AN OPTIMAL DESIGN METHOD

FOR

MRI TEARDROP GRADIENT WAVEFORMS

AN OPTIMAL DESIGN METHOD

FOR

MRI TEARDROP GRADIENT WAVEFORMS

By

TINGTING REN, hB.Eng., M.Eng.

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfillment of the Requirements

for the Degree

Master of Science

McMaster University

MASTER OF SCIENCE(2005) McMaster University

COMPUTING AND SOFTWARE Hamilton, Ontario

TITLE: An Optimal Design Method for MRI Teardrop Gradient Waveforms

AUTHOR: Tingting Ren, M.ENG.

SUPERVISORS: Dr. Christopher Kumar Anand and Dr. Tamás Terlaky

NUMBER OF PAGERS: xiii, 138

Abstract

This thesis presents an optimal design method for MRI (Magnetic Resonance Imaging) teardrop gradient waveforms in two and three dimensions. Teardrop in two dimensions was introduced at ISMRM 2001 by Anand et al. to address the need for a high efficiency balanced k-space trajectory for real-time cardiac SSFP (Steady State Free Precession) imaging.

We have modeled 2D and 3D teardrop gradient waveform design as nonlinear convex optimization problems with a variety of constraints including global constraints (e.g., moment nulling for motion insensitivity). Commercial optimization solvers can solve the models efficiently. The implementation of AMPL models and numerical testing results with the solver MOSEK are provided. This optimal design procedure produces physically realizable teardrop waveforms which enable real-time cardiac imaging with equipment otherwise incapable of doing it, and optimally achieves the maximum resolution and motion artifact reduction goals. The research may encompass other waveform design problems in MRI and has built a good foundation for further research in this area.

Acknowledgements

I would first like to offer sincere thanks to my supervisors Dr. Christopher Kumar Anand and Dr. Tamás Terlaky, for their patience, guidance and continual support during the preparation of this thesis. I am truly grateful for what I have been able to learn from them.

I would also like to express gratitude towards the members of the Advanced Optimization Laboratory for their great help and support.

Finally, I want to thank my family for their support and understanding. Special thanks go to my husband Zhanyue Qi. Without his patience and encouragement, none of this would have been possible.

Contents

List of Figures

Chapter 1

Introduction

Conventional MRI has a serious drawback that the scan time is long compared with time constants of cardiac, peristaltic, and respiratory motion in the patient, giving rise to motion artifacts in the images. This long scan time is caused by the time it takes to allow the spins after acquisition to relax back to the equilibrium situation before the next excitation pulse is applied. Fast scanning is necessary in these cases.

Real-time cardiac imaging has been demonstrated by using Steady State Free Precession (SSFP) [3], making use of high performance gradients. The two challenges faced by SSFP are banding artifacts and the requirement for powerful gradients, which involve high rate of change of magnetic field (dB/dt). We will design a novel non-raster gradient waveform for k-space

(Fourier transform space where signals of MRI come from) readout which does not suffer from banding in the phase-encode direction and which is more efficient.

SSFP requires that all gradient activity be rewound, or balanced. Conventionally, this is done by adding read, slice and phase rewinders. While these rewinders can be at least partially overlapped in time, it is not practical to collect data during this time, and in any case, they expose the patient to high dB/dt. By using a non-raster trajectory beginning and ending in the center of k-space, a teardrop readout requires neither read nor phase dephase/rephase lobes, increasing scan-time efficiency.

The name teardrop comes from the trajectory in k-space: leave the center of k-space on a radial trajectory, turn around and return on a radial trajectory (Fig. 1.1). The actual waveform is numerically generated and may circle the center of k-space one or more times, but is still referred to as a teardrop readout. By resampling the center of k-space at the beginning of every shot, the reconstruction can compensate for approach to steady state, and the sequence is less sensitive to motion artifacts.

The teardrop gradient waveform is in fact a continuous family of waveforms, one extreme of which integrates to describe a teardrop shaped k-space trajectory. Rotating the associated gradient profiles is equivalent to rotating

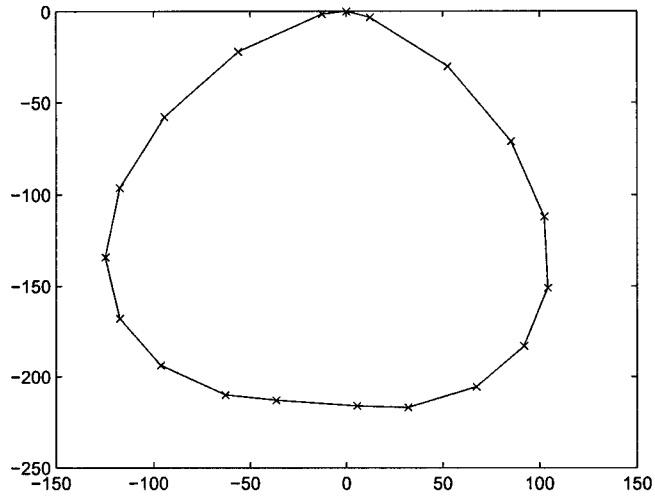

Figure 1.1: One teardrop trajectory in k-space.

this trajectory around the center of k-space. Together the combined views cover k-space, as evidenced by a partial set of views (Fig. 1.2).

An MRI technician could design a gradient waveform interactively to match the requested repetition time(TR) and resolution. The teardrop shape above is suitable for short TR and many interleaves. Scan-time efficiency can be increased by increasing the TR, and covering a larger region of k-space per repetition. This leads to increased sampling duty cycle and higher frame rates.

A number of techniques have been put forth to perform the complicated task of designing nulled moment gradient waveforms for motion compensation

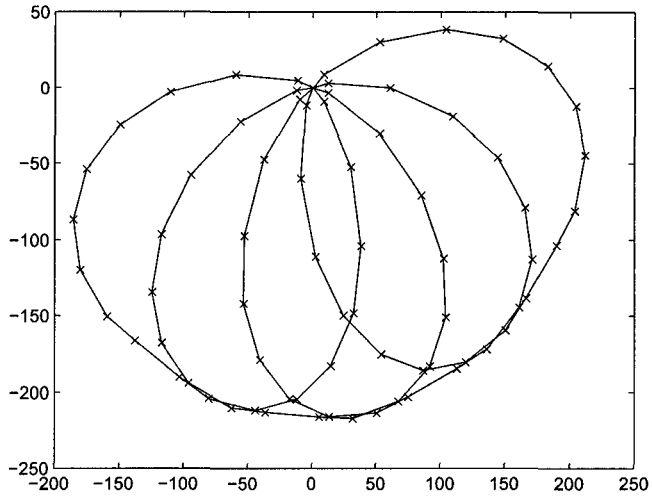

Figure 1.2: Rotating one teardrop trajectory around the center of k-space.

[10, 13, 17, 18, 22-24]. These are algebraic design methods. They take advantage of the linear relationship between gradient waveform time moments and lobe amplitudes to generate motion compensated or sensitized waveforms. There are several potential deficiencies that may limit their effectiveness in gradient waveform design. First, because the physical limitations of gradients and coils are not explicitly included in algebraic design techniques, they may prevent the attainment of specific imaging goals. Some computationally ineffective methods [7, 9, 22] iterate through algebraic solutions until feasible solutions are found. These methods do not guarantee that a realizable solution

will be found even if one exists. The resulting waveforms may not be optimal in any sense either.

It has been recognized that gradient waveform design can be generalized as a problem of non-linear constrained optimization [22], and there are some previous works have presented different methods to optimize gradient waveforms in different situations [2, 4, 6, 12, 15, 19, 20]. But many of these methods are limited to the design of trapezoidal pulses, and most have been studied for one dimensional (1D) gradient design. Up to now, no design formalism has been presented for optimal teardrop waveform design.

The goal of our project is to describe a general formalism for 2D and 3D optimal gradient waveform design as a convex-optimization problem using teardrop waveforms as examples, for which very efficient solution methods exist. This technique offers the following immediate practical benefits: it incorporates as much a priori knowledge of the gradient system as possible; it generates waveforms which are optimal for the imaging goals; it generates waveform shapes which are constrained only by the limitations of the hardware system; it guarantees the feasibility of the gradient waveforms. Finally, it can potentially reduce waveform design time by avoiding iterative or trial and error methods for determining feasible solutions.

The structure of this thesis is as follows. Chapter 2 reviews some basic

MRI concepts that are needed to understand this project. The methods of formulating optimization models of MRI teardrop gradient waveforms are presented in Chapter 3. This discussion includes: teardrop waveform parametrization, the design objective, constraints, the original optimization model, and the improved optimization models. Following this, Chapter 4 details the implementation of the models in AMPL. Computational results are presented in Chapter 5. Chapter 6 closes the thesis with conclusions and recommendations for future work.

Chapter 2

MRI Basics

In this chapter, we review some basic MRI concepts that will be useful for understanding the ideas that follow in the subsequent chapters. More detail is available in texts on MRI, including [16].

2.1 Classical Description of MRI

Atoms with an odd number of protons or odd number of neutrons possess a nuclear spin angular momentum, and therefore exhibit the Magnetic Resonance phenomenon. These nucleons can be visualized as spinning charged spheres that give rise to a small magnetic moment. We refer to these MR-relevant nuclei as spins. Because the body is composed of tissues primarily consisting of water and fat, both of which contain hydrogen, in biological specimens,

hydrogen (^{1}H), with a single proton, is the most abundant spin and therefore produces the strongest signal. That is why we usually assume (^{1}H) imaging.

The principle of MRI is based on the interaction of these spins with three types of magnetic fields: 1) the main field B_0, 2) the radio frequency field B_1, and 3) the gradient fields G.

In the absence of an external magnetic field, the spins are oriented randomly and the net magnet momentic is zero. However, in the presence of an external magnetic field B_0, the spins will align with or align against the external magnetic field. But the net magnetic moment vector is an alignment with the external field. Also, the nuclear spins exhibit resonance at the Larmor frequency ω and

$$\omega = \gamma B, \tag{2.1}$$

where B is the applied magnetic field, γ is called the gyromagnetic ratio, a known constant unique for each type of atom. For ^{1}H, $\gamma/2\pi = 42.58$ MHz/Tesla.

To obtain an MR signal, a radiofrequency (RF) magnetic pulse B_1 tuned to the resonant frequency of the spins is applied in the transverse plane to excite these spins out of equilibrium. B_1 applies a torque that rotates the magnetization vector by a prescribed angle dependent on the strength and duration of B_1. For example, if the excitation is set to a 90^o tip angle, then upon turning the excitation off, the tipped vector precess in the transverse plane at

the Larmor frequency generating an electromotive force (EMF) signal in an RF receiver coil. The generated time signal is called a free induction decay (FID). In general, a set of FIDs will be recorded and processed to reconstruct and MR image.

In the context of imaging, the body can be visualized as an ensemble of tiny oscillators inducing RF signals. The objective of MR imaging then is to map the spatial distribution for their amplitudes. Spatial localization is based on applying gradient fields to control the relative phases and frequencies of these oscillators.

2.2 The k-Space and Basic 2D k-Space Patterns

By MR theory [16], we can derive the equations for the MR signals:

$$s(t) = \int_x \int_y m(x,y) e^{-i2\pi[k_x(t)x + k_y(t)y]} \, \mathrm{d}x\mathrm{d}y, \tag{2.2}$$

where

$$k_x(t) = \frac{\gamma}{2\pi} \int_0^t G_x(\tau) \, \mathrm{d}\tau \tag{2.3}$$

$$k_y(t) = \frac{\gamma}{2\pi} \int_0^t G_y(\tau) \, \mathrm{d}\tau \tag{2.4}$$

$m(x, y)$ is the transverse nuclear magnetization, and G_x, G_y are gradient fields in the x and y directions respectively.

Comparing the signal equation (2.2) with the 2D Fourier transform of $m(x, y)$,

$$\mathcal{M}(k_x, k_y) = \int_x \int_y m(x, y) e^{-i2\pi[k_x x + k_y y]} \, \mathrm{d}x\mathrm{d}y, \tag{2.5}$$

we can see that

$$s(t) = \mathcal{M}(k_x, k_y), \tag{2.6}$$

or

$$s(t) = \mathcal{M}\left(\frac{\gamma}{2\pi}\int_0^t G_x(\tau)\,\mathrm{d}\tau, \frac{\gamma}{2\pi}\int_0^t G_y(\tau)\,\mathrm{d}\tau\right). \tag{2.7}$$

Thus, k_x and k_y are in units of spatial frequency, typically cycles/cm.

This is the most important relationship in MR imaging. At any given time t, $s(t)$ equals the value of the 2D Fourier transform of $m(x, y)$ at some spatial frequency. The total recorded signals $s(t)$ therefore maps directly to a trajectory through the spatial-frequency (Fourier transform) space as determined by the time integrals of the applied gradient waveforms $G_x(t)$ and $G_y(t)$. In the MR literature [14] and [21], the Fourier transform space is often called k-space, where k represents the spatial-frequency variable. To form an image, the trajectories given by $\{s(t)\}$ should cover a sufficient part of the k-space to allow reconstruction of $m(x, y)$.

The MRI signal provides information about the spatial frequency content of the image, rather than directly about the spatial positioning of information in the image. A computer has to be used to sample the information and apply the Fourier transform to the obtained signal to produce the image. A variety of patterns have been developed for sampling and image reconstruction.

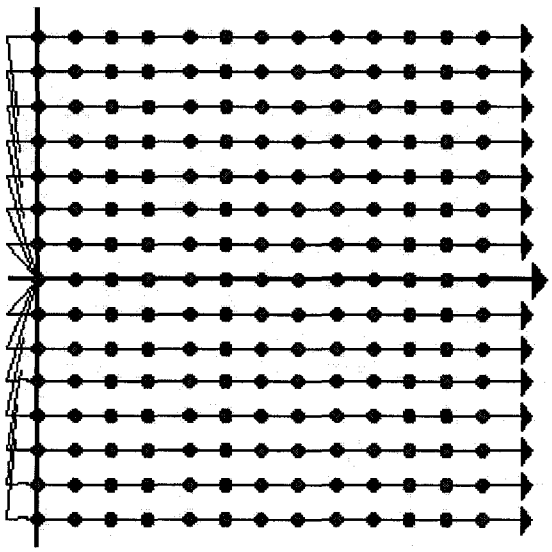

Figure 2.1: Raster scanning pattern.

The rectangular raster scan (Fig. 2.1) is used by standard two-dimensional Fourier transform (2DFT) imaging. Beginning at the origin, the k-space trajectory moves along the k_x direction as the signal is read out. A change in the amplitude of the G_y gradient leads to a different line in the k-space. By indexing to a set of k_y-positions, we can assemble sufficient measurements to fill the 2D k-space, and simply perform an inverse 2D Fourier transform to reconstruct the image.

Because of the specific roles that these gradients play, the G_y gradient is often referred to as the *phase-encoding gradient*, the G_x is called the *readout gradient* because it is on during data acquisition. The third gradient G_z is called the *slice-select gradient.*

Projection reconstruction imaging was the first method of MRI and is still used in some special applications. The sampling method for projection reconstruction is called the *radial pattern.* The *radial pattern* is shown in Fig. 2.2. The measurements are along diameters of a circle. The scan lines pass through the origin and are equally spaced in angle.

There are a number of relatively exotic methods of scanning the k-space. For example, *spiral scanning* and *square spiral scanning* (Fig. 2.3). In general, these data are resampled onto a rectangular grid and then reconstructed by using the inverse Fourier transform.

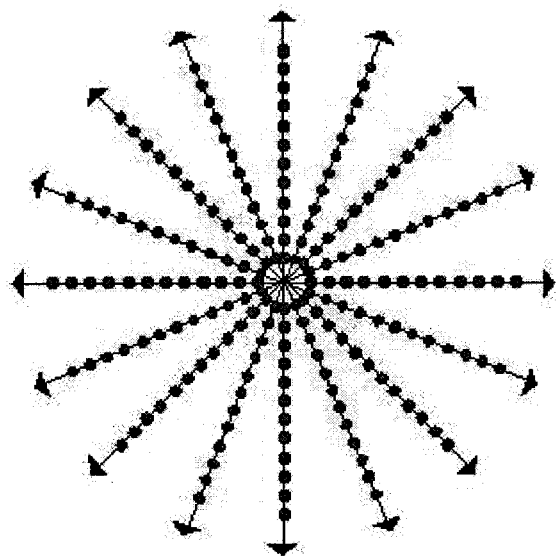

Figure 2.2: Radial scanning pattern.

Because the location of each measurement is controlled by the magnetic field gradients, the gradient system must be able to generate the chosen trajectories.

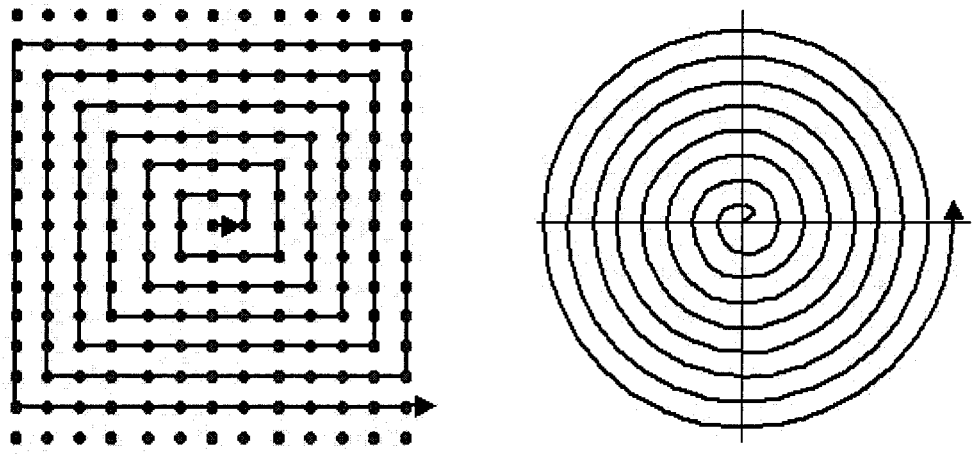

Figure 2.3: Square spiral and spiral scanning patterns.

2.3 Sampling Requirements and Artifacts in 2DFT Imaging

In practice, the continuous MRI signals are sampled as shown in Fig. 2.4. The sampling periods are Δk_x and Δk_y.

Let's only consider one-dimensional sampling here. From sampling theory we know that

$$\frac{1}{\Delta_{k_x}} = \text{FOV}. \tag{2.8}$$

Equation (2.8) means that the uniform spacing between data points Δ_{k_x} equals to 1/FOV, where FOV is the spatial interval over which the reconstructed

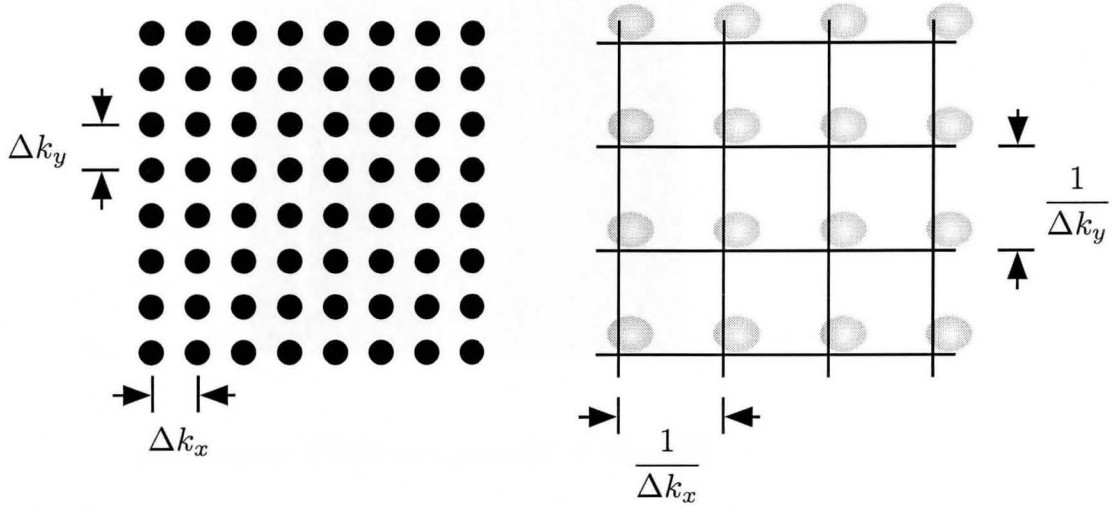

Figure 2.4: (a) Sampling in k-space (b) The replication in object domain.

image repeats itself. This interval is called the Field-Of-View (FOV).

The two most common artifacts particular to 2DFT images are the consequences of undersampling and motion. Undersampling means that not enough cycles are used in the data collection. This results in the samples not being sufficiently close in k-space. When this happens, the bottom of the image appears to wrap around the top (Fig. 2.5), this artifact is called aliasing. The cure for this is to increase the number of cycles in the data collection stage or reduce the hardware zoom factor.

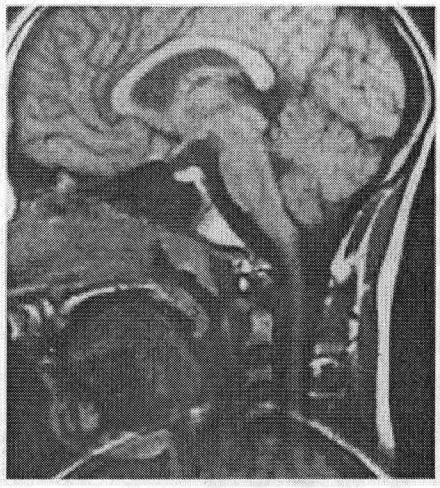

Figure 2.5: Aliasing in a 2DFT image.

To avoid aliasing artifact, the Nyquist sampling criterion has to be met.

$$\Delta_k < \frac{1}{\mathrm{S}}, \tag{2.9}$$

where S is the size of the object imaged. In one dimension, inequality 2.9 means that the FOV must be larger than S, the size of the object imaged.

The effects of the subjects motion on the 2DFT image are complicated. Fig. 2.6 shows the effects of motion on 2DFT images, where A is an ungated cardiac image, and B is a gated cardiac image. Cardiac gating is a kind of method that synchronizes the heartbeat with the beginning of the TR (repetition time) in order to minimize motion artifacts. The gated image shows

improvement in the motion artifacts.

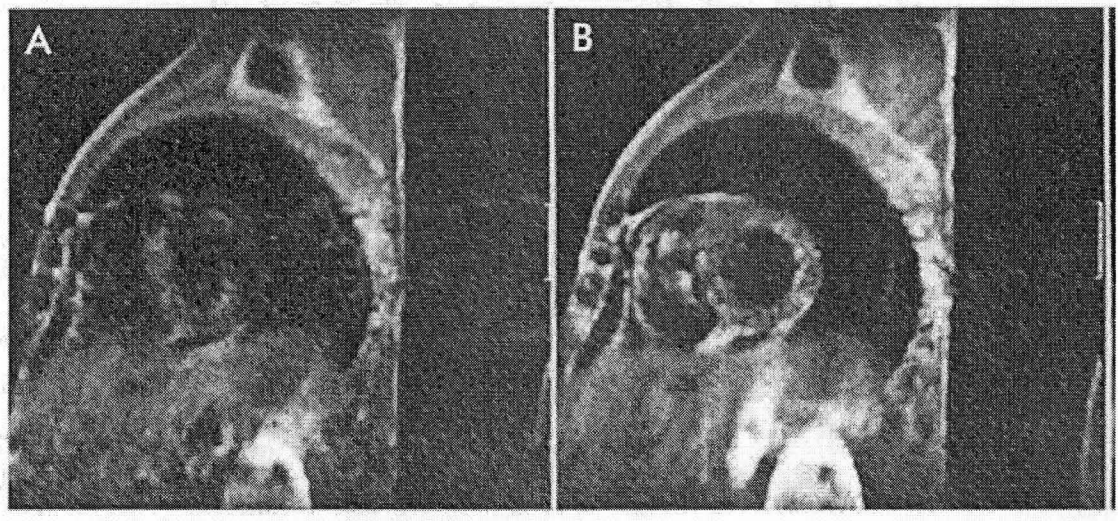

Figure 2.6: The effects of motion on 2DFT images.

Chapter 3

Nonlinear Optimization

A goal of gradient waveform design is to optimally meet some specific imaging performance criteria without violating the constraints imposed by hardware limitations of the gradient system and the sequence constraints, such as the desired shape of the scanning trajectory in k-space etc. So the waveform design problem can be viewed as a constrained optimization problem. Formulation of any constrained optimization problem involves four basic steps:

1. Select a variable set.

2. Define the objective.

3. Specify the constraints.

4. Determine a suitable means of solution.

In this chapter, the first three steps will be applied to the teardrop gradient waveform design, in other words, optimization models will be developed for this problem. The suitable means of solutions will be discussed in Chapter 5.

3.1 Teardrop Waveform Parametrization

The design objectives and constraints must be expressed in terms of a variable set which fully describes the gradient waveform. The ability to achieve a feasible solution is highly dependent on the choice of a variable set.

Since gradient waveforms are computer generated discrete functions and applied to the gradient coils via digital-to-analog (D/A) converters, the most obvious and general variable set is a discrete series of point-by-point gradient amplitudes. This set may be expressed in vector form as

$$\vec{g} = [g_1, g_2, \cdots, g_i, \cdots, g_{n+1}]^{\mathrm{T}} \in \mathcal{R}^{2n+2}, \tag{3.1}$$

where $g_i \in \mathcal{R}^2$ indicates the amplitude of the gradient waveform at time $t_i = i\triangle t$, and $\triangle t$ is the sample interval. The amplitudes at the $n+1$ points define the gradient waveform. Any gradient waveform shape can be expressed by such a gradient amplitude vector, thereby removing artificial constraints on shape and this way expanding the feasible solution space to the limits of the gradient hardware system.

To present objective and constraints conveniently and efficiently in the teardrop gradient waveform design, we add another variable set $\vec{k}$ which stands for a set of points in k-space.

$$\vec{k} = [k_1, k_2, \cdots, k_i, \cdots, k_n]^{\mathrm{T}} \in \mathcal{R}^{2n}, \tag{3.2}$$

where

$$k_i = \sum_{j=1}^{i} g_j \quad i \in [1, \cdots, n]. \tag{3.3}$$

Observe that there is one more point in the gradient amplitude space than in k-space.

3.2 Objective: Maximize Resolution

The goal of waveform design can be expressed by an objective function in terms of a variable set. In k-space, low frequencies are near the center of k-space, higher spatial frequencies are towards the edges. We know that small structures and fine details of an image contain high spatial frequencies. So higher spatial frequencies give better spatial resolution. Thus, if we want a sharp image, we have to measure not only the low spatial frequencies but higher ones as well.

Fig. 3.1 displays the same image of a head measured with different

numbers of spatial frequencies. It is obvious that the way to improve the resolution is to measure higher spatial frequencies.

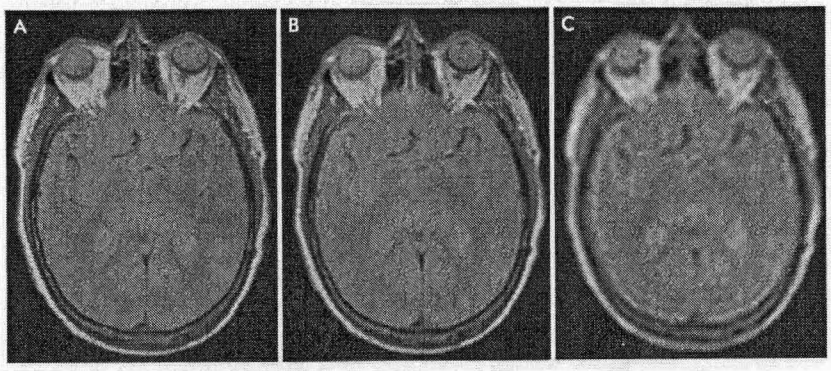

Figure 3.1: 256, 128, and 64 phase-encoding step acquisitions of a transverse image of a head.

By MRI theory, we know that the distance from the center of k-space to the farthest point in k-space equals the resolution of the image. Our design goal is to maximize resolution. So our objective function can be expressed as

$$\max \quad ||k_{n/2}||_2 \quad \text{if } n \text{ is even} \tag{3.4}$$

$$\text{or} \quad \max \quad ||k_{(n-1)/2}||_2 + ||k_{(n+1)/2}||_2 \quad \text{if } n \text{ is odd} \tag{3.5}$$

3.3 Constraints

The constraints of an optimization problem are the conditions which define the feasible set of solutions. According to our specific teardrop gradient waveform

design, the gradient system hardware limitations, waveform characteristics required to generate a teardrop k-space trajectory, and first moment nulling form a set of constraints for the design of teardrop gradient waveforms.

3.3.1 Gradient System Hardware Limitations

Amplitude Limits

Gradient amplifiers have peak current limits which restrict the maximum absolute value of gradient waveform amplitude. These limits can be expressed as inequality range constraints on each of the $n + 1$ points in the discrete waveform sequence as

$$||g_i||_2 \leq G_{\max}, \quad i \in [1, \cdots, n+1], \tag{3.6}$$

where $G_{\max}$ is the maximum allowable gradient amplitude.

Slew or Rise Time Limits

Gradient amplifiers also have limits on slew rate or rate of change of amplitude. This can be approximated as an inequality constraint on the first-order differences between adjacent discrete points as

$$||g_{i+1} - g_i||_2 \leq S_{\max}\Delta t, \quad i \in [1, \cdots, n+1], \tag{3.7}$$

where Δt is the sample interval.

Gradient Start and End Amplitudes

Because gradient amplifiers are switched off at the beginning and at the end, so gradient amplitudes are zeros at start and end.

$$g_1 = 0 \tag{3.8}$$

$$g_{n+1} = 0 \tag{3.9}$$

3.3.2 Constraints to Keep a Teardrop Shape of k-Space Trajectory

The Beginning and the End of k-space Trajectory

The name teardrop comes from the shape of the trajectory in k-space: it leaves the center of k-space, become tangent to a circle at the required resolution, and returns on the mirror-image trajectory to the center of k-space. So the beginning and the end of a k-space trajectory are at the center of k-space, i.e.,

$$k_1 = 0 \tag{3.10}$$

$$k_n = 0. \tag{3.11}$$

The constraint (3.11) is required for SSFP imaging.

The Spiral Constraint

This constraint family is meant to insure that the interleaves of the trajectories in k-space do not become too separated from their neighbors. Suppose the designed trajectory is along a standard spiral which can be described by a polar equation

$$r = \alpha\theta, \tag{3.12}$$

where r is the radial distance, θ is the polar angle, and α is a constant. When applying the spiral constraint to the trajectory, the constraint should be expressed as follows:

$$r' \leq \alpha\theta'. \tag{3.13}$$

It means that the ratio of the radial derivative and the angular derivative (measured in radians) should be bounded by a constant.

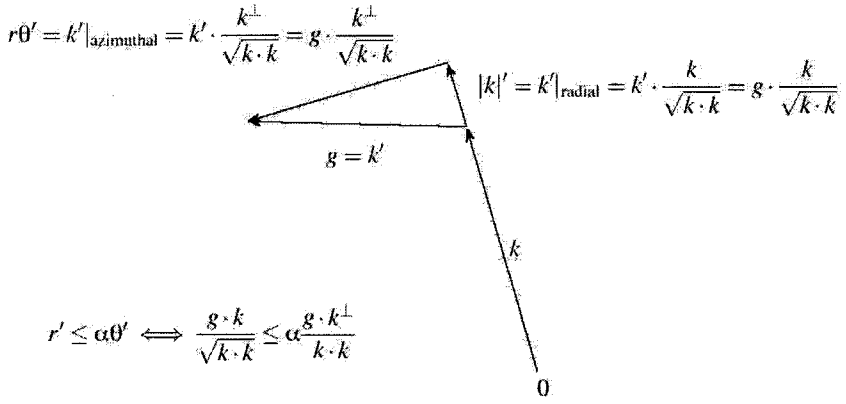

Figure 3.2: The deduction of the spiral constraint.

From Fig. 3.2, we know that the radial derivative r' at a point $k = (k_x, k_y)$ is $k \cdot g/||k||$ where $g = k'$ and $k \cdot g$ indicates the dot-product of the vectors k and g. The angular derivative is $k^{\perp} \cdot g/||k||^2$, where $k^{\perp} = (-k_y,\ k_x)$ is the perpendicular vector of k. Substituting these in the constraint (3.13), we get

$$r' \leq \alpha\theta' \Leftrightarrow \frac{k \cdot g}{\sqrt{k \cdot k}} \leq \alpha \frac{k^{\perp} \cdot g}{k \cdot k}. \tag{3.14}$$

By squaring both sides of this formula, the constraint family can be written as

$$(k_i \cdot g_i)(k_i \cdot g_i)(k_i \cdot k_i) \leq \alpha^2 (k_i^{\perp} \cdot g_i)(k_i^{\perp} \cdot g_i) \quad i \in [1, \cdots, n]. \tag{3.15}$$

3.3.3 First Moment Nulling

In SSFP imaging techniques, we do not dephase (destroy) the magnetization from one readout to the next, but keep modifying it with new RF pulses. One important fact about SSFP imaging is that, since the magnetization is never reset as it is done in conventional techniques, errors will build up over time, which means that motion artifacts can be more of a problem. To make the readout gradient motion-insensitive, we should zero the first moment, which is a global constraint

$$\sum_{i=1}^{n+1} i g_i = 0. \tag{3.16}$$

3.4 The Original Optimization Model

Combining the objective and all the constraints together, we get the following optimization model when n is even:

$$\max \quad ||k_{n/2}||_2 \tag{3.17a}$$

$$\text{subject to } k_1 = 0 \tag{3.17b}$$

$$k_n = 0 \tag{3.17c}$$

$$g_1 = 0 \tag{3.17d}$$

$$g_{n+1} = 0 \tag{3.17e}$$

$$k_i = \sum_{j=1}^{i} g_j \quad i \in [1, \cdots, n] \tag{3.17f}$$

$$||g_i||_2 \leq G_{\max}, \quad i \in [1, \cdots, n+1] \tag{3.17g}$$

$$||g_{i+1} - g_i||_2 \leq S_{\max}\Delta t, \quad i \in [1, \cdots, n+1] \tag{3.17h}$$

$$\sum_{i=1}^{n+1} i g_i = 0 \tag{3.17i}$$

$$(k_i \cdot g_i)(k_i \cdot g_i)(k_i \cdot k_i) \leq \alpha^2 (k_i^{\perp} \cdot g_i)(k_i^{\perp} \cdot g_i) \quad i \in [1, \cdots, n], \tag{3.17j}$$

where n is an even number, Δt is the sample interval, and α is a constant.

When n is odd, the objective becomes:

$$\max \quad ||k_{(n-1)/2}||_2 + ||k_{(n+1)/2}||_2 \tag{3.18}$$

while the constraints are the same.

These are the original models. Our goal is to maximize the resolution while satisfying all the constraints imposed by the hardware limitations of the gradient system, the teardrop shape of the scanning trajectory in k-space and first moment nulling.

3.5 The Improved Optimization Model

The most complicated constraint in the original model is the last one: the spiral constraint (3.17j). By using the arctangent function, we can simplify this constraint significantly. From Fig. 3.3, we know that

$$k_{i-1}^{\perp} \cdot k_i = |k_{i-1}||k_i| \cos(90^o - \Delta\theta) = |k_{i-1}||k_i| \sin \Delta\theta, \tag{3.19}$$

$$k_{i-1} \cdot k_i = |k_{i-1}||k_i| \cos \Delta\theta, \tag{3.20}$$

and

$$\Delta\theta = \text{atan}(k_{i-1}^{\perp} \cdot k_i,\ k_{i-1} \cdot k_i), \tag{3.21}$$

where atan represents inverse tangent function.

Then we have

$$r' \leq \alpha\theta' \Rightarrow \Delta r \leq \alpha\Delta\theta, \tag{3.22}$$

$$\Delta r \leq \alpha\Delta\theta \Leftrightarrow ||k_i||_2 - ||k_{i-1}||_2 \leq \alpha \text{ atan}(k_{i-1}^{\perp} \cdot k_i, k_{i-1} \cdot k_i). \tag{3.23}$$

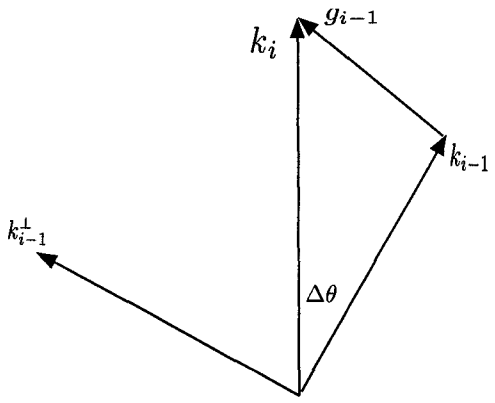

Figure 3.3: The illustration of the simplification of the spiral constraint.

As a result, the new constraints in the improved optimization model that replace (3.17j) are

$$||k_i||_2 - ||k_{i-1}||_2 \leq \alpha \operatorname{atan}(k_{i-1}^{\perp} \cdot k_i, k_{i-1} \cdot k_i), \quad i \in [1, \cdots, n], \tag{3.24}$$

while all other constraints and the objective function are the same. When n is odd, we should use formula (3.18) as the objective function.

3.6 The Second Improvement

The spiral constraints (3.24) from the improved optimization model are not convex and can not be generalized to the 3D case, so we want to improve our model again by modifying the spiral constraint which will require an iterative

algorithm to solve a sequence of nonlinear problems. To replace the spiral constraint in the previous models, we need to add another set of points in k-space as our variables:

$$\vec{k''} = [k''_1, k''_2, \cdots, k''_i, \cdots, k''_n]^{\mathrm{T}} \in \mathcal{R}^{2n}. \tag{3.25}$$

Then we can use the following two constraints to replace the spiral constraints:

$$k''_i = \begin{pmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{pmatrix} k_i, \tag{3.26}$$

where $\phi = \frac{2\pi}{\text{the number of interleaves}}$. This constraint means that one trajectory in k-space is the rotation of another trajectory. The second constraint is:

$$\min_j \text{ distance}(k''_i, \overline{k_j k_{j+1}}) = \beta_{i,j} \leq \text{constant}. \tag{3.27}$$

We can explain this constraint by Fig. 3.4 . Here $\beta_{i,j}$ is the distance from the

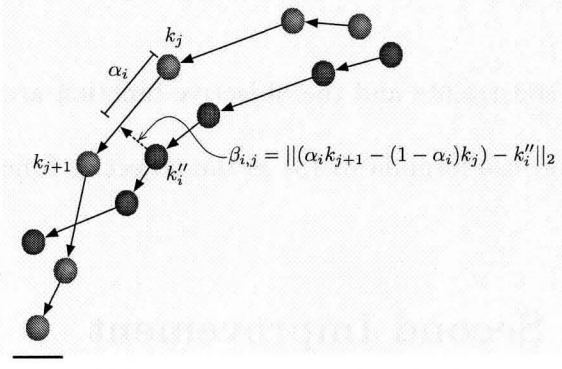

Figure 3.4: Limit the distances between two trajectories.

point k_i'' to the segment $\overline{k_j k_{j+1}}$ in another trajectory and

$$\beta_{i,j} = ||\alpha_i \, k_{j+1} + (1 - \alpha_i) \, k_j - k_i''||_2, \tag{3.28}$$

where

$$\alpha_i = \begin{cases} \frac{(k''_{i,old} - k_{j,old}) \cdot (k_{j+1,old} - k_{j,old})}{||k_{j+1,old} - k_{j,old}||_2} & \text{when } 0 < \alpha_i < 1 \\ 1 & \text{when } \alpha_i \geq 1 \\ 0 & \text{when } \alpha_i \leq 0 \end{cases} \tag{3.29}$$

and $k''_{i,old}$, $k_{j,old}$ $\quad i, j \in [1, \cdots, n]$ are solutions of the previous iteration or the initials for the first iteration. If $\alpha_i \geq 1$ or $\alpha_i \leq 0$, then the closest point to k_i'' on the line through $\overline{k_j k_{j+1}}$ is not on the segment. In these cases, we choose the distance from k_i'' to the closest endpoint k_j ($\alpha_i = 0$) or k_{j+1} ($\alpha_i = 1$) to be the distance $\beta_{i,j}$.

Keeping all other constraints and the objective function the same as before, we can get our new model when n is even:

$$\max \quad ||k_{n/2}||_2 \tag{3.30a}$$

$$\text{subject to } k_1 = 0 \tag{3.30b}$$

$$k_n = 0 \tag{3.30c}$$

$$g_1 = 0 \tag{3.30d}$$

$$g_{n+1} = 0 \tag{3.30e}$$

$$k_i = \sum_{j=1}^{i} g_j \quad i \in [1, \cdots, n] \tag{3.30f}$$

$$||g_i||_2 \leq G_{\max}, \quad i \in [1, \cdots, n+1] \tag{3.30g}$$

$$||g_{i+1} - g_i||_2 \leq S_{\max}\Delta t, \quad i \in [1, \cdots, n+1] \tag{3.30h}$$

$$\sum_{i=1}^{n+1} i g_i = 0 \tag{3.30i}$$

$$k_i'' = \begin{pmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{pmatrix} k_i, \quad i \in [1, \cdots, n] \tag{3.30j}$$

$$||\alpha_i \, k_{j+1} + (1 - \alpha_i) \, k_j - k_i''||_2 \leq d, \tag{3.30k}$$

$$i \in [2, \cdots, n], \quad j \in [1, \cdots, n],$$

where n is an even number, Δt is the sample interval, and d is a constant.

When n is odd, you should use formula (3.18) as your objective function.

Now we design an iterative algorithm to solve a sequence of nonlinear problems (3.30). The general idea is that of at each iteration we choose the previous solutions as our starting point, then for each point k_i'' in one trajectory, we look for the segment in another trajectory which is nearest to the point k_i'' based on the results from the previous iteration, and we limit this distance to be less than or equal to a constant. This constant can be decided by the Nyquist sampling theory and field of view to avoid aliasing artifacts. The purpose of these constraints are to avoid big holes between neighboring trajectories. By repeating the process, we expect the distances between two trajectories to become almost the same everywhere, and the objective function

will become larger and larger until the results converge to an optimal value. See Fig. 3.5 for a pseudocode description.

3.7 The Third Improvement

We cannot guarantee the result of Model (3.30) and the iterative algorithm (Fig. 3.5) is a global optimal solution because the model is not convex. In this section, we present how to change the subproblem Model (3.30) to a convex problem while keeping the iterative algorithm the same as before. To reformulate the model as a convex model, we need to change the non-convex objective function in the model.

Another modification we make is to reduce the design variables by half because of the symmetry of teardrop trajectories. Suppose the number of g_i is $n = 2N + 1$ (odd number), then k_N is the middle point of the teardrop trajectory. In our new model, we only need to optimize $N + 1$ points (half of the trajectory), then we get the another half of the trajectory by reflecting.

By introducing a new variable τ, we can replace the old objective function by

$$\max \quad \tau \tag{3.31a}$$

$$\text{subject to } \tau \leq k_N \cdot (k_{N,old}/||k_{N,old}||_2), \tag{3.31b}$$

Input:

initial values $k_{i,old},\ k''_{i,old} \quad i \in [1, n]$;

tolerance: a parameter which stands for the tolerance of the objective function.

begin

iter := 1;

previous-obj := 0;

$k_i := k_{i,old},\ k''_i := k''_{i,old} \quad i \in [1, \cdots, n]$;

calculate α_i by using Formula (3.29);

solve;

repeat

{ $k_i := k_{i,old},\ k''_i := k''_{i,old} \quad i \in [1, \cdots, n]$;

recalculate α_i by using Formula (3.29);

previous-obj := current-obj;

solve the optimization problem (3.30);

$i := i + 1$;

} until current-obj <= previous-obj + tolerance.

end.

end.

Figure 3.5: The pseudocode of the iterative non-linear method.

where $k_{N,old}$ is k_N from the last iteration.

The new objective function and the constraint is linear. This new objective expression and the constraint together have the same purpose as the old objective expression, they are trying to expand the trajectory in k-space.

To get another half trajectory by reflecting, we have to add a symmetric constraint: $||k_N||_2 = ||k_{N+1}||_2$. This is a non-convex constraint. We can change the symmetric constraint to a convex constraint by using the following method.

Let

$$\bar{k} = \frac{1}{2}(k_{N,old} + k_{N+1,old}). \tag{3.32}$$

where $k_{N,old}$, $k_{N+1,old}$ are the solutions from the previous iteration (or the initial values for the first iteration). Instead of using $||k_N||_2 = ||k_{N+1}||_2$ in the symmetric constraint, we use

$$k_N \cdot \bar{k} = k_{N+1} \cdot \bar{k}. \tag{3.33}$$

Then this constraint is linear.

Keeping all other constraints the same as before, we can get the following convex model for the 2D case:

$$\max \quad \tau \tag{3.34a}$$

$$\text{subject to } \tau \leq k_N \cdot (k_{N,old}/||k_{N,old}||_2) \tag{3.34b}$$

$$k_1 = 0 \tag{3.34c}$$

$$g_1 = 0 \tag{3.34d}$$

$$k_i = \sum_{j=1}^{i} g_j, \quad i \in [1, \cdots, N] \tag{3.34e}$$

$$||g_i||_2 \leq G_{\max}, \quad i \in [1, \cdots, N+1] \tag{3.34f}$$

$$||g_{i+1} - g_i||_2 \leq S_{\max}\Delta t, \quad i \in [1, \cdots, N+1] \tag{3.34g}$$

$$k_N \cdot \bar{k} = k_{N+1} \cdot \bar{k} \tag{3.34h}$$

$$\sum_{i=1}^{2N} i g_i = 0 \tag{3.34i}$$

$$k_i'' = \begin{pmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{pmatrix} k_i, \quad i \in [1, \cdots, N] \tag{3.34j}$$

$$||\alpha_i \cdot k_{j+1} + (1 - \alpha_i)\, k_j - k_i''||_2 \leq d, \tag{3.34k}$$

$$i \in [2, \cdots, N], \quad j \in [1, \cdots, N],$$

where Δt is the sample interval, d is a constant, and $\bar{k} = \frac{1}{2}(k_{N,old} + k_{N+1,old})$. This model is a good foundation for the optimization model in the three dimensional space. We will develop the 3D model in next section.

3.8 The Optimization Model for the 3D Case

3.8.1 Review of 3D k-Space Acquisition

There are a variety of volumetric imaging methods available in MRI. Some methods get volumetric data through the acquisition of many 2D slices while other methods obtain data in 3D k-space directly by appropriate modulation of all three gradients.

The extension of the signal equation from 2D imaging to 3D imaging is straightforward. With all three gradients involved, the resultant signal can be expressed as

$$s(t) = \int_x \int_y \int_z m(x,y,z) e^{-i2\pi[k_x(t)x + k_y(t)y + k_z(t)z]} \,\mathrm{d}x\mathrm{d}y\mathrm{d}z, \tag{3.35}$$

where

$$k_x(t) = \frac{\gamma}{2\pi} \int_0^t G_x(\tau)\,\mathrm{d}\tau \tag{3.36}$$

$$k_y(t) = \frac{\gamma}{2\pi} \int_0^t G_y(\tau)\,\mathrm{d}\tau \tag{3.37}$$

$$k_z(t) = \frac{\gamma}{2\pi} \int_0^t G_z(\tau)\,\mathrm{d}\tau. \tag{3.38}$$

The trajectory through k-space is now along all three dimensions. There are a lot of possible ways of filling 3D k-space. Let us review some specific 3D k-space trajectories here. The most common 3D k-space imaging method is three-dimensional Fourier transform (3DFT). A simple extension of 2DFT,

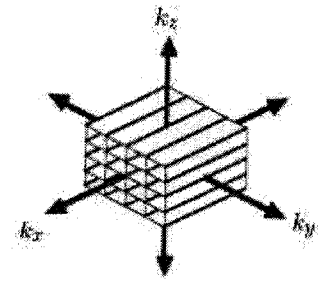

Figure 3.6: 3DFT k-space trajectory.

3DFT employs phase-encoding gradients along two axes, and a readout gradient along the third axis. From a k-space perspective, the G_y and G_z phase encoding lobes create movement to some (k_y, k_z) location prior to movement along the k_x axis when the G_x readout gradient and data acquisition turn on. The k-space trajectory consists of lines parallel to the k_x axis in a cartesian orientation (Fig. 3.6), convenient for image reconstruction through a 3DFT. Whereas 3DFT acquires data in a cartesian coordinate structure, 3D

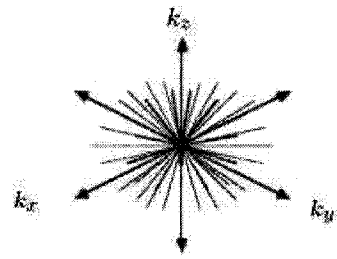

Figure 3.7: 3DPR k-space trajectory.

projection reconstruction (3DPR) acquires data in a 3D spherical coordinate structure. The readout direction varies in a manner to fill k-space with radial lines, typically uniformly distributed over a sphere, each line passes through the origin (Fig. 3.7).

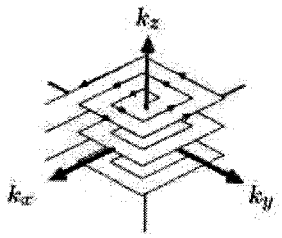

Figure 3.8: Hybrid spiral-scan 3D trajectory.

Given any 2D k-space acquisition for imaging a slice, one effective way to extend the acquisition into 3D k-space is to apply a phase-encoding gradient along the third axis. The 3D k-space filling therefore consists of a planar trajectory that is replicated in the third dimension, one for each phase encoding that is applied. Fig. 3.8 shows the 3D k-space trajectory for the example of a spiral-scan sequence. The 3D reconstruction operation is simply the appropriate 2D reconstruction operation for each of the 2D planes in k-space followed by Fourier transformation along the phase encoding axis with the set of 2D reconstructions.

One disadvantage of the above 3D imaging methods is that they require

rewinders for SSFP imaging, which reduce their sampling efficiency. Just like in 2D case, we still need to design our 3D teardrop gradient waveforms and k-space trajectories to obtain an optimal SSFP method in 3D.

3.8.2 The Optimization Model for 3D Teardrop Gradient Waveform Design

We can apply most of the ideas of the 2D model (3.34) to the 3D model straightforwardly except for constraints (3.34j) and (3.34k). The constraint (3.34j) means the trajectory k_i'' is the rotation of another trajectory k_i, and the constraint (3.34k) insures that these two trajectories are not separated by more than a fixed amount. But in the 3D case, the concept of rotation is not applicable. So we cannot design two neighboring trajectories then rotate them to get all the interleaves in the 3D case. To replace these two constraints in the 3D case, we have to design all the interleaves together in the 3D model.

Suppose we have N_t trajectories in the 3D k-space all together. Then we take the following set of points as our design variables:

$$\vec{k}^1 = [k_1^1, k_2^1, \cdots, k_i^1, \cdots, k_{N+1}^1]^{\mathrm{T}}, \tag{3.39}$$

$$\vec{k}^2 = [k_1^2, k_2^2, \cdots, k_i^2, \cdots, k_{N+1}^2]^{\mathrm{T}}, \tag{3.40}$$

$$\cdots$$

$$\vec{k}^{N_t} = [k_1^{N_t}, k_2^{N_t}, \cdots, k_i^{N_t}, \cdots, k_{N+1}^{N_t}]^{\mathrm{T}}. \tag{3.41}$$

For each pair of two neighboring trajectories (for example, k_i^{ℓ}, k_i^{p} are neighboring to each other), we add the following constraint (see Fig. 3.4):

$$\min_j \text{ distance}(k_i^{\ell}, \overline{k_j^p k_{j+1}^p}) \leq \text{constant}. \tag{3.42}$$

This constraint has the same meaning as constraint (3.27) for the 2D model. Its purpose is just to avoid big gaps between the neighboring trajectories. The constant can be decided by Nyquist sampling theory to avoid alias artifacts.

Then the optimization model for the 3D case is:

$$\max \quad \tau \tag{3.43a}$$

$$\text{subject to } \tau \leq k_N^m \cdot (k_{N,old}^m / ||k_{N,old}^m||_2), \quad m \in [1, \cdots, N_t] \tag{3.43b}$$

$$k_1^m = 0, \quad m \in [1, \cdots, N_t] \tag{3.43c}$$

$$g_1^m = 0, \quad m \in [1, \cdots, N_t] \tag{3.43d}$$

$$k_i^m = \sum_{j=1}^{i} g_j^m, \quad i \in [1, \cdots, N], \quad m \in [1, \cdots, N_t] \tag{3.43e}$$

$$||g_i^m||_2 \leq G_{\max}, \quad i \in [1, \cdots, N+1], \quad m \in [1, \cdots, N_t] \tag{3.43f}$$

$$||g_{i+1}^m - g_i^m||_2 \leq S_{\max}\Delta t, \quad i \in [1, \cdots, N+1], \quad m \in [1, \cdots, N_t] \tag{3.43g}$$

$$k_N^m \cdot \bar{k}^m = k_{N+1}^m \cdot \bar{k}^m, \quad m \in [1, \cdots, N_t] \tag{3.43h}$$

$$\sum_{i}^{2N} ig_i^m = 0, \quad m \in [1, \cdots, N_t] \tag{3.43i}$$

$$||\alpha_i \, k_{j+1}^p + (1 - \alpha_i) \, k_j^p - k_i^\ell||_2 \leq d, \tag{3.43j}$$

$$\forall i, j \in [2, \cdots, N], \; p, \ell \in [1, \cdots, N_t],$$

where Δt is the sample interval, d is a constant, $\bar{k}^m = \frac{1}{2}(k_{N,old}^m + k_{N+1,old}^m)$, where $m \in [1, \cdots, N_t]$, and trajectory p and trajectory ℓ are neighboring to each other.

This is a convex model of the subproblem in the 3D case. The design variables represent points in three dimensions. The iterative algorithm can be applied analogously as before. The idea of this model is almost the same as the 2D model, but the implementation is much more complicated. In Chapter 4, some details of implementation issues will be discussed.

Chapter 4

Implementation

To solve the models built in Chapter 3 by optimization software, we need to implement those models in a modeling language. We have implemented all the models in AMPL [8]. AMPL has been developed at Bell Laboratories. It allows the implementation of numerical experiments with familiar mathematical notation and concepts. Further, AMPL offers an interactive command environment for setting up and solving optimization problems. A flexible interface allows a user to choose from several solvers and to select options that improve the solver's performance. AMPL offers also various options to format data, for browsing or printing results.

In Chapter 3, we have presented all the models we have developed for this project. Model (3.34) and model (3.43) are the final convex models for the

2D and 3D cases. So only the implementation issues of these two models and the iterative algorithm of Fig. 3.5 involved in the numerical experiments will be addressed in this chapter. Our implementation is based on the formulations and equations that were detailed in the previous chapter.

4.1 The AMPL Model teardrop2D.mod

4.1.1 Setting up the Model

In this section, we will explain how to implement the objective function and all the constraints of the model (3.34) and its iterative algorithm (Fig. 3.5) in the AMPL code teardrop2D.mod.

The expressions in the objective and constraints necessarily involve variables and parameters. So we need to define those variables and parameters first in AMPL.

In model (3.34), we have the following variables: g_i, k_i, k_i'' and τ. In the AMPL model teardrop2D.mod, we declared these variables by:

```
var x{1..N+1};  # x of gi
var y{1..N+1};  # y of gi
var x1{1..N+1}; # x of ki
var y1{1..N+1}; # y of ki
```

```
var x2{1..N+1}; # x of k''i, rotation of x1

var y2{1..N+1}; # y of k''i, rotation of y1

var tao;  # the supplementary variable of objective function
```

These declarations create the indexed collection of variables for g_i, k_i and k_i' and a variable for τ.

There are two supplementary variables for setting up the quadratic constraints:

```
var minumber; # supplementary variable

var beta{2..N,1..N}; # distances between two trajectories,

                     # intermediate variables
```

They will be explained later.

The following are the parameter declarations in this model.

```
param N integer > 0; # total number of gi is 2*N+1

param pi >0; # the ratio constant of circumference to diameter

param K > 0; # a constant used in the initial condition

param M integer > 0; #number of interleaves

param mx ; #scalar in the initial condition

param TR >0;#readout interval for SSFP sequences, the unit is

            # s(second)
```

```
param GMax;  # maximum gradient of MRI machine, the unit is T/m

param SMax;  # maximum slew rate of MRI machine, the unit

             # is T/m/s

param gama;  # gyromagnetic ratio

param DelT := TR/(2*N+1); # the sampling interval, the unit is

                          # s(second)

param G1 := (gama * GMax * DelT)^2; # maximum gradient of

                                    # calculation

param S1 := (gama * SMax * DelT * DelT)^2;#maximum slew

                                          #rate of calculation

param x10{1..N+1}; # the initial value for x of ki

param y10{1..N+1}; # the initial value for y of ki

param x20{1..N+1}; # the initial value for x of k''i

param y20{1..N+1}; # the initial value for y of k''i

param alfa{2..N,1..N};

param beta1{1..N};

param jj;

param previous_obj default 0; #the value of previous

                              # objective function

param tolerance; # a tolerance for the objective function
```

```
param d; # the distance between two neighboring trajectories
param iter; # iteration number
```

There is a brief explanation for each parameter after the # in each declaration. Now we explain some more complicated parts here. Let us look at what `G1`, `S1` stand for first.

Let $t_0, t_1, \cdots, t_n$ represent the discrete time, then we use the notation

$$k_i = k(t_i). \tag{4.1}$$

In model (3.34), by (3.34e)we define the relation

$$k_i = \sum_{j=1}^{i} g_j.$$

Further,

$$g_i = k_i - k_{i-1}. \tag{4.2}$$

However, from MRI theory (Section 2.2), we know that

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

So we have

$$\begin{aligned} g_i &= k_i - k_{i-1} \\ &= \tfrac{\gamma}{2\pi} \textstyle\int_0^{t_i} G(\tau)\,\mathrm{d}\tau - \tfrac{\gamma}{2\pi} \int_0^{t_{i-1}} G(\tau)\,\mathrm{d}\tau \\ &= \tfrac{\gamma}{2\pi} \textstyle\int_{t_i}^{t_{i-1}} G(\tau)\,\mathrm{d}\tau \\ &\doteq \tfrac{\gamma}{2\pi} g(t_i)(t_i - t_{i-1}) \\ &= \tfrac{\gamma}{2\pi} g(t_i)\Delta t. \end{aligned}$$

Thus

$$g(t_i) \doteq \frac{2\pi}{\gamma}\frac{g_i}{\Delta t}. \tag{4.3}$$

Suppose that $G_{\max}$ represents the maximum gradient of a MRI machine, then constraint (3.34f) of model (3.34) should be

$$||g(t_i)||_2 \leq G_{\max}. \tag{4.4}$$

Then

$$\begin{aligned} ||g(t_i)||_2 \leq G_{\max} \quad &\Rightarrow \quad \tfrac{2\pi}{\gamma}\tfrac{||g_i||_2}{\Delta t} \leq G_{\max} \\ &\Rightarrow \quad ||g_i||_2 \leq \tfrac{\gamma}{2\pi} G_{\max}\Delta t, \end{aligned}$$

so

$$g_i^2 \leq (\frac{\gamma}{2\pi}G_{\max}\Delta t)^2. \tag{4.5}$$

In the AMPL model teardrop2D.mod, the formulation of (4.5) was used for the implementation of this constraint. In the AMPL model, we declared parameter `gama`=$\frac{\gamma}{2\pi}$ = 42576000HzT^{-1}, further parameter `GMax` represents $G_{\max}$, and parameter `Delt` represents Δt. We also defined another parameter `G1`=$(\frac{\gamma}{2\pi}G_{\max}\Delta t)^2$. So the constraint in AMPL is:

```
subject to gg{i in 1..N+1}:
        x[i]^2 + y[i]^2 <= G1 ;
```

Let $S_{\max}$ represent the maximum slew of a MRI machine, then the

actual meaning of constraint (3.34g) of model (3.34) is

$$||g(t_{i+1}) - g(t_i)||_2 \leq S_{\max}\Delta t. \tag{4.6}$$

Then

$$\begin{aligned} ||g(t_{i+1}) - g(t_i)||_2 \leq S_{\max}\Delta t \quad &\Rightarrow \quad \tfrac{2\pi}{\gamma}\tfrac{1}{\Delta t}(||g_{i+1} - g_i||_2) \leq S_{\max}\Delta t \\ &\Rightarrow \quad ||g_{i+1} - g_i||_2 \leq \tfrac{\gamma}{2\pi}S_{\max}(\Delta t)^2, \end{aligned}$$

so

$$(g_{i+1} - g_i)^2 \leq (\frac{\gamma}{2\pi}S_{\max}(\Delta t)^2)^2. \tag{4.7}$$

In the AMPL model teardrop2D.mod, the formulation of (4.7) was used for the implementation of this constraint. In the AMPL model, the parameter `SMax` represents $S_{\max}$, we also defined another parameter `S1`=$(\frac{\gamma}{2\pi}S_{\max}(\Delta t)^2)^2$. So the constraint in AMPL is:

```
subject to vv{i in 2..N+1}:
        (x[i]-x[i-1])^2 + (y[i]-y[i-1])^2 <= S1 ;
```

After defining the variables and parameters, the objective function and constraints (3.34a)-(3.34e), (3.34h) and (3.34j) can be expressed in AMPL straightforwardly. The AMPL code is as follows:

```
#objective function
maximize k:
tao;
```

```
subject to ob:
tao <= (x1[N]*x10[N]+y1[N]*y10[N])/sqrt(x10[N]*x10[N]+
        y10[N]*y10[N]);

# teardrop starts from the center of k-space
subject to x0:
    x1[1]=0;
subject to y0:
    y1[1]=0;

#the first point of gi is zero
subject to s0:
    x[1] = 0;
subject to sy0:
    y[1] = 0;

# define k[i] in terms of g[i]
subject to x1_def{i in 2..N+1}:
    x1[i]=x[i]+x1[i-1];
```

```
subject to y1_def{i in 2..N+1}:
        y1[i]=y[i]+ y1[i-1];

# For odd points of gi, ki is even. The symmetric constraint: kN+1=kN.
subject to symmetr:
x1[N]*(x10[N]+x10[N+1])+y1[N]*(y10[N]+y10[N+1])
=x1[N+1]*(x10[N]+x10[N+1])+y1[N+1]*(y10[N]+y10[N+1]);

#define x2,y2 in terms of x1,y1
subject to x2_def{i in 1..N+1}:
    x2[i] = x1[i]*cos(2*pi/M) - y1[i]*sin(2*pi/M);
subject to y2_def{i in 1..N+1}:
    y2[i] = x1[i]*sin(2*pi/M) + y1[i]*cos(2*pi/M);
```

Some further work is needed for efficient implementation of the first moment nulling constraint (3.34i). We introduce two vectors $\bar{k}$ and its perpendicular vector $\bar{k}^{\perp}$ here. $\bar{k}$ can be calculated by formula (3.32). Because of the first moment nulling constraint and the trajectory is symmetric, we know

$$g_{i+N} \cdot \bar{k} = -g_{N+2-i} \cdot \bar{k}, \tag{4.8}$$

and

$$g_{i+N} \cdot \bar{k}^{\perp} = g_{N+2-i} \cdot \bar{k}^{\perp}. \tag{4.9}$$

To satisfy the constraint (3.34i), it is necessary and sufficient to satisfy the following equalities:

$$\bar{k} \cdot (\sum_{i=1}^{2N} ig_i) = 0, \tag{4.10}$$

$$\bar{k}^{\perp} \cdot (\sum_{i=1}^{2N} ig_i) = 0. \tag{4.11}$$

Then we can make the following deduction from formula (4.10):

$$\begin{aligned}
\bar{k} \cdot (\sum_{i=1}^{2N} ig_i) &= \sum_{i=1}^{2N} ig_i \cdot \bar{k} \\
&= \sum_{i=1}^{N} ig_i \cdot \bar{k} + (N+1)g_{N+1} \cdot \bar{k} + \sum_{i=N+2}^{2N} ig_i \cdot \bar{k} \\
&= \sum_{i=1}^{N} ig_i \cdot \bar{k} + (N+1)g_{N+1} \cdot \bar{k} + \sum_{j=2}^{N} (N+j)g_{N+j} \cdot \bar{k} \\
&= \sum_{i=1}^{N} ig_i \cdot \bar{k} + (N+1)g_{N+1} \cdot \bar{k} - \sum_{j=2}^{N} (N+j)g_{N+2-j} \cdot \bar{k} \\
&= \sum_{i=1}^{N} ig_i \cdot \bar{k} + (N+1)g_{N+1} \cdot \bar{k} - \sum_{i=2}^{N} (N+i)g_{N+2-i} \cdot \bar{k}
\end{aligned} \tag{4.12}$$

We can make the same deduction from formula (4.11), finally we can get:

$$\bar{k}^{\perp} \cdot (\sum_{i=1}^{2N} ig_i) = \sum_{i=1}^{N} ig_i \cdot \bar{k}^{\perp} + (N+1)g_{N+1} \cdot \bar{k}^{\perp} + \sum_{i=2}^{N} (N+i)g_{N+2-i} \cdot \bar{k}^{\perp}. \tag{4.13}$$

Finally, we use formulas (4.12) and (4.13) in our implementation. The AMPL codes are as follows:

```
#First moment nulling constrain1

subject to null1:

sum{i in 1..N} i*(x[i]*(x10[N]+x10[N+1])+y[i]*
```

```
(y10[N]+y10[N+1])) + (N+1)*(x[N+1]*(x10[N]+x10[N+1])
+y[N+1]*(y10[N]+y10[N+1]))-sum{i in 2..N} (N+i)*(x[N+2-i]
*(x10[N]+x10[N+1])+y[N+2-i]*(y10[N]+y10[N+1]))=0;

#First moment nulling constrain2
subject to null2:
sum{i in 1..N} i*(x[i]*(y10[N]+y10[N+1])-y[i]*
(x10[N]+x10[N+1])) + (N+1)*(x[N+1]*(y10[N]+y10[N+1])
-y[N+1]*(x10[N]+x10[N+1]))+sum{i in 2..N} (N+i)*(x[N+2-i]
*(y10[N]+y10[N+1])-y[N+2-i]*(x10[N]+x10[N+1]))=0;
```

The hardest part is how to express the quadratic constraint (3.34k) in AMPL. In Section 3.6 we already described the meaning of this constraint. For each $k_i^{''\ell}$ $i \in [2, \cdots, N]$ in a trajectory ℓ, look for the segment $\overline{k_j^p k_{j+1}^p}$ which is the nearest to the point in another trajectory p, then we limit the distance to be less than a constant, i.e.,

$$\beta_{i,j} = ||\alpha_i \, k_{j+1}^p + (1 - \alpha_i) \, k_j^p - k_i^{''\ell}||_2 \leq d.$$

To look for the nearest segment $\overline{k_j^p k_{j+1}^p}$ of trajectory p to $k_i^{''\ell}$, we defined two dimensional parameters `alfa{2..N,1..N}` to represent α_i. For example, `alfa[2,10]` equals to α_i that is used to calculate the distance from

the point $k_2^{''\ell}$ to the segment $\overline{k_{10}^p k_{11}^p}$. We also defined two dimensional variables `beta{2..N,1..N}` to represent $\beta_{i,j}$. For example, `beta[2,10]` equals the distance $\beta_{2,10}$ which is the distance from the point $k_2^{''\ell}$ to the segment $\overline{k_{10}^p k_{11}^p}$. The following code is used to calculate all the alfas, i.e., for all the point $k_i^{''\ell}$, $i \in [2, \cdots, N]$ (except for the first point) in a trajectory ℓ, we calculate all the `alfa[i,j]`, `j` $\in [1, \cdots, N]$ to all the segments on another trajectory p.

```
#caculate alfa[i,j]
let {i in 2..N, j in 1..N} alfa[i,j] :=
((x20[i]-x10[j])*(x10[j+1]-x10[j])+(y20[i]-y10[j])*(y10[j+1]-y10[j]))/
((x10[j+1]-x10[j])^2+(y10[j+1]-y10[j])^2);

for {i in 2..N} {
  for{j in 1..N} {
  if alfa[i,j]>1 then let alfa[i,j] :=1;
  if alfa[i,j]<0 then let alfa[i,j] :=0;
  }
}
```

If `alfa[i, j]` > 1 or `alfa[i, j]` < 0, then the closest point to $k_i^{''\ell}$ on the line through $\overline{k_j^p k_{j+1}^p}$ is not on the segment. So we chose a larger distance to

the closest endpoint, like if `alfa[i, j]` > 1, we let `alfa[i, j]` $= 1$, it indicates that the distance $\beta_{i,j}$ will be the distance of $k_i^{''\ell}$ to k_{j+1}^p. On the other hand, if `alfa[i, j]` < 0, we let `alfa[i, j]` $= 0$, what means that the distance $\beta_{i,j}$ will be the distance of $k_i^{''\ell}$ to k_j^p.

Then we can calculate all the distances from each point $k_i^{''\ell}$, $i \in [2, \cdots, N]$ to all the segments $\overline{k_j^p k_{j+1}^p}$, $j \in [1, \cdots, N]$ by using the following code:

```
#caculate beta[i,j]
let {i in 2..N, j in 1..N} beta[i,j] :=
(alfa[i,j]*x10[j+1]+(1-alfa[i,j])*x10[j]-x20[i])^2+(alfa[i,j]
*y10[j+1]+(1-alfa[i,j])*y10[j]-y20[i])^2;
```

We do not calculate the distance from the first point $k_1^{''\ell}$ to the segments on another trajectory because the first point is always on the center of the k-space.

After we get all the distances `beta[i,j]`, we need to determine which one is the smallest to each point $k_i^{''\ell}$ $i \in [2, \cdots, N]$. For example, for $i = 3$, we suppose `beta[3,5]` is the smallest element in `beta[3,j]` `j` $\in [1, \cdots, N]$. This means that the nearest segment to $k_3^{''\ell}$ is $\overline{k_5^p k_6^p}$. To do this, we need to define another intermediate variable `minumber` and another series of parameters `beta1[1..N]` to store the indices (j's) of the nearest segment $\overline{k_j^p k_{j+1}^p}$. Contin-

uing the above example, let us suppose `beta[3,5]` is the smallest element of `beta[3,j]` `j` $\in [1, N]$, then `beta1`$[3] = 5$.

We used the following code to search for the smallest `beta[i,j]` and `beta1[1..N]`.

```
for {i in 2..N}{
let minumber :=beta[i,1];
let beta1[1] :=0;
let beta1[i] :=1;
    for{j in 2..N}{
    if beta[i,j] < minumber
    then {let minumber:=beta[i,j];
          let beta1[i]:=j;}
         }
       if beta1[i-1]=40
        then {let jj:= i-1;
              break;}
    }
```

It is still necessary to explain what `jj` means here. The counter `jj` stands for how many points of $k_i^{''\ell}$ should be subject to the quadratic constraint. For example, if the nearest segment to the point $k_{33}^{''\ell}$ is $\overline{k_{40}^p k_{41}^p}$ (the last

segment in a trajectory) at the previous iteration, then `jj` $= 33$, it means that to get a better result, we don't need to apply the quadratic constraints to the points k''^{ℓ}_{34}-k''^{ℓ}_{40} anymore.

For the constraint (3.34k) in model (3.34), the variables and parameters have the following relationship with the variables and parameters in the AMPL model teardrop2D.mod.

$$\begin{aligned} k_j &= (\texttt{x1[beta1[i]]}, \texttt{y1[beta1[i]]}) \\ k_{j+1} &= (\texttt{x1[beta1[i + 1]]}, \texttt{y1[beta1[i + 1]]}) \\ k''_i &= (\texttt{x2[i]}, \texttt{y2[i]}) \\ \alpha_i &= (\texttt{alfa[i, beta1[i]]}). \end{aligned}$$

Thus, the constraint can be expressed in the following way in AMPL.

```
#Quadratic constraint
subject to qu3{i in 4..jj}:
(alfa[i,beta1[i]]*x1[beta1[i]+1]+(1-alfa[i,beta1[i]])*x1[beta1[i]]
-x2[i])^2+(alfa[i,beta1[i]]*y1[beta1[i]+1]+(1-alfa[i,beta1[i]])*
y1[beta1[i]]-y2[i])^2<=d^2;
```

For the first two points k''^{ℓ}_2, k''^{ℓ}_3, we always limit the distance from k''^{ℓ}_2 to $\overline{k^p_1 k^p_2}$ and the distance from k''^{ℓ}_3 to $\overline{k^p_3 k^p_4}$ for the purpose of getting a good shape of a teardrop as follows:

```
# Quadratic constraints for the first two points
```

```
subject to qu1:

(alfa[2,1]*x1[2]+(1-alfa[2,1])*x1[1]-x2[2])^2+

(alfa[2,1]*y1[2]+(1-alfa[2,1])*y1[1]-y2[2])^2<=d^2;

subject to qu4:

(alfa[3,3]*x1[4]+(1-alfa[3,3])*x1[3]-x2[3])^2+

(alfa[3,3]*y1[4]+(1-alfa[3,3])*y1[3]-y2[3])^2<=d^2;
```

4.1.2 Specifying Data

There is a distinction between an AMPL model for an optimization problem, and data values that define a particular instance of the problem. After setting up the model, we need to provide the data that convert a model into a specific problem instance, then we use an optimization software to solve the problem. All the implementation issues about specifying data for the parameters and initial values of variables are addressed in this section.

The following code is used to specify data for most of the parameters in the model.

```
data;

param N :=40;

param K := 800;

param M := 4;
```

```
param mx := 20;

param pi := 3.14159;

param TR := 0.004;

param GMax := 0.04;

param SMax := 150;

param gama := 42576000;

param jj :=40;

param tolerance := 1;

param d := 63.25;

param iter := 1;
```

The data command initiates a data mode in AMPL. AMPL will read data statements after this command. The values for the parameters `TR`, `GMax` and `SMax` are from a real MRI machine, `GMax` and `SMax` will change from model to model. The value for the parameter `gama` is from MRI theory, and we give reasonable values to the parameter `tolerance` and `d` accordingly.

We give the initial values for the parameter `iter` and `jj`, they will change after the first iteration. We can modify the value of `N` to get different results for different scanning points. If you give a bigger value of `N`, you would get a smoother trajectory in k-space, but it would take more time to solve the problem.

The parameters `K`, `M` and `mx` are used for specifying data for the parameters `x10`, `y10`, `x20` and `y20` — the initial values for the corresponding variables in the first iteration. After the first iteration, the parameters `x10`, `y10`, `x20` and `y20` will hold the values of the corresponding variables from the last iteration. These values will be used to calculate the `alfa` and `beta` needed in the quadratic constraints.

We used the Archimedean spirals to generate the initial values for variables `x1`, `y1`, `x2` and `y2` just as the following code does:

```
#intial values for parameters
let {i in 1..N+1} x10[i] :=K/mx*((pi*(i-1)/N*M)*cos(pi*(i-1)/N*M));
let {i in 1..N+1} y10[i] :=
K/mx*((pi*(i-1)/N*M)*sin(pi*(i-1)/N*M));
let {i in 1..N+1} x20[i] :=x10[i]*cos(2*pi/M) - y10[i]*sin(2*pi/M);
let {i in 1..N+1} y20[i] :=
x10[i]*sin(2*pi/M) + y10[i]*cos(2*pi/M);


#intial values for variables
let {i in 2..N+1} x1[i] := x10[i];
let {i in 2..N+1} y1[i] := y10[i];
let {i in 2..N+1} x2[i] := x20[i];
```

```
let {i in 2..N+1} y2[i] := y20[i];

let {i in 2..N+1} x[i] := x1[i]-x1[i-1];

let {i in 2..N+1} y[i] := y1[i]-y1[i-1];
```

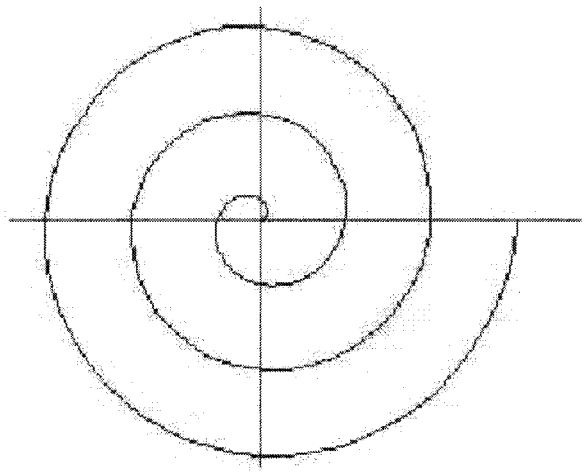

Figure 4.1: Archimedean spiral.

An Archimedean spiral (Fig. 4.1) has the property that successive arms have a fixed distance which is an attractive property for our trajectories in the k-space. An Archimedean spiral can be described as

$$r = a + b\theta. \tag{4.14}$$

In our AMPL program, we let $a = 0$, $b =$ `K/mx`, where `K` and `mx` are two parameters.

4.1.3 Command Environment

This section explains the commands that invoke solvers, display results and commands for modifying and resolving the model.

The commands used for the first run (first iteration) of the model are as follows:

```
option solver mosek;

option show_stats 1;

solve;

display k ;

display x1, y1 ;

display x2, y2  ;

display x,y;

display iter ;

display jj;
```

The command

```
option solver mosek;
```

chose the solver MOSEK. If you have more than one solver, you can choose your favorite one by changing this option. We used the command

```
option show_stats 1;
```

to have a summary of the actions of presolve and of the size of the resulting program. AMPL simplifies constraints as it prepares the model and data for handing them to a solver. For example, it may combine linear terms involving the same variable, move variables from one side of a constraint to the other side, or eliminate variables fixed at a value. Entire constraints may be dropped because a mathematical test shows that they are implied by other constraints. This work is carried out by a presolve phase.

The `solve` command finally sends the generated optimization problem to the solver of your choice. The `display` commands are used to exhibit the results after the solver solves the problem.

In the iterative algorithm (Fig. 3.5), we have a "repeat structure". The repeat structure means we want to solve the problem again and again after the first iteration until a stop criterion is satisfied. At each iteration, the problem is almost the same as the problem of the last iteration, except for the initial values for the variables and the quadratic constraints. The values for the parameters `alfa` and `beta1` change after each iterations because they are based on the results of the last iteration. In other words, after each iteration we recalculate `alfa`, `beta` and `beta1`, and then solve the problem again. Although we do not change the model, actually the quadratic constraints change because the value of their parameters change.

There are some commands provided in AMPL to help us to change the current optimization problem. The commands that implements the repeat structure and changes the data of `alfa`, `beta` and `beta1` are as follows:

```
repeat{
#after the first iteration
#give the results of last iteration to parameters
let {i in 1..N+1} x10[i] :=x1[i];
let {i in 1..N+1} y10[i] :=y1[i];
let {i in 1..N+1} x20[i] :=x2[i];
let {i in 1..N+1} y20[i] :=y2[i];
#caculate alfa[i,j] again for this iteration
let {i in 2..N, j in 1..N} alfa[i,j] :=
((x20[i]-x10[j])*(x10[j+1]-x10[j])+(y20[i]-y10[j])*
(y10[j+1]-y10[j]))/((x10[j+1]-x10[j])^2+
(y10[j+1]-y10[j])^2);
for {i in 2..N} {
  for{j in 1..N} {
  if alfa[i,j]>1 then let alfa[i,j] :=1;
  if alfa[i,j]<0 then let alfa[i,j] :=0;
  }
```

```
}
#caculate beta[i,j] again for this iteration
let {i in 2..N, j in 1..N} beta[i,j] :=
(alfa[i,j]*x10[j+1]+(1-alfa[i,j])*x10[j]-x20[i])^2+
(alfa[i,j]*y10[j+1]+(1-alfa[i,j])*y10[j]-y20[i])^2;
for {i in 2..N}{
let minumber :=beta[i,1];
let beta1[i] :=1;
    for{j in 2..N}{
    if beta[i,j] < minumber
    then {let minumber:=beta[i,j];
          let beta1[i]:=j;
           }
        }
        if beta1[i] < beta1[i-1]
          then {let jj:=i-1;  break;}
       if beta1[i-1]=40
        then {let jj:= i-1;
             break;}
    }
```

```
let previous-obj :=k;

solve; display k ;

display x1, y1 ;

display x2, y2  ;

display x,y;

display jj;

let iter := iter+1;

display iter; } until k <= previous-obj+tolerance;
```

After some data values have been changed, the current values of the variables no longer give an optimal solution. So the command `solve` and the `display` commands for displaying results are issued a second time.

Our stop criterion is `k <= previous-obj+tolerance`, where `k` stands for the value of the current objective function. If the trajectories in the k-space do not expand any more or only expand a little, then the algorithm stops.

Putting all the code that we explained in this chapter together, we have our model teardrop2D.mod as presented in Appendix A.1.

We performed the numerical experiments using this model. We will describe the results in the next chapter.

4.2 The AMPL Model teardrop3D.mod

The AMPL model teardrop3D.mod is the implementation of model (3.43). The 3D optimization model (3.43) is almost a straight-forward adoption of the 2D model (3.34) except that all the interleaves of trajectories are design variables in the 3D model.

In the model teardrop3D.mod, we designed 20 trajectories in k-space. This idea is based on the dodecahedron.

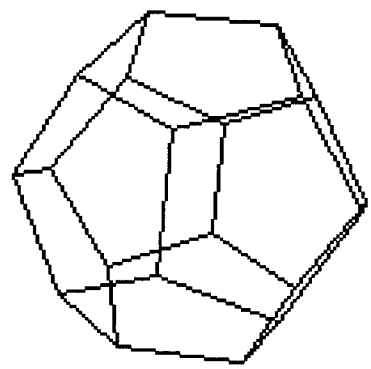

Figure 4.2: A dodecahedron in 3D.

Fig. 4.2 shows a dodecahedron. It is composed of 20 polyhedron vertices, 30 polyhedron edges, and 12 pentagonal faces. Suppose the center of k-space is on the center of a dodecahedron, we will design 20 trajectories starting from the center of k-space, and the starting directions of the trajectories

are from the center to each vertex.

The overall structure of teardrop3D.mod is the same as the structure of teardrop2D.mod. Because we are designing 20 trajectories now, instead of 2 trajectories, we need some two-dimensional variables in the model teardrop3D.mod.

The variables `x`, `y` and `z` represent the x-, y- and z-coordinates of k, `xdt`, `ydt` and `zdt` represent x-, y- and z-coordinates of g. They are all two-dimensional variables. For example, `x[2,3]` stands for the x-coordinate of the third point on the second trajectory. Variable `zdt[20,11]` means the z-coordinate of the 11th point of the 20th g (gradient waveform).

The variables `betaaij{2..N,1..N}` correspond to the `beta`'s in the teardrop2D.mod from trajectory i to p. For example, `betaa23[3,4]` equals the distance from the third point on the trajectory 2 to the segment $\overline{k_4k_5}$ on the trajectory 3.

The parameters `alfaij{2..N,1..N}` correspond to the `alfa`'s in the teardrop2D.mod from trajectory i to j. For example, `alfa23[2,10]` stands for the α value that is used when calculating the distance from the second point of trajectory 2 to the segment $\overline{k_{10}k_{11}}$ on the trajectory 3.

The parameters `betaij{1..N}` correspond to the `beta1`'s in the teardrop2D.mod from trajectory i to j. For example, suppose that `betaa23[3,5]` is the smallest

one of `betaa23[3,j]`, `j` $\in [1, N]$, then `beta23[3]=5`, i.e., the nearest segment to the third point of trajectory 2 is $\overline{k_5 k_6}$ on the trajectory 3.

In the implementation of constraint (3.43b), we let

$$\mathtt{k}_{\mathtt{N,old}} = (\mathtt{xx0}, \mathtt{yy0}, \mathtt{zz0}).$$

The purpose of doing this is to avoid that all the trajectories get close to each other after several iterations. The parameters `xx0`,`yy0` and `zz0` are initial values for the variables x, y and z in the first iteration.

We generated initial values `x0`,`y0` and `z0` for the variables `x`,`y` and `z` by using Maple. Fig. 4.3 shows the initial values of the 20 trajectories.

Just like in the 2D model, the most difficult part of implementing the 3D model is how to express the quadratic constraints (3.43j) in AMPL. For each pair of two neighboring trajectories, we need to apply these quadratic constraints to avoid big holes between the trajectories.

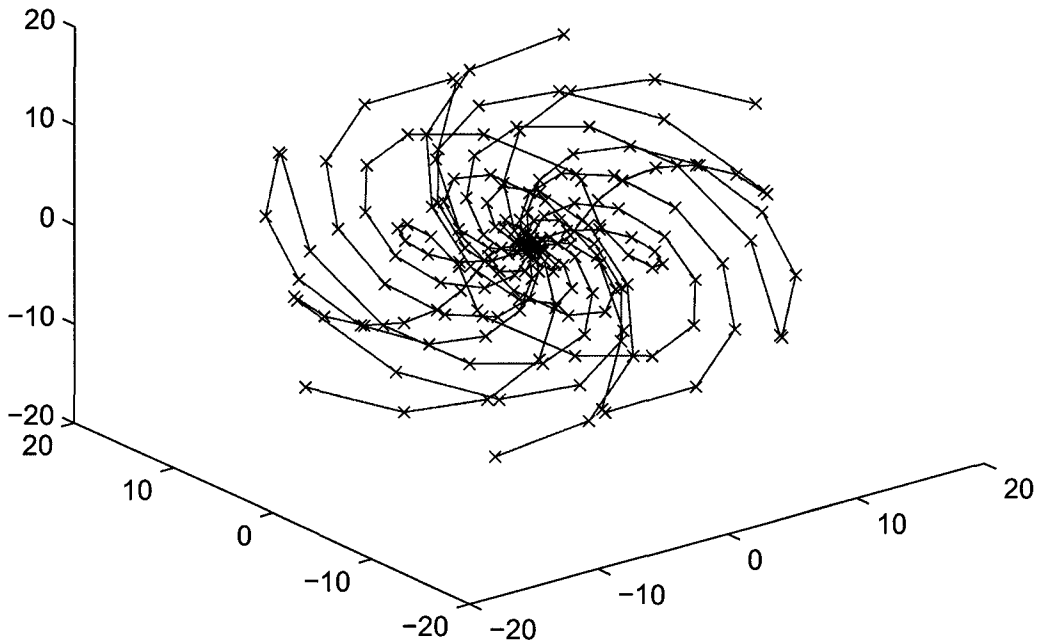

Figure 4.3: The initial values of the 20 trajectories.

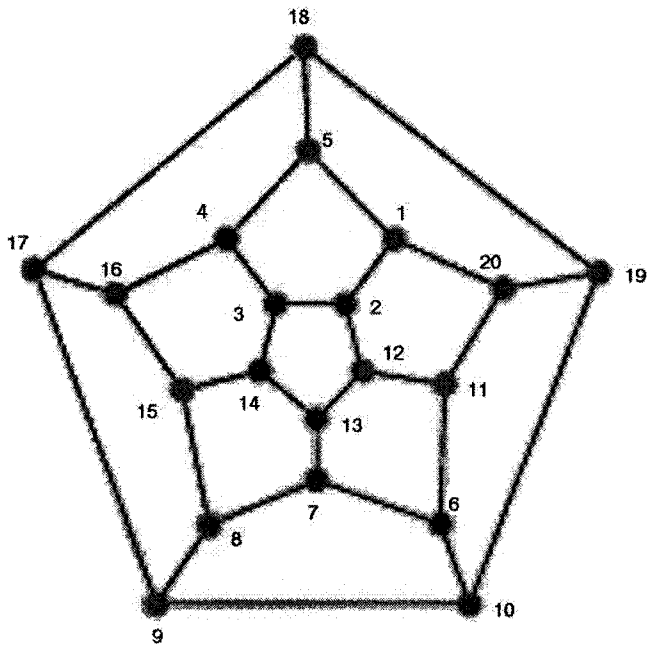

Figure 4.4: A platonic dodecahedron graph.

To implement these constraints, the first thing is to make clear the connectivity of the the dodecahedron, i.e, we need to know which trajectories are neighboring to each other. The Platonic graph (Fig. 4.4) of a dodecahedron corresponds to the connectivity of the vertices of a dodecahedron. It is a symmetric cubic planar graph. For example, trajectory 2 is surrounded by trajectory 1, 3 and 12. So we need to limit the distances from trajectory 2 to trajectory 1, 3 and 12 to be less than a constant, as the following code does:

```
#Quadratic constraint of tra. 2 to tra. 1 subject to qu3{i in
2..10}:
(alfa21[i,beta21[i]]*x[1,beta21[i]+1]+(1-alfa21[i,beta21[i]])*
x[1,beta21[i]]-x[2,i])^2+(alfa21[i,beta21[i]]*y[1,beta21[i]+1]+
(1-alfa21[i,beta21[i]])*y[1,beta21[i]]-y[2,i])^2+
(alfa21[i,beta21[i]]*z[1,beta21[i]+1]+(1-alfa21[i,beta21[i]])*
z[1,beta21[i]]-z[2,i])^2<=d^2;

#Quadratic constraint of tra. 2 to tra. 3 subject to qu4{i in
2..10}:
(alfa23[i,beta23[i]]*x[3,beta23[i]+1]+(1-alfa23[i,beta23[i]])*
x[3,beta23[i]]-x[2,i])^2+(alfa23[i,beta23[i]]*y[3,beta23[i]+1]+
(1-alfa23[i,beta23[i]])*y[3,beta23[i]]-y[2,i])^2+
```

```
(alfa23[i,beta23[i]]*z[3,beta23[i]+1]+(1-alfa23[i,beta23[i]])*
z[3,beta23[i]]-z[2,i])^2<=d^2;


#Quadratic constraint of tra. 2 to tra. 12 subject to qu5{i in
2..10}:
(alfa212[i,beta212[i]]*x[12,beta212[i]+1]+(1-alfa212[i,beta212[i]])*
x[12,beta212[i]]-x[2,i])^2+(alfa212[i,beta212[i]]*y[12,beta212[i]+1]+
(1-alfa212[i,beta212[i]])*y[12,beta212[i]]-y[2,i])^2+
(alfa212[i,beta212[i]]*z[12,beta212[i]+1]+(1-alfa212[i,beta212[i]])*
z[12,beta212[i]]-z[2,i])^2<=d^2;
```

We can implement all the quadratic constraints similarly in AMPL.

Part of the AMPL code of the model teardrop3D.mod is in Appendix A.2. We do not include all the quadratic constraints in the Appendix because the whole program is too long.

Chapter 5

Computational Results

In this chapter we present the computational results of the teardrop2D.mod and teardrop3D.mod models solved by MOSEK [8]. The MOSEK optimization software is designed to solve large scale mathematical optimization problems. It can solve linear, convex quadratic and general convex mathematical programs. An interior-point optimizer is available for all supported problem classes. All numerical experiments were performed on an IBM RS/6000 44P Model 270, 375 MHz, with 8 GB memory Workstation.

5.1 Results with teardrop2D.mod

Fig. 5.1 to Fig. 5.4 present the 2D k-space trajectories obtained by solving the model teardrop2D.mod without the first moment nulling constraint. After

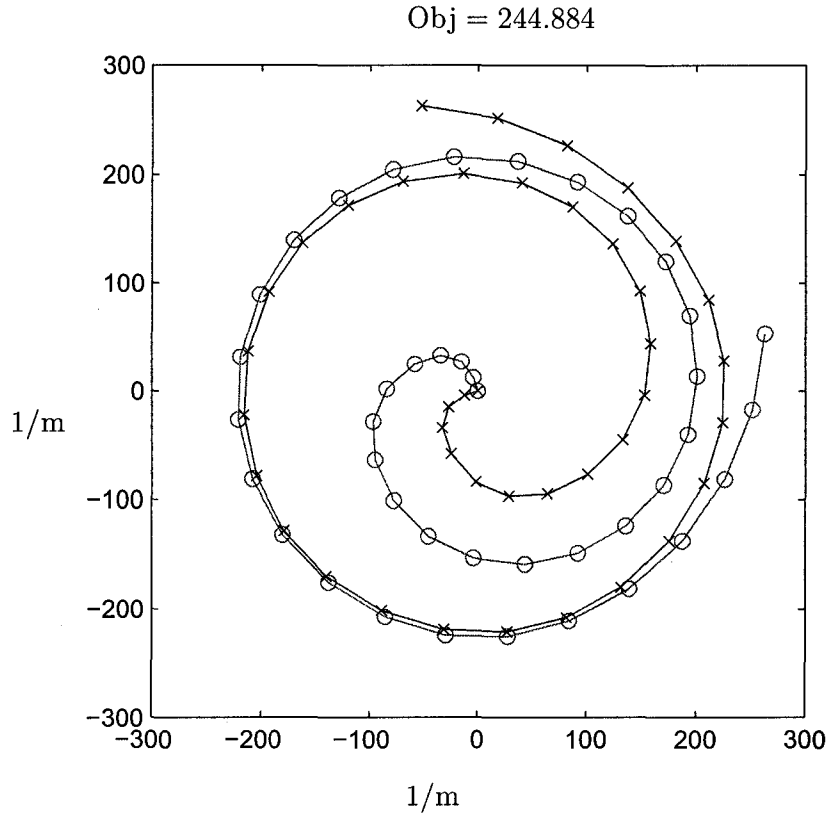

Figure 5.1: 2D k-space trajectories after the first iteration.

ten iterations, we got the optimal result (Fig.5.4). We only put the results of four iterations here to save some space. From these figures, we can see that the objective function value is increasing from iteration to iteration, in the meantime, the trajectories are becoming more and more evenly spaced which is a perfect property for trajectories in the k-space.

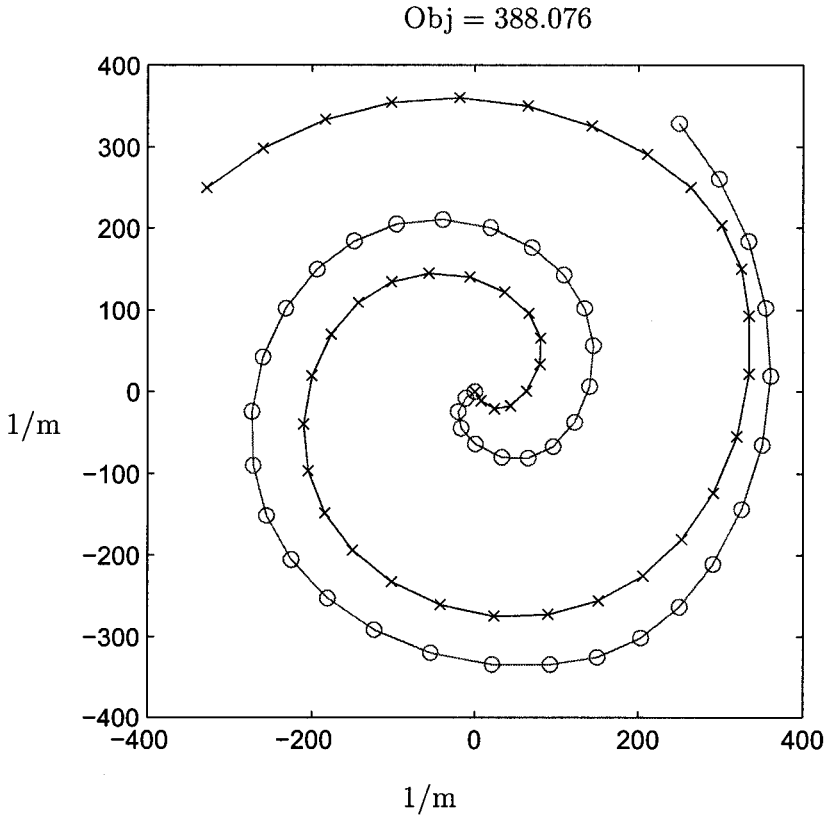

Figure 5.2: 2D k-space trajectories after the fourth iteration.

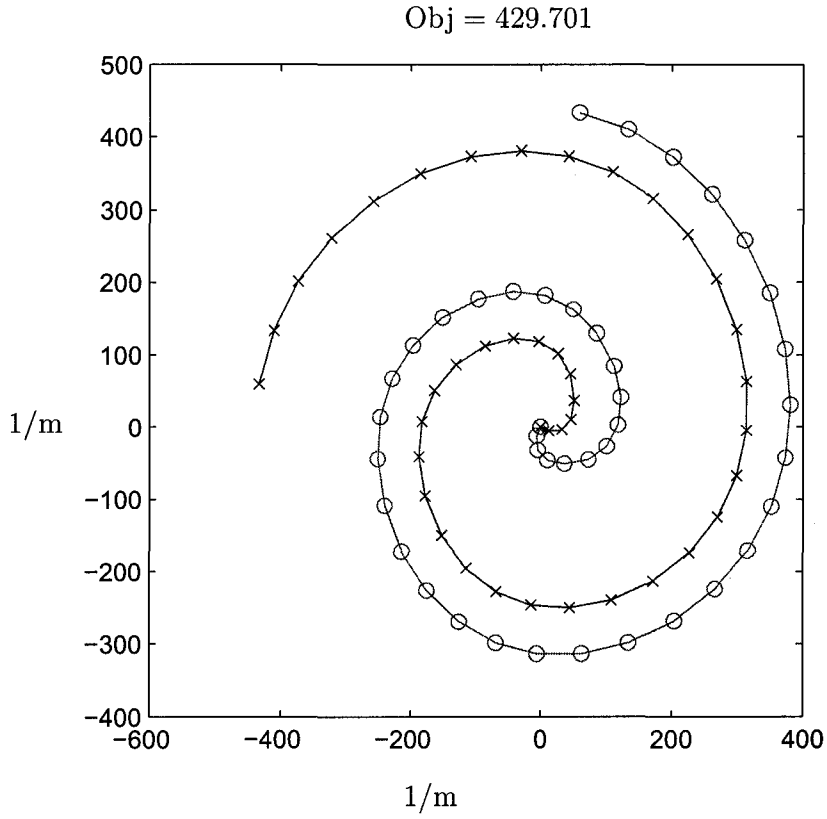

Figure 5.3: 2D k-space trajectories after the eighth iteration.

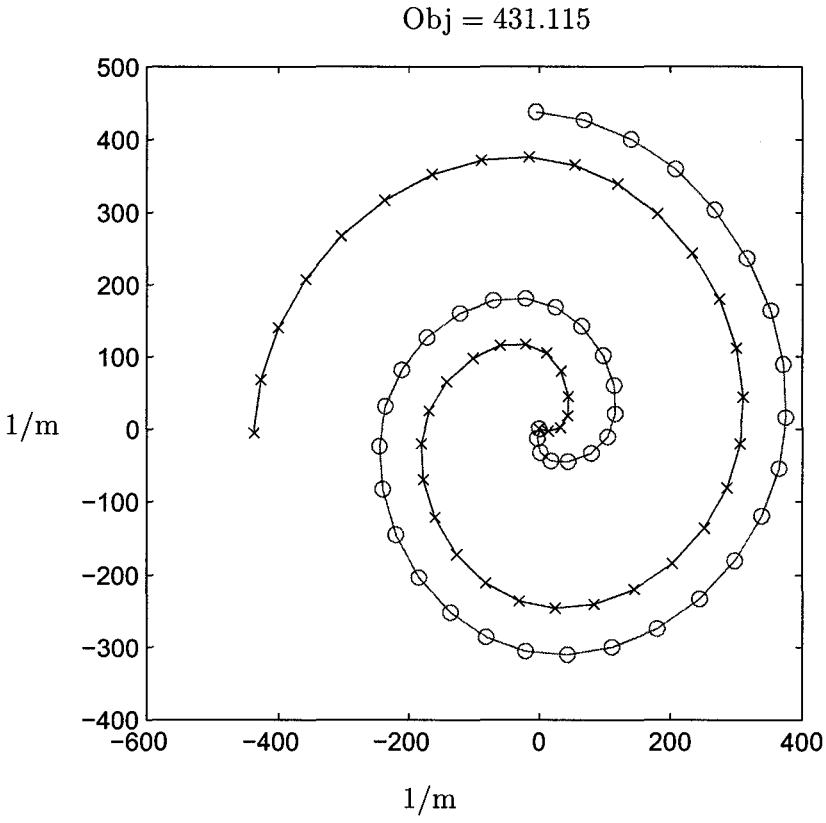

Figure 5.4: 2D k-space trajectories after ten (the last) iteration.

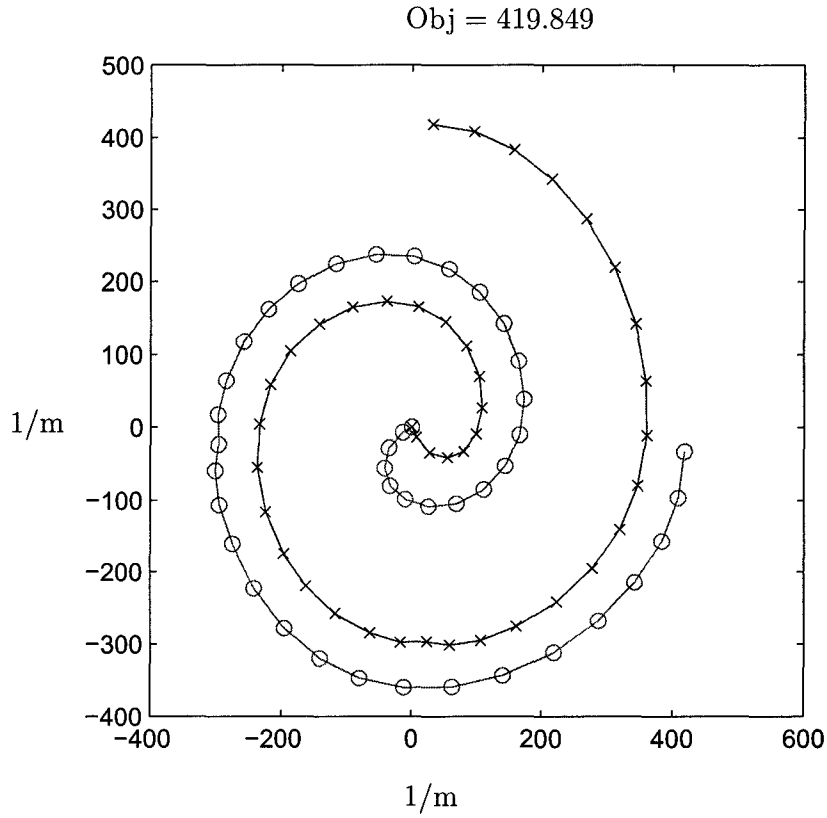

Figure 5.5: 2D k-space optimal trajectories with first moment nulling.

Fig.5.5 presents the the optimal trajectories obtained by solving the model teardrop2D.mod with first moment nulling. We got the optimal result after twelve iterations. Comparing this result to the result without first moment nulling (Fig.5.4), we can see the objective decreases a little, however as first moment nulling is required for SSFP imaging, the better results are more realistic.

The k-space trajectories we got from the model are only half teardrops. We can get the another half by symmetry of the designed teardrop trajectories. For example, Fig. 5.6 shows one of the whole teardrop trajectories from Fig. 5.4.

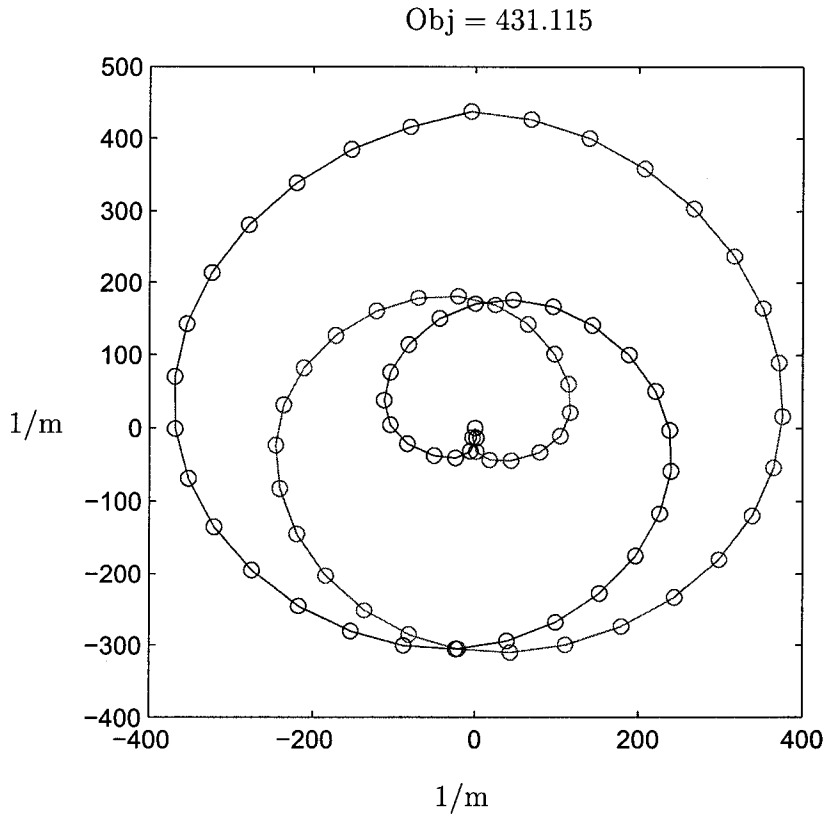

Figure 5.6: One optimal teardrop trajectory.

Rotating the associated gradient profiles is equivalent to rotating the trajectory around the center of k-space. Together the combined views can cover the two-dimensional k-space completely, as evidenced by the set of views in Fig.5.7. There are four teardrops in the figure.

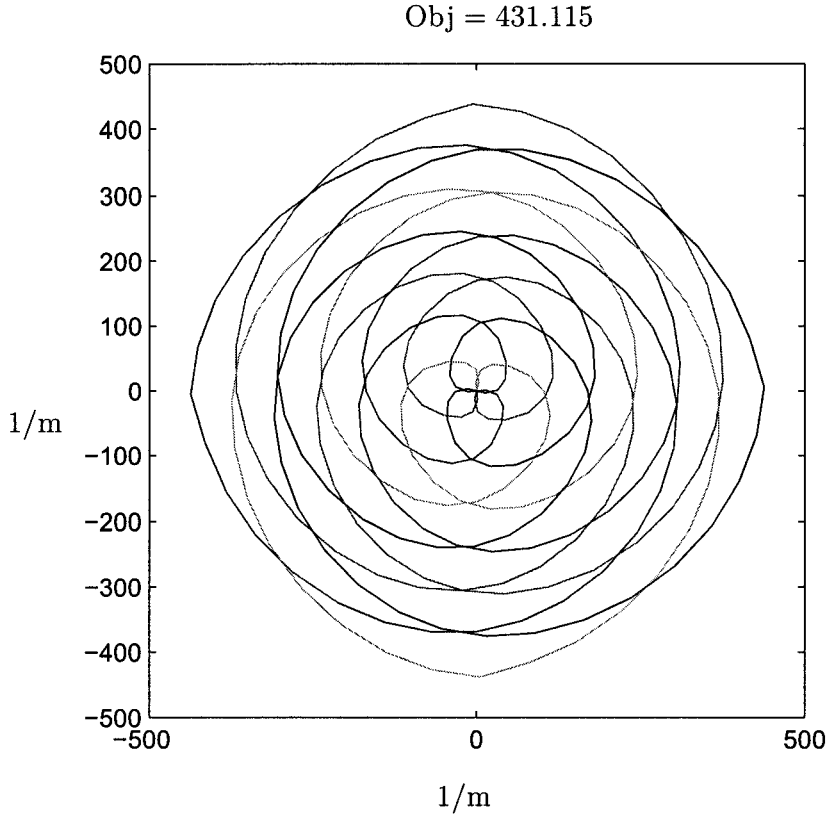

Figure 5.7: Rotate the optimal teardrops to cover the whole k-space.

Fig. 5.8 and Fig. 5.9 present the corresponding optimal gradient waveforms in the x- and y-direction which generate one of the trajectories of Fig. 5.4. Waveforms for the other interleaves are linear combination of these two.

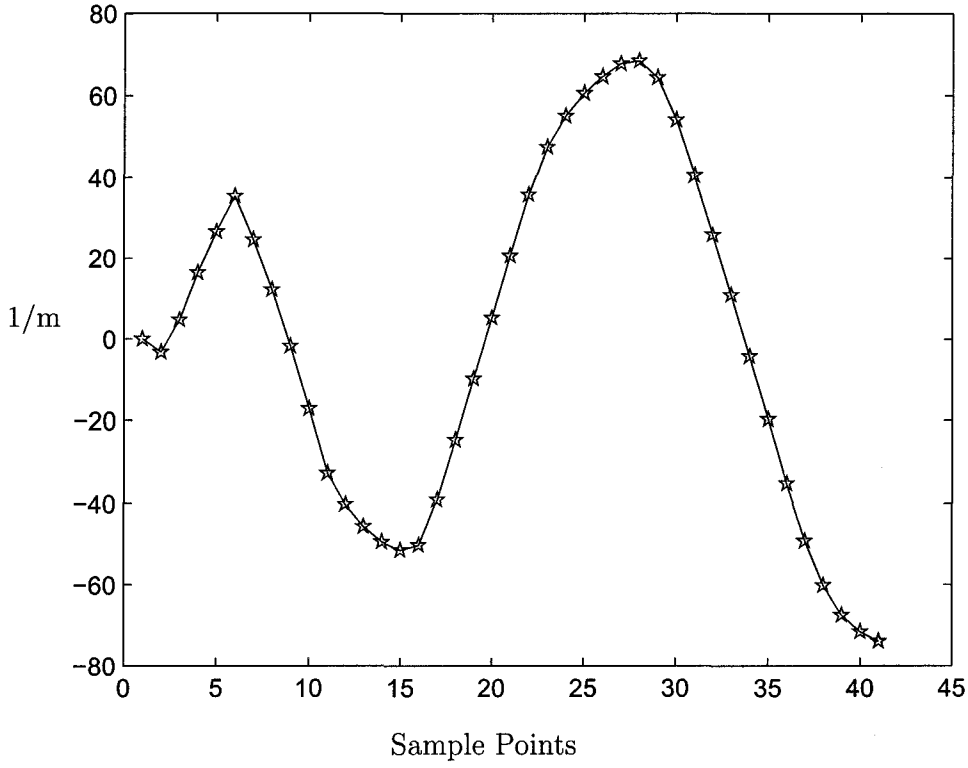

Figure 5.8: Gradient waveform G_x in the x-direction of the 2D k-space.

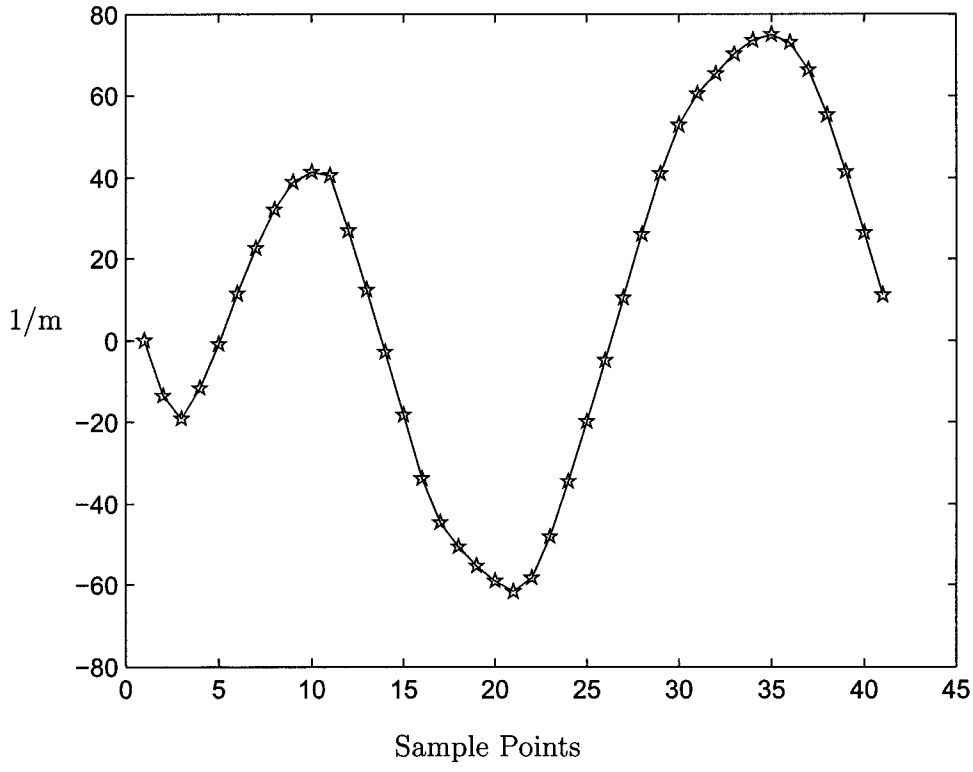

Figure 5.9: Gradient waveform G_y in the y-direction of the 2D k-space.

5.2 Results with teardrop3D.mod

Fig. 5.10 to Fig. 5.13 present the results of the 3D k-space trajectories obtained by solving model teardrop3D.mod. From these figures, we can see that the value of the objective function is increasing from iteration to iteration, this means that the resolution is increasing while all the constraints are satisfied. After seven iterations, we got the optimal twenty trajectories (Fig.5.13). Although the results are promising, they are still not perfect. The trajectories in the 3D k-space are not as evenly spaced in the 3D k-space as we expected. To illustrate this behavior, we show two trajectories in Fig. 5.14. The two trajectories starting from the origin show similar behavior initially as in the 2D case, while the last two points jump unexpectedly. This strange behavior may indicate where the model should be improved to get better results. Further improvements and experimentation are the subject of future research.

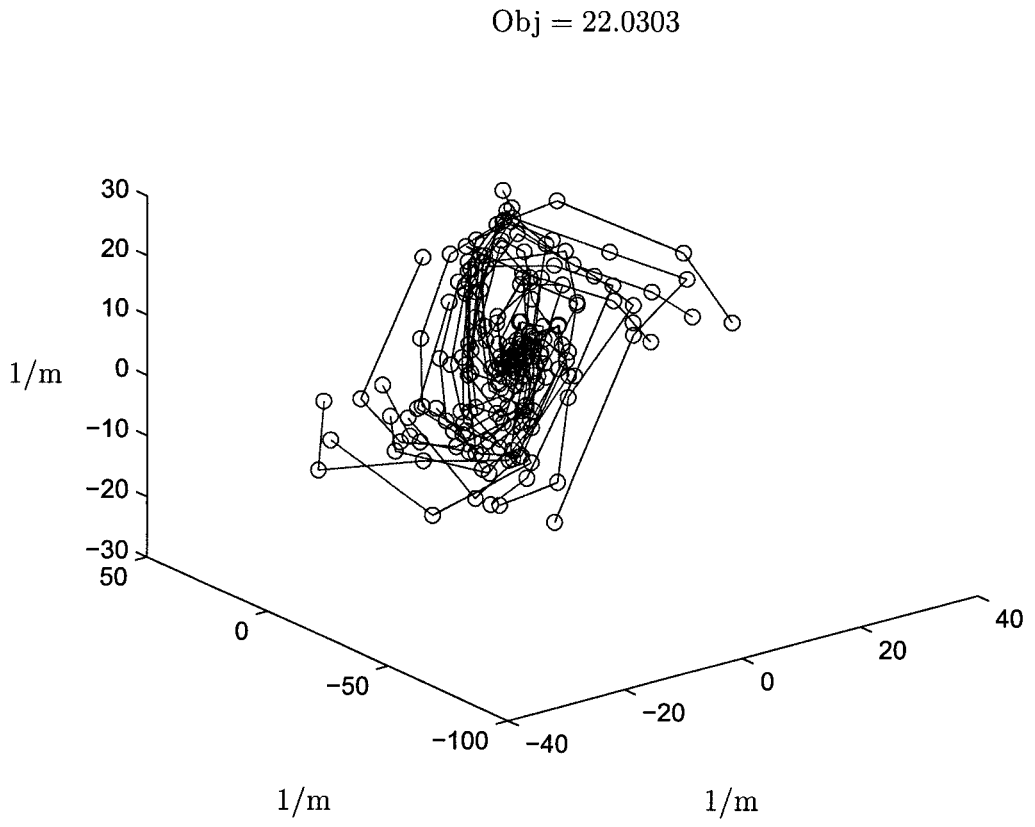

Figure 5.10: 3D k-space trajectories after the first iteration.

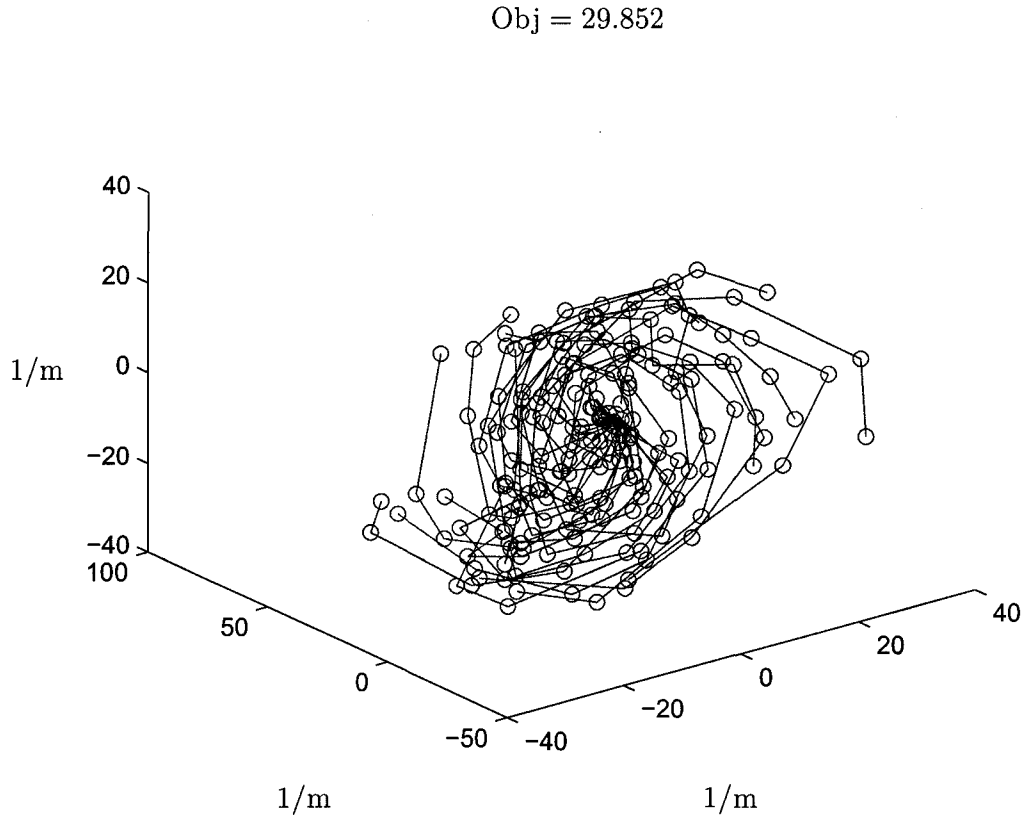

Figure 5.11: 3D k-space trajectories after the third iteration.

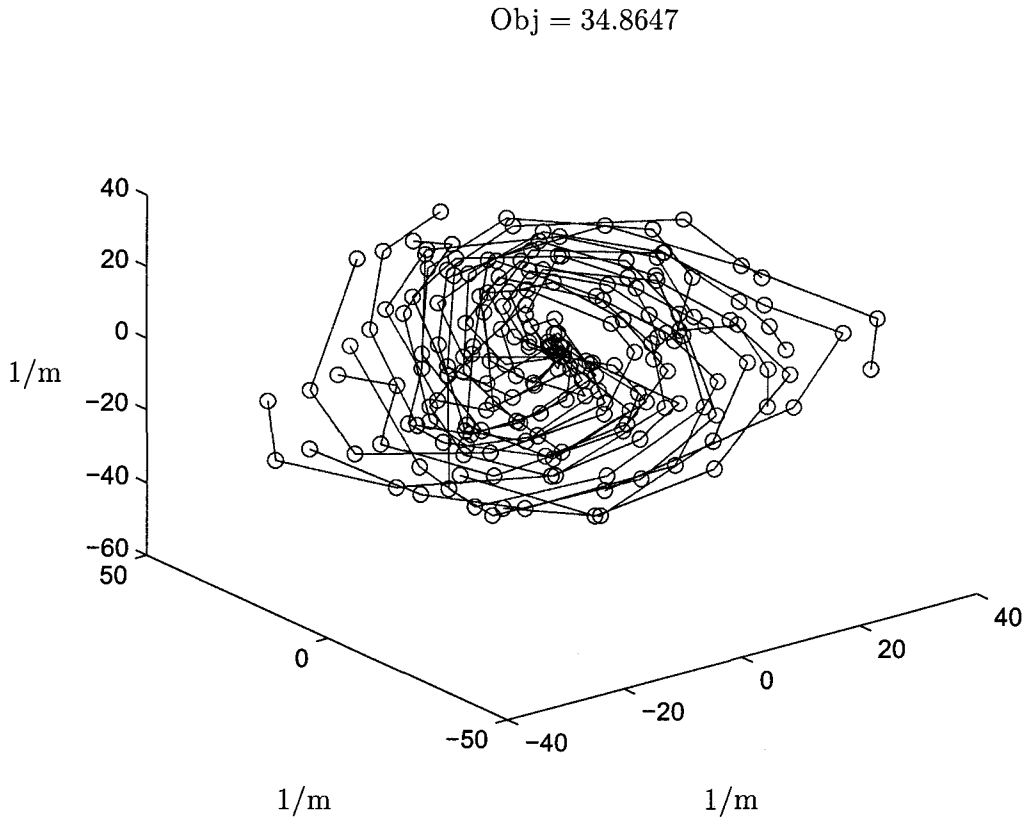

Figure 5.12: 3D k-space trajectories after the fifth iteration.

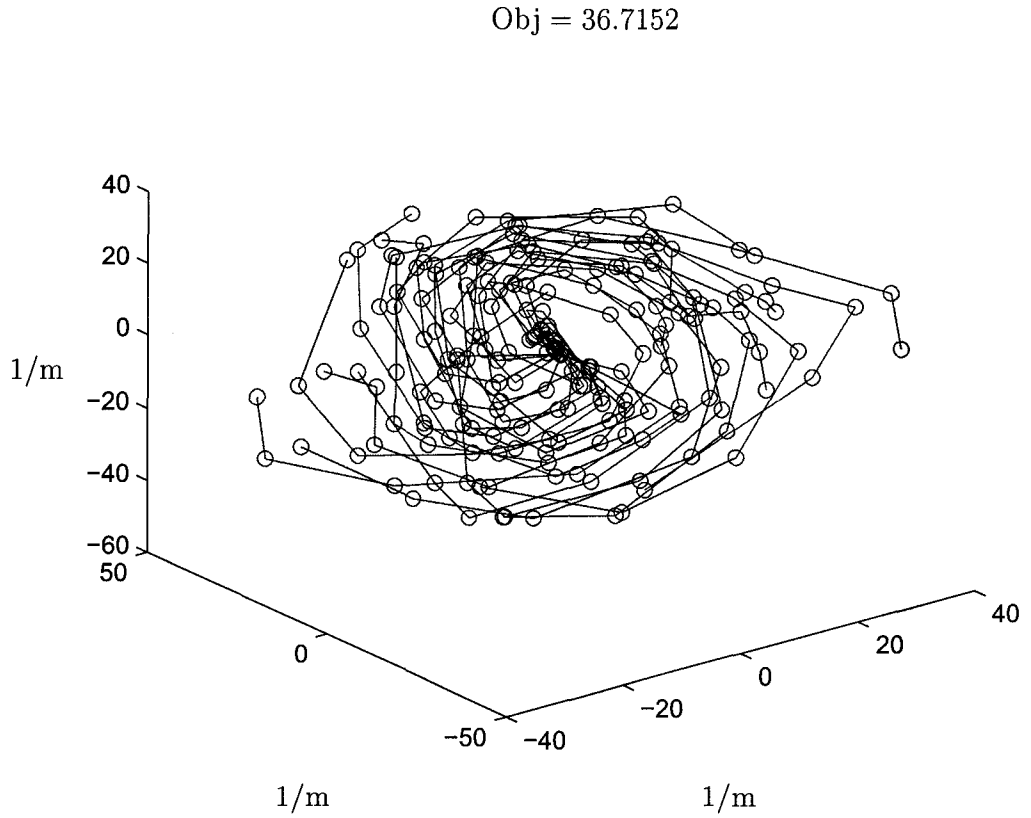

Figure 5.13: 3D k-space trajectories after the seventh(last) iteration.

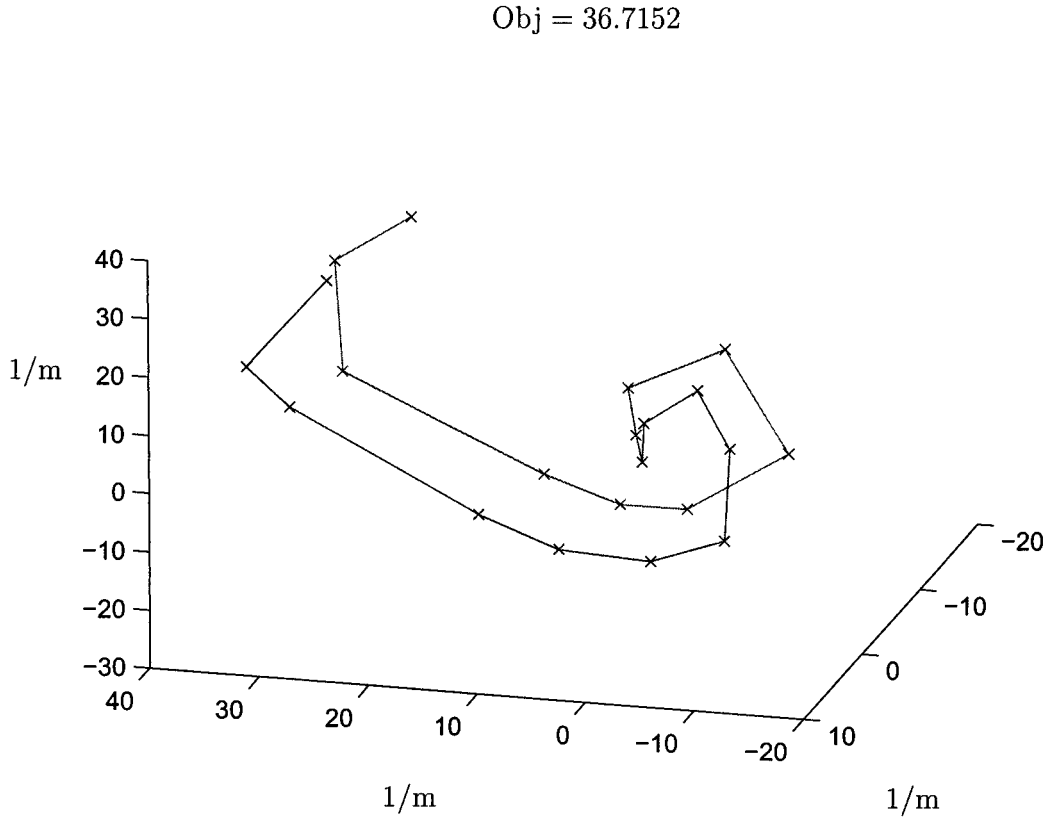

Figure 5.14: Two neighboring trajectories in 3D k-space.

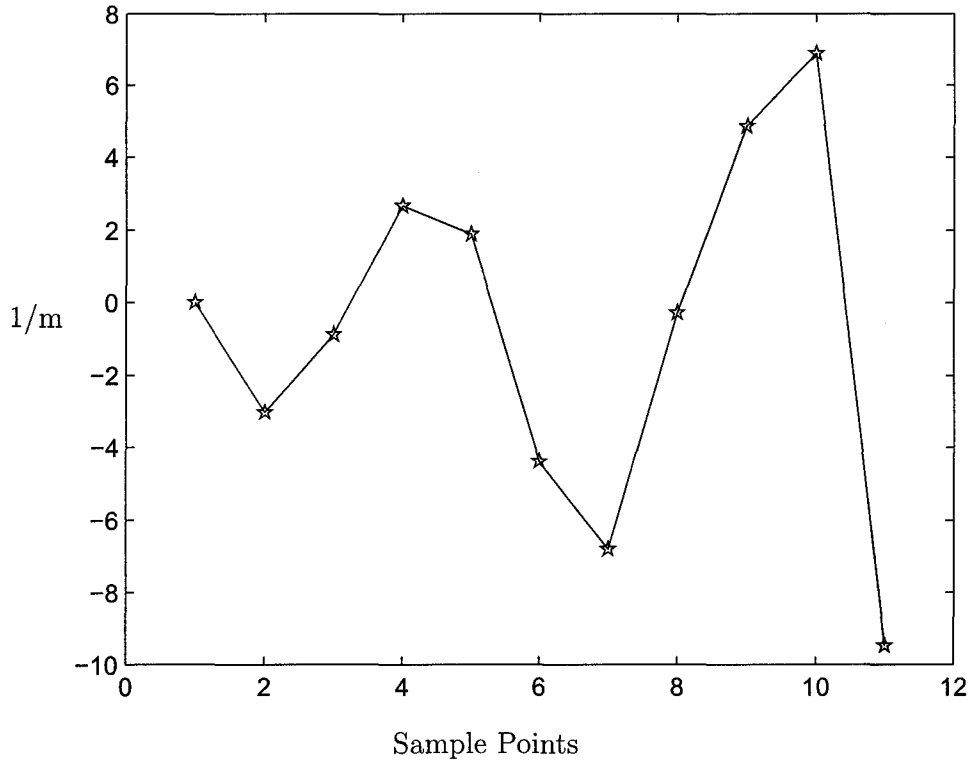

Figure 5.15: Gradient waveform G_x for trajectory 1 in the 3D k-space.

Fig's. 5.15, 5.16 and 5.17 display the corresponding optimal gradient waveforms in the x-, y- and z-direction to generate the first trajectory of Fig. 5.13. The other optimal gradient waveforms are similar, so we do not present them.

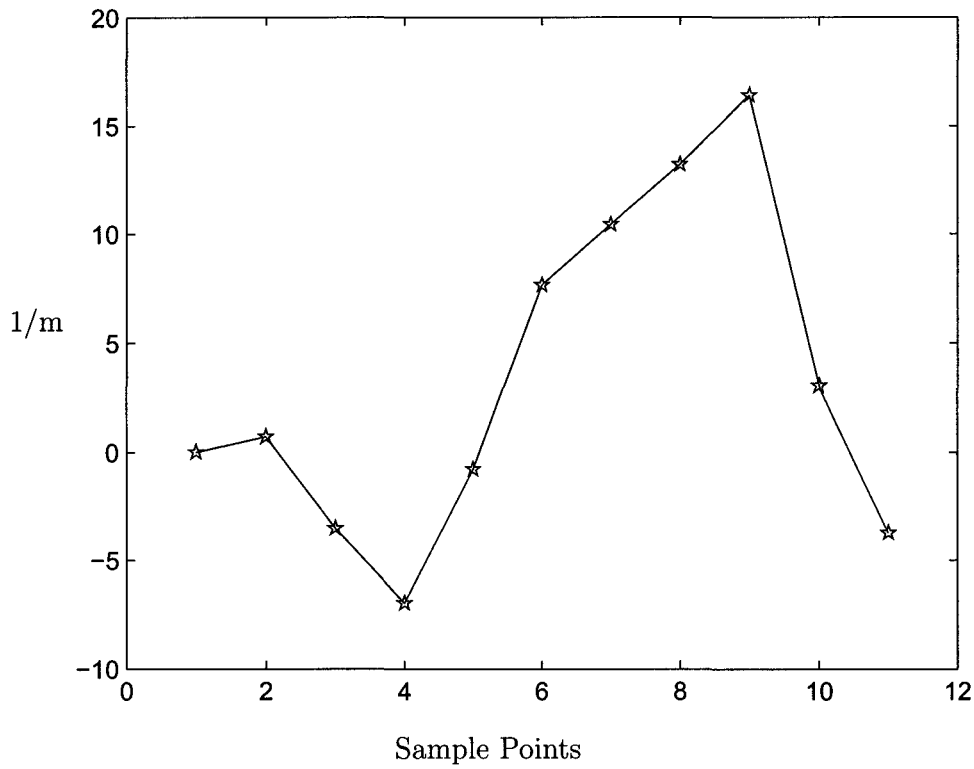

Figure 5.16: Gradient waveform G_y for trajectory 1 in the 3D k-space.

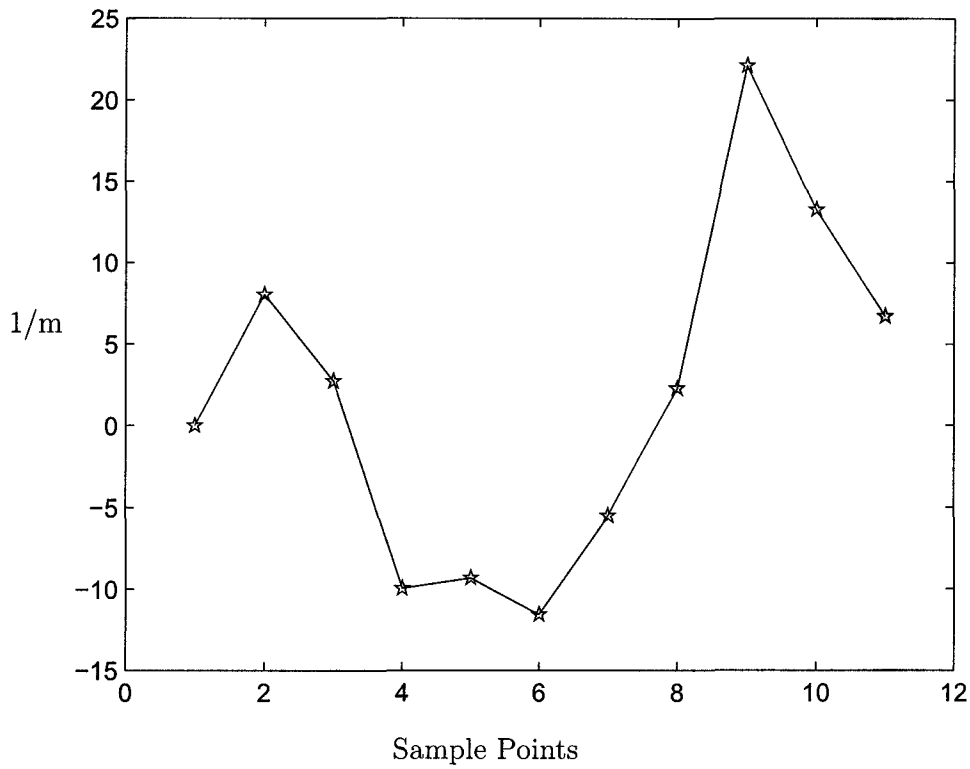

Figure 5.17: Gradient waveform G_z for trajectory 1 in the 3D k-space.

The exact shapes of teardrop trajectories and gradient waveforms vary with the number of interleaves, Field Of View and resolution. Various FOV, TRs, interleaves and many different initial solutions were tested and alternative solvers were used as well. In this thesis we represent only the best result derived by using MOSEK.

In the paper [1], Anand et al. implemented interleaved teardrop readouts designed for a PD250 (27 mt/m, 72 T/m/s) on an in-house research Marconi Eclipse 1.5 T scanner. The readout gradient waveform was numerically optimized using a proprietary Marconi algorithm. They showed a frame of transverse, TR 5.2ms, 50 interleave cine (movie), 36cm field of view, 3mm in-plane resolution. The image (Fig.5.18) displays good blood/septem (muscle) contrast to noise ratio. They have demonstrated that the nonstandard readout trajectories—teardrops can be used to acquire SSFP images. We expect our optimal readout to offer increased scan-time efficiency, which will yield higher frame rates and increased signal to noise ratio.

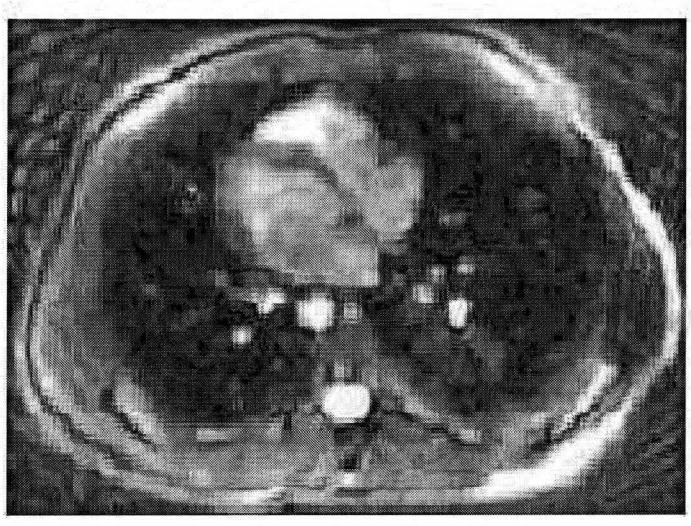

Figure 5.18: Gradient waveform G_z for trajectory 1 in 3D k-space.

Chapter 6

Conclusions and Future Work

Teardrop gradient waveform design is an increasingly important part of MRI sequencing because teardrop readout enables us to acquire SSFP images efficiently. We have shown that teardrop gradient waveform design can be cast as a convex-optimization problem for which efficient solution methods exist. In particular, we described how to build nonlinear convex optimization models for 2D and 3D teardrop gradient waveform design. The results of our numerical experiments indicate that the described optimization procedure can provide an effective, scientifically sound method of ensuring feasible and optimal teardrop gradient waveform for the stated design goal and the constraints while requiring short computation time. The unusual shapes exhibited by some numerical results suggest that conventional design methods may be far from optimal.

Due to the limitations, there are still some aspects of the 3D model that can be improved and areas left for further investigation. A few of the issues that should be taken into account for future research are:

- Improve the 3D model to get the resulting trajectories more evenly spaced in the 3D k-space (see Fig. 5.14 and the discussion around it);
- Add more constraints to the 2D and 3D models, like the second moment nulling which would reduce the image artifacts caused by pulsatile flow. The second moment nulling can be implemented the same efficient way as the first moment nulling (see page 51 to page 53) ;
- Test on MRI machine.

The work which has been done with the teardrop gradient waveform design has built an excellent foundation for future developments. This research illustrates the design of MRI sequences should benefit from optimal design.

Appendix A: The AMPL Codes

A.1 Model Teardrop2D.mod

```
#This is teardrop 2D convex model with only linear and quardratic
#  constraints
#parameters and variables, (xi, yi) corresponds to gi
#(x1i,y1i) corresponds to ki, and (x2i,y2i) corresponds to k'i
param N integer > 0; # total number of gi is 2*N+1
param pi >0; # the ratio constant of circumference to diameter
param K > 0; # a constant used in the initial condition
param M integer > 0; #number of interleaves
param mx ; #scalar in the intial condition
param TR >0; # readout interval for SSFP sequences, the unit
             #  is s(second)
param GMax;  # maximium gradient of MRI machine, the unit is T/m
param SMax;  # maximium slew rate of MRI machine, the unit is
```

```
                # T/m/s
param gama;  # gyromagnetic ratio
param DelT := TR/(2*N+1); # the sampling interval, the unit is
                          # s(second)
param G1 := (gama * GMax * DelT)^2; # maximum gradient of
                                    # calculation
param S1 := (gama * SMax * DelT * DelT)^2; # maximum slew rate
                                           #of calculation
param x10{1..N+1}; # the initial value for x of ki in each iteration
param y10{1..N+1}; # the initial value for y of ki in each iteration
param x20{1..N+1}; # the initial value for x of k'i in each iteration
param y20{1..N+1}; # the initial value for y of k'i in each iteration
param alfa{2..N,1..N};
param beta1{1..N};
param jj; # the number of points are subject to quadratic
          #    constraints
param previous-obj default 0; #the value of previous objective
                              # function
param tolerance; # a tolerance for the objective function
param d; # the distance between two neighboring trajectories
```

```
param iter; # iteration number

var x{1..N+1};  # x of gi

var y{1..N+1};  # y of gi

var x1{1..N+1}; # x of ki

var y1{1..N+1}; # y of ki

var x2{1..N+1}; # x of k''i, rotation of x1

var y2{1..N+1}; # y of k''i, rotation of y1

var tao;  # the supplementary variable of objective function

var minumber; # supplementary variable

var beta{2..N,1..N}; # distances between two trajectories,

                     #   intermediate variables


#objective function

maximize k:

tao;


subject to ob:

tao <= (x1[N]*x10[N]+y1[N]*y10[N])/sqrt(x10[N]*x10[N]+

y10[N]*y10[N]);
```

```
# teardrop starts from the center of k-space

subject to x0:

    x1[1]=0;

subject to y0:

    y1[1]=0;


#the first point of gi is zero

subject to s0:

    x[1] = 0;

subject to sy0:

    y[1] = 0;


# define k[i] in terms of g[i]

subject to x1_def{i in 2..N+1}:

    x1[i]=x[i]+x1[i-1];

subject to y1_def{i in 2..N+1}:

        y1[i]=y[i]+ y1[i-1];


#Gmax constraint

subject to gg{i in 1..N+1}:
```

```
        x[i]^2 + y[i]^2 <= G1 ;


#Smax constraint
subject to vv{i in 2..N+1}:
        (x[i]-x[i-1])^2 + (y[i]-y[i-1])^2 <= S1 ;


# For odd points of gi, ki is even. The symmetric constraint:
#   kN+1=kN.
subject to symmetr:
x1[N]*(x10[N]+x10[N+1])+y1[N]*(y10[N]+y10[N+1])=
x1[N+1]*(x10[N]+x10[N+1])+y1[N+1]*(y10[N]+y10[N+1]);


#define x2,y2 in terms of x1,y1
subject to x2_def{i in 1..N+1}:
    x2[i] = x1[i]*cos(2*pi/M) - y1[i]*sin(2*pi/M);
subject to y2_def{i in 1..N+1}:
    y2[i] = x1[i]*sin(2*pi/M) + y1[i]*cos(2*pi/M);



# Quadratic constraints for the first two points
```

```
subject to qu1:

(alfa[2,1]*x1[2]+(1-alfa[2,1])*x1[1]-x2[2])^2+

(alfa[2,1]*y1[2]+(1-alfa[2,1])*y1[1]-y2[2])^2<=d^2;

subject to qu4:

(alfa[3,3]*x1[4]+(1-alfa[3,3])*x1[3]-x2[3])^2+

(alfa[3,3]*y1[4]+(1-alfa[3,3])*y1[3]-y2[3])^2<=d^2;


#First moment nulling constrain1

subject to null1:

sum{i in 1..N} i*(x[i]*(x10[N]+x10[N+1])+y[i]*

(y10[N]+y10[N+1])) + (N+1)*(x[N+1]*(x10[N]+x10[N+1])

+y[N+1]*(y10[N]+y10[N+1]))-sum{i in 2..N} (N+i)*

(x[N+2-i]*(x10[N]+x10[N+1])+y[N+2-i]*(y10[N]+y10[N+1]))=0;


#First moment nulling constrain2

subject to null2:

sum{i in 1..N} i*(x[i]*(y10[N]+y10[N+1])-y[i]*

(x10[N]+x10[N+1])) + (N+1)*(x[N+1]*(y10[N]+y10[N+1])-

y[N+1]*(x10[N]+x10[N+1]))+sum{i in 2..N} (N+i)*(x[N+2-i]

*(y10[N]+y10[N+1])-y[N+2-i]*(x10[N]+x10[N+1]))=0;
```

```
data;

param N :=40;

param K := 800;

param M := 4;

param mx := 20;

param pi := 3.14159;

param TR := 0.004;

param GMax := 0.04;

param SMax := 150;

param gama := 42576000;

param jj :=40;

param tolerance := 1;

param d := 63.25;

param iter := 1;


#intial values for parameters

let {i in 1..N+1} x10[i] :=K/mx*((pi*(i-1)/N*M)*cos(pi*(i-1)/N*M));

let {i in 1..N+1} y10[i] :=

K/mx*((pi*(i-1)/N*M)*sin(pi*(i-1)/N*M));
```

```
let {i in 1..N+1} x20[i] :=x10[i]*cos(2*pi/M) - y10[i]*sin(2*pi/M);

let {i in 1..N+1} y20[i] :=

x10[i]*sin(2*pi/M) + y10[i]*cos(2*pi/M);


#intial values for variables

let {i in 2..N+1} x1[i] := x10[i];

let {i in 2..N+1} y1[i] := y10[i];

let {i in 2..N+1} x2[i] := x20[i];

let {i in 2..N+1} y2[i] := y20[i];

let {i in 2..N+1} x[i] := x1[i]-x1[i-1];

let {i in 2..N+1} y[i] := y1[i]-y1[i-1];


#caculate alfa[i,j]

let {i in 2..N, j in 1..N} alfa[i,j] :=

((x20[i]-x10[j])*(x10[j+1]-x10[j])+(y20[i]-y10[j])*

(y10[j+1]-y10[j]))/((x10[j+1]-x10[j])^2+(y10[j+1]-y10[j])^2);


for {i in 2..N} {

  for{j in 1..N} {

  if alfa[i,j]>1 then let alfa[i,j] :=1;
```

```
  if alfa[i,j]<0 then let alfa[i,j] :=0;
  }
}

#caculate beta[i,j]
let {i in 2..N, j in 1..N} beta[i,j] :=
(alfa[i,j]*x10[j+1]+(1-alfa[i,j])*x10[j]-x20[i])^2+
(alfa[i,j]*y10[j+1]+(1-alfa[i,j])*y10[j]-y20[i])^2;

for {i in 2..N}{
let minumber :=beta[i,1];
let beta1[1] :=0;
let beta1[i] :=1;
    for{j in 2..N}{
    if beta[i,j] < minumber
    then {let minumber:=beta[i,j];
          let beta1[i]:=j;}
         }
       if beta1[i-1]=40
        then {let jj:= i-1;
```

```
            break;}
    }

#Quadratic constraint
subject to qu3{i in 4..jj}:
(alfa[i,beta1[i]]*x1[beta1[i]+1]+(1-alfa[i,beta1[i]])*x1[beta1[i]]
-x2[i])^2+(alfa[i,beta1[i]]*y1[beta1[i]+1]+(1-alfa[i,beta1[i]])*
y1[beta1[i]]-y2[i])^2<=d^2;



option solver mosek;
option show_stats 1;
solve;
display k ;
display x1, y1 ;
display x2, y2  ;
display x,y;
display iter ;
display jj;
```

```
repeat{
#after the first iteration
#take the results of last iteration as the initial values
let {i in 1..N+1} x10[i] :=x1[i] ;
let {i in 1..N+1} y10[i] :=y1[i];
let {i in 1..N+1} x20[i] :=x2[i];
let {i in 1..N+1} y20[i] :=y2[i];

#caculate alfa[i,j] again for this iteration
let {i in 2..N, j in 1..N} alfa[i,j] :=
((x20[i]-x10[j])*(x10[j+1]-x10[j])+(y20[i]-y10[j])*(y10[j+1]
-y10[j]))/((x10[j+1]-x10[j])^2+(y10[j+1]-y10[j])^2);

for {i in 2..N} {
  for{j in 1..N} {
  if alfa[i,j]>1 then let alfa[i,j] :=1;
  if alfa[i,j]<0 then let alfa[i,j] :=0;
  }
}
```

```
#caculate beta[i,j] again for this iteration

let {i in 2..N, j in 1..N} beta[i,j] :=

(alfa[i,j]*x10[j+1]+(1-alfa[i,j])*x10[j]-x20[i])^2+(alfa[i,j]*

y10[j+1]+(1-alfa[i,j])*y10[j]-y20[i])^2;


for {i in 2..N}{

let minumber :=beta[i,1];

let beta1[i] :=1;

    for{j in 2..N}{

    if beta[i,j] < minumber

    then {let minumber:=beta[i,j];

          let beta1[i]:=j;

           }


        }

        if beta1[i] < beta1[i-1]

          then {let jj:=i-1;  break;}

       if beta1[i-1]=40

        then {let jj:= i-1;

             break;}
```

```
    }


let previous-obj :=k;

solve;


display k;

display x1, y1;

display x2, y2 ;

display x,y;

display jj;


let iter := iter+1;

display iter;

} until k <= previous-obj+tolerance;
```

A.2 Part of Model Teardrop3D.mod

```
#This is teardrop 3D convex model with only linear and quadratic

# constraints


#parameters and variables
```

```
param N integer > 0; # total number of gi is 2*N+1

param NT integer > 0; # number of trajectories

param pi >0; # the ratio constant of circumference to diameter

param K > 0; # a constant used in the initial condition

param M integer > 0; # number of interleaves

param mx ; # scaler in the initial condition

param TR >0; # readout interval for SSFP sequences, the unit is

             #  s(second)

param GMax;  # maximum gradient of MRI machine, the unit is T/m

param SMax;  # maximum slew rate of MRI machine, the unit is T/m/s

param gama;  # gyromagnetic ratio

param DelT := TR/(2*N+1); # the sampling interval, the unit is

                          #  s(second)

param G1 := (gama * GMax * DelT)^2; # maximum gradient of

                                    #  calculation

param S1 := (gama * SMax * DelT * DelT)^2; # maximum slew

                                           #rate of calculation

param x0{1..NT,1..N+1}; # the initial values for x of k[i,j]

                        #  in each iteration

param y0{1..NT,1..N+1}; # the initial values for y of k[i,j]
```

```
                        #  in each iteration
param z0{1..NT,1..N+1}; # the initial values for z of k[i,j]
                        #  in each iteration
# the initial values for x of k[i,j] in the first iteration
param xx0{1..NT,1..N+1};
# the initial values for y of k[i,j] in the first iteration
param yy0{1..NT,1..N+1};
# the initial values for z of k[i,j] in the first iteration
param zz0{1..NT,1..N+1};
param previous-obj default 0; #the value of previous objective
                              # function
param tolerance; # a tolerance for the objective function
param d; # the distance between two neighboring trajectories
param iter; # iteration number
param c1; # a costant from the last itertion
param c2; # a costant from the last itertion
param c3; # a costant from the last itertion
# alfaij stand for alfa's from trajectory i to trajectory j
param alfa21{2..N,1..N};
param alfa23{2..N,1..N};
```

```
param alfa212{2..N,1..N};

param alfa15{2..N,1..N};

param alfa120{2..N,1..N};

param alfa34{2..N,1..N};

param alfa314{2..N,1..N};

param alfa1211{2..N,1..N};

param alfa1213{2..N,1..N};

param alfa137{2..N,1..N};

param alfa1314{2..N,1..N};

param alfa1415{2..N,1..N};

param alfa2019{2..N,1..N};

param alfa2011{2..N,1..N};

param alfa116{2..N,1..N};

param alfa610{2..N,1..N};

param alfa67{2..N,1..N};

param alfa78{2..N,1..N};

param alfa54{2..N,1..N};

param alfa518{2..N,1..N};

param alfa416{2..N,1..N};

param alfa1617{2..N,1..N};
```

```
param alfa1615{2..N,1..N};

param alfa89{2..N,1..N};

param alfa815{2..N,1..N};

param alfa1817{2..N,1..N};

param alfa1819{2..N,1..N};

param alfa179{2..N,1..N};

param alfa1910{2..N,1..N};

param alfa910{2..N,1..N};

# betaij stand for beta's from trajectory i to trajectory j

param beta21{1..N};

param beta23{1..N};

param beta212{1..N};

param beta15{1..N};

param beta120{1..N};

param beta34{1..N};

param beta314{1..N};

param beta1211{1..N};

param beta1213{1..N};

param beta137{1..N};

param beta1314{1..N};
```

```
param beta1415{1..N};

param beta2019{1..N};

param beta2011{1..N};

param beta116{1..N};

param beta610{1..N};

param beta67{1..N};

param beta78{1..N};

param beta54{1..N};

param beta518{1..N};

param beta416{1..N};

param beta1617{1..N};

param beta1615{1..N};

param beta89{1..N};

param beta815{1..N};

param beta1817{1..N};

param beta1819{1..N};

param beta179{1..N};

param beta1910{1..N};

param beta910{1..N};
```

```
var x{1..NT,1..N+1};     #x of k[i,j]

var y{1..NT,1..N+1};     #y of k[i,j]

var z{1..NT,1..N+1};     #z of k[i,j]

var xdt{1..NT,1..N+1};   #x of g[i,j]

var ydt{1..NT,1..N+1};   #y of g[i,j]

var zdt{1..NT,1..N+1};   #z of g[i,j]

var tao; # the supplementary variable of objective function

var minumber;

# betaaij stand for betaa's from trajectory i to trajectory j

var betaa21{2..N,1..N}; #tra 2 to tra 1

var betaa23{2..N,1..N}; #tra 2 to tra 3

var betaa212{2..N,1..N}; #tra 2 to tra 12

var betaa15{2..N,1..N}; #tra 1 to tra 5

var betaa120{2..N,1..N}; #tra 1 to tra 20

var betaa34{2..N,1..N}; #tra 3 to tra 4

var betaa314{2..N,1..N}; #tra 3 to tra 14

var betaa1211{2..N,1..N}; #tra 12 to tra 11

var betaa1213{2..N,1..N}; #tra 12 to tra 13

var betaa137{2..N,1..N};

var betaa1314{2..N,1..N};
```

```
var betaa1415{2..N,1..N};

var betaa2019{2..N,1..N};

var betaa2011{2..N,1..N};

var betaa116{2..N,1..N};

var betaa610{2..N,1..N};

var betaa67{2..N,1..N};

var betaa78{2..N,1..N};

var betaa54{2..N,1..N};

var betaa416{2..N,1..N};

var betaa518{2..N,1..N};

var betaa1617{2..N,1..N};

var betaa1615{2..N,1..N};

var betaa89{2..N,1..N};

var betaa815{2..N,1..N};

var betaa1817{2..N,1..N};

var betaa1819{2..N,1..N};

var betaa179{2..N,1..N};

var betaa1910{2..N,1..N};

var betaa910{2..N,1..N};
```

```
#objective function

maximize k:

          tao;


subject to ob1{i in 1..20}:

      tao <= ( x[i,N]*xx0[i,N] + y[i,N]*yy0[i,N]+ z[i,N]*zz0[i,N])

      /sqrt(xx0[i,N]*xx0[i,N]+yy0[i,N]*yy0[i,N]+zz0[i,N]*zz0[i,N]);


# trajectories start from the center of k-space

subject to x00{i in 1..NT}:

    x[i,1]=0;

subject to y00{i in 1..NT}:

    y[i,1]=0;

subject to z00{i in 1..NT}:

    z[i,1]=0;



#define k[i,j] in terms of g[i,j]

subject to x_def{i in 1..NT,j in 2..N+1}:

    x[i,j]=xdt[i,j]+x[i,j-1];
```

```
subject to y_def{i in 1..NT,j in 2..N+1}:
        y[i,j]=ydt[i,j]+ y[i,j-1];


subject to z_def{i in 1..NT,j in 2..N+1}:
        z[i,j]=zdt[i,j]+ z[i,j-1];



#Gmax constraint
subject to gg{i in 1..NT,j in 1..N+1}:
         (xdt[i,j])^2 + (ydt[i,j])^2+ (zdt[i,j])^2 <= G1;


#Smax constraint
subject to vv{i in 1..NT,j in 2..N+1}:
         (xdt[i,j]-xdt[i,j-1])^2 + (ydt[i,j]-ydt[i,j-1])^2+
         (zdt[i,j]-zdt[i,j-1])^2 <= S1;


# symmetric constraints
subject to symetr{i in 1..NT}:
x[i,N]*(x0[i,N]+x0[i,N+1])+y[i,N]*(y0[i,N]+y0[i,N+1])+z[i,N]*
```

```
(z0[i,N]+z0[i,N+1])
=x[i,N+1]*(x0[i,N]+x0[i,N+1])+y[i,N+1]*(y0[i,N]+y0[i,N+1])+
z[i,N+1]*(z0[i,N]+ z0[i,N+1]);


#the first point is zero, g1=0
subject to xdt0{i in 1..NT}:
    xdt[i,1] = 0;
subject to ydt0{i in 1..NT}:
    ydt[i,1] = 0;
subject to zdt0{i in 1..NT}:
    zdt[i,1] = 0;


data;
param K := 800;
param N :=10;
param NT :=20;
param M := 4;
param mx := 20;
param pi := 3.14159;
param TR := 0.004;
```

```
param GMax := 0.04;

#param SMax := 150;

param SMax := 200;# ONLY FOR TEST

param gama := 42576000;

param iter :=1;

param tolerance := 1;

param d := 40;


#initial values for parameters

param x0 (tr):

   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15

   16  17  18  19  20 :=

1 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

0.0 0.0 0.0 0.0 0.0

2 0.356822 -0.356822 -0.577350 0.000000 0.577350 0.000000

-0.577350 -0.356822  0.356822 0.577350 0.000000 -0.577350

-0.934172 -0.934172 -0.577350 0.000000  0.577350 0.934172

0.934172 0.577350

3 0.949309 -0.990296 -1.558363 0.030158 1.579985 -0.030158

-1.579985 -0.949309  0.990296 1.558363 -0.045814 -1.605316
```

```
-2.553486 -2.524468 -1.533032 0.045814  1.605316 2.553486
2.524468 1.533032
4 1.500012 -1.817725 -2.692316 0.084895 2.675895 -0.084895
-2.675895 -1.500012  1.817725 2.692316 -0.206250 -2.872253
-4.398577 -4.287370 -2.495958 0.206250  2.872253 4.398577
4.287370 2.495958
5 1.815029 -2.894922 -4.000879 0.025554 3.619982 -0.025554
-3.619982 -1.815029  2.894922 4.000879 -0.438036 -4.287393
-6.253944 -6.076868 -3.333468 0.438036  4.287393 6.253944
6.076868 3.333468
6 1.843217 -4.237355 -5.610385 -0.378393 4.228187 0.378393
-4.228187 -1.843217  4.237355 5.610385 -0.536086 -5.707845
-7.989689 -7.929455 -4.130727 0.536086  5.707845 7.989689
7.929455 4.130727
7 1.728043 -5.678274 -7.562535 -1.320755 4.421138 1.320755
-4.421138 -1.728043  5.678274 7.562535 -0.188099 -6.862515
-9.478677 -9.911313 -5.121158 0.188099  6.862515 9.478677
9.911313 5.121158
8 1.820851 -6.852362 -9.709559 -2.802191 4.323994 2.802191
-4.323994 -1.820851  6.852362 9.709559 0.880325 -7.433639
```

```
-10.650085 -12.056681 -6.599914 -0.880325  7.433639 10.650085
12.056681 6.599914
9 2.587227 -7.294918 -11.688779 -4.522189 4.300868 4.522189
-4.300868 -2.587227  7.294918 11.688779 2.724011 -7.210381
-11.551995 -14.319797 -8.779266 -2.724011
7.210381 11.551995 14.319797 8.779266
10 4.419062 -6.632597 -13.002187 -5.887151 4.879773 5.887151
-4.879773 -4.419062  6.632597 13.002187 5.041656 -6.247813
-12.379593 -16.554026 -11.634147
-5.041656 6.247813 12.379593 16.554026 11.634147
11 7.414612 -4.799456 -13.190688 -6.162686 6.572090 6.162686
-6.572090 -7.414612  4.799456 13.190688 7.161587 -4.955834
-13.443713 -18.533132 -14.806943  -7.161587 4.955834
13.443713 18.533132 14.806943;



param y0 (tr):
   1   2   3   4   5   6   7   8   9   10  11  12  13  14   15
   16  17  18  19  20 :=
```

```
1 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0
2 -0.934172 -0.934172 -0.577350 -0.356822 -0.577350 0.356822
0.577350 0.934172  0.934172 0.577350 -0.356822 -0.577350
0.000000 0.000000 0.577350 0.356822
0.577350 0.000000 0.000000 -0.577350
3 -2.228747 -2.228911 -2.129645 -2.068131 -2.129380 2.068131
2.129380 2.228747  2.228911 2.129645 0.365457 -0.625605 0.464559
-0.464988 0.625340 -0.365457
0.625605 -0.464559 0.464988 -0.625340
4 -2.340837 -2.368639 -3.736944 -4.554800 -3.691959 4.554800
3.691959 2.340837  2.368639 3.736944 2.755940 0.781342 1.359834
-1.432620 -0.826327 -2.755940
-0.781342 -1.359834 1.432620 0.826327
5 -0.392183 -0.538345 -4.220845 -6.350593 -3.984349 6.350593
3.984349 0.392183  0.538345 4.220845 5.995163 3.409251 2.166500
-2.549159 -3.645747 -5.995163
-3.409251 -2.166500 2.549159 3.645747
6 3.485024 3.053895 -2.860754 -6.085079 -2.163173 6.085079
2.163173 -3.485024  -3.053895 2.860754 8.582724 6.204447
```

2.236946 -3.365656 -6.902028 -8.582724

-6.204447 -2.236946 3.365656 6.902028

7 8.244427 7.303293 0.321334 -3.052619 1.844122 3.052619

-1.844122 -8.244427 -7.303293 -0.321334 8.991319 7.764897

1.068226 -3.532149 -9.287685 -8.991319

-7.764897 -1.068226 3.532149 9.287685

8 12.247921 10.556335 4.488467 2.429904 7.225510 -2.429904

-7.225510 -12.247921 -10.556335 -4.488467 6.280547 6.868296

-1.478907 -2.949722 -9.605338

-6.280547 -6.868296 1.478907 2.949722 9.605338

9 13.815566 11.172721 8.200902 9.007062 12.477115 -9.007062

-12.477115 -13.815566 -11.172721 -8.200902 0.537614 2.966495

-5.077050 -1.842009 -7.242709 -0.537614 -2.966495 5.077050

1.842009 7.242709

10 11.837207 8.130507 9.860997 14.637200 15.858565 -14.637200

-15.858565 -11.837207 -8.130507 -9.860997 -7.010212 -3.517839

-8.986422 -0.717847 -2.479729 7.010212 3.517839 8.986422

0.717847 2.479729

11 6.243495 1.473231 8.255132 17.216842 15.973582 -17.216842

-15.973582 -6.243495 -1.473231 -8.255132 -14.269315 -11.204383

```
-12.257678 -0.231036  3.485933 14.269315 11.204383 12.257678
0.231036 -3.485933;


param z0 (tr):
   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15
   16  17  18  19  20 :=
1 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0
2 0.000000 0.000000 0.577350 0.934172 0.577350 -0.934172
-0.577350 0.000000 0.000000 -0.577350 -0.934172 -0.577350
-0.356822 0.356822 0.577350 0.934172
0.577350 0.356822 -0.356822 -0.577350
3 -1.233107 -1.200135 0.652054 1.763797 0.598704 -1.763797
-0.598704 1.233107  1.200135 -0.652054 -2.693213 -2.102531
-0.808052 0.894374 2.155880 2.693213
2.102531 0.808052 -0.894374 -2.155880
4 -3.733217 -3.570979 -0.673168 0.955539 -0.935676 -0.955539
0.935676 3.733217 3.570979 0.673168 -3.745493 -3.578566 -0.685444
1.110190 3.841073 3.745493
3.578566 0.685444 -1.110190 -3.841073
```

5 -6.364405 -5.939983 -3.182958 -1.903444 -3.869686 1.903444

3.869686 6.364405 5.939983 3.182958 -2.796414 -3.734843 0.385033

0.726117 4.421572 2.796414

3.734843 -0.385033 -0.726117 -4.421572

6 -7.661883 -6.853163 -5.881068 -6.089000 -7.189604 6.089000

7.189604 7.661883 6.853163 5.881068 0.544746 -1.781187 2.325561

-0.208304 3.089724 -0.544746

1.781187 -2.325561 0.208304 -3.089724

7 -6.448826 -5.192610 -7.432959 -10.073788 -9.465560 10.073788

9.465560 6.448826 5.192610 7.432959 5.627155 2.270757 4.643022

-1.354205 -0.238156 -5.627155

-2.270757 -4.643022 1.354205 0.238156

8 -2.348551 -0.672060 -6.664898 -12.045167 -9.377517 12.045167

9.377517 2.348551 0.672060 6.664898 10.891396 7.510677 6.575049

-2.185939 -4.798058 -10.891396

-7.510677 -6.575049 2.185939 4.798058

9 3.947022 5.924278 -3.043065 -10.562444 -6.242333 10.562444

6.242333 -3.947022 -5.924278 3.043065 14.332945 12.343132 7.342858

-2.166334 -9.143864 -14.332945

-12.343132 -7.342858 2.166334 9.143864

```
10 10.760837 12.858721 3.024412 -5.151410 -0.370037 5.151410
0.370037 -10.760837 -12.858721 -3.024412 14.173279 14.967727 6.436854
-0.944521 -11.573278 -14.173279 -14.967727 -6.436854
0.944521 11.573278
11 15.867786 17.903607 10.178551 3.368382 6.884523 -3.368382
-6.884523 -15.867786 -17.903607 -10.178551 9.531142 13.987346
3.841907 1.487942 -10.693318 -9.531142
-13.987346 -3.841907 -1.487942 10.693318;

#intial values for variables
let {i in 1..NT,j in 2..N+1} x[i,j] := x0[i,j];
let {i in 1..NT,j in 2..N+1} y[i,j] := y0[i,j];
let {i in 1..NT,j in 2..N+1} z[i,j] := z0[i,j];

let {i in 1..NT,j in 2..N+1} xdt[i,j] := x[i,j]-x[i,j-1];
let {i in 1..NT,j in 2..N+1} ydt[i,j] := y[i,j]-y[i,j-1];
let {i in 1..NT,j in 2..N+1} zdt[i,j] := z[i,j]-z[i,j-1];

let {i in 1..NT,j in 2..N+1} xx0[i,j] := x0[i,j];
let {i in 1..NT,j in 2..N+1} yy0[i,j] := y0[i,j];
```

```
let {i in 1..NT,j in 2..N+1} zz0[i,j] := z0[i,j];


let c1:= (x0[1,N]+x0[1,N+1])/sqrt(x0[1,N]*x0[1,N]+
y0[1,N]*y0[1,N]+z0[1,N]*z0[1,N]);
let c2:= (y0[1,N]+y0[1,N+1])/sqrt(x0[1,N]*x0[1,N]+
y0[1,N]*y0[1,N]+z0[1,N]*z0[1,N]);
let c3:= (z0[1,N]+z0[1,N+1])/sqrt(x0[1,N]*x0[1,N]+
y0[1,N]*y0[1,N]+z0[1,N]*z0[1,N]);


#First moment nulling constrain1
subject to null1:
sum{i in 1..N} i*(xdt[1,i]*c1+ydt[1,i]*c2+zdt[1,i]*c3)
+ (N+1)*(xdt[1,N+1]*c1+ydt[1,N+1]*c2+zdt[1,N+1]*c3)-
sum{i in 2..N} (N+i)*(xdt[1,N+2-i]*c1+ydt[1,N+2-i]*
c2+zdt[1,N+2-i]*c3)=0;


#First moment nulling constrain2
subject to null2:
sum{i in 1..N} i*(xdt[1,i]-(c1*xdt[1,i]+c2*ydt[1,i]+
c3*zdt[1,i])*c1)+(N+1)*(xdt[1,N+1]-(c1*xdt[1,N+1]+c2*
```

```
ydt[1,N+1]+c3*zdt[1,N+1])*c1)+sum{i in 2..N} (N+i)*
(xdt[1,N+2-i]-(c1*xdt[1,N+2-i]+c2*ydt[1,N+2-i]+c3*
zdt[1,N+2-i])*c1)=0;

#First moment nulling constrain3
subject to null3:
sum{i in 1..N} i*(ydt[1,i]-(c1*xdt[1,i]+c2*ydt[1,i]+
c3*zdt[1,i])*c2)+(N+1)*(ydt[1,N+1]-(c1*xdt[1,N+1]+
c2*ydt[1,N+1]+c3*zdt[1,N+1])*c2)+sum{i in 2..N}
(N+i)*(ydt[1,N+2-i]-(c1*xdt[1,N+2-i]+c2*ydt[1,N+2-i]
+c3*zdt[1,N+2-i])*c2)=0;

#First moment nulling constrain4
subject to null4:
sum{i in 1..N} i*(zdt[1,i]-(c1*xdt[1,i]+c2*ydt[1,i]+c3*
zdt[1,i])*c3)+(N+1)*(zdt[1,N+1]-(c1*xdt[1,N+1]+c2*
ydt[1,N+1]+c3*zdt[1,N+1])*c3)+sum{i in 2..N} (N+i)*
(zdt[1,N+2-i]-(c1*xdt[1,N+2-i]+c2*ydt[1,N+2-i]+c3*
zdt[1,N+2-i])*c3)=0;
```

```
# calculate all the alfaij's

# calculate alfa21[i,j]

let {i in 2..N, j in 1..N} alfa21[i,j] :=

((x0[2,i]-x0[1,j])*(x0[1,j+1]-x0[1,j])+(y0[2,i]-y0[1,j])*

(y0[1,j+1]-y0[1,j])+(z0[2,i]-z0[1,j])*(z0[1,j+1]-z0[1,j]))

/((x0[1,j+1]-x0[1,j])^2+(y0[1,j+1]-y0[1,j])^2+(z0[1,j+1]-

z0[1,j])^2);


for {i in 2..N} {

  for{j in 1..N} {

  if alfa21[i,j]>1 then let alfa21[i,j] :=1;

  if alfa21[i,j]<0 then let alfa21[i,j] :=0;

  }

}


#  ... (calculate all other alfa's the similar way

        as above)



# calculate all the betaaij's and betaij's
```

```
let {i in 2..N, j in 1..N} betaa21[i,j] :=
(alfa21[i,j]*x0[1,j+1]+(1-alfa21[i,j])*x0[1,j]-x0[2,i])^2+(
alfa21[i,j]*y0[1,j+1]+(1-alfa21[i,j])*y0[1,j]-y0[2,i])^2
+(alfa21[i,j]*z0[1,j+1]+(1-alfa21[i,j])*z0[1,j]-z0[2,i])^2;


for {i in 2..N}{
let minumber :=betaa21[i,1];
let beta21[i] :=1;
    for{j in 1..N}{
    if betaa21[i,j] < minumber
    then {let minumber:=betaa21[i,j];
          let beta21[i]:=j;}
         }
    }


#  ... (calculate all other betaa's and beta's the similar way
       as above)


#Quadratic constraints for all the neighbouring trajectories
#Quadratic constraint of tra. 2 to tra. 1
```

```
subject to qu3{i in 2..10}:
(alfa21[i,beta21[i]]*x[1,beta21[i]+1]+(1-alfa21[i,beta21[i]])*
x[1,beta21[i]]-x[2,i])^2+(alfa21[i,beta21[i]]*y[1,beta21[i]+1]+
(1-alfa21[i,beta21[i]])*y[1,beta21[i]]-y[2,i])^2+
(alfa21[i,beta21[i]]*z[1,beta21[i]+1]+
(1-alfa21[i,beta21[i]])*z[1,beta21[i]]-z[2,i])^2<=d^2;


#Quadratic constraint of tra. 2 to tra. 3
subject to qu4{i in 2..10}:
(alfa23[i,beta23[i]]*x[3,beta23[i]+1]+(1-alfa23[i,beta23[i]])*
x[3,beta23[i]]-x[2,i])^2+(alfa23[i,beta23[i]]*y[3,beta23[i]+1]+
(1-alfa23[i,beta23[i]])*y[3,beta23[i]]-y[2,i])^2+
(alfa23[i,beta23[i]]*z[3,beta23[i]+1]+
(1-alfa23[i,beta23[i]])*z[3,beta23[i]]-z[2,i])^2<=d^2;


#  ... (apply all other quadratic constraints the similar
     #   way as above)


option solver mosek;
option show_stats 1;
```

```
solve;


option display_width 120;

option gutter_width 1;


display k;

display x;

display y;

display z;

display xdt;

display ydt;

display zdt;

display iter;


repeat{

# the second and after iteration

# take the results of last iteration as the initial values

let {i in 1..NT,j in 1..N+1} x0[i,j] :=x[i,j] ;

let {i in 1..NT,j in 1..N+1} y0[i,j] :=y[i,j];

let {i in 1..NT,j in 1..N+1} z0[i,j] :=z[i,j];
```

```
# recalculate all the alfaij's for this iteration
# calculate alfa21[i,j]
let {i in 2..N, j in 1..N} alfa21[i,j] :=
((x0[2,i]-x0[1,j])*(x0[1,j+1]-x0[1,j])+(y0[2,i]-y0[1,j])*
(y0[1,j+1]-y0[1,j])+(z0[2,i]-z0[1,j])*(z0[1,j+1]-z0[1,j]))/
((x0[1,j+1]-x0[1,j])^2+(y0[1,j+1]-y0[1,j])^2
+(z0[1,j+1]-z0[1,j])^2);

for {i in 2..N} {
  for{j in 1..N} {
  if alfa21[i,j]>1 then let alfa21[i,j] :=1;
  if alfa21[i,j]<0 then let alfa21[i,j] :=0;
  }
}

#  ... (calculate all other alfa's the similar way as above)
# recalculate all the betaaij's and betaij's for this iteration
let {i in 2..N, j in 1..N} betaa21[i,j] :=
(alfa21[i,j]*x0[1,j+1]+(1-alfa21[i,j])*x0[1,j]-x0[2,i])^2+
```

```
(alfa21[i,j]*y0[1,j+1]+(1-alfa21[i,j])*y0[1,j]-y0[2,i])^2+
(alfa21[i,j]*z0[1,j+1]+(1-alfa21[i,j])*z0[1,j]-z0[2,i])^2;


for {i in 2..N}{
let minumber :=betaa21[i,1];
let beta21[i] :=1;
    for{j in 1..N}{
    if betaa21[i,j] < minumber
    then {let minumber:=betaa21[i,j];
          let beta21[i]:=j;}
         }
    }


#  ... (recalculate all other betaa's and beta's the similar way
     #     as above)
let iter:=iter+1;
let previous-obj :=k;
solve;
option display_width 120;
option gutter_width 1;
```

```
display k;

display x;

display y;

display z;

display xdt;

display ydt;

display zdt;

display iter;

}until k <= previous-obj+tolerance;
```

Bibliography

[1] C.K. Anand, M. Thompson, D.H. Wu, and T. Cull. Teardrop, a novel non-raster readout for true FISP. *ISMRM*, vol. 9, p.1804, 2001.

[2] E. Atalar, and E.R. McVeigh. Minimizing dead-periods in MRI pulse sequences for imaging oblique planes. *Magn. Reson. Med.*, vol. 32, pp. 773-777, 1994.

[3] K.S. Nayak, and B.S. Hu. The Future of Real-time Cardiac Magnetic Resonance Imaging. *Current Cardiology Reports*, vol. 7, pp. 45-51, 2005.

[4] B.D. Bolster, and E. Atalar Jr. Minimizing dead-periods in flow-encoded or compensated pulse sequences while imaging in oblique planes. *J. Magn. Reson. Imaging*, vol. 10, pp. 183-192, 1999.

[5] S.C. Bushong. *Magnetic Resonance Imaging: Physical and Biological Principles.* Mosby, Toronto, 2nd edition, 1996.

[6] B.M. Dale, and J.L. Duerk. Time-optimal control of gradients. In: *Proceedings of the 10th Annual Meeting of ISMRM, Honolulu*, p. 2361, 2002.

[7] J.L. Duerk. Magnetic resonance imaging gradient modulation technique for motion artifact reduction and flow quantification. *Ph.D. Thesis*, Dept. Biomed. Eng., Case Western Reserve Univ., Cleveland, OH, 1986.

[8] R. Fourer, D.M. Gay, and B.W. Kernighan. *AMPL: A Modeling Language For Mathematical Programming.* The Scientific Press, 1993.

[9] J.P. Groen, P. van Dijk, and J.J.E. In den Kleef. Design of flow adjustable gradient waveforms. In: *Book of Abstracts*, Soc. Magnetic Resonance in Medicine, Sixth Annual Meeting, p. 868, 1987.

[10] E.M. Haacke, and G.W. Lenz. Improving MR image quality in the presence of motion by using rephasing gradients. *American Journal Roentgenology*, vol. 148, pp. 1251-1258, 1987.

[11] E.M. Haacke, R.W. Brown, M.R. Thompson, and R. Venkatesan. *Magnetic Resonance Imaging: Physical Principles and Sequence Design.* John Wiley and Sons, Toronto, 1999.

[12] O. Heid. The fastest circular k-space trajectories. In: *Proceedings of the 10th Annual Meeting of ISMRM*, Honolulu, p. 2364, 2002.

[13] P.J. Keller, and F.W. Wehrli. Gradient moment nulling through the N-th moment. Application of Binomial Expansion Coefficients to Gradient Amplitudes. *Journal of Magnetic Resonance*, vol. 78, pp. 145-149, 1988.

[14] S. Ljunggren. A simple graphical representation of fourier-based imaging methods. *Journal of Magnetic Resonance*, vol. 54, pp. 338-343, 1983.

[15] V.L. Morgan, R.R. Price, and C.H. Lorenz. Application of linear optimization techniques to MRI phase contrast blood flow measurements. *Magnetic Resonance Imaging*, vol. 14, pp. 1043-1051, 1996.

[16] D.G. Nishimura. *Principles of Magnetic Resonance Imaging.* Dept. of Electrical Engineering, Stanford University, CA, 1996.

[17] P.M. Pattany, J.J. Phillips, L.C. Chiu, J.D. Lipcamon, J.L. Duerk, J.M. McNally, and S.N. Mohapatra. Motion artifact suppression technique (MAST) for MR imaging. *Journal of Computer Assisted Tomogrraphy*, vol. 11, no. 3, pp 369-377, 1987.

[18] J.G. Pipe, and T.L. Chenevert. A progressive gradient moment nulling design technique. *Magnetic Resonance in Medicine*, vol. 19, pp. 175-179, 1991.

[19] O.P. Simonetti, J.L. Duerk, and V. Chankong. An optimal design method

for magnetic resonance imaging gradient waveforms. *IEEE Transactions on Medical Imaging*, vol. 12, pp. 350-360, 1993.

[20] O.P. Simonetti, J.L. Duerk, and V. Chankong. MRI gradient waveform design by numerical optimization. *Magnetic Resonance in Medicine*, vol. 29, pp. 498-504, 1993.

[21] D.B. Twieg. The k-trajectory formulation of the NMR imaging process with applications in analysis and synthesis of imaging methods. *Medical Physics*, vol. 10, pp. 610-621, 1983.

[22] R.E. Wendt III. Interactive design of motion compensated gradient waveforms with a personal computer spreadsheet program. *Journal of Magnetic Resonance*, vol. 1, pp. 87-92, 1991.

[23] Q.S. Xiang, and O. Nalcioglu. A formalism for generating multiparametric encoding gradients in NMR tomography. *IEEE Transactions on Medical Imaging*, vol. MI-6, no. 1, pp. 14-20, 1987.

[24] H. Yamagata, M.H. Bounocore, B. Telford, and J.A. Sanders. Optimized gradient pulses for MR quantitative flow imaging. *Radiology*, vol. 173(P), p. 162, 1989.