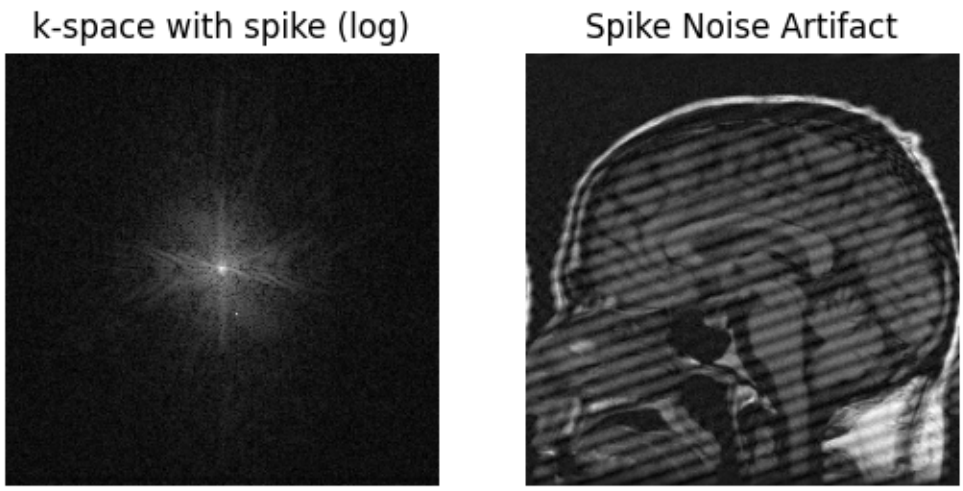
Axial brain image with diffuse ghosting motion artifacts that are the result of the whole head moving.

25.8 Unwanted signals - Spike noise, Zipper

25.8.1 Spike Noise

This artifact is due to intermittent unwanted signals that are received during the MRI acquisition. These can be caused by loose connections, broken components, or other hardware issues. The result is a spike or bright spot in the k-space data that appears as distinct spatial frequencies on top of the image.

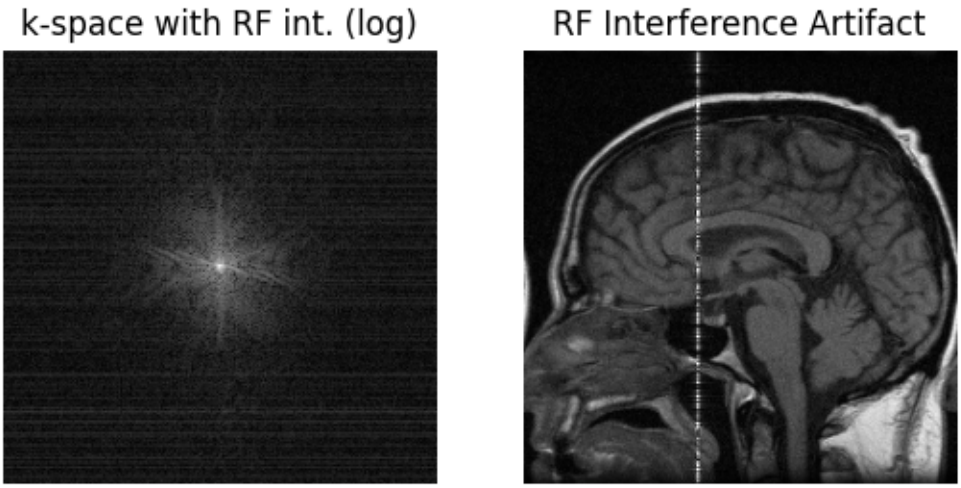
```
# Spike noise simulation
spike_location = [0.6, 0.53]

data_spike = np.zeros_like(kdata)
data_spike[round(S[0]*spike_location[0]), round(S[1]*spike_location[1])] = np.
 ↪max(kdata)/.5
im_spike = ifft2c(kdata + data_spike)

plt.subplot(1, 2, 1)
plt.imshow(np.log(np.abs(kdata + data_spike) + 1e1), cmap='gray', aspect='equal')
plt.axis('off')
plt.title('k-space with spike (log)')
plt.subplot(1, 2, 2)
plt.imshow(np.abs(im_spike), cmap='gray', aspect='equal', vmin=0, vmax=np.max(np.
 ↪abs(im_original)))
plt.axis('off')
plt.title('Spike Noise Artifact')
plt.show()
```



25.8.2 RF interference/Zipper

This artifact is due to unwanted frequencies that are received during the MRI acquisition. These can be caused by RF interference from other devices, such as lights or other electronics, or can originate from outside the MRI scan room especially if the door is not close. Receiving specific frequencies in k-space leads to lines or "zippers" in the image.

```
# RF interference - simulate amplitude modulated interference
relative_RF_frequency = 0.2
rfint = np.exp(1j * 2 * np.pi * np.arange(S[0]) / S[0] * S[0] * relative_RF_frequency␣
↪/ 2) * np.max(np.abs(kdata)) / S[0] / 3
data_rfint = (np.random.randn(S[0], 1)) * rfint  # amplitude modulation
im_rfint = ifft2c(kdata + data_rfint)

plt.subplot(1, 2, 1)
plt.imshow(np.log(np.abs(kdata + data_rfint) + 1e1), cmap='gray', aspect='equal')
plt.axis('off')
plt.title('k-space with RF int. (log)')
plt.subplot(1, 2, 2)
plt.imshow(np.abs(im_rfint), cmap='gray', aspect='equal', vmin=0, vmax=np.max(np.
↪abs(im_original)))
plt.axis('off')
plt.title('RF Interference Artifact')
plt.show()
```



25.9 MRI hardware limitations - RF shading, gradient non-linearity

25.9.1 RF Shading

The progation of radiofrequency waves in tissue as well as inhomogeneous RF coils lead to variations in intensity or shading across the image.

When transmitting RF pulses, there is some variation in the RF amplitude across the body. This is partially due to inhomogeneous transmit coils, but also is fundamentally limited by the wavelength of MRI RF waves in tissue. This wavelength is determined by the dielectric coefficient of tissue, so this is sometimes referred to as dielectric shading. The variation in RF amplitude leads to variations in flip angle that, in turn, change the resulting image intensity.

RF receive coil arrays are designed to provide high sensitivity and approximately uniform coverage. However, they arrays have varying sensitivity across the body, leading to variations in image intensity.

RF shading from these sources is typically corrected by applying a weighting to the originally reconstructed image to create more uniform image intensities.

25.9.2 Gradient Non-Linearity

Gradient coils in MRI are designed to create linear changes in the magnetic field. However, these are not perfect, particularly near the edges and closer to the open ends of the magnet bore. In these regions, the gradients are non-linear. Since MRI assumes linear gradient encoding, this non-linearity results in distortion of the images.

This is corrected for by using measurements of the magnetic fields created by the gradient coils to undo the distortions of the image.

25.10 Partial Volume and Boundary Artifacts

The concept of partial volume effects refers to when an imaging voxel contains a mixture of materials, such as different tissues, fat, fluids, or air. This occurs typically and the boundaries of organs or structures.

The strongest manifestation of this type of artifact is when then phase of the transverse magnetization, M_{XY}, varies between these materials and leads to signal loss within a voxel.

This can occur at boundaries between fat and other tissues, when the TE creates out-of-phase behavior due to the chemical shift difference of fat. This is the same principle used for *fat/water imaging*. To avoid this, in-phase TEs should be chosen.

This can also occur in Inversion Recovery sequences, where the inversion time results in the longitudinal magnetization, M_{Z}, being positive for some materials and negative for others.

Part X

Summary

CHAPTER

TWENTYSIX

KEY MRI CONCEPTS AND EQUATIONS

26.1 Background Material

MRI Math Concepts

- Linear Algebra concepts
- Complex numbers and representations
- Fourier Transforms

Signals and Systems

- Point Spread Functions

Electricity and Magnetism

- Magnetic fields

26.2 MR Spin Physics

Resonance - nuclear spins in a magnetic field precess at a frequency proportional to the magnetic field strength

$$f = \bar{\gamma} |\vec{B}|$$

Polarization - equilibrium magnetization

$$M_0(\vec{r}) = \frac{N(\vec{r}) \bar{\gamma}^2 h^2 I_Z (I_Z +1) B_0}{3 k T}$$

Net Magnetization at Equilibrium

$$\vec{M}(\vec{r},0) = \begin{bmatrix} 0 \ 0 \ M_0(\vec{r}) \end{bmatrix}$$

Excitation

- Apply magnetic field at resonant frequency to rotate net magnetization out of alignment with static magnetic field

Relaxation

$$M_{XY}(\vec{r},t) = M_{XY}(\vec{r},0) e^{-t/T_2(\vec{r})}$$

$$M_Z(\vec{r},t) = M_Z(\vec{r},0)e^{-t/T_1} + M_0(\vec{r})(1- e^{-t/T_1(\vec{r})})$$

26.3 MRI System

1. Main magnet - B_0
2. Radiofrequency (RF) coils
    - transmit RF coil - $B_1^+(\vec{r},t)$: provide homogeneous excitation
    - receive RF coil - $B_1^-(\vec{r},t)$: detect signal with high sensitivity
3. Magnetic field gradient coils - $\vec{G}(t)$

Intuition from magnetic fields and sensitivity profiles of loop coils

26.4 MRI Experiment

1. Polarization
2. Excitation
3. Signal Acquisition

- Gradients during Excitation and Acquisition for spatial encoding
- Repeat Excitation and Acquisition as needed

Experiment described by a “Pulse Sequence”

26.5 MR Contrasts

Contrast weightings

- T1-weighted - short TE, short TR
- T2-weighted - long TE, long TR
- Proton Density (PD)-weighted - short TE, long TR

spoiled GRE contrast

$$S \propto M_0 \sin(\theta) \exp(-TE/T_2) \frac{1- \exp(-TR/T_1)}{1- \cos(\theta) \exp(-TR/T_1)}$$

Ernst angle - flip angle for maximum SNR

$$\theta_{optimal} = \cos^{-1}(\exp(-TR/T_1))$$

Magnetization Preparation Methods: Inversion Recovery

$$S_{IR} \propto M_0 \exp(-TE/T_2) (1 - 2\exp(-TI/T_1) + \exp(-TR/T_1))$$

26.6 In Vivo Spin Physics

Magnetic susceptibility effects

- magnetic susceptibility is inherent property of materials
- differences in magnetic susceptibility lead to distortions of the magnetic field
- in vivo sources include: iron, oxygenated versus deoxygenated blood, air-tissue interfaces

Chemical Shift

- chemcial environment of an atom creates variations in local magnetic field
- in vivo consideration: “fat”, assumed to have a -3.5 ppm chemcial shift from water protons

26.7 In Vivo Contrasts

Phase - chemical shift and off-resonance (e.g. magnetic susceptibility effects) create phase differences in M_{XY} and the MR signal

T2*

- intra-voxel dephasing due to magnetic field inhomogeneity
- largely driven by magnetic susceptibility
- eliminate with spin-echo
- create susceptibility contrast

Fat

- fat/water imaging - separate fat and water images based on multiple echo times
- fat suppression - spectrally-selective RF pulses and/or inversion recovery

Contrast Agents

- Gadolinium (Gd)-based contrast agents - most common, primarily shortens T_1
- Iron-based contrast agents - less common, shortens T_1 but also can shorten T_2

26.8 RF Pulses

Pulse Characteristics

- flip angle

$$\theta = \gamma \int_0^{T_{rf}} b_1(\tau) d\tau $$

- Time-bandwidth product - constant for a given pulse shape

$$ TBW = T_{rf} \cdot BW_{rf} $$

- SAR

$$ SAR \propto \int_0^{T_{rf}} |b_1(\tau)|^2 d\tau $$

- pulse profile - describes the flip angle or net magnetization response across a range of frequency and/or position. This is approximately proportional to the Fourier Transform of the pulse shape for small (less than 60-degree) flip angles

Slice Selection

- Slice thickness

$$ \Delta z = \frac{BW_{rf}}{\bar{\gamma} G_{Z,SS}} $$

- Slice shifting

$$ f_{off} = \bar{\gamma} G_{Z,SS} \ z_{off} $$

26.9 Spatial Encoding

Frequency encoding - turn on gradient during data acquisition to map frequency to position

$$ x = \frac{f}{\bar \gamma G_{xr}}$$

Phase encoding - perform step-wise frequency encoding, which appears in the phase versus position of the signals. This measurement is repeated for $n = 1, \ldots, N_{PE}$

$$ \Phi(n) = \gamma (-G_{y,PE} + (n-1) G_{yi}) t_y y$$

k-space - define spatial encoding based on the cumulative sum of the gradients (i.e. gradient areas) applied after excitation

$$\vec{k}(t) = \frac{\gamma}{2\pi} \int_0^t \vec{G}(\tau) d\tau$$

- Formulates image reconstruction as an inverse Fourier Transform

$$s(t) = \mathcal{F}{m(\vec{r}) } |_{\vec{k} = \vec{k}(t)} = M(\vec{k}(t))$$

- describes all MRI acquisitions including frequency and phase encoding
- effects of gradients can be refocused
- supports 2D and 3D imaging

26.10 Image Characteristics

$$SNR \propto f_{seq}\ \mathrm{Voxel\ Volume}\ \sqrt{T_{meas}}$$

$$ FOV = \frac{1}{\Delta k}$$

$$ \delta = \frac{1}{2 k_{max}}$$

Scan Time

$$ T_{scan} = TR \cdot N_{TR} \cdot NEX$$

where N_{TR} is the total number of TR periods sampled. Including fast sequences and acceleration methods:

$$N_{TR} = \frac{N_{PE,total}}{ETL \cdot R}$$

26.11 FT Imaging Sequence

Typical acquisition uses frequency and phase encoding

See *Pulse Sequence* for a typical 2D gradient-echo sequence

Can convert between sequence parameters (e.g. timings, gradient amplitudes) and the FOV, resolution and scan time, as well as predict relative SNR

26.12 Fast Imaging Pulse Sequences

Volumetric coverage

- 2D multislice imaging - interleave multiple slices within a single TR
- 3D imaging - cover 3D k-space with 2 phase-encoding dimensions

Echo-planar imaging (EPI)

- k-space trajectory that covers multiple k-space lines per excitation
- Echo spacing (t_{esp}), echo train length (ETL)
- echo time, $TE = TE_{eff}$, defined when data closest to center of k-space is acquired

Multiple Spin-echo imaging (FSE/TSE/RARE)

- multiple spin-echoes per excitation used to acquire different k-space lines
- Echo spacing (t_{esp}), echo train length (ETL)
- echo time, $TE = TE_{eff}$, defined when data closest to center of k-space is acquired. Used to create different contrasts

Gradient Echo methods

- Contrast can be changed based on whether transverse magnetization is available or refocused in a subsequent TR
- Variations based on whether RF and/or gradient spoiling are used
- Magnetization Prepared gradient-echo imaging methods

26.13 Accelerated Imaging Methods

Partial Fourier

- Why does it work? MRI approximately satisfies conjugate symmetry property of k-space data
- How does it work? Only sample slightly more than half of k-space

Parallel Imaging - k-space undersampling

- Why does it work? RF coil arrays with different elements provide spatial encoding
- How does it work? Skip k-space data in the direction(s) that have variation in RF coil element sensitivity profiles
- Key variations: May require measurement of coil sensitivity maps, also autocalibrated methods
- SNR considerations: Geometry factor (g-factor, $g>1$) due noise amplification in parallel imaging reconstruction, $SNR \propto \frac{1}{g}$

Parallel Imaging - Simultaneous Multi-slice

- Why does it work? RF coil arrays with different elements provide spatial encoding
- How does it work? Excite multiple slices simultaneously
- SNR considerations: Geometry factor (g-factor, $g>1$) due noise amplification in parallel imaging reconstruction, $SNR \propto \frac{1}{g}$

Compressed Sensing

- Why does it work? MRI data has typical patterns that can be predicted are represented by sparse coefficients
- How does it work? Skip k-space data with a pseudo-random pattern. Define a sparsity domain

Deep Learning Reconstruction

- Why does it work? MRI data has typical patterns that can be learned
- How does it work? Skip k-space data. Train a neural network using a large MRI dataset.

26.14 Artifacts

See *Artifacts* for high-level comparison

CHAPTER

TWENTYSEVEN

MRI QUESTIONS

Note that questions may have multiple possible answers.

27.1 MRI System

Clinical MRI systems (i.e. 3T and 1.5T) operate in the same frequency range as ..

- Xray
- Visible light
- FM
- AM

The main magnetic field, B_0, is for …

- polarization
- excitation
- acquisition

Which coil(s) generates a magnetic field(s) that are perpendicular to the main field B_0?

- RF coils
- gradient coils in the x direction
- gradient coils in the y direction
- gradient coils in the z direction

(T/F) In MRI, the imaging plane is determined by proper assignment on the x, y, and z gradients

Match the following components with the approximate amplitude of their magnetic field

- Components: B_0, RF transmit coil, RF receive coil, magnetic field gradients
- frequency of magnetic field: 0 Hz, 1 kHz, 100 MHz
- amplitude of magnetic field: 1 μT , 10 μT, 10 $m T$, 1 $ T$

Which of the following steps are conducted in a calibration scan?

- Center frequency / B0 calibration
- RF power, transmit gain
- RF receiver gain
- Gradient calibration

27.2 MR Physics

Which of the following isotopes cannot be imaged with MRI?

- ^1H
- ^{12}C
- ^{13}C
- ^{19}F
- ^2H

Human tissue is …

- diamagnetic
- paramagnetic
- ferromagnetic

(T/F) When placed in a magnetic field, all the protons ($^1 H$) in the body will line up with the field.

(T/F) When placed in a magnetic field, spins immediately are preferentially aligned with that field.

(T/F) A spin's precession frequency (Larmor frequency) is proportional to the magnetic field it's in.

(T/F) The net magnetization, $\vec{M}$, does not precess at equilibirum.

To create signals in MRI, we need to apply an RF pulse that is …

- perpendicular to the main field B_{0}, oscillating at frequency 0 Hz (static)
- parallel to the main field B_{0}, oscillating at frequency 0 Hz (static)
- parallel to the main field B_{0}, oscillating at a frequency γB_0
- perpendicular to the main field B_{0}, oscillating at a frequency γB_0

(T/F) The z-axis is the same in lab and rotating frame.

27.3 RF Coils

(T/F) The RF receive coils detect magnetic flux directly when collecting signals.

What are the benefit(s) of using phased array receive coils?

- Increase FOV
- Increase SNR
- Enables parallel imaging acceleration
- Enables compressed sensing acceleration

27.4 Contrast

(T/F) Proton density represents the density of all the protons in a given tissue.

The recovery of longitudinal magnetization is proportional to …

- e^{t/T_1}
- $1- e^{t/T_1}$
- e^{t/T_2}
- $1-e^{t/T_2}$

The decay of the transverse magnetization is proportional to …

- e^{t/T_1}
- $1- e^{t/T_1}$
- e^{t/T_2}
- $1-e^{t/T_2}$

(T/F) The amplitude of the net magnetization doesn't change during relaxation.

Which of the following equations is correct?

- $T_2 \ge T_2^* \ge T_1$
- $T_2^* \ge T_2 \ge T_1$
- $T_1 \ge T_2 \ge T_2^*$
- $T_1 \ge T_2^* \ge T_2$

T2* relaxation depends on …

- magnetic field inhomogeneities
- spin-spin interactions
- spin-lattice interactions
- longitudinal relaxation

(T/F) M_Z doesn't contribute to the MR signal.

Calculate the signal in a spin-echo sequence with a 90-degree flip angle $M_0(1-e^{-TR/T1})e^{-TE/T2}$ when TR = ∞

- M_0
- 0
- $M_0(1-e^{-TR/T1})$
- $M_0 e^{-TE/T2}$

Calculate the signal in a spin-echo sequence with a 90-degree flip angle $M_0(1-e^{-TR/T1})e^{-TE/T2}$ when TE = ∞

- M_0
- 0
- $M_0(1-e^{-TR/T1})$
- $M_0 e^{-TE/T2}$

A longer TR …

- increases T1 weighting
- reduces T1 weighting
- increases T2 weighting
- reduces T2 weighting

A longer TE …

- increases T1 weighting
- reduces T1 weighting
- increases T2 weighting
- reduces T2 weighting

1. What flip angle gives the highest SNR for a spoiled gradient echo pulse sequence?
    - 45-degrees
    - 90-degrees
    - 180-degrees
    - $\cos^{-1} (\exp(-TR/T_1))$

The signal of an inversion recovery pulse sequence in the steady-state is proportional to …

- $M_{0}\cdot(1-2e^{-TI/T1})$
- $M_{0}\cdot(1-e^{-TR/T1})$
- $M_{0}\cdot(1-e^{-TI/T1}+e^{-TR/T1})$
- $M_{0}\cdot(1-2e^{-TI/T1}+e^{-TR/T1})$

(T/F) In a spin-echo sequence, the dephasing of spins in the transverse plane is only eliminated at the spin-echo.

Which of the following image contrasts can be generated by a spin-echo pulse sequence?

- PDw
- T1w
- T2w
- T2*w

What magnetic resonance property is used to perform fat/water (Dixon) imaging?

- Proton density
- T1
- T2
- T2*
- Chemical Shift

What are the minimum measurements required to create separate fat and water images?

- In-phase TE
- Out-of-phase TE
- In-phase TE & Out-of-phase TE

What is “magnetization preparation” used for?

- polarization
- data acquisition
- create additoinal contrast
- tissue suppression

(T/F) Multiple readouts can be used following a magnetization preparation pulse to improve efficiency

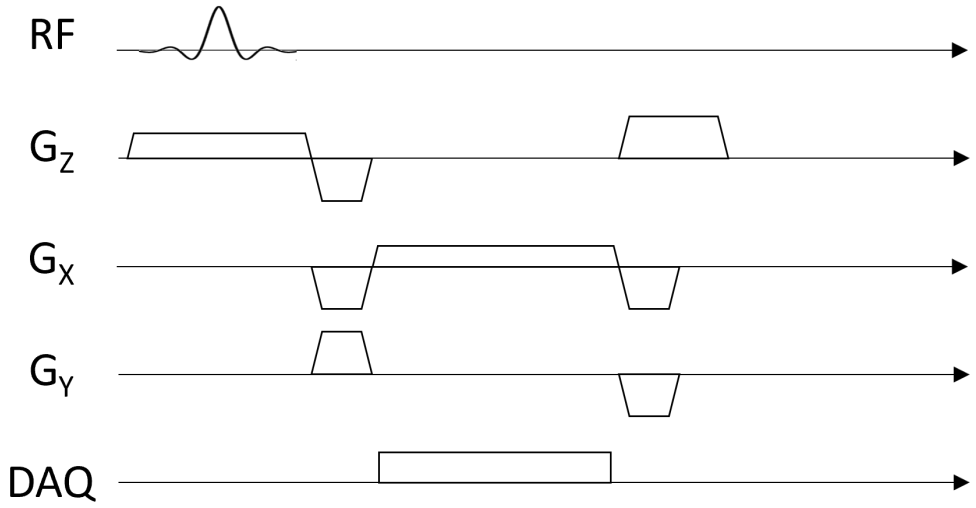
27.5 Pulse Sequence

Which of the following are typically components of a pulse sequence diagram?

- RF Pulses
- Main field (B_0)
- Magnetic field gradients (G_X, G_Y, G_Z)
- Data acquisition (DAQ)

In a typical pulse sequence, identify the gradients that serve the following functions:

1. spoil transverse magnetization
2. refocus Mxy phase across the slice
3. move to the edge of k-space



Which of the following statement is true for the slice select refocusing gradient?

- The slice select refocusing gradient must have the same gradient amplitude with the slice selective excitation pulse.
- The slice select refocusing gradient must have the same gradient area with the slice selective excitation pulse.
- The slice select refocusing gradient can overlap with the prewinder of the frequency encoding gradient.
- The slice select refocusing gradient can overlap with the gradient echo.

(T/F) For 2D FT imaging, gradient spoiling is usually applied in the slice select direction because the voxel size is larger thus more dephasing can happen within a voxel in that direction.

(T/F) For a GRE or SE with no phase encoding gradient, the k-space position at TE equals 0 (i.e, $\vec{k}(TE) = 0$).

27.6 RF Pulses

What is the flip angle θ (in radians) of a constant amplitude (hard) RF pulse with an amplitude $B_{1,0}$ and duration τ?

- $\theta = \gamma B_{10} \tau$
- $\theta = \gamma B_{0} \tau$
- $\theta = B_{10} \tau$
- $\theta = \gamma \tau$

(T/F) When describing the RF excitation, we usually neglect the relaxation terms in the Bloch equations because the RF pulse duration is usually much shorter than the relaxation time constants.

(T/F) A constant amplitude 180° pulse that is the same duration as a constant amplitude 90° pulse requires twice the power.

In slice-selective excitation, thinner slices can be achieved by

- decreasing the RF pulse bandwidth, BW_{RF}
- decreasing the receive BW
- decreasing the slice-select gradient strength
- increasing the slice-select gradient strength

27.7 Spatial Encoding

(T/F) During frequency encoding, the MRI signal is in the time domain, corresponding to the spatial frequency domain. The Fourier Transform of the signal is in the frequency domain, which corresponds to the image domain.

When is slice selective gradient turned on?

- during the RF excitaiton pulse
- between the RF excitation and the readout
- during the echo (readout)

When is the frequency encoding gradient turned on?

- during the RF excitaiton pulse
- between the RF excitation and the readout
- during the echo (readout)

When is phase encoding gradient turned on?

- during the RF excitaiton pulse
- between the RF excitation and the readout
- during the echo (readout)

When is DAQ turned on during a 2D FT sequence?

- When the slice selective gradient is on
- When the RF is on
- When the phase encoding gradient is on

- When the frequency encoding gradient is on

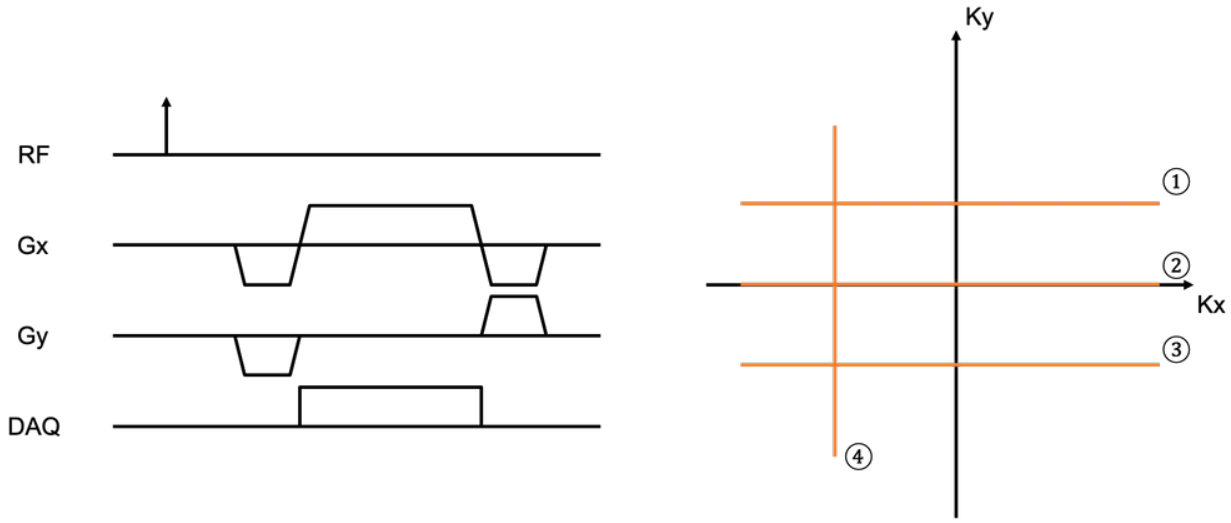
According to the Nyquist theorem, to avoid aliasing:

- The sampling frequency must be at least half the highest frequency in the signal
- The sampling frequency must be at least twice the lowest frequency in the signal
- The sampling frequency must be at least twice the highest frequency in the signal
- The sampling frequency must be at least half the lowest frequency in the signal

(T/F) Aliasing occurs because of oversampling.

(T/F) During 2D multislice imaging, acquisition for one slice must be completed before the acquisition for the next slice can start.

Which k-space line is acquired by the following magnetic field gradients?



In 3DFT imaging, the gradient added to the slice encoding axis (compared to a 2DFT) is a …

- frequency encoding gradient
- phase encoding gradient
- either frequency or phase encoding gradient

27.8 Image Reconstruction

For a real-valued image $m(x,y)$, which of the following equation holds for its k-space data $M(k_x,k_y)$?

- $M(k_x,k_y) = M(-k_x,-k_y)$
- $M(k_x,k_y) = -M(-k_x,-k_y)$
- $\mathcal{Real}{M(k_x,k_y)} = \mathcal{Real}{ M(-k_x,-k_y) }$
- $\mathcal{Imag}{M(k_x,k_y) } = \mathcal{Imag}{M(-k_x,-k_y) }$

(T/F) In MRI we only look at the magnitude images.

(T/F) The center of k-space always contains the maximum signal.

27.9 Image Characteristics (FOV and Resolution)

The field of view is inversely proportional to…

- receiver BW (RBW)
- readout gradient strength (Gxr)
- TR
- TE

The field of view is directly proportional to…

- receiver BW (RBW)
- readout gradient strength (Gxr)
- TR
- TE

(T/F) An anti-aliasing filter can be applied in the phase encoding direction.

The resolution in frequency encoding direction (δ_x) is equal to …

- $ \frac{1}{W_{kx}} $
- $ \frac{1}{\frac{\gamma}{2\pi} G_{xr} t_{read}} $
- $ \frac{FOV_x}{N_{FE}} $
- None of the above

27.10 SNR

SNR can be increased by …

- decreasing voxel size
- increasing total time
- increasing NEX

27.11 Artifacts

Chemical shift displacement artifact is characterized by …

- signal stretch and pile-up
- bright and dark bands
- Gibbs ringing
- ghosting

Susceptibility displacement artifact is characterized by …

- signal stretch and pile-up
- bright and dark bands
- Gibbs ringing

- ghosting

Truncation artifacts can be reduced by …

- improving resolution
- filtering in k-space
- increasing FOV
- increasing NEX

(T/F) Motion artifacts occur only along the phase-encoding direction.

27.12 Fast Imaging Pulse Sequences

What does it mean to use a multiple spin echo pulse sequence?

- multiple spin echoes are created following a single excitation pulse
- multiple k-space lines acquired sequentially
- multiple gradient-echo repetitions after a magnetization preparation pulse
- fully refocused gradients and no spoiling in every TR

What types of contrast can be created with a multiple spin echo pulse sequence?

- proton density weighted
- T1 weighted
- T2 weighted
- T2* weighted

What are the limitations of multiple spin echo pulse sequences?

- Chemical shift and susceptibility displacement artifacts
- T2 blurring artifacts
- ghosting
- SAR

What does it mean to use echo planar imaging (EPI)?

- multiple spin echoes are created following a single excitation pulse
- multiple k-space lines acquired sequentially
- multiple gradient-echo repetitions after a magnetization preparation pulse
- fully refocused gradients and no spoiling in every TR

What are the advantages of EPI?

- Create additional T1, T2, and/or T2* contrast
- Rapidly acquire k-space
- Robust to motion
- Repeated refocusing of intravoxel dephasing

What are the artifacts associated with EPI?

- Chemical shift displacement
- Distortion due to magnetic susceptibility differences
- T2* blurring
- ghosting

What does it mean to use balanced steady-state free-precession (bSSFP)?

- multiple spin echoes are created following a single excitation pulse
- multiple k-space lines acquired sequentially
- multiple gradient-echo repetitions after a magnetization preparation pulse
- fully refocused gradients and no spoiling in every TR

27.13 Accelerated Imaging Methods

Match the acceleration methods

- Partial Fourier
- Parallel Imaging
- Compressed Sensing
- Deep Learning

with the following concept they rely on:

- conjugate symmetry in k-space
- spatial encoding from receive coil arrays
- a sparse representation of the image
- training on prior images to learn expected patterns

(T/R) Partial Fourier, parallel imaging, and compressed sensing or deep learning reconstructions can be used simultaneously.

How is coil sensitivity information gathered for parallel imaging?

- It is stored in a database on the scanner
- A separate scan to measure coil sensitivity maps
- Using fully-sampled data from the center of k-space
- Using fully-sampled data from outer k-space

What is the “g-factor” in parallel imaging?

- How much faster scan can be performed
- Describes noise amplification
- Describes SNR loss
- Describes magnetic field gradients

Compared to the SNR a fully sampled acquistion (SNR_{full}), the SNR Of a parallel imaging acquisition (SNR_{PI}) with an acceleration factor, R, and g-factor is

- $SNR_{PI} = SNR_{full}$

- $SNR_{PI} = SNR_{full} / \sqrt{R}$
- $SNR_{PI} = SNR_{full} / g(\vec{r}) $
- $SNR_{PI} = SNR_{full} / (g(\vec{r}) \sqrt{R})$

(T/F) Parallel imaging undersampling can be performed in any direction, regardless of the RF coil configuration.

Simultaneous multi-slice parallel imaging

- enables acceleration in the slice direction
- requires no modifications to the pulse sequence
- requires RF pulses that excite multiple slices
- requires coil sensitivity information

What type of k-space sampling is required for compressed sensing?

- full sampling
- equally spaced undersampling
- equally spaced undersampling with fully-sampled center of k-space
- pseudo-random undersampling

At least how many training datasets are typically required to develop deep learning MRI reconstruction methods?

- 1
- 10-100
- 1000-10,000
- 100,000-1,000,000

Generalization problems can arise in deep learning MRI reconstruction methods when applied to situations that area different from the training data in which of the following ways:

- different anatomy
- different contrasts
- different sampling patterns
- different B0

Which type of architecture is commonly used for physics-based deep learning MRI reconstruction networks?

- Encoder-decoder
- Unet
- Unrolled
- Recurrant

Part XI

Reference Material

CHAPTER

TWENTYEIGHT

MRI NOTATION AND TERMINOLOGY

The following is the standardized notation used in this book:

Notation	Definition	SI Units
$\vec{r} = [x, y, z]^T$	position in space	[m]
$\vec{B}(\vec{r},t)$	Magnetic field	[T]
$\bar{\gamma}$	gyromagnetic ratio	[Hz/T]
B_0	Main magnetic field	[T]
$B_1^+(\vec{r},t)$	RF transmit magnetic field	[T]
$B_1^-(\vec{r},t)$	RF receive magnetic field	[T]
$\vec{G}(t) = [G_X(t), G_Y(t), G_Z(t)]^T $	magnetic field gradients	[T/m]
$\vec{M}(\vec{r},t)$	Net Magnetization	[T]
$M_{XY}(\vec{r}, t) = M_X(\vec{r}, t) + i M_Y(\vec{r}, t)$	transverse magnetization	[T]
$M_0(\vec{r})$	Equilibrium magnetization	[T]
$T_1(\vec{r}), T_2(\vec{r}), T_2^*(\vec{r})$	relaxation rates	[s]

28.1 Definitions

28.1.1 Position in space

$$\vec{r} = \begin{bmatrix} x\ y \ z \end{bmatrix}$$

28.1.2 Net magnetization

$$\vec{M} = \begin{bmatrix} M_X \ M_Y \ M_Z \end{bmatrix}$$

It can vary over space, $\vec{r} = [x,y,z]^T$ and time, t:

$$\vec{M}(\vec{r},t) = \begin{bmatrix} M_X(\vec{r},t) \ M_Y(\vec{r},t) \ M_Z(\vec{r},t) \end{bmatrix}$$

Shorthand for transverse magnetization using complex numbers

$$M_{XY}(\vec{r},t) = M_X(\vec{r},t) + i M_Y(\vec{r},t)$$

28.1.3 Magnetic field

$$\vec{B} = \begin{bmatrix} B_X \ B_Y \ B_Z \end{bmatrix}$$

It can vary over space, $\vec{r}$ and time, t:

$$\vec{B}(\vec{r},t) = \begin{bmatrix} B_X(\vec{r},t) \ B_Y(\vec{r},t) \ B_Z(\vec{r},t) \end{bmatrix}$$

28.2 Acronyms

MRI is full of jargon and acronyms, which can be intimidating and impede understanding. Luckily, there are a couple good resources to help you out:

Radiopaedia - general list: https://radiopaedia.org/articles/mri-pulse-sequence-abbreviations

MR-TIP - structured lists comparing acronyms across vendors

- https://mr-tip.com/serv1.php?type=cam
- https://mr-tip.com/serv1.php?type=cam&sub=1

28.3 Glossary

Spins - Nuclei with non-zero spin so they have a net magnetic moment

Polarization - Alignment of spins in a magnetic field, resulting in a net magnetization

RF - Radiofrequency

Gradients - Magnetic field gradients

Excitation - Flipping the net magnetization into the transverse plane with an RF pulse

Refocusing - A 180 degree RF pulse used to refocus the effects of off-resonance in spin-echo sequences

Voxel - Volume element

CHAPTER

TWENTYNINE

MRI MATH CONCEPTS

29.1 Complex Numbers

$$c = a + i b = A\exp(i \phi)$$ $$ i = \sqrt{-1}, , \exp(i \phi) = \cos \phi + i \sin\phi$$ $$ |c| = A = \sqrt{a^2+b^2}$$ $$ \phi = \tan^{-1} (b/a)$$

29.2 Impulse Function

The impulse (or delta) funcion, $\delta(x)$, can be used to represent sampling of a continuous signal, and has the following properties:

- $\delta(x) = 0$ for all $x \ne 0$
- $\delta(x) \rightarrow \infty$ at $x =0$
- $\int_{-\infty}^\infty \delta(x) dx = 1$
- $\delta(ax) = \frac{1}{|a|} \delta(x)$
- $f(x) \delta(x) = f(0) \delta(x)$
- $f(x) \ast \delta(x) = f(x)$

29.2.1 Impulse train sampling function

To represent discrete sampling, a train of impulse functions is used. This is known as Dirac Comb function, also the sha, impulse train, or sampling function:

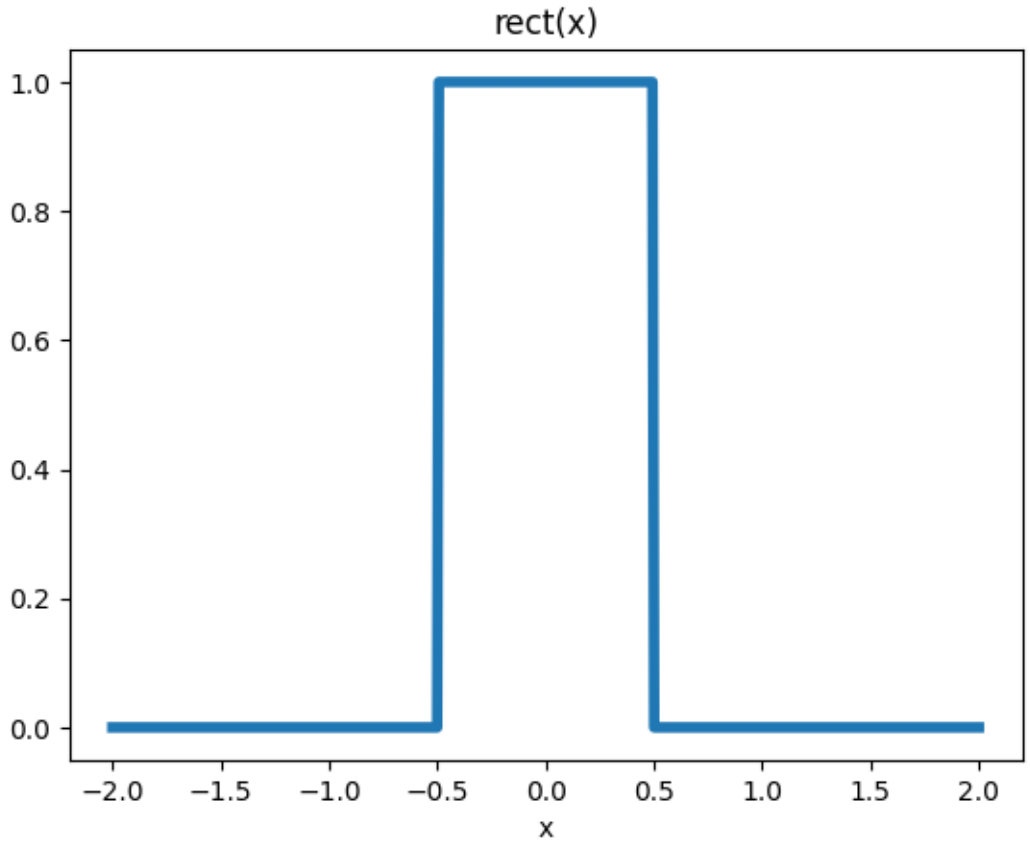
$$Ш(x) = \sum_{n = -\infty}^\infty \delta(x - n)$$

29.3 Other Functions

Rectangle function

$$\mathrm{rect}(x) = \sqcap(x) = \left{ \begin{array}{l l} 1 & \quad |x| < 1/2 \ 1/2 & \quad |x| = 1/2 \ 0 & \quad |x| > 1/2 \ \end{array} \right. $$

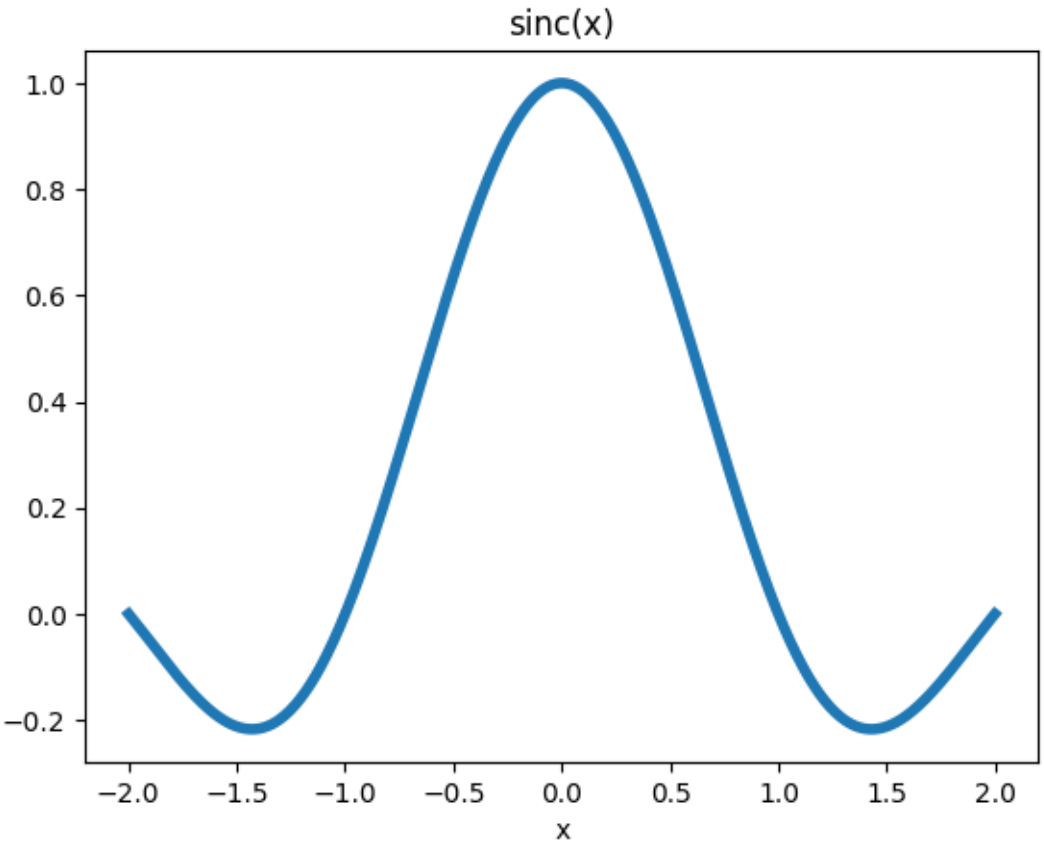
```
import numpy as np

import matplotlib.pyplot as plt

x = np.linspace(-2, 2, 500)
rect = np.ones_like(x)
rect[np.abs(x) > 0.5] = 0
rect[np.abs(x) == 0.5] = 0.5

plt.plot(x, rect, linewidth=4)
plt.xlabel('x')
plt.title('rect(x)')
plt.show()
```



Sinc (normalized)

$$\mathrm{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$$

```
plt.plot(x, np.sinc(x), linewidth=4)
plt.xlabel('x')
plt.title('sinc(x)')
plt.show()
```

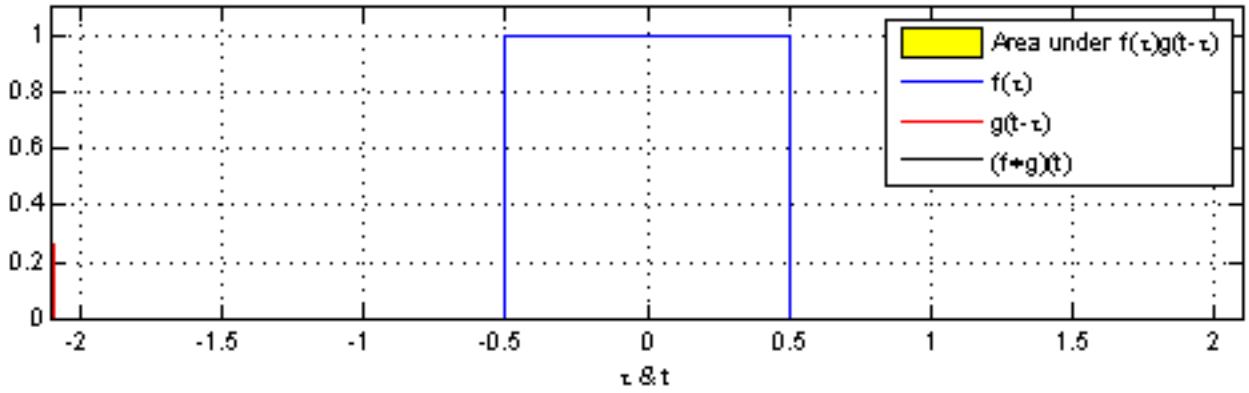


29.4 Convolution

The convolution operation $\ast$, is defined as:

$$ g(x) \ast h(x) = \int_{-\infty}^\infty g(x-\tau) h(\tau) d\tau$$

The intuition of convolution is that one of functions is reversed, slid in the convolution dimension relative to the other function, and the area of the functions multiplied at each location is the output, which is nicely illustrated in this example from https://en.wikipedia.org/wiki/Convolution



29.5 Fourier Transforms

The Fourier transform allows for determination of the frequencies present in the original function. The Fourier transform (FT) operation, $ \mathcal{F} { \cdot }$, of a complex-valued function, $f(x)$, is:

$$ \mathcal{F} { f(x) } = F(k_x) = \int_{-\infty}^\infty f(x) \exp(-i 2 \pi k_x x) dx$$

(Unitary, ordinary frequency)

Note that the pair of variables used - x used to represent space and k_x used to represent spatial frequency, can be swapped for other variables. Most commonly the other variables are t used to represent time and f used to represent frequency.

29.5.1 Inverse Fourier Transform:

$$ \mathcal{F}^{-1} { F(k_x) } = f(x) = \int_{-\infty}^\infty F(k_x) \exp(+i 2 \pi x k_x) dk_x$$

29.5.2 2-D Fourier Transform

$$ F(k_x,k_y) = \int_{-\infty}^\infty \int_{-\infty}^\infty f(x,y) \exp(-i 2 \pi (k_x x + k_y y)) dx dy$$

29.5.3 N-D Fourier Transform

$$ F(\vec{k}) = \int_{-\infty}^\infty f(\vec{x}) \exp(-i 2 \pi \vec{k}\cdot \vec{x}) d\vec{x}$$

29.5.4 Duality

$$\mathcal{F}{f(x)} = F(k_x)$$ $$\mathcal{F}{F(x)} = f(-k_x)$$ $$\mathcal{F}{F(-x)} = f(k_x)$$

29.5.5 Separable

If $f(x,y) = f_x(x) f_y(y)$, then $\mathcal{F}{f(x,y)} = \mathcal{F}{f_x(x)} \mathcal{F}{f_y(y)}$

29.6 Fourier Transform Identities and Pairs

For $\mathcal{F}{f(x)} = F(k_x), \mathcal{F}{g(x)} = G(k_x)$:

Function	Fourier Transform
$\delta(x)$	1
1	$\delta(k_x)$
$Ш(x)$	$Ш(k_x)$
$\mathrm{rect}(x)$	$\mathrm{sinc}(k_x)$
$f(ax)$	$\frac{1}{\|a\|} F(\frac{k_x}{a})$
$a \cdot f(x) + b \cdot g(x)$	$a \cdot F(k_x) + b \cdot G(k_x)$
$f(x-a)$	$\exp(-i 2\pi a k_x) F(k_x)$
$\exp(i 2\pi a x) f(x)$	$F(k_x - a)$
$\delta(x-a)$	$\exp(-i 2\pi a k_x) $
$\exp(i 2\pi a x) $	$\delta(k_x - a)$
$f(x) \ast g(x)$	$F(k_x) G(k_x)$
$f(x) g(x)$	$F(k_x) \ast G(k_x)$

29.7 Fourier Transform Example - Truncated k-space data

Based on the identities above, we can derive that the Fourier Transform of a sinc() function is a rect() function. This is relevant for MRI as we can use a rect() function to represent finite sampling of the data:

$$\mathcal{F}{\mathrm{sinc(x)}} = \mathrm{rect}(k_x)$$

In MRI, we sample data in the spatial frequency domain, or k-space, in this case we are trying to measure $F(k_x)$. However, we can only sample a limited amount of k-space data. This can be modeled by multiplying our kspace data by a rect function with a width W_{kx}:

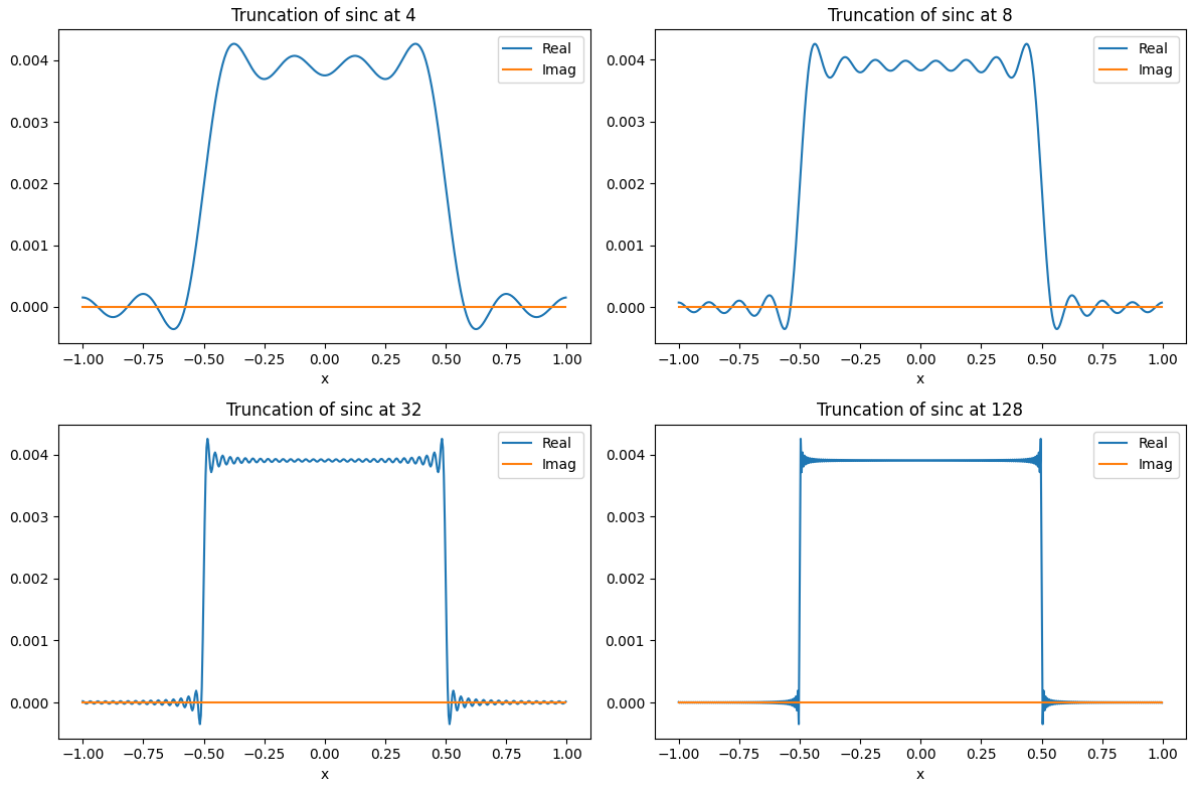
$$ \hat{F}(k_x) = F(k_x) \mathrm{rect(k_x/W_{kx})}$$

According the Fourier theory, the resulting image will be convolved with a sinc function

$$ \hat{f}(x) = \mathcal{F}^{-1}{\hat{F}(k_x)} = f(x) \ast W_{kx} \mathrm{sinc}(W_{kx} x) $$

This convolution most notably leads to “ringing” artifacts near sharp edges.
The example below uses the Fourier Transform to illustrate the ringing artifacts from this necessary “truncation” of k-space data.

```
N = 512
kxmax = 128
kx = np.linspace(-N/2, N/2-1, N) * (2 * kxmax) / N
x = np.linspace(-N/2, N/2-1, N) / (2 * kxmax)

fig, axs = plt.subplots(2, 2, figsize=(12, 8))
kxmax_truncate_list = [4, 8, 32, 128]
for idx, kxmax_truncate in enumerate(kxmax_truncate_list):
    F = np.sinc(kx)
    F[np.abs(kx) > kxmax_truncate] = 0

    f = np.fft.fftshift(np.fft.ifft(np.fft.ifftshift(F)))
    ax = axs[idx // 2, idx % 2]
    ax.plot(x, np.real(f), label='Real')
    ax.plot(x, np.imag(f), label='Imag')
    ax.set_title(f'Truncation of sinc at {kxmax_truncate}')
    ax.set_xlabel('x')
    ax.legend()
plt.tight_layout()
plt.show()
```



29.8 Point Spread function (PSF) Analysis

Point spread function (PSF) analysis is a powerful tool for characterizing linear systems, including MRI. The basic process of PSF analysis is to input an ideal "point" object, represented by a delta function, into the system, here denoted by $\mathcal{H}()$, and observe the resulting output, which is the PSF.

$$ psf(x) = \mathcal{H}(\delta(x)) $$

In MRI, PSF analysis is often performed starting in the spatial frequency, or k-space, domain, since this is where data is sampled and this is where we typically need to characterize the MRI system. Since the Fourier Transform of the input point object is $\mathcal{f}(\delta(x)) = 1$, then the Fourier Transform of the PSF is simply

$$ PSF(k_x) = \mathcal{F}{psf(x)} $$

Fourier Transform identities then show that, when imaging a real object, $f(x)$, it will be convolved by the point spread function in the actual image reconstructed, $\hat{f}(x)$:

$$ \hat{f}(x) = \mathcal{F}^{-1}{\hat{F}(k_x)} = \mathcal{F}^{-1}{F(k_x) PSF(k_x)} = f(x) \ast psf(x) $$

29.9 Discrete Fourier Transform

The discrete Fourier transform (DFT) is used to perform Fourier analysis of the frequencies present in a signal from discretely sampled data. It transforms a discrete set of values, $f[x_n], x_n = 0, 1, 2, … N-1 $, into their frequency coefficients, $F[k_n], k_n = 0, 1, 2, … N-1 $ with the following formula.

$$F[k_n] = \sum_{x_n=0}^{N-1} f[x_n] \exp(-i 2 \pi k_n x_n/N)$$

The formulation is extremely similar to the Fourier transform, and thus many of the Fourier transform identities and pairs still apply when using the DFT. It is also extended similarly into multiple dimensions.

Practically, the DFT is computed using the Fast Fourier Transform (FFT) algorithm.

The DFT can be analyzed starting from the (continuous) FT by incorporating the effects of sampling using the impulse train function, $Ш(x)$.

CHAPTER

THIRTY

MRI SOFTWARE RESOURCES

This is just a small sample of MRI software resources available! I've selected a few of my favorites that are focused on education and general pulse sequence/reconstruction methods.

https://github.com/kmjohnson3/Intro-to-MRI

- Python Jupyter notebooks introducting MRI concepts, including examples of advanced reconstructions (variational networks, compressed sensing) and distortion artifacts

https://github.com/mribri999/MRSignalsSeqs

- MRI signals and systems course code and lectures, and builds of introductory MRI courses

https://github.com/markus-nilsson/mri-simulator

- Simulator in MATLAB for Educational Purposes with nice net magnetization illustrations

https://github.com/mikgroup/sigpy

- Python package for signal processing, with powerful MRI reconstruction and MRI pulse design tools

https://github.com/mrirecon/bart

- Popular MRI Reconstruction toolbox with many features for basic and advanced reconstructions

https://github.com/qMRLab/qMRLab

- quantitative MRI processing methods, from T1 and T2 fitting to susceptibility mapping, diffusion and magnetization transfer

https://github.com/ISMRM/mrhub

- Hub for the entire range of MRI software resources available. Worth browsing especially when delving into more advanced topics

Part XII

Advanced and Fun Topics

CHAPTER

THIRTYONE

EXCITATION K-SPACE AND MULTIDIMENSIONAL RF PULSES

RF pulse analysis and design can be performed in the excitation k-space framework, which can be used for all types of pulses. It is particularly useful for multi-dimensional and spectral-spatial RF pulses.

31.1 Learning Goals

1. Describe how images are formed
    - Understand the theory of excitation k-space
    - Describe how excitation k-space is used for RF pulse design and analysis
    - Describe multidimensional and spectral-spatial RF pulses

31.2 Excitation k-space

Excitation k-space is similar to acquisition k-space in many ways, so many intuitions, theoretical properties, and gradient designs can be applied to both. Acquisition k-space describes the measured spatial frequencies of the object being imaged, while excitation k-space describes the excited spatial frequencies. They are defined as

$$\vec{k}_{acq}(t) = \frac{\gamma}{2\pi} \int_0^t \vec{G}(\tau) d\tau$$

$$\vec{k}*{ex}(t) = -*frac{\gamma}{2\pi}* *int_t^{t*{end}} \vec{G}(\tau) d\tau$$

As is shown, they are inverted in several ways from each other. Acquisition k-space is defined by the accumulated gradient area, whereas excitation k-space is defined by the remaining gradient area. Acquisition k-space always starts at the center of k-space, while excitation k-space always ends at the center of k-space.

The RF profile is approximately proportional to Fourier Transform of the RF weighting that is created in excitation k-space. This holds true under the small-tip approximation. Thus, the profile can be controlled by the RF pulse shape and gradients that create the weighting in excitation k-space.

To use this principle, excitation k-space, the desired profile (e.g. rect) can be Fourier Transformed to create the desired weighting (e.g. sinc). The RF pulse shape is this weighting, modulated by the k-space sampling density. This last step is analogous to density compensation in non-Cartesian MRI reconstruction methods.

The excitation k-space trajectory defines limitations of the resulting excitation profile, analogously to acquisition k-space:

- Resolution - the resolution or sharpness of the excitation profile is determined by the maximum extent of the excitation k-space trajectory.
- Field-of-view (FOV) - the FOV of the excitation profile is determined by the sample spacing of the excitation k-space trajectory. Beyond this FOV, there will be aliasing of excitation profile.

31.3 Multidimensional spatially-selective Pulses

To create multidimensional spatially-selective pulses, first a k-space trajectory and a desired profile are chosen. Then the RF pulse shape is designed based on the desired profile (using Fourier Transform with small-tip approximation) and the density of the k-space trajectory. For example, 2D pulses are commonly designed with a spiral or echo-planar imaging (EPI) gradient waveforms. The spacing of the spiral arms and EPI lines determine the excitation FOV.

Some use cases of 2D pulses are fast navigator acquisitions and reduced FOV acquisitions.

The major limitations when using these pulses are pulse duration required to traverse 2D k-space, off-resonance distortions that result during these long pulses, and gradient fidelity that can lead to distortions of the weighting in excitation k-space.

31.4 Spectral k-space

The concept of "Spectral k-space" can be used to determine the spectral or off-resonance profile of RF pulses. This is particularly useful for spectral-spatial RF pulses, but can be applied to any RF pulse.

Spectral k-space, k_f, defines the accumulation of encoding of the spectral or off-resonance domain. This information simply accumulates over time, so the spectral k-space trajectory is linear with time.

In acquisitions, the accumulation of information in spectral k-space is equivalent to the so-called free-induction decay readout in time used in MR spectroscopy.

In excitations, spectral accumulates during an RF pulse. RF pulses without a gradient are spectrally-selective pulses, and can be interpreted as traveling in spectral excitation k-space. Considering spectral k-space, slice-selective RF pulses travel on an angled trajectory in spectral-spatial k-space. This angle leads to the artifact of chemical shift or off-resonance slice displacement.

31.5 Spectral-Spatial RF Pulses

Spectral-spatial RF pulses aim to provide both

- spectral selection, typically to only excite protons in water molecules, and not protons in lipids
- spatial selection, for slice-selective imaging These are particularly common in fMRI and diffusion with echo-planar imaging (EPI) in order to eliminate chemical shift displacement artifacts in the image, and are also used for water and/or fat suppression in MR spectroscopy.

These pulses can be designed by considering the spectral-spatial excitation k-space trajectory, where the RF energy deposited can be Fourier Transformed to reveal the approximate spectral-spatial profile created. The gradient applied traverses spatial excitation k-space, while time traverses the spectral excitation k-space dimension.

The most common approach is to use a repeated gradient and RF shape, a so-called sub-pulse, which is repeated multiple times but with modulations to create the spectral k-space weighting. The design then consists of With this approximation, we can formulate the goal of the RF pulse design is to create and combine two pulses:

- A spectral pulse filter, $H_f(k_f)$, that should be applied in k_f direction to create a profile $h_f(f)$
- A spatial pulse filter, $H_z(k_z)$, that should be applied in the k_z direction to create a profile of $h_z(z)$ The resulting spectral-spatial profile is:

$$ h(f,z) = h_f(f) h_z(z)$$

The pulse is created by repeating the spatial sub-pulse, which is modulated by the spectral pulse filter over time in the repeated sub-pulses to create a spectral-spatial profile.

The major limitations when using these pulses are pulse duration required to traverse spectral-spatial k-space, small frequency FOVs that are limited by the subpulse switching rate, and gradient fidelity that can lead to distortions of the weighting in excitation k-space.

CHAPTER

THIRTYTWO

SPECTRAL-SPATIAL RF PULSES

Spectral-spatial RF pulses aim to provide both

- spectral selection, typically to only excite protons in water molecules, and not protons in lipids
- spatial selection, for slice-selective imaging

These are particularly common in fMRI and diffusion with echo-planar imaging (EPI) in order to eliminate chemical shift displacement artifacts in the image, and are also used for water and/or fat suppression in MR spectroscopy.

32.1 Learning Goals

1. Describe how images are formed
    - Understand how spectral-spatial RF pulses work
2. Identify artifacts and how to mitigate them
    - Determine when using spectral-spatial RF pulses would help remove chemical shift related artifacts

32.2 Spectral-Spatial RF Pulse Design

This is done by creating a 2D excitation profile in both the spectral and spatial dimensions. This is most easily interpreted through excitation k-space, where the RF energy deposited can be Fourier Transformed to reveal the approximate spectral-spatial profile created.

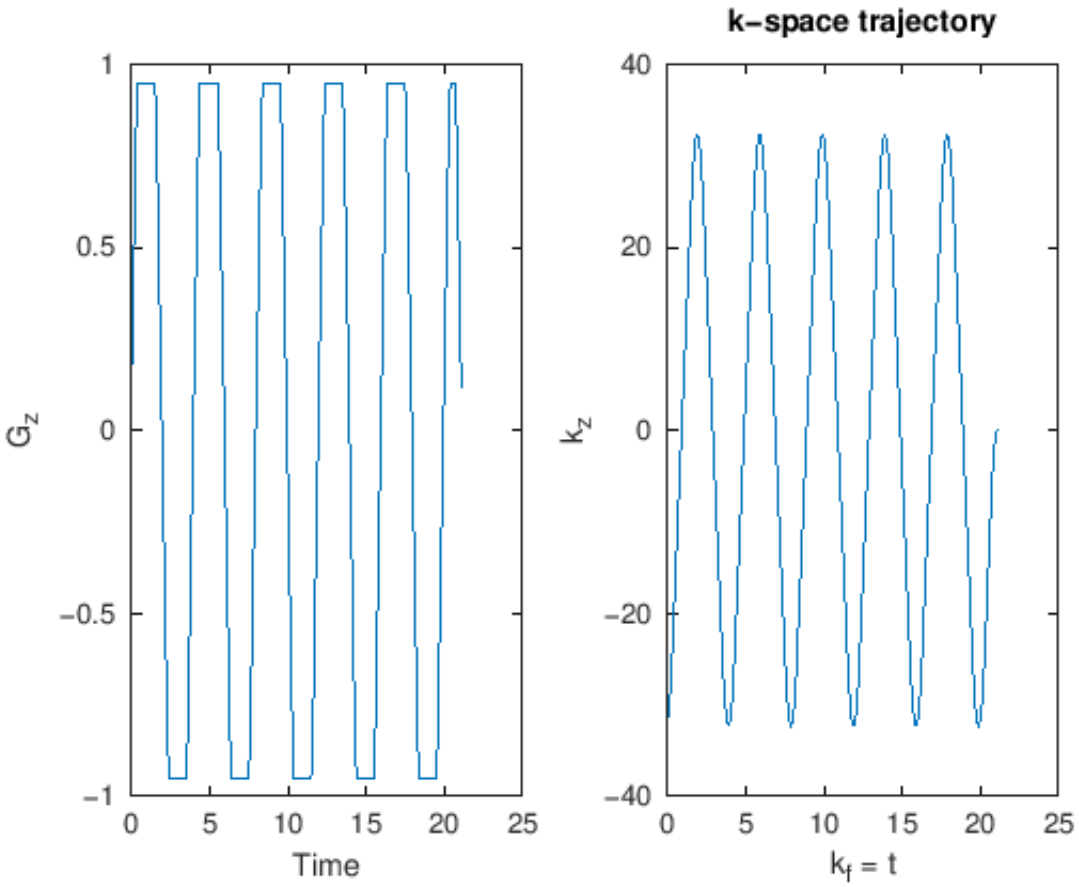
The gradient applied traverses spatial excitation k-space, while time traverses the spectral excitation k-space dimension, as demonstrated by the gradient and k-psace plot below

```
% setup MRI-education-resources path and requirements
cd ../
startup

% trick to compile abr() mex file
abr([1 1], [0 0], 0);
```

```
loading image
loading signal
```

```
error: MEX file now compiled, re-run code
error: called from
    abrx at line 31 column 1
    abr at line 31 column 12
```

```
% Demonstrate gradient and spectral-spatial k-space trajectoy

GAMMA = 42.58;
dt = 0.1;

Nlobe = 20;
Nspec = 10;
Nramp = 4;

ramp = [.5:Nramp-.5]/Nramp;
g1 = [ramp, ones(1,Nlobe-2*Nramp), ramp(end:-1:1)];

gamp = 0.95;
g = [];
for k = 1:Nspec
   g = [g, (mod(k,2)*2 - 1)*gamp*g1];
end

Nrw = round((Nlobe - Nramp)/2 + Nramp);
grw = [ramp, ones(1,Nrw-2*Nramp), ramp(end:-1:1)];

g = [g, (mod(Nspec+1,2)*2-1)*gamp*grw];

t = [0:length(g)-1]*dt;
subplot(121)
plot(t, g)
ylabel('G_z')
xlabel('Time')

subplot(122)
plot(t, GAMMA*dt*(cumsum(g) - sum(g)))
xlabel('k_f = t'), ylabel('k_z')
title('k-space trajectory')
```



To design the RF pulse, we can approximate the pulse and profile shape as separable function in frequency and space:

$$ h(f,z) = h_f(f) h_z(z)$$ $$ \mathcal{F}{ h(f,z) } = H(k_f, k_z) = H_f(k_f) H_z(k_z)$$

With this approximation, we can formulate the goal of the RF pulse design is to create and combine two pulses:

- A spectral pulse filter, $H_f(k_f)$, that should be applied in k_f direction
- A spatial pulse filter, $H_z(k_z)$, that should be applied in the k_z direction

Then design typically consists of 3 main steps

1. Design spectral pulse (envelope)
2. Design spatial sub-pulse
3. Combine spectral pulse envelope and spatial subpulses with an oscillating gradient

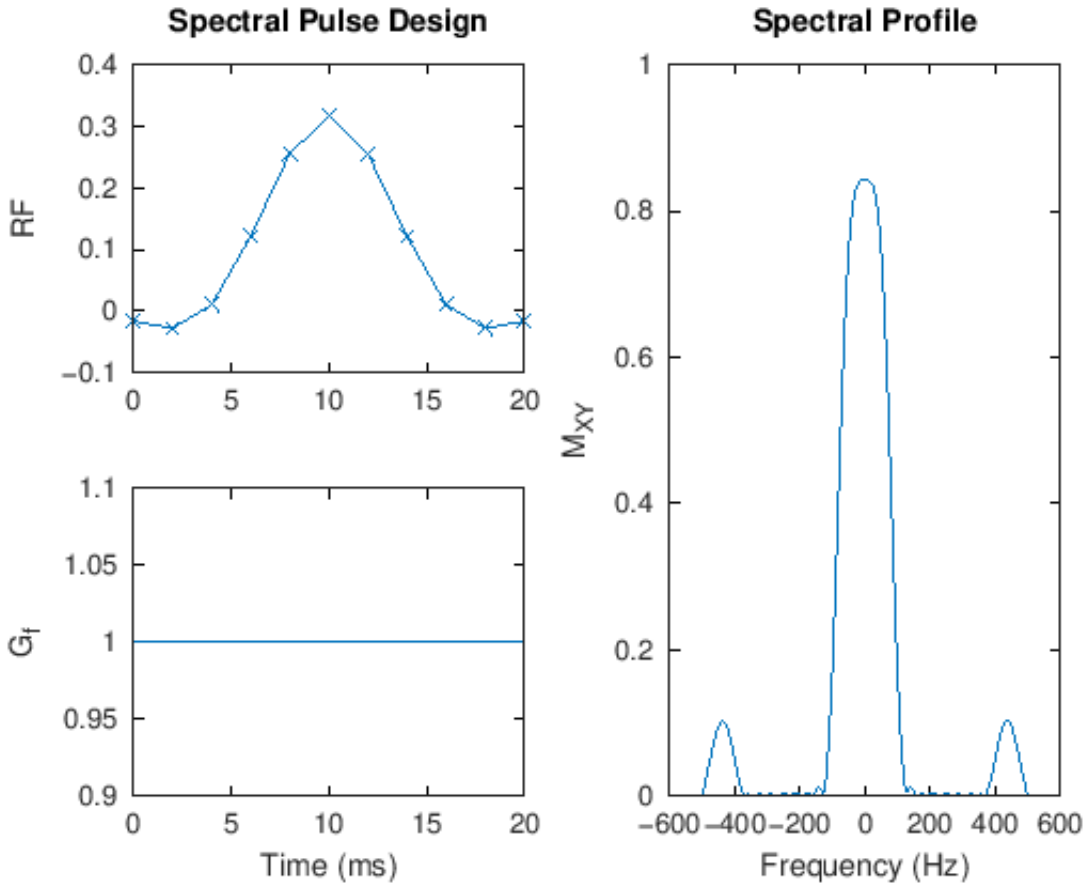
```
% Design parameters
GAMMA = 4258; % Hz/G
slewmax = 15e3; % G/cm/s
gmax = 4; % G/cm
T = 20e-3; % ms
TBW = 3;
SBW = 6;
dz = 0.5; % cm, max spatial resolution
Nspec = 10;
DT = T/Nspec;
dt = 10e-6; % us
flip = pi/4;
```

(continues on next page)

(continued from previous page)

```
% gradient parameters
verse_frac = 0.8;
Nlobe = DT/dt;
Nspat = round(Nlobe * verse_frac);
Nwait = (Nlobe - Nspat)/2;

z = linspace(-0.8, 0.8, 201);
df = linspace(-500, 500, 201);
```

```
% 1. Spectral Pulse Design

rfspec = dzrf(Nspec, TBW);
gf = 2*pi*DT * ones(1,length(rfspec));
mxy_f = ab2ex(abr(rfspec, gf, df));

t = [0:length(rfspec)-1]*DT*1e3;
subplot(221)
plot(t, real(rfspec), '-x')
ylabel('RF')
title('Spectral Pulse Design')
subplot(223)
plot(t, ones(1,length(rfspec)))
ylabel('G_f')
xlabel('Time (ms)')
subplot(122)
plot(df, abs(mxy_f))
xlabel('Frequency (Hz)'), ylabel('M_{XY}')
title('Spectral Profile')
```

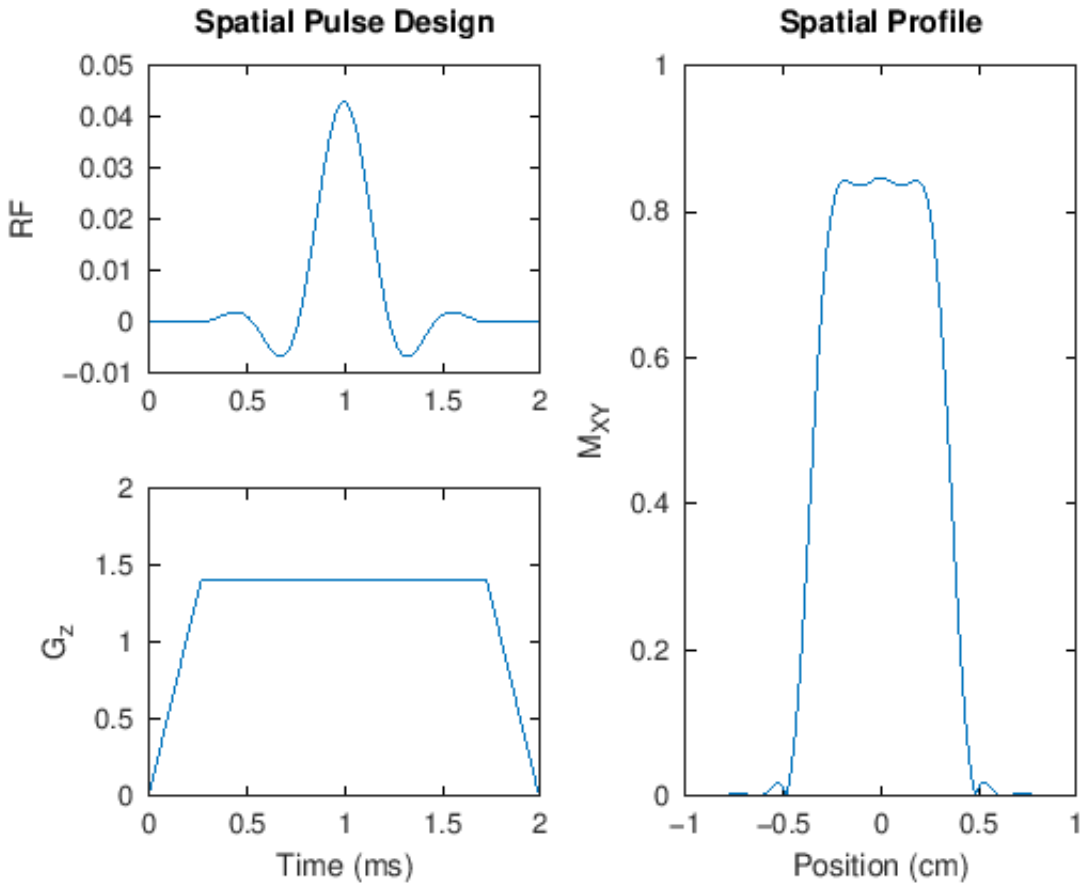



```
% 2. Spatial pulse design

rfspat = dzrf(Nspat, SBW);

% gradients (including ramp sampling)
gamp = (SBW/DT)/(GAMMA*dz); % gradient for spatial resolution

% Uses a slew rate that can't be violated
Nramp = min( ceil(gmax/slewmax / dt), floor(Nlobe/2));
ramp = [.5:Nramp-.5]/Nramp;
g1 = [ramp, ones(1,Nlobe-2*Nramp), ramp(end:-1:1)];
% apply VERSE algorithm for time-varying gradient
rfspatv = verse(g1, [zeros(1, floor(Nwait)), rfspat(:).', zeros(1,ceil(Nwait))]).';
rfspatv(find(isnan(rfspatv))) = 0;

g1z = 2*pi*GAMMA*dt * gamp* g1;
mxy_z = ab2ex(abr(rfspatv, g1z, z));

t = [0:length(g1)-1]*dt*1e3;
subplot(221)
plot(t, real(rfspatv))
ylabel('RF')
title('Spatial Pulse Design')

subplot(223)
plot(t, gamp*g1)
```

(continues on next page)

(continued from previous page)

```
ylabel('G_z')
xlabel('Time (ms)')

subplot(122)
plot(z, abs(mxy_z))
xlabel('Position (cm)'), ylabel('M_{XY}')
title('Spatial Profile')
```

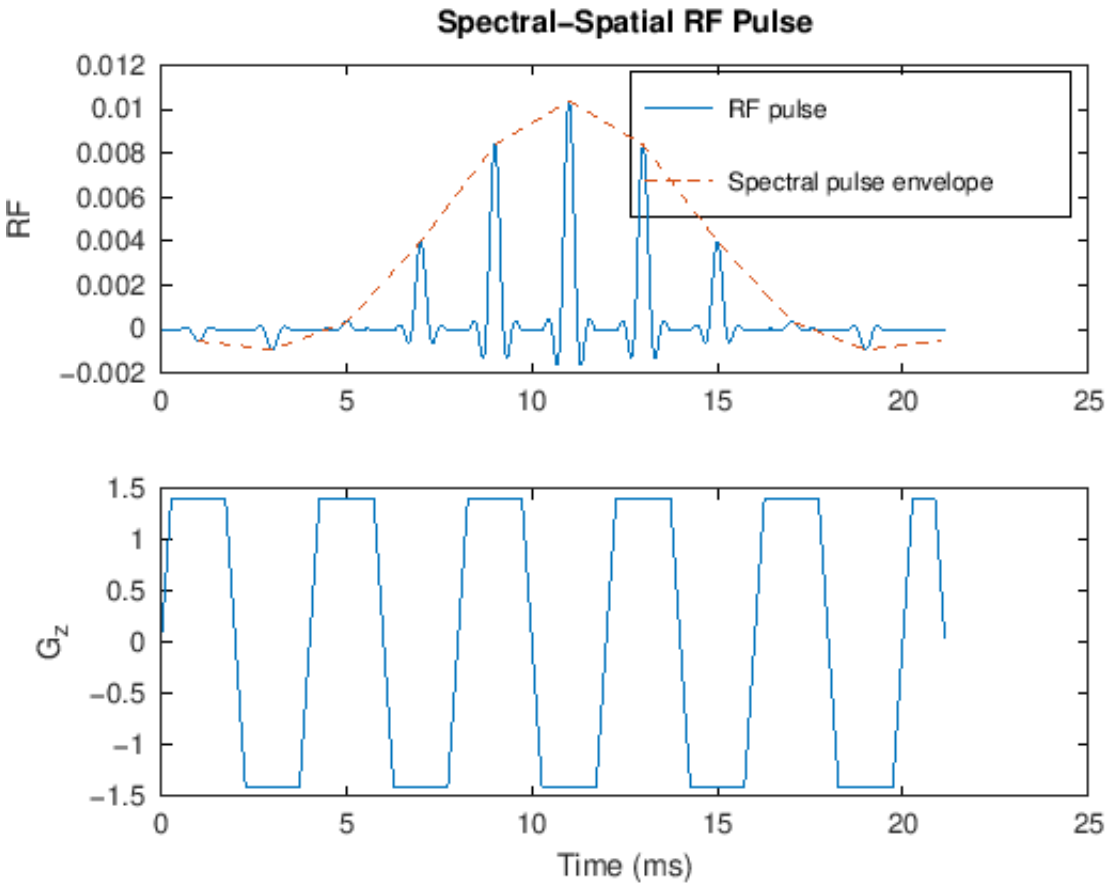


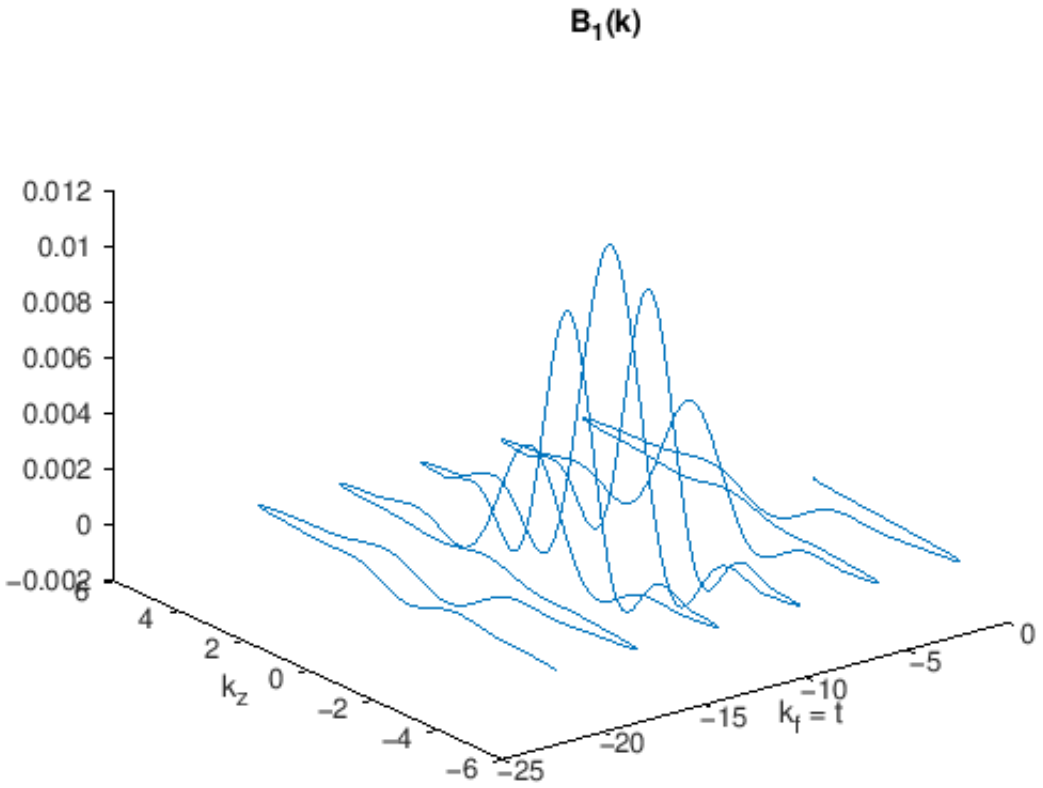

```
% 3. Combine spectral and spatial pulse designs

rf = []; g = [];
for k = 1:Nspec
   rf = [rf, rfspatv*rfspec(k)];  % weight spatial sub-pulses by the spectral filter
   g = [g, (mod(k,2)*2 - 1)*gamp*g1];  % oscillate gradient
end

% add spatial refocusing gradient
Nrw = round((Nlobe - Nramp)/2 + Nramp);
grw = [ramp, ones(1,Nrw-2*Nramp), ramp(end:-1:1)];

g = [g, (mod(Nspec+1,2)*2-1)*gamp*grw];
rf = [rf, zeros(1,Nrw)];

% scale RF pulse
flip_scale = flip/sum(rf);
```

(continues on next page)

(continued from previous page)

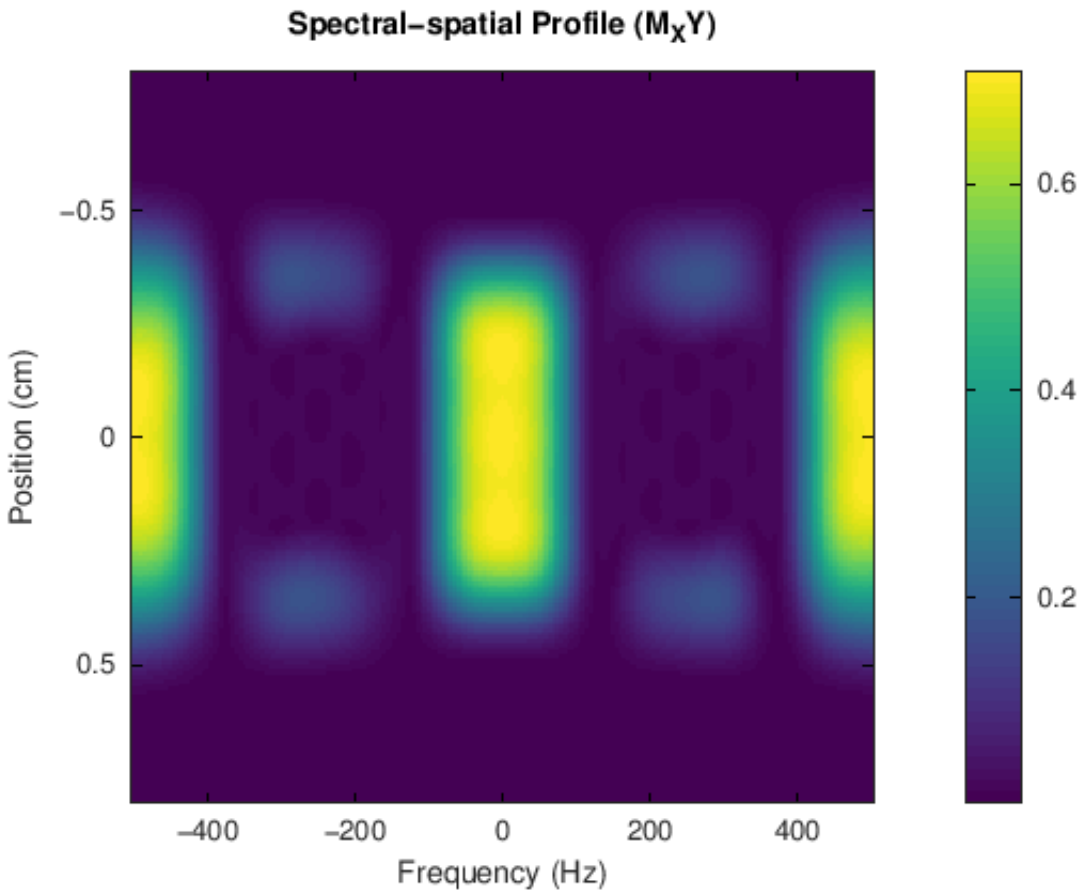
```
rf = flip_scale*rf;
rfs = rfscaleg(rf, T);

figure
subplot(211)
plot([0:length(rf)-1]*dt*1e3, real(rf), ...
    [0.5:1:length(rfspec)-0.5]*DT*1e3, real(rfspec)/max(real(rfspec)) * max(real(rf)),
 ↪ '--')
    legend('RF pulse', 'Spectral pulse envelope')
ylabel('RF')
title('Spectral-Spatial RF Pulse')
subplot(212)
plot([0:length(rf)-1]*dt*1e3, g)
ylabel('G_z')
xlabel('Time (ms)')

figure
plot3([-length(rf)+1:0]*dt*1e3, GAMMA*dt*(cumsum(g) - sum(g)), real(rf))
xlabel('k_f = t'), ylabel('k_z')
title('B_1(k)')
```



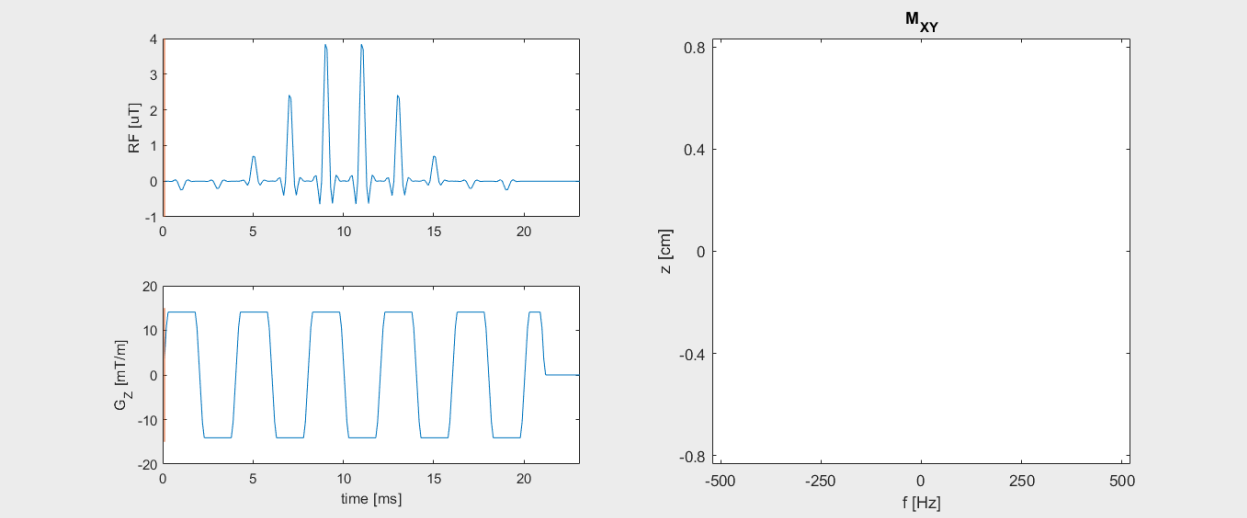


The spectral pulse, $H_f(k_f)$, creates the overall envelope, and is applied as a weighting for repeated versions of the spatial pulse, $H_z(k_z)$. This creates the excitation k-space profile, $B_1(k_f, k_z)$ shown above, which results in the spectral-spatial profile shown below:

```
gz = 2*pi*GAMMA*dt * g;
gf = 2*pi*dt * ones(1,length(rf));

mxy_2d = ab2ex(abr(rf, gz + i*gf, z, df));


figure
imagesc(df,z,abs(mxy_2d))
xlabel('Frequency (Hz)'), ylabel('Position (cm)')
title('Spectral-spatial Profile (M_XY)')
%colormap(gray)
colorbar
```



Watch how the transverse magnetization evolves during the spectral-spatial RF pulse:



CHAPTER

THIRTYTHREE

FUN WITH RF PULSES?

Yes, you heard me right. Maybe it takes a special type of person, but I think RF pulses can be fun! Check out some interesting examples below.

Movies created with SpinBench https://heartvista.ai/spinbench

33.1 Learning Goals

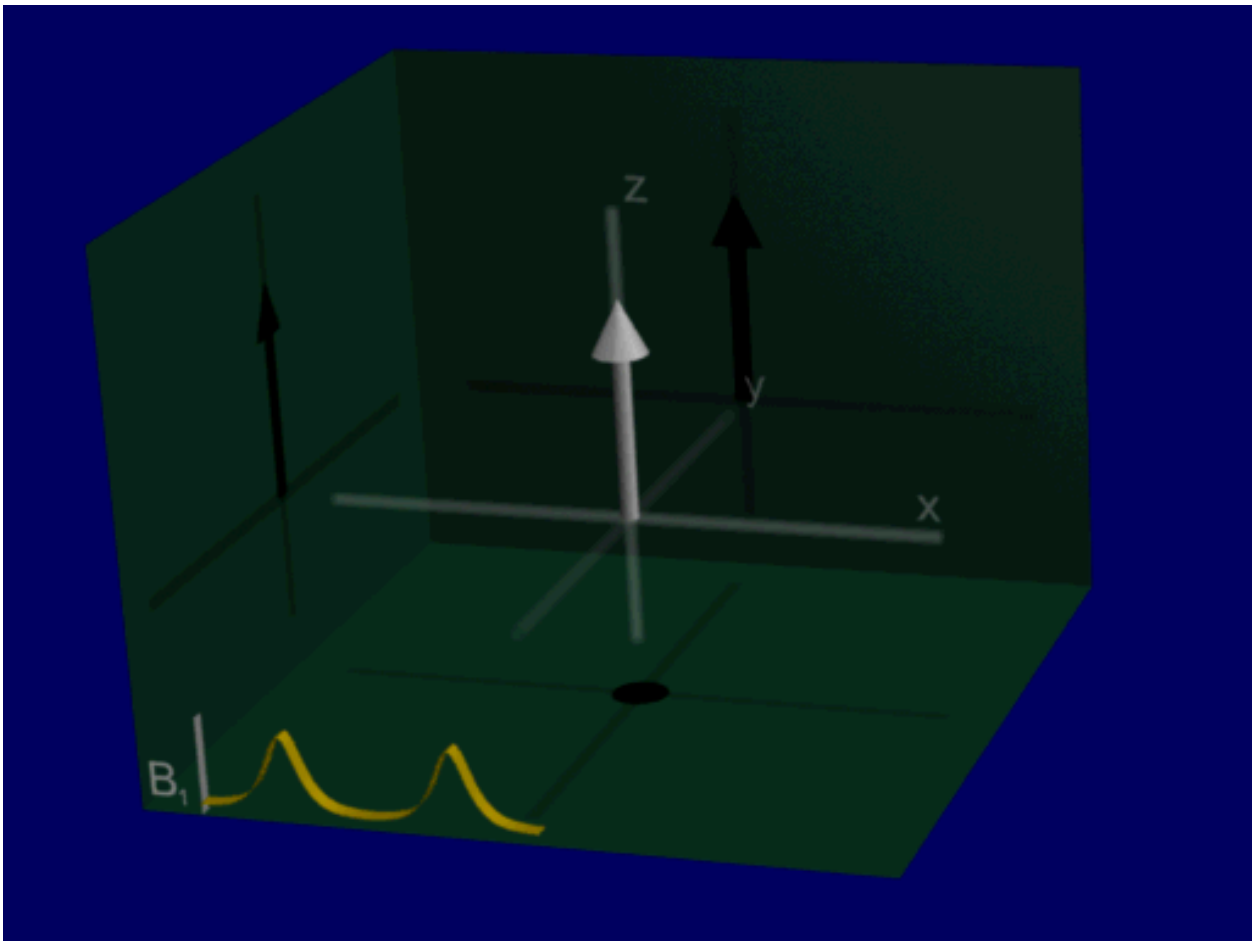
1. Have fun

33.1.1 Adiabatic Pulses

Adiabatic pulses are a class of pulses that are basically insensitive to $B_1^+(\vec{x})$ field inhomongeneities. In other words, they perform similarly even if there is variation in the amplitude of the magnetic field created by the RF transmit coil.

These are mostly effective for 180-degree flip angles, for purposes of inversion or spin-echoes (although typically 2 adiabatic 180-degree pulses are required for refocusing unwanted phase when used as a spin-echo).

The example below shows a simulation of two 180-degree inversion adiabatic pulses. This movie shows how these pulses "sweep" the magnetization from the +z axis to the -z axis, and then back again with the second 180-degree pulse.



Yellow - B1 (RF) vector

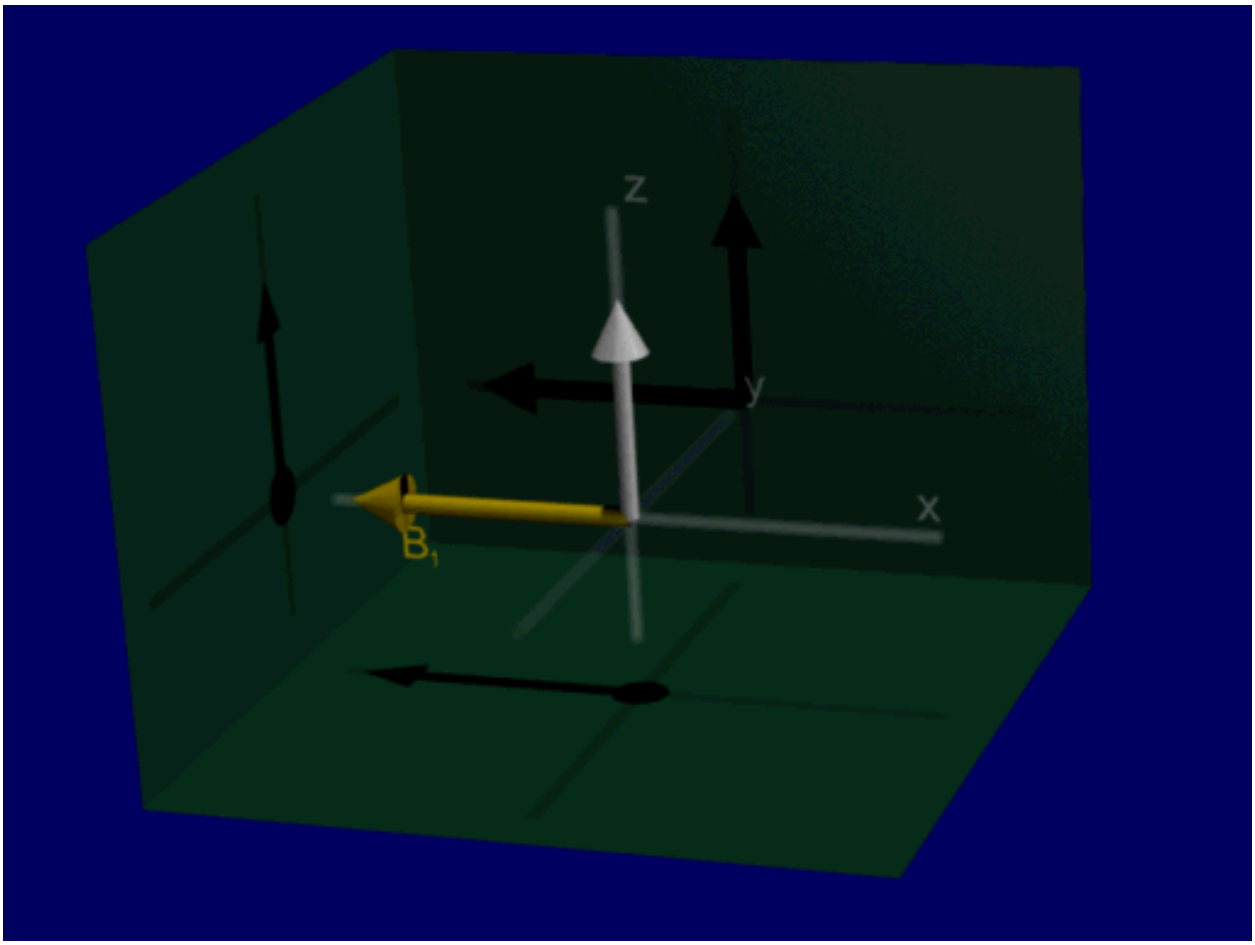
White - net magnetizations for a range of frequencies around the Larmor frequency

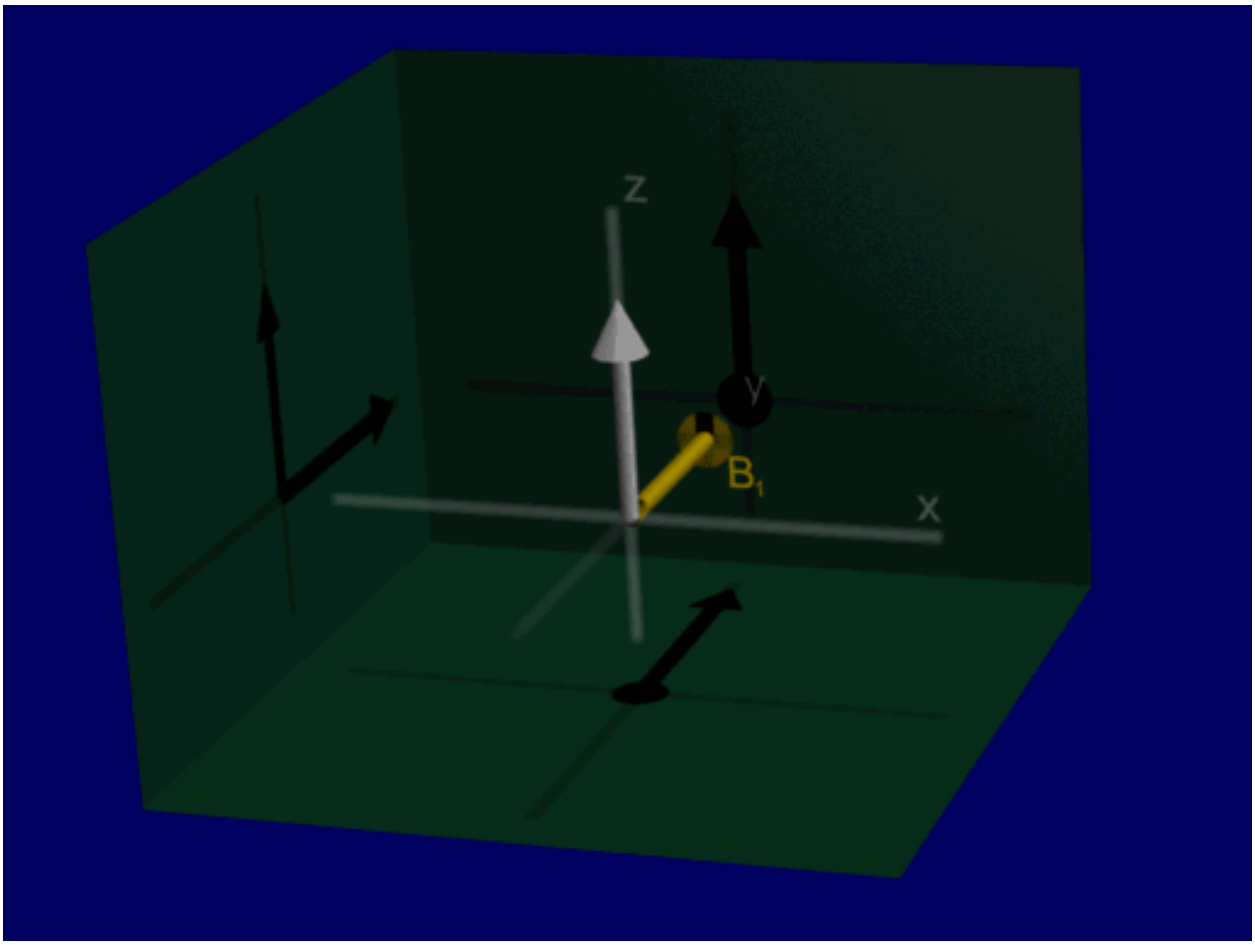
The amplitude of the adiabatic pulses is plotted in the bottom left corner

33.1.2 Pretty Pictures

These two examples apply continuous RF pulse (also called a constant amplitude or hard pulse), and show the resulting net magnetization for a range of off-resonance frequnecies around the Larmor frequency. Here, the off-resonance changes the angle of rotation.

One movie is the RF applied along the x-direction, the other RF is applied along the y-direction.


z
y
x
B1


z
y
B_1
x

CHAPTER

THIRTYFOUR

ADVANCED SNR

Simulations of SNR efficiency, challenges to the fundamental assumptions of SNR (e.g. T2* decay during the readout)

34.1 SNR and T_2^* decay

The SNR equation alone implies that the readout time should be as long as possible, as this leads to more total measurement time, but practically this is limited by the $T_2^$ decay rate, since at some point there will no longer be any signal to measure. To incorporate the effect of this decay, we need to account for the signal modulations during the readout by integrating the signal area, accounting for $T_2^$, which will offset signal gains from extending the readout

$$SNR \propto \frac{ \int_0^{T_{read}} \exp(-\tau / T_2^*) d\tau } { \sqrt{T_{read}} }$$

T_{read} is the readout duration, the numerator the signal acquired during this readout time, and the demoninator captures how the total noise scales with the square root of measurement time.

The above equation applieds for a one-sided decay, as in a gradient echo, while the following applies in the case of two-sided T2* decay, as in a spin echo:

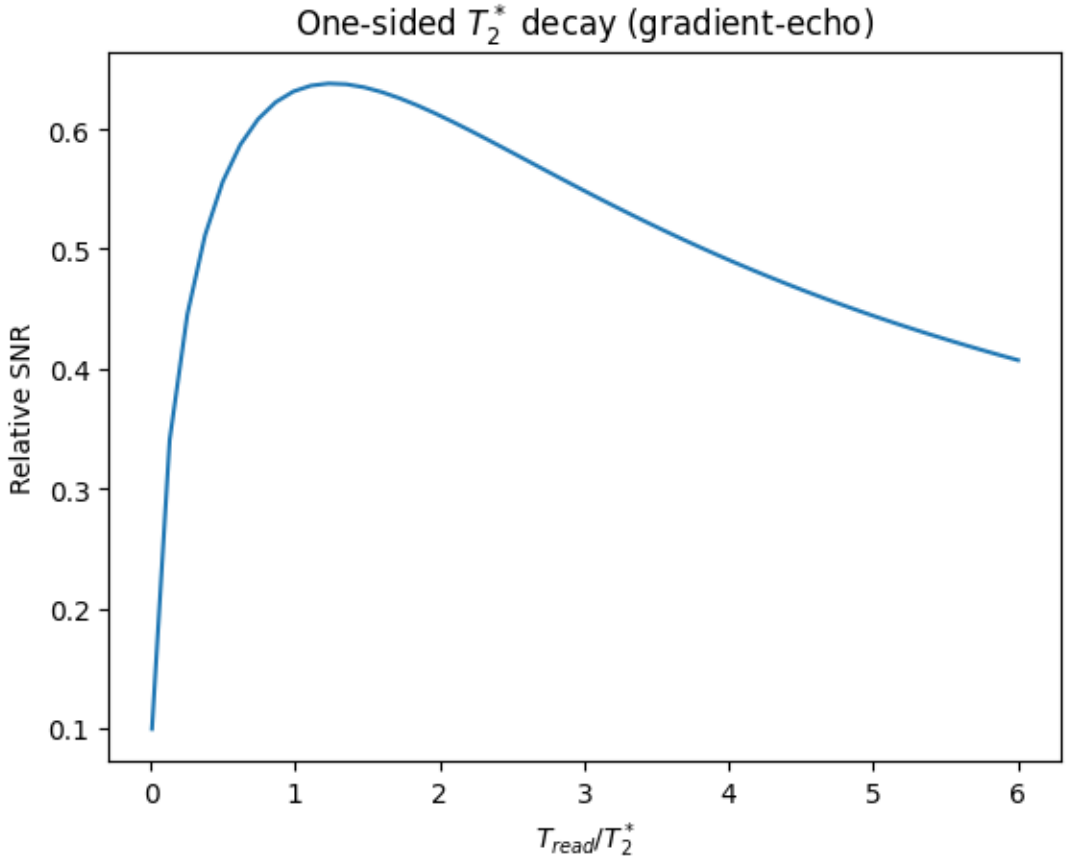
$$SNR \propto \frac{\int_{-T_{read}/2}^{T_{read}/2} | \exp(-\tau / T_2^*) | d\tau }{\sqrt{T_{read}}}$$

```
import numpy as np
import matplotlib.pyplot as plt

Tread = np.linspace(0.01, 6)
T2s = 1

SNR_T2s_onesided = T2s * (1 - np.exp(-Tread / T2s)) / np.sqrt(Tread)
Imax_onesided = np.argmax(SNR_T2s_onesided)

SNR_T2s_twosided = T2s * 2 * (1 - np.exp(-Tread / 2 / T2s)) / np.sqrt(Tread)
Imax_twosided = np.argmax(SNR_T2s_twosided)

plt.plot(Tread, SNR_T2s_onesided)
plt.xlabel(r'$T_{read} / T_2^*$')
plt.ylabel('Relative SNR')
plt.title(r'One-sided $T_2^*$ decay (gradient-echo)')
plt.show()

print(
    f'Maximum for one-sided decay: '
    f'T_read / T_2* = {Tread[Imax_onesided]:.2f}'
)

plt.figure()
```

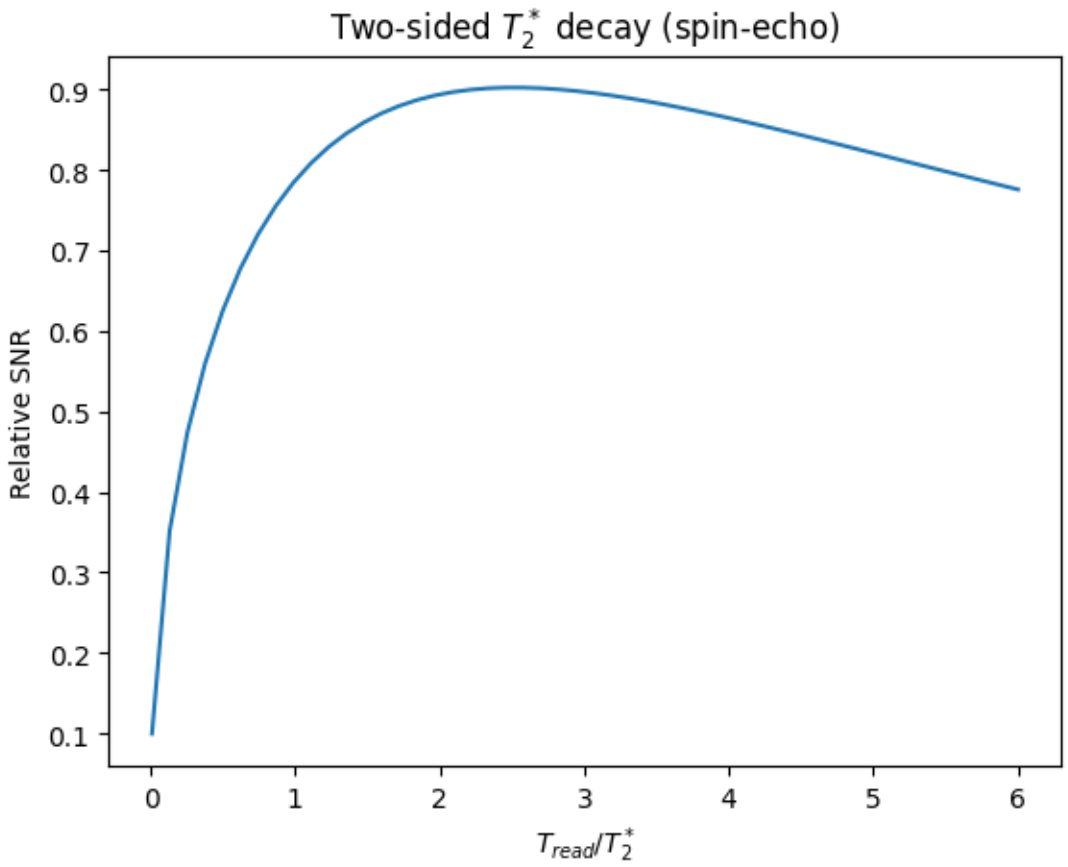
(continues on next page)

(continued from previous page)

```
plt.plot(Tread, SNR_T2s_twosided)
plt.xlabel(r'$T_{read} / T_2^*$')
plt.ylabel('Relative SNR')
plt.title(r'Two-sided $T_2^*$ decay (spin-echo)')
plt.show()

print(
    f'Maximum for two-sided decay: '
    f'T_read / T_2* = {Tread[Imax_twosided]:.2f}'
)
```




```
Maximum for one-sided decay: T_read / T_2* = 1.23
```




```
Maximum for two-sided decay: T_read / T_2* = 2.45
```

From this result, we can see that the optimal choice of T_{read} from a SNR point of view is approximately $ T_{read} / T_2^* = 1.3$ (one-sided) and $ T_{read} / T_2^* = 2.5$ (two-sided).

However, this relationship does not take into account the effects of resolution loss that will typically occur due to decaying T_2^* weighting in k-space. (See "MRI Signal Equation" for information on this effect).

34.2 SNR efficiency versus TR

The SNR efficiency is useful to examine the choice of TR. It is especially interesting for spoiled gradient-echo sequences where when we can choose to use the optimal flip angle ("Ernst angle") based on a T_1 value of interest. For these sequences, the optimal flip angle is

$$\theta_{optimal} = \cos^{-1} \exp(-TR/T_1)$$

And $f_{seq,GRE}$ is defined above.

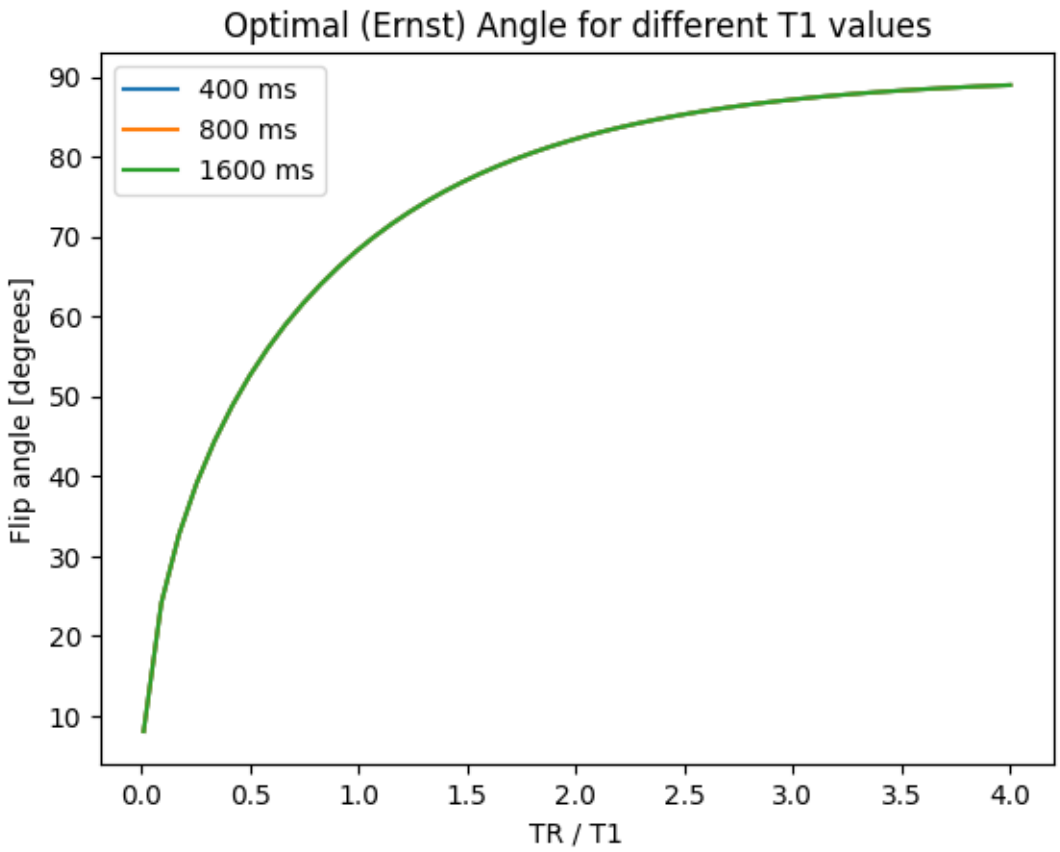
Under this condition, we see the following intriguing result. Once the optimal flip angle begins to approach 90-degrees, the SNR efficiency begins to drop. But for the shorter TRs, the SNR efficiency is relatively stable for different TRs.

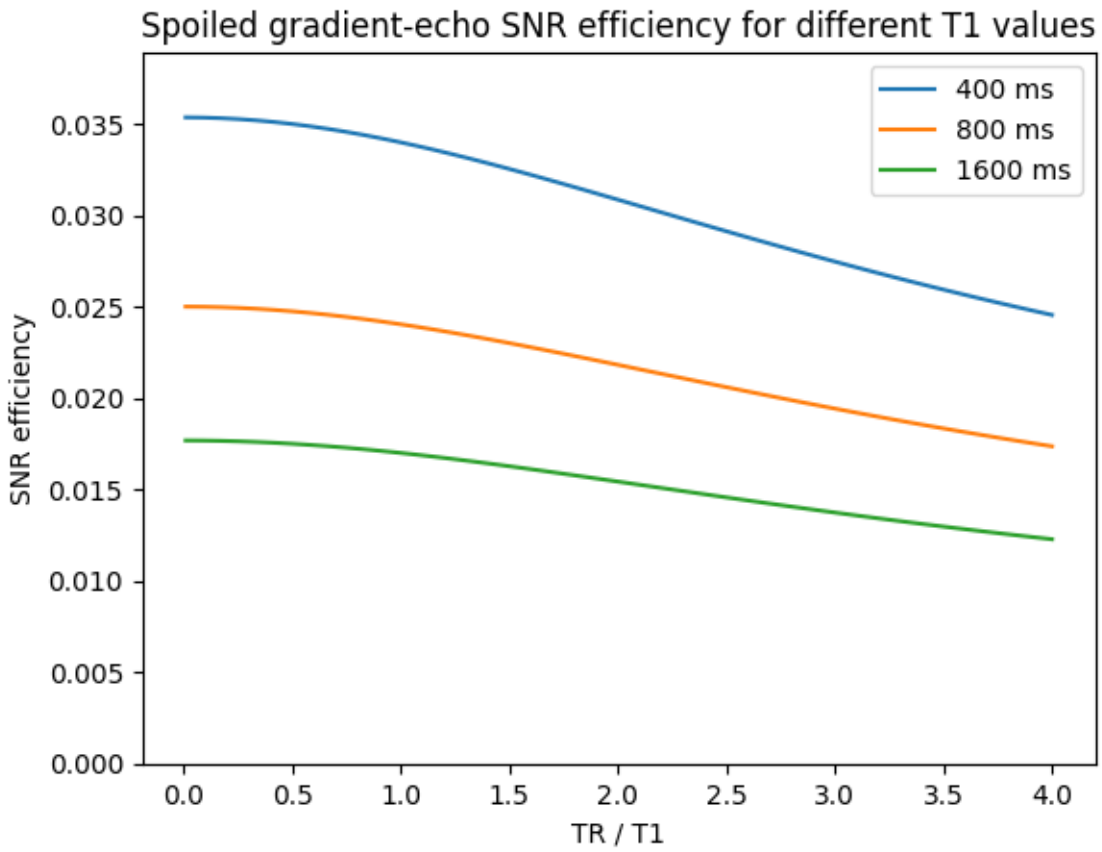
```
import numpy as np
import matplotlib.pyplot as plt

# SNR efficiency for spoiled gradient-echo pulse sequences
```

(continues on next page)

(continued from previous page)

```
TRdT1 = np.linspace(0.01, 4)
T1 = np.array([400, 800, 1600])

# To remove TE or T2 effects
TE = 0
T2 = np.inf
M0 = 1

theta_optimal = np.zeros((len(T1), len(TRdT1)))
SNR_efficiency = np.zeros((len(T1), len(TRdT1)))

for IT1, T1_value in enumerate(T1):
    theta_optimal[IT1, :] = np.arccos(np.exp(-TRdT1))

    # Spoiled gradient-echo signal divided by sqrt(TR)
    signal = (
        M0
        * np.sin(theta_optimal[IT1, :])
        * (1 - np.exp(-TRdT1))
        / (1 - np.exp(-TRdT1) * np.cos(theta_optimal[IT1, :]))
    )
    SNR_efficiency[IT1, :] = signal / np.sqrt(TRdT1 * T1_value)

plt.figure()
for IT1, T1_value in enumerate(T1):
    plt.plot(
        TRdT1,
        np.degrees(theta_optimal[IT1, :]),
        label=f'{T1_value} ms',
    )
plt.xlabel('TR / T1')
plt.ylabel('Flip angle [degrees]')
plt.title('Optimal (Ernst) Angle for different T1 values')
plt.legend()

plt.figure()
for IT1, T1_value in enumerate(T1):
    plt.plot(TRdT1, SNR_efficiency[IT1, :], label=f'{T1_value} ms')
plt.ylim(0, 1.1 * np.max(SNR_efficiency))
plt.legend()
plt.xlabel('TR / T1')
plt.ylabel('SNR efficiency')
plt.title('Spoiled gradient-echo SNR efficiency for different T1 values')
plt.show()
```


Optimal (Ernst) Angle for different T1 values
400 ms
800 ms
1600 ms
Flip angle [degrees]
90
80
70
60
50
40
30
20
10
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
TR / T1


Spoiled gradient-echo SNR efficiency for different T1 values
400 ms
800 ms
1600 ms
SNR efficiency
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
TR / T1

CHAPTER

THIRTYFIVE

ADVANCED IMAGE RECONSTRUCTION

This section introduces a more general formulation of image reconstruction in MRI. This formulation can then be used to describe a variety of basic and advanced image reconstruction methods, several of which are described.

35.1 General Formulation of MRI Reconstruction

For advanced image reconstruction methods, it is helpful to reformulate the reconstruction problem as a linear system and using standard mathematical notation describing linear systems as:

$$ y = \mathbf{E}x + n $$

where y is the acquired data, $\mathbf{E}$ is the encoding matrix, x is the spatial distribution of the transverse magnetization (e.g. image), and n is noise. In this formulation, the image is vectorized.

35.2 Fourier Transform Reconstruction

The definition of $\mathbf{E}$ is critical, and this can be derived from the MRI signal equation. If we only consider the effects of magnetic field gradients, then

$$s(t) = \int m(\vec{r}) \exp(-i2\pi \vec{k}(t) \cdot \vec{r}) \ d\vec{r}$$

Our measurements, y, taken from $s(t)$ at discrete time points. Discretizing this relationship for Cartesian sampled data:

$$ y = \left[\begin{array}{c} s_1(t_1) \ s_1(t_2) \ \vdots \ s_1(t_{Nx}) \ s_2(t_1) \ \vdots \ s_{Ny}(t_{Nx})\ \end{array} \right]$$

where $s_m(t_n)$ is the data from the repetition/TR m at sample n.

We want to reconstruct the following image on a grid:

$$ x = \left[\begin{array}{c} m(x_1,y_1) \ m(x_2,y_1) \ \vdots \ m(x_{Nx},y_1) \ m(x_{1},y_2) \ \vdots \ m(x_{Nx},y_{Ny}) \end{array} \right]$$

where x_i and y_i are equally spaced locations in image space.

In this most simple case, the encoding matrix, $\mathbf{E}$, includes just a discrete Fourier Transform matrix, $\mathbf{F}$, describing the fact that our data is the Fourier Transform of the transverse magnetization, evaluated at k-space locations, as shown in the above signal equation. For this matrix, the entries are defined by k-space location being sampled, $\vec{k}_i$ and the spatial locations, $\vec{r}_j$:

$$ \mathbf{f}_{i,j} = \exp(-i 2 \pi \vec{k}_i \cdot \vec{r}_j)$$

In this case, the measurement is the discrete Fourier Transform of the image

$$ y = \mathbf{F}x + n $$

and image reconstruction is performed by inverse discrete Fourier Transform, which for fully sampled 2D FT imaging is well defined by matrix inversion:

$$ \hat{x} = \mathbf{F^{H}} y $$

This can equivalently be computed using the Fast Fourier Transform (FFT) algorithm when the data is fully sampled on a Cartesian grid.

35.3 Parallel Imaging

For parallel imaging (PI), we take advantage and include in our model the coil sensitivity profiles, $\mathbf{C}_q$, for each RF coil. These are included in the encoding matrix along with a Fourier Transform encoding matrix, $\mathbf{F}$. For performing undersampled reconstruction, a k-space sub-sampling operator, $\mathbf{S}$, is also added to the encoding matrix, acknowledging the fact we know the sampling criteria are not satisfied, and allowing $\mathbf{F}$ and $\mathbf{F^H}$ to be computed with the FFT. With these factors, the measurements from each RF coil, $\mathbf{y}_q$ are:

$$y_q = \mathbf{E}_q x + n_q = \mathbf{S} \mathbf{F} \mathbf{C_q} x + n_q$$

The coil sensitivity profiles $\mathbf{C}_q$ is a diagonal matrix with entries corresponding to the coil sensitivity profile at each location in $\mathbf{x}$.

The k-space sub-sampling operator $\mathbf{S}$, is a diagonal matrix with entries of 1 or 0, describing whether an expect grid location was sampled.

Here we can return to our original formulation by creating augmented matrices, concatenating along the coil elements dimension for example as:

$$\mathbf{y} = [y_1 \ y_2 \ldots y_N]$$

$$\mathbf{E} = [\mathbf{E}_1 \ \mathbf{E}_2 \ldots \mathbf{E}_N]$$ $$\mathbf{n} = [n_1 \ n_2 \ldots n_N]$$

resulting in:

$$ \mathbf{y} = \mathbf{Ex} + \mathbf{n}$$

35.3.1 Model-based Methods

Model-based parallel imaging methods use some simulation of the MRI system and include the methods of SENSE, ESPIRiT, and NLINV. These reconstruct a single underlying image by solving an optimization problem. The popular SENSE methods solves

$$\mathbf{\hat{x}}*{PI} = *arg* *min*\mathbf{x} \frac{1}{2} | \mathbf{y} - \mathbf{Ex} |^2_2$$

This can be solved directly by a pseudo-inverse, but requires measurements of the coil sensitivity profiles to calculate $\mathbf{E}$:

$$\mathbf{\hat{x}}_{PI} = (\mathbf{E}^H\mathbf{E})^{-1} \mathbf{E}^H \mathbf{y}$$

35.3.2 Linear Predictability Methods

Linear predictability parallel imaging methods assume that each k-space sample is a linear combination of other k-space samples and inlcude the methods of AUTO-SMASH, GRAPPA, and SPIRiT. These utilize a calibration kernel, computed from the data itself and captured in the matrix $\mathbf{G}$, and can be generally formulated as the following optimization problem

$$\mathbf{\hat{x}}*{PI} = *arg* *min*\mathbf{x} | \mathbf{y} - \mathbf{Ex} |^2_2 + \lambda | (\mathbf{G} - \mathbf{I}) \mathbf{x} |^2_2$$

Where λ is a tuning or regularization parameter and $\mathbf{I}$ is the identity matrix. This simultaneously enforces data consistency (first term) as well as self-consistency of the calibration (second term).

The calibration kernel, $\mathbf{G}$, can either be directly measured from the data as in GRAPPA, or iteratively estimated while solving the optimization, as in SPIRiT.

35.4 Compressed Sensing Methods

Compressed Sensing is formulated as the following optimization problem, specifically using the ℓ_1-norm ($|x|*1* = *sum*{i=1}^n |x_i|$) that promotes sparsity in the solution:

$$\mathbf{\hat{x}}*{CS} = *arg* *min*\mathbf{x} \frac{1}{2} | \mathbf{y} - \mathbf{Ex} |^2_2 + \lambda_{CS} | \mathbf{Wx} |_1 $$

which includes a data consistency term where the data multiplied by the encoding matrix must match the reconstructed image, and a regularization term that enforces that the image is sparse using the ℓ_1 norm in some domain through the sparsifying transform, $\mathbf{W}$. There is a regularization factor, λ_{CS}, that must be chosen to balance the data consistency and sparsity terms.

Popular sparsifying transforms, $\mathbf{W}$, include total variation (TV), total generalized variation (TGV), and wavelets, where MRI data is typically sparse.

This regularization term, $| \mathbf{Wx} |_1$ can also be combined with iterative parallel imaging reconstruction methods as well, for example in the ℓ_1-ESPIRiT method.

35.5 Deep Learning Reconstructions

Deep learning (DL) methods can often be considered general solutions to a regularized least-squares objective

$$\mathbf{\hat{x}}*{reg} = *arg* *min*\mathbf{x} \frac{1}{2} | \mathbf{y} - \mathbf{Ex} |^2_2 + \mathcal{R}(\mathbf{x}) $$

where $\mathcal{R}(\cdot)$ can be a parallel imaging or compressed sensing regularizer as above, but can also be implicitly implemented via deep learning techniques. This means that the deep learning reconstruction is attempting to constrain or regularize the reconstruction.

The most straightforward approach using deep learning is to train a CNN to invert the encoding operation in image space, e.g. after applying the inverse Fourier Transform to the data:

$$\mathbf{\hat{x}}*{DL} = g*{DNN}(\mathbf{E^{H} y})$$

The learned deep neural network, $g_{DNN}()$, includes many weights that must be trained prior to being able to perform the reconstruction.

However, so-called un-rolled networks have become more popular for MRI reconstruction, as they can better incorporate parallel imaging information, data consistency, and also are often more compact. They are designed to mimic iterations

of classical algorithms to solve optimization problems, and each iteration has been "un-rolled" into a series of cascaded neural networks. Some popular methods include MoDL and Variational Networks.

Variational Networks (VarNets) for undersampled MRI reconstruction are formulated as the above regularlized least-squares objective, but models the architecture after a single gradient update in iterative gradient descent (used to solve optimization problems) and uses a CNN (e.g. UNet) within the iterations

The Model-based Deep Learning (MoDL) method uses the formulation where the regularizer is a CNN that estimates the noise and aliasing patterns

$$ \mathcal{R}_{MoDL}(\mathbf{x}) = \lambda | \mathcal{N}_w(\mathbf{x}) |^2$$

and the estimate depends on a set of learned parameters, w, in the neural network, $\mathcal{N}_\mathbf{w}(\cdot)$. It's structure is overall similar to VarNets, as both are unrolled architectures, but with slightly different formulations.

35.5.1 Deep Learning Requirements

1. Data
    - Generally a lot of data (10^4 in many radiology applications, but also depends a lot on the question being answered and data quality)
    - Fully sampled data typically required for supervised learning of undersampled MRI reconstruction
    - Data augmentation – realistic modifications of existing data to increase training data size
    - For reconstruction, fastMRI provides thousands of high quality training data: http://fastmri.med.nyu.edu, https://doi.org/10.48550/arXiv.1811.08839
2. Architecture
    - U-net, ResNet popular for working in image-domain
    - Un-rolled networks popular for image reconstruction.
    - Generative models possible
    - Transformers
3. Computation
    - GPUs are essential for training of deep networks
4. Training
    - Send input data forward through network, compare to the expected output (e.g. fully sampled image or data) update the network weights through back-propagation
    - Choose loss function, such as l1 or l2
    - Choose batch sizes, Learning rate, etc

35.5.2 Reference

For a recent, comprehensive reference on these methods I reccommend:

Hammernik K, Küstner T, Yaman B, Huang Z, Rueckert D, Knoll F, Akçakaya M. Physics-Driven Deep Learning for Computational Magnetic Resonance Imaging: Combining physics and machine learning for improved medical imaging. IEEE Signal Process Mag. 2023 Jan;40(1):98-114. doi: 10.1109/msp.2022.3215288. Epub 2023 Jan 2. PMID: 37304755; PMCID: PMC10249732.

https://arxiv.org/pdf/2203.12215.pdf