Politecnico di Milano
Scuola di Ingegneria Industriale e dell'Informazione
Corso di Laurea Magistrale in Ingegneria Informatica
Dipartimento di Elettronica, Informazione e Bioingegneria

Computing intrinsic signal to noise ratio using Dyadic Green's function for application on magnetic resonance imaging coil design

Relatore: Prof. Pierluca LANZI

Tesi di laurea di:
Matteo MARINELLI Matr. 864595

Anno Accademico 2019–2020

Abstract

Accurate modeling of electromagnetic effects on patients and objects is becoming increasingly important as higher magnetic field strengths are used in magnetic resonance systems. Numerical simulations have good result in this area, but their execution time make the exploration of the space of possibility extremely small. Exists anyway a study able to make rigorous and fast simulations using mode expansions with dyadic Green's functions. This study had amazing result in the magnetic resonance systems coil design, and the tool to perform the simulations is widely used in this field of research.

The aim of the work is analyzing the program distributed to the universities to perform simulation and presenting eventual enhancements.

Is presented the program, with the theory behind it and the algorithms that implement this theory. Some modification are descripted explaining also which kind of enhancements they bring. Finally some experiments are performed to compare the two version reported in this work.

What emerge from this work is that the original tool is too much specialized and not really user friendly, aspects solved by the second version. The performances anyway, even if them are enhanced by the revised code, have a gain not really meaningful, but still all considered the modification of the original offer a valid substitute.

Sommario

La costruzione di modelli accurati degli effetti elettromagnetici sta diventando sempre più importante in quanto campi magnetici di forza sempre maggiore vengono utilizzati nei sistemi di risonanza magnetica. L'interazione di campi elettromagnetici con tessuti biologici ad alta frequenza produce perturbazioni nelle linne di campo specifiche da tessuto a tessuto, richiedendo uno studio accurato nel design delle coils per migliorare la qualità dele immagini e evitare effetti indesiderati nei pazienti. Le thecniche di parallel magnetic resonance imaging tuttavia promettono di trovare soluzione a uesto genere di problemi. Di risposta, il più alto repporto segnale rumore disponibile per alti valori di forza del campo permetterebbe maggiori gradi di accellerazione nel parallel magnetic resonance imaging, quindi produrre immagini di qualità con meno irradiazione di energia e limitando l'interazione con oggetti e altro materiale che potrebbe disturbare il segnale. Essendo il numero di canali disponibile per la risonanza magnetica incrementato per permettere un'acquisizione più veloce e eccitazioni multiple delle coils, costruire prototipi per le coil è diventato difficile e costoso; quindi, il design di queste viene affidato prevalentemente a simulazioni elettrodinamiche. Le simulazioni numeriche tramite tecniche come il metodo delle differenze finite sono normalmente utilizzate nelle analisi elettromagnetiche con modelli dettagliati del corpo umano. anche se questi approcci sono rigorosi e hanno presentato una buona concordanza con dati sperimentali, essi sono estremamente lunghi nell'esecuzione e la loro complessità nimerica cresce rapidamente con il numero delle coils simulate. La durata delle simulazioni restringe anche i numero di configurazioni che possono essere realisticamente esplorate, limitando la generalità dei risultati. Infatti è stato dimostrato che c'è una forte correlazione fra il rapporto segnale rumore e fattori fisici e geometrici, così come con forma, dimensione e proprietà elettriche dei tessuti. Quindi assumono un ruolo importante approcci rigorosi ma rapidi per simulazioni elettrodinamiche che operino su queste dipendenze fondamentali usand semplici modelli geometrici comparativi. Un esempio di questo tipo è presentato dal Professor Lattanzi in un suo studio, il cui metodo usa l'espansione modale e la dyadic Green's function per caratterizzare un campo magnetico completo in un dielettrico sferico. Per lo studio, è stato scritto un programma che implementa il metodo teorico per contribuire al design delle coils paragonando il loro rapporto sengale rumore con

l'ultimate intrinsic signal to noise ratio, il più alto rapporto segnale rumore ottenibile da una determinanta configurazione. Questo strumento riscuote molto succeso nei dipartimenti di radiologia, tanto da spingere gli autori a iniziare lo sviluppo di un applicazione che lo renda facile da usare.

Lo scopo di questo lavoro è analizzare e presentare il codice scritto dal Professor Lattanzi e, consecutivamente, proporre modifiche utili per generare una versione del programma che sia facile da usare e capire, ottimizzata, generalizzata e più performante. L'algoritmo originale è pensato per scopi di ricerca e calcola il rapporto segnale rumore ottimale e quello per una determinata configurazione geometrica. Il paragone fra i valori ottenuti indica la qualità della configurazione e consequenzialmente dell'immagine. Attualmente le simulazioni sono disponibili per un qualsiasi numero di coils ma limitate a una sfera a tre strati che simula il cranio umano.

Nel primo capitolo viene descritto il codice originale, partendo dall'input, passando poi alla logica e terminando con l'output. La logica è separata in teoria e metodi di implementazione della stessa. Il capitolo termina esponendo le limitazioni del programma e gli effetti che causano.

Il secondo capito è dedicato alla presentazione di modifiche con lo scopo di risolvere i problemi esistenti. Viene proposta una nuova struttura, non più orientata semplicemente alla ricerca, ma più user friendly, identificando come users possibili non solo ricercatori esperti nel settore ma anche eventuali utenti non specializzati. Vengono descritte delle modifiche al codice consistenti in un approccio a oggetti per la gestione della sfera e in un approccio ricorsivo per il calcono dei coefficienti della dyadic Green's function, spiegando anche i benefici apportati al codice. Una sezione invece tratta di ciò che del codice originale può essere riutilizzato in una versione innovativa, spiegando le motivazioni che giustificano il mantenimento. Viene esposta anche una descrizione degli esperimenti di validazione del codice scritto seguendo le proposte del capitolo. Infine i due codici vengono paragonati, sottolineando cosa la nuova versione apporti alla vecchia.

L'ultimo capitolo parla dei test effettuati per verificare le prestazioni in termini memoria utilizzata, tempo di esecuzione e precisione. Viene anche sollevato un argomento su un possibile approccio ripetitivo del codice originale, producendo una serie di versioni specializzate nella processazione di uno specifico numero di layers. Una breve sezione è dedicata ai dati utilizzati nello svolgimento dei vari test. Infine una sezione riassume e valuta i risultati ottenuti. Il capitolo e il lavoro si chiudono con una sezione sulle tecnologie utilizzate.

Contents

Introduction

Accurate modeling of electromagnetic (EM) effects is becoming increasingly important as higher magnetic field strengths are used in magnetic resonance (MR) systems. The interactions of the EM field with biological tissues at high frequencies result in tissue-specific perturbations of EM field patterns, requiring appropriate coil designs to improve image quality and to avoid adverse effects in patients. Parallel MR imaging (MRI) [5,18,20] techniques are promising solutions to address these issues. In reception, the increased signal-to-noise ratio (SNR) available at higher field strengths allows for higher degrees of acceleration in parallel MRI, therefore enabling efficient imaging with reduced energy deposition and also limiting susceptibility artifacts and other time-dependent signal perturbations. As the number of channels available in MR systems has increased to enable faster acquisitions and multiple coil excitation, building prototypes of coil arrays has become more difficult and expensive; therefore, the design of coil arrays has relied ever more upon electrodynamic simulations. Numerical simulations with techniques such as the finite difference time domain technique are normally used for EM analyses with detailed heterogeneous models of the human body [4,8,9]. Although these approaches are rigorous and the results have shown good agreement with experimental data, they are time consuming and their numerical complexity grows rapidly as the number of modeled coils increases. The duration of these simulations also restricts the number of different coil-sample configurations that can be realistically explored, limiting the generality of the results. In fact, it has been shown that there is a strong dependency of SNR upon geometrical and physical factors [13,25], such as shape and dimensions of the object and the conductors, or electrical properties of the tissues. There is therefore a valuable role for rapid but rigorous electrodynamic simulation approaches that may yield insight into these fundamental dependencies using comparatively simple geometrical models. An example of rapid and rigorous electrodynamic simulation approach is the solution presented by Professor Lattanzi in [12], that use mode expansions with dyadic Green's functions (DGFs) [21] to characterize the fullwave EM field in a dielectric sphere. For the study [12] was written a program (P1) which perform the mode expansions with DGFs to help and guide the design of MRI coils according to the ultimate intrinsic SNR (UISNR), the best possible SNR in an ob-

ject. This algorithm had a great success in the radiology departments, so its authors choose to start the development of an application to make P1 easy to use for research reasons.

The purpose of this work is to analyze and present P1 and, consequentially, present some useful modification that produced another program (P2) to make the code more easy to use, more general, more optimized and more performing. P1 is a code that, thanks to mode expansions with DGFs, calculate the UISNR and the SNR of coils in simulated position. The comparison between those two values indicate how good is the coils configuration. Actually the simulations are available for any number of coil but the body to scan is limited to a sphere of three layers that wants to be a sample of an human head.

In the first chapter is presented P1. The chapter starts with a section on the input set. The approach is not just from a algorithmic point of view, but is given also a representation of what the input try to simulate. Following there is a digression on the theory P1 is based on. It is divided in sections about the computation of DGF and EM fields, the computation of UISNR and the relative current patterns, the coil configuration SNR and relative current patterns. There is also an illustration of the implementation of the theory in the code, where there are some highlights on crucial parts of the program. The list of section on the description of the code terminate with the output of P1. The last section of the chapter is about the limitation of the code and the reason why P1 has to be enhanced.

The second chapter instead is totally about P2, with an approach oriented to underline the differences from P1 and why those modifications are better respect to the previous solution. The chapter open with the description of a new structure designed not thinking at the research but user oriented, individuating as user not only experts of the field but in general also people may be in contact with P2 but without any notions of electromagnetism. The chapter continue with the presentation of two modifications in the algorithm itself: the adoption of an object oriented solution to manage the sphere, its layers and EM features, and a recursive approach to calculate the DGFs coefficients. Those two modifications bring a new level of optimization and generalization to P2. The description of P2 terminates with what was chosen to keep from P1 and the explanations of the choice. The chapter continue with the validation of P2, presenting the experiments necessary to the purpose. The last section of the chapter about P2 make a comparison with P1 considering the modifications apported tho the code.

Last chapter has the purpose to make some performances comparisons between P1 and P2, in order to see if P2 is a valid replacement to P1. The chapter open with an introduction to the experiments and an argument about the possibility to give generality to P1 with several equivalent codes specialized each one on configurations with different number of layers. Than are presented the experiments to test per-

formances and their results. In order are analyzed memory consumption, execution time and precision. After the experiments there is a short section to present the small dataset used to perform the simulations. The data are about the tissues of the human body simulated in the sphere and the coil characteristics. After the dataset is present a summary on the results of the experiments on the performances. The chapter and the work, before the conclusion, end with a description of the technology used in the development of P1 and P2 and to perform the various testing.

Chapter 1

Current solution

1.1 Input and Preparation to Calculations

P1 is a system which perform the calculation of Optimal Current Patterns for the UISNR and for circular surface coils, starting from the mathematical reproduction of the setting of an MRI. Inputs needed to represent properly the setting concerns the characteristics of the body to scan, the asset of the coils, EM conditions created by the machine and information for the creation of the field of view (FOV). As inputs are asked also a set of flags to indicate which part of the calculation perform and some options to organize the graphical output, this parts will not be described in this chapter since are just related to the execution of the code and not to its logic.

The body, up to now, is limited to an approximation of an human head, described by a three concentric layers sphere that want to represent respectively brain (L3), skull's bones (L2) and skin (L1) of the head. The innermost layer L3 radius has to be indicated in the inputs in meters, instead for the following two has to be indicated the radius increment, from the sum of the previous layer radius and the increment is derived the layer radius. All L3, L2 and L1 also require the dielectric constant ε and the conductivity σ in $1/(\Omega$*m) to describe the EM properties of tissues.

The coils are supposed as circular loops of copper and represented by a nx2 matrix that express the rotations of each of the n coils respect the center of the three layers sphere, this data is completed by a radius that indicate the distance form the center of the sphere and the radius of the coil itself. Distance from the center of the sphere and coil radius are useful to calculate the effective distance of the coil copper wire and the center of the sphere as $\sqrt{d^2 + R^2}$, where d and R are the two distances. Those three elements describe a fourth layer, a spherical surface where the coils are distributed, still concentric with the dielectric sphere. The radius of this layer has to be grater than the radius of the last layer of the sphere.

The MRI machine generate the EM field enabling current to flow trough the coils. This situation produces EM waves with a certain frequency chose by the user, from

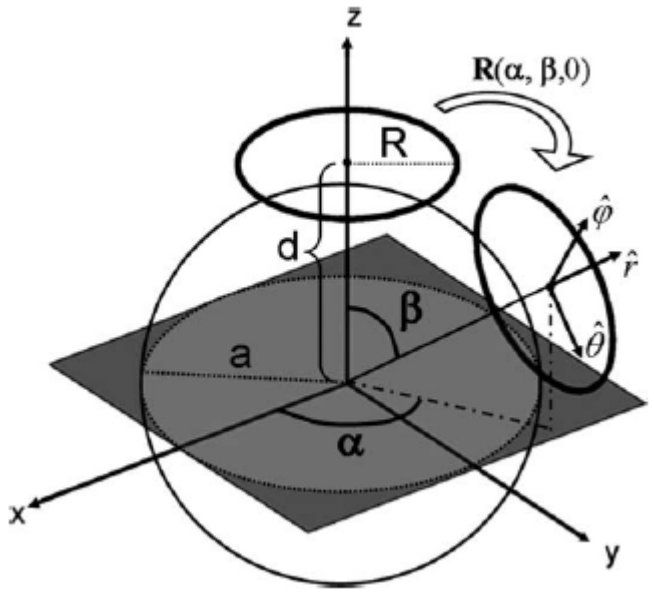

Figure 1.1: Schematic representation of the spherical sample geometry, with two exemplary loop coils arranged on a spherical surface at distance $\sqrt{d^2 + R^2}$ from the center of the sphere. SNR calculation is performed on a transverse plane through the center of the object. Picture from [12].

which is possible to calculate the field strength. P1 is also able to simulate parallel imaging, so is necessary specify a vector nx2 of accelerations in the two dimensions (a vector 1x2 means just one imaging). Frequency and acceleration, together with some other fields like mode of expansion and the image output size, are part of the input that describe the activity of the machine. The input has also to specify which current basis functions has to be included in the calculation (1 = divergence-free magnetic dipole, 2 = curl-free electric dipole).

The FOV is the region of the sample that will be seen at the end of calculation, so also the part where has to be performed the calculation. With some parameters like an offset and an angle, both three elements vectors, it describes a plane passing through the center of the sphere and on this plane can be identified, thanks to their coordinates, the points where the user want to check the UISNR. In order to generate the output images, are also inputs the patient position, can be 'head first' o 'feet first' according to which part the machine will meet first, and patient orientation, can be 'supine', 'prone', 'ldecub' (on the left side), or 'rdecub' (on the right side).

Before starting the calculation, from the inputs are calculated all the quantities that can be derived from those, this in order to reduce the input as much a possible. In this section are managed all the issues about the consistency of the flag and the execution of some adjustments, like conversion in spherical coordinates and the assemble of physical quantities and inputs in thematic structures. The approach to the memory consumption is a static one: all arrays and/or matrix are defined as zeros

arrays of a fixed dimension avoiding any kind of threaten deriving from a dynamic allocation of memory. Once all the preparation procedures are solved, according to the flag specification, the calculation will start. In the following subsections the topics will be the relevant parts of the calculation and the theory behind them. P1 is based on four papers where all the calculation is fully described that are Ref.s [11–13, 15], the presentation of the theory is helped taking some hints from those works in order to make the explanation as clear as possible.

1.2 Electromagnetic Field Expansion in a Dielectric Sphere with Dyadic Green's Functions

Finding the DGF is the first checkpoint for the calculation. From Ref. [15], to construct the EM DGFs in the layered media (sphere) the method adopted in P1 is the scattering superposition [21], using the vector wave functions to describe it. In the paper is given the DGF:

$$\overline{G}_e^{(f,s)}(r,r') = \overline{G}_{0e}(r,r')\delta_f^s + \overline{G}_{es}^{(f,s)}(r,r') \tag{1.1}$$

where the scattering dyadic Green's function $\overline{G}_{es}^{(f,s)}(r,r')$ describes an additional contribution of the multiple reflection and transmission waves in the presence of the boundary of dielectric media while the unbounded dyadic Green's function$\overline{G}_{0e}(r,r')$ represents the contribution of the direct waves from radiation sources in an unbounded medium, and the superscript (f,s) denotes the layers where the field point and source point locate, respectively, while the subscript s identifies the scattering dyadic Green's functions. Applying the method of contour integration in the complex h-plane, is obtained the dyadic Green's function in the unbounded medium as a result of the residue theorem, given by:

$$\begin{aligned}\overline{G}_{0e}(r,r') = &\frac{\widehat{rr}}{k_s^2}\delta(r-r') \\ &+ \frac{ik_s}{4\pi}\sum_{l=0}^{\infty}\sum_{m=0}^{l}(2-\delta_m^0)\frac{2l+1}{l(l+1)}\frac{(m-l)!}{(l+m)!} \\ &\times \begin{cases} M_{l,m}^{(1)}(k_s)M_{l,m}'(k_s) + N_{l,m}^{(1)}(k_s)N_{l,m}'(k_s) & r \geq r' \\ M_{l,m}(k_s)M_{l,m}'^{(1)}(k_s) + N_{l,m}(k_s)N_{l,m}'^{(1)}(k_s) & r \leq r' \end{cases}\end{aligned} \tag{1.2}$$

where the prime denotes the coordinates $(r',\theta',\phi')$of the current source J_f, l and m are the expansion indices, and $M_{l,m}(k) = j_l(kr)X_{l,m}(\theta,\varphi)$ stands for the electric field of the $TE_{l,m}$ mode, with $X_{l,m}$ a vector spherical harmonic of order (l, m), while $N_{l,m}(k) = \frac{1}{k}\nabla \times M_{l,m}(k)$ represents that of the $TM_{l,m}$mode. Under the spherical coordinates, the electromagnetic fields usually consist of the radial wave

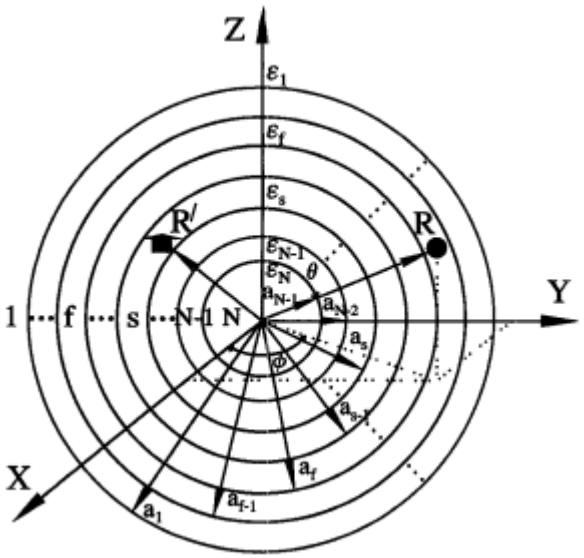

Figure 1.2: Geometry of spherically multilayered medium

modes propagating outwards and inwards. Hence, under the assumption the current source is located in the layer s, the scattering dyadic Green's function for the layer f (= 1,2,..., N) is constructed among the multi layers with the spherical Bessel and Hankel functions as follows:

$$\begin{aligned}
\overline{G}_{es}^{(fs)}(r,r') &= \frac{\widehat{rr}}{k_s^2}\delta(r-r') \\
&+ \frac{ik_s}{4\pi}\sum_{l=0}^{\infty}\sum_{m=0}^{l}(2-\delta_m^0)\frac{2l+1}{l(l+1)}\frac{(m-l)!}{(l+m)!} \\
&\times\{(1-\delta_f^N)M_{l,m}^{(1)}(k_f)[(1-\delta_s^1)A_M^{fs}M'_{l,m}(k_s)+(1-\delta_f^N)B_M^{fs}M'^{(1)}_{l,m}(k_s)] \\
&+(1-\delta_f^N)N_{l,m}^{(1)}(k_f)[(1-\delta_s^1)A_N^{fs}N'_{l,m}(k_s)+(1-\delta_s^N)B_N^{fs}N'^{(1)}_{l,m}(k_s)] \\
&+(1-\delta_f^1)M_{l,m}(k_f)[(1-\delta_s^1)C_M^{fs}M'_{l,m}(k_s)+(1-\delta_s^N)D_M^{fs}M'^{(1)}_{l,m}(k_s)] \\
&+(1-\delta_f^1)N_{l,m}(k_f)[(1-\delta_s^1)C_N^{fs}N'_{l,m}(k_s)+(1-\delta_s^N)D_N^{fs}N'^{(1)}_{l,m}(k_s)]\}
\end{aligned} \tag{1.3}$$

where $A_{M,N}^{fs}$,$B_{M,N}^{fs}$,$C_{M,N}^{fs}$ and $D_{M,N}^{fs}$are the coefficients of scattered dyadic Green's function to be solved, N represents the number of the layers of the spherical medium. The superscript (1) denotes that the third-type spherical Bessel function or the first-type spherical Hankel function should be chosen in the expression of the spherical wave vector functions. For the rest of the vector wave functions, is chosen the normal first-type spherical Bessel function because this type of function can be used to represent both out-going and in-coming waves.

Find the coefficients $A_{M,N}^{fs}$,$B_{M,N}^{fs}$,$C_{M,N}^{fs}$ and $D_{M,N}^{fs}$ is possible thanks to a system that can be found starting from the boundary conditions satisfied by the electric

type of the DGF:

$$\widehat{r} \times \overline{G}_e^{(f,s)} = \widehat{r} \times \overline{G}_e^{[(f+1),s]} \tag{1.4}$$

$$\frac{1}{\mu_f}\widehat{r} \times \nabla \times \overline{G}_e^{(f,s)} = \frac{1}{\mu_{f+1}}\widehat{r} \times \nabla \times \overline{G}_e^{[(f+1),s]} \tag{1.5}$$

Calculation, absent in the code (you can find them in [15]), is skipped and is immediately shown the coefficients system:

$$\begin{bmatrix} A_{M,N}^{(f+1)s} + \delta_{f+1}^{s} & B_{M,N}^{(f+1)s} \\ C_{M,N}^{(f+1)s} & D_{M,N}^{(f+1)s} \end{bmatrix} = \begin{bmatrix} \frac{1}{T_{Ff}^{H,V}} & \frac{R_{Ff}^{H,V}}{T_{Ff}^{H,V}} \\ \frac{R_{Pf}^{H,V}}{T_{Pf}^{H,V}} & \frac{1}{T_{Pf}^{H,V}} \end{bmatrix} \cdot \begin{bmatrix} A_{M,N}^{fs} & B_{M,N}^{fs} \\ C_{M,N}^{fs} & D_{M,N}^{fs} + \delta_{f}^{s} \end{bmatrix} \tag{1.6}$$

where $T_{(P,F)f}^{H}$ and $R_{(P,F)f}^{H}$ represent the centripetal and centrifugal transmission and reflection contributions from TE waves (corresponding to the superscript H) while $T_{(P,F)f}^{V}$ and $R_{(P,F)f}^{V}$ represent the centripetal and centrifugal transmission and reflection contributions from TM waves (corresponding to the superscript V). All the contributions $T_{(P,F)f}^{H,V}$ and $R_{(P,F)f}^{H,V}$ are easy to find since from the input is possible to compute everything needed [15]. Is not easy instead finding all the coefficients $A_{M,N}^{fs}$,$B_{M,N}^{fs}$,$C_{M,N}^{fs}$ and $D_{M,N}^{fs}$, in fact the system shows how each of them depends form the coefficients of the layer directly under. Since P1 is dealing with a MRI, according to [15], $A_{M,N}^{Ns}$,$B_{M,N}^{Ns}$,$C_{M,N}^{1s}$ and $D_{M,N}^{1s}$ are considered equal to zero since is possible to assume f=N, , where the current distribution is placed for sure outside the sphere, is also possible assuming s=1 and consequentially all $A_{M,N}^{fs}$ and $C_{M,N}^{fs}$ coefficients are equal to zero as well. Under those assumption the unknown coefficients will be $B_{M,N}^{f1}$ with f= (1,2,3,...(N-1)) and $D_{M,N}^{f1}$ with f= (2,3,...N) and the system can be simplified as follow:

$$\begin{cases} B_{M,N}^{21} = \frac{1}{T_{F1}^{H,V}} \left[B_{M,N}^{11} + R_{F1}^{H,V} \right] \\ D_{M,N}^{21} = \frac{1}{T_{P1}^{H,V}} \left[1 + R_{P1}^{H,V} B_{M,N}^{11} \right] \end{cases} \tag{1.7}$$

$$\begin{cases} B_{M,N}^{i1} = \frac{1}{T_{F(i-1)}^{H,V}} \left[B_{M,N}^{(i-1)1} + R_{F(i-1)}^{H,V} D_{M,N}^{(i-1)1} \right] \quad i = (2...(N-1)) \\ D_{M,N}^{i1} = \frac{1}{T_{P(i-1)}^{H,V}} \left[R_{P(i-1)}^{H,V} B_{M,N}^{(i-1)1} + D_{M,N}^{(i-1)1} \right] \end{cases} \tag{1.8}$$

$$\begin{cases} 0 = \frac{1}{T_{F(N-1)}^{H,V}} \left[B_{M,N}^{(N-1)1} + R_{F(N-1)}^{H,V} D_{M,N}^{(N-1)1} \right] \\ D_{M,N}^{N1} = \frac{1}{T_{P(N-1)}^{H,V}} \left[R_{P(N-1)}^{H,V} B_{M,N}^{(i-1)1} + D_{M,N}^{(i-1)1} \right] \end{cases} \tag{1.9}$$

This is a 2N equations with 2N unknown parameters linear system, so admit a solution. Now all needed for expression 1.1 calculation is known, than all the calculation showed up to now are performed in P1 to find the DGF.

As shown in [12] the DGF formalism enables calculation of the electric field

resulting from any spatial current distribution J(r) as:

$$\mathbb{E}(r) = i\omega\mu_0 \iiint_V \overline{G}(r, r') \cdot J(r')dV' \tag{1.10}$$

where i is the imaginary unit, ω is the angular frequency, μ_0 is the magnetic permeability in free space.

The current is constrained to flow only on a spherical surface of radius b (which need not coincide with the surface of the dielectric sphere itself):

$$J(r, \theta, \varphi) = K(\theta, \varphi)\frac{\delta(r-b)}{b^2 sin\theta} \tag{1.11}$$

In the most general case, the surface current density K may consist of both magnetic-type (divergence-free) and electric-type (curl-free) components, indicated with the superscript (M) and (E), respectively, and in P1 is expressed with a mode expansion. The generic surface current mode then takes the form of:

$$K_{l,m}(\theta, \varphi) = -i\sqrt{l(l+1)}\left[W_{l,m}^{(M)} X_{l,m}(\theta, \varphi) + W_{lm}^{(E)} \hat{r} \times X_{l,m}(\theta, \varphi)\right] \tag{1.12}$$

where, $W_{l,m}^{(M)}$ and $W_{lm}^{(E)}$ are the series expansion coefficients representing divergence-free and curl-free surface current contributions, respectively, $X_{l,m}$is a vector spherical harmonic of order (l, m). The DGF method shown at the beginning of this section allows solution of the scattering problem to determine the EM field impressed upon the sphere by the vector source current density K. From expression 1.10, is derived (Appendix A of [12]) an expression for the electric field inside the sphere and then, using Maxwell's equations, the corresponding expression for the magnetic field is calculated:

$$\begin{aligned} \mathbb{E}(r) &= \tfrac{\omega\mu_0}{k_{in}} \textstyle\sum_{l=0}^{+\infty} \sum_{m=-l}^{+l} \left[M_{l,m}(k_{in}, r)V_{l,m}^M + N_{l,m}(k_{in}, r)V_{l,m}^N\right] \\ \mathbb{B}(r) &= i\mu_0 \textstyle\sum_{l=0}^{+\infty} \sum_{m=-l}^{+l} \left[N_{l,m}(k_{in}, r)V_{l,m}^M + M_{l,m}(k_{in}, r)V_{l,m}^N\right] \end{aligned} \tag{1.13}$$

Note that harmonic time variation is assumed in everything to follow, and a common factor of $e^{-i\omega t}$ is omitted but assumed for all fields and currents throughout the theoretical derivation. The complex wave number inside the sphere is calculated as $k_{in}^2 = \omega\varepsilon_r\varepsilon_0\mu + i\omega\mu_0\sigma$, where ω is the angular frequency; ε_r and σ are the relative permittivity and the electrical conductivity of the dielectric material, respectively; μ_0 and ε_0 are the magnetic permeability and the electric permittivity in free space, respectively. The weighting coefficients $V_{l,m}^M$ and $V_{l,m}^N$ are derived by multiplying the expansion coefficients of the current density with a transformation matrix T that

accounts for boundary conditions at the surface of the sphere:

$$\begin{aligned} V &= \begin{pmatrix} V_{l,m}^{M} \\ V_{l,m}^{N} \end{pmatrix} \\ &= \begin{pmatrix} -i\sqrt{l(l+1)}h_l^{(1)}(k_0 b)C_l & 0 \\ 0 & \frac{-i\sqrt{l(l+1)}}{k_0 b}\frac{\partial\left[rh_l^{(1)}(k_0 b)\right]}{\partial r}\Big|_{r=b} D_l \end{pmatrix} \\ &\cdot \begin{pmatrix} W_{l,m}^{(M)} \\ W_{l,m}^{(e)} \end{pmatrix} = T^T \cdot W \end{aligned} \tag{1.14}$$

where the superscript T indicates the transpose of the matrix, $h_l^{(1)}$ is the spherical Hankel function of the first kind of order l, and the complex wave number outside the sphere is calculated as $k_0^2 = \omega^2 \mu_0 \varepsilon_0$. The coefficients C_l and D_l in the matrix T are determined by applying the Dirichlet boundary conditions at the surface of the sphere. Note that the vectors V, T, and W are defined for given (l, m).

It should be noted that the individual EM field modes in expression 1.13, matched to their associated surface current modes, can be used as if they correspond to hypothetical coil current distributions. This observation make possible in P1 to find optimal weighting coefficients that allow combinations of modes for maximum SNR. Ideal current patterns are computed as a weighted sum of the current modes in expression 1.12, using the same coefficients.

1.3 Calculation of Ideal Current Patterns for Ultimate Intrinsic SNR

Ultimate intrinsic SNR (UISNR) [16,17,19,25], the theoretical best possible SNR independent of any particular coil geometry, is calculated in P1 using the EM modes in the basis set to maximize the general SNR expression. In the case of Cartesian SENSE parallel imaging reconstructions [18] followed in the code to perform the calculation, the combination of modes that results in the highest possible SNR is found by solving a constrained minimization problem, which yields the following solution for the optimal series expansion coefficients [18]:

$$W^{opt} = (S^H \Psi_{mode}^{-1} S)^{-1} S^H \Psi_{mode}^{-1} \tag{1.15}$$

where S is the sensitivity matrix, and Ψ_{mode} is the modes' noise covariance matrix. The sensitivity matrix contains the complex signal sensitivities associated with each

mode at the target position r_0 and at all aliased positions:

$$S(r) = \begin{pmatrix} S_1(r_0) & \dots & S_1(r_{R-1}) \\ \dots & \dots & \dots \\ S_{L_{mode}}(r_0) & \dots & S_{L_{mode}}(r_{R-1}) \end{pmatrix} \tag{1.16}$$

where R is the acceleration (or reduction) factor and $L_{mode} = 2(l_{max}+1)^2$ is the total number of modes corresponding to the expansion order l_{max}, which is chosen to ensure convergence of the UISNR calculations. The elements of S is derived by applying the principle of reciprocity [6], which allows calculation of the receive sensitivity of a coil in terms of the left circularly polarized component of the RF magnetic field that would be transmitted at the same position by a unit current flowing around the coil:

$$\begin{aligned} \mathbb{B}_x(r) - i\mathbb{B}_y(r) &= i\mu_0 \sum_{l=0}^{+\infty} \sum_{m=-l}^{+l} (V_{l,m}^M V_{l,m}^N) \begin{pmatrix} N_{l,m}(k_{in},r)_x - iN_{l,m}(k_{in},r)_y \\ M_{l,m}(k_{in},r)_x - iM_{l,m}(k_{in},r)_y \end{pmatrix} \\ &= \sum_{l=0}^{+\infty} \sum_{m=-l}^{+l} V^T F(r) = \sum_{l=0}^{+\infty} \sum_{m=-l}^{+l} W^T T F(r) = \sum_{l=0}^{+\infty} \sum_{m=-l}^{+l} W^T S(r) \end{aligned} \tag{1.17}$$

The only type of noise contributing to ultimate intrinsic SNR is, by definition, sample noise, which means that any conductor is assumed to be perfect (i.e., to have zero resistance and zero reactance) and that the modes' noise covariance matrix Ψ_{mode} in expression 1.15 is calculated by simply integrating electric field products over the volume of the sphere:

$$\begin{aligned} R_{mode} &= \sigma \iiint_V \mathbb{E}(r) \cdot \mathbb{E}^*(r) dV \\ &= \sigma \left| \frac{\omega\mu_0}{k_{in}} \right|^2 \iiint_V \sum_{l=0}^{+\infty} \sum_{m=-l}^{+l} (V_{l,m}^M V_{l,m}^N) \\ &\times \begin{pmatrix} M_{l,m}(k_{in},r) \\ N_{l,m}(k_{in},r) \end{pmatrix} (M_{l'',m'}^*(k_{in},r) N_{l'',m'}^*(k_{in},r)) \begin{pmatrix} V_{l'',m'}^M \\ V_{l'',m'}^N \end{pmatrix}^* dV \\ &= \sum_{l=0}^{+\infty} \sum_{m=-l}^{+l} V^T R_L V^{T*} = \sum_{l=0}^{+\infty} \sum_{m=-l}^{+l} W^T T R_L T^{T*} W^* = \sum_{l=0}^{+\infty} \sum_{m=-l}^{+l} W^T \Psi_{mode} W^* \end{aligned} \tag{1.18}$$

where

$$R_L = \begin{pmatrix} \int_0^a |j_l(k_{in}r)|^2 \, r^2 dr & 0 \\ 0 & \frac{1}{|k_{in}|^2}\left\{\int_0^a \left[\left|\frac{\partial r j_{l1}(k_{in}r)}{\partial r}\right|^2 + l(l+1)\,|j_l(k_{in}r)|^2\right] dr\right\} \end{pmatrix} \tag{1.19}$$

and a and σ are the radius and the electrical conductivity of the sphere, respectively. The UISNR received at any particular position r_0 inside the sphere is calculated using the weights in expression 1.15 as [25]:

$$\begin{aligned} \tilde{\zeta}(r_0) &= \frac{\omega_0 M_0 \sum_{l=0}^{+\infty}\sum_{m=-l}^{+l}(W^{opt})^T S(r)}{\sqrt{4k_B T_S \sum_{l=0}^{+\infty}\sum_{m=-l}^{+l}((W^{opt})^T \Psi_{mode}(W^{opt})^*)_{0,0}}} \\ &= \frac{\omega_0 M_0}{\sqrt{4k_B T_S \cdot (S(r)^H \Psi_{mode}^{-1} S(r))_{0,0}^{-1}}} \end{aligned} \tag{1.20}$$

where M_0 is the equilibrium magnetization, ω_0 is the Larmor frequency, k_B is Boltzmann's constant, and T_S is the absolute temperature of the sample. The "0,0" subscript in the denominator indicates the diagonal element of the matrix in parentheses with an index associated with the target position r_0.

Ideal surface current patterns associated with the ultimate intrinsic SNR are found by applying the optimal weights in expression 1.15 to the individual current modes in expression 1.12 and adding them up:

$$\begin{aligned} I_{Rx}^{ideal}(\theta,\varphi) &= \sum_{l=0}^{+\infty}\sum_{m=-l}^{+l} K_{l,m}^{opt}(\theta,\varphi) \\ &= -i\sum_{l=0}^{+\infty}\sum_{m=-l}^{+l}\sqrt{l(l+1)}(W^{opt})^T \begin{pmatrix} X_{l,m}(\theta,\varphi) \\ \hat{r}\times X_{l,m}(\theta,\varphi) \end{pmatrix} \end{aligned} \tag{1.21}$$

The subscript Rx indicates that the currents refer to the signal reception case.

1.4 Optimal Currents for Circular Surface Coils

After deriving the ultimate intrinsic limit, in this section is described the method used in P1 and explained in [12] for comparing UISNR with particular coil array performance. In the previous sections, is showed that is possible to use a complete set of basis functions to simulate the optimal SNR theoretically achievable with any possible combination of coils. The performance of any actual coil can be simulated with the same formalism by applying the appropriate weighting functions to the general current distribution in expression 1.12. Note that this is possible because the DGF formalism solves the scattering problem rigorously, accounting for boundary conditions associated with the object geometry and its relation to the coil currents.

Consider a loop coil of radius R positioned outside the dielectric sphere as described in the input, with its axis along the z-axis and at a distance d from the center of the sphere in fig.1.1. The surface current distribution for this coil can be defined as:

$$K_z^{coil}(\theta,\varphi) = \hat{\varphi} I \frac{sin\theta}{\sqrt{d^2+R^2}} \delta(r - \sqrt{d^2+R^2}) \delta(cos\theta - \frac{d}{\sqrt{d^2+R^2}} \tag{1.22}$$

where I is the current circulating in the coil and $\delta(r - \sqrt{d^2+R^2}) = 1$, as the current is defined only on the spherical surface of radius $b = \sqrt{d^2+R^2}$. The proportionality factor $\frac{sin\theta}{\sqrt{d^2+R^2}}$ guarantees that the flux of K_z^{coil} through any half plane of constant ϑ (polar angle) is equal to I. The use of delta functions in expression 1.22 amounts to modeling surface current patterns as infinitely thin conductor wires. However, in the practical implementation of our DGF computations, is used a finite number of basis functions (enough to ensure convergence), resulting in circulating surface current patterns and corresponding effective coil conductors with a finite width that depends on the expansion order. The known bunching of current toward the edges of flat conductive ribbons may be simulated using a sufficiently large number of modes, but this measure is quite expensive so, for computational reasons, was considered not necessary for the derivation of key physical insights, since field distributions converge relatively rapidly with mode number, and since the remaining effects of current distribution within conductor cross sections generally result in a simple scaling of effective conductor resistivity. For simplicity, expression 1.22 also assumes a uniform distribution of current azimuthally around the coil loop—an assumption that admittedly does not hold rigorously for sufficiently long conductor paths and at sufficiently high frequency that charge separation becomes significant. Once again, this limitation is not a fundamental impediment to the generation of basic physical insights into coil and field behavior [7,10], and, as is the case for cross-sectional current distribution, if the true azimuthal current distribution is known a priori, it may certainly be simulated with our DGF formalism, using appropriate current weights. The coil current distribution in expression 1.22 can be also expressed as a weighted combination of the basis functions in 1.12, where the curl-free component is set to zero, as currents can only flow in closed patterns for the case of a loop coil:

$$K(\theta,\varphi) = \sum_{l=0}^{+\infty} \sum_{m=-l}^{+l} W_{l,m}^{(M)} (-i\sqrt{l(l+1)} X_{l,m}(\theta,\varphi)). \tag{1.23}$$

Equations 1.22 and 1.23 must be equivalent and, by comparing them, is possible

ot find an expression for $W_{l,m}^{(M)}$ associated with the particular loop coil:

$$W_{l,0}^{coil,(M)} = \frac{-2\pi R}{(l+1)} \left(cot\theta Y_l^0(\theta,\varphi) - csc\theta \sqrt{\frac{2l+1}{2l-1}} Y_{l-1}^0(\theta,\varphi) \right)_{\theta = arccos \frac{d}{\sqrt{d^2+R^2}}} . \tag{1.24}$$

I=1 is substituted, as, for the purposes of modeling coil sensitivities by reciprocity, is interesting the unit current. The current density for a loop coil rotated to an arbitrary position on the sphere fig.1.1 has the same functional form as that of the loop coil along the z-axis, but in a coordinate system rotated with respect to the reference coordinate system:

$$K_{rot}^{coil}(\theta,\varphi) \equiv K_{z'}^{coil}(\theta',\varphi') = \sum_{l=0}^{+\infty} \left(W_{l,0}^{coil,(M)} \right)_{rot} \left[-i\sqrt{l(l+1)} X'_{l,m}(\theta',\varphi') \right], \tag{1.25}$$

where the weights $\left(W_{l,0}^{coil,(M)} \right)_{rot}$ are obtained from expression 1.24 by substituting θ', φ' for θ, φ. Rotated vector spherical harmonics can always be represented as a linear superposition of unrotated vector spherical harmonics [1]:

$$X'_{l,0}(\theta',\varphi') = \sqrt{\frac{4\pi}{2l+1}} \sum_{m=-l}^{+l} Y_l^{m*}(\beta,\alpha) X_{l,m}(\theta,\varphi), \tag{1.26}$$

where α and β define the angular position of the center of the rotated coil on the sphere (Fig. 1). Substituting in expression 1.25, we obtain an expression for the current density of a rotated loop coil in the reference coordinate system:

$$\begin{aligned} K_{rot}^{coil}(\theta,\varphi) &= \sum_{l=0}^{+\infty} \sum_{m=-l}^{+l} \left(W_{l,0}^{coil,(M)} \right)_{rot} \sqrt{\frac{4\pi}{2l+1}} Y_l^{m*}(\beta,\alpha) \left[-i\sqrt{l(l+1)} X_{l,m}(\theta,\varphi) \right] \\ &= \sum_{l=0}^{+\infty} \sum_{m=-l}^{+l} W_{l,m}^{coil,(M)} \left[-i\sqrt{l(l+1)} X_{l,m}(\theta,\varphi) \right], \end{aligned} \tag{1.27}$$

The optimal SNR for an array of L_{coil} receive loop coils is calculated with expression 1.20, after applying the weights $W_{l,m}^{coil,(M)}$ to the receive sensitivity of each mode and to the elements of the noise resistance matrix:

$$\mathbb{B}_{x,c}^{coil}(r) - i\mathbb{B}_{y,c}^{coil}(r) = \sum_{l=0}^{+\infty} \sum_{m=-l}^{+l} \begin{pmatrix} W_{(l,m),c}^{coil,(M)} & 0 \end{pmatrix} S(r) \tag{1.28}$$

$$\mathbb{B}_{x,c}^{coil}(r_n) + i\mathbb{B}_{y,c}^{coil}(r_n) = \sum_{l=0}^{+\infty} \sum_{m=-l}^{+l} \begin{pmatrix} W_{(l,m),c}^{coil,(M)} & 0 \end{pmatrix} C_n \tag{1.29}$$

$$R_{cc'}^{coil} = \sum_{l=0}^{+\infty} \sum_{m=-l}^{+l} \begin{pmatrix} W_{(l,m),c}^{coil,(M)} & 0 \end{pmatrix} \tilde{\Psi}_{mode} \begin{pmatrix} W_{(l,m),c'}^{coil,(M)} \\ 0 \end{pmatrix}^*, \tag{1.30}$$

where c is the coil index, ranging from 1 to L_{coil}. In receive, $\tilde{\Psi}_{mode}$ should include all sources of noise affecting SNR. In P1, in addition to the intrinsic noise due to the sample (see expression 1.18), is modeled coil noise, which is the second largest noise contribution and can be calculated in a straightforward manner with the DGF formalism. Thus, for SNR calculations $\tilde{\Psi}_{mode} = \Psi_{mode} + R_{A=TR_LT^{T*}+R_A}$, where R_L is defined in expression 1.19 and the additional term, R_A, which accounts for resistive power losses in the coil conductors, is calculated by integrating the current distribution on the spherical surface A' with radius b, where the coil lies:

$$\begin{aligned}
\tilde{R}_A &= \frac{1}{\sigma_c d_c} \iint_{A'} K(r') \cdot K^*(r') dA' \\
&= \frac{1}{\sigma_c d_c} \iint_{A'} \sum_{l=0}^{+\infty} \sum_{m=-l}^{+l} \sum_{l'=0}^{+\infty} \sum_{m'=-l'}^{+l'} \begin{pmatrix} W_{l,m}^{coil,(M)} & 0 \end{pmatrix} \begin{pmatrix} K_{l,m}^{(M)} \\ 0 \end{pmatrix} \\
&\begin{pmatrix} K_{l',m'}^{*(M)} & 0 \end{pmatrix} \begin{pmatrix} W_{l',m'}^{coil,(M)} \\ 0 \end{pmatrix}^* dA' \\
&= \sum_{l=0}^{+\infty} \sum_{m=-l}^{+l} \begin{pmatrix} W_{l,m}^{coil,(M)} & 0 \end{pmatrix} \frac{l(l+1)}{\sigma_c d_c} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} W_{l',m'}^{coil,(M)} \\ 0 \end{pmatrix}^* \\
&= \sum_{l=0}^{+\infty} \sum_{m=-l}^{+l} \begin{pmatrix} W_{l,m}^{coil,(M)} & 0 \end{pmatrix} R_A \begin{pmatrix} W_{l',m'}^{coil,(M)} \\ 0 \end{pmatrix}^*.
\end{aligned} \tag{1.31}$$

We see that the current flowing in the conductors causes a power loss inversely proportional to the electrical conductivity (σ_c) and thickness (d_c) of the coil material. In calculating UISNR, is assumed perfect conductors with infinite conductivity and therefore $R_A$becomes zero.

1.5 Output

Output of P1 is strictly dependent from the FOV size and position. If the FOV is placed far from the coils, the result of the UISNR in those points will result lower than if the FOV was near a group of coils.

In the case the FOV is just one point, the output of the program will be a panel like fig.1.3:

In the panel are present information about the the magnetic field intensity, the radius where the coils are placed, the amount of mode expansions and the current basis functions included in the calculation, in this case both divergence-free magnetic dipole and curl-free electric dipole. At the top of this infos there is the value of UISNR

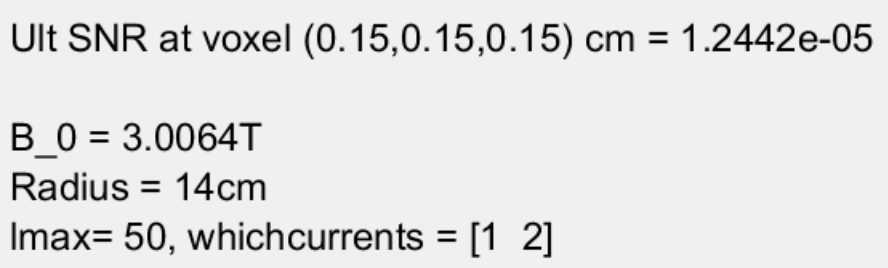

Figure 1.3: UISNR output panel with FOV equal to a single point

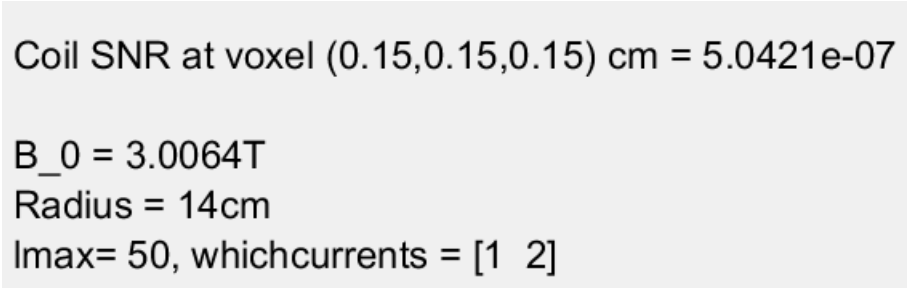

Figure 1.4: Coil SNR output panel for FOV equal to a single point

in the voxel considered by the FOV.

Same structure is for the panel for the coil SNR, with the obvious replacement of the UISNR with the coil SNR at the chosen voxel (fig.1.4).

In case of bidimensional FOV the panel is quite different. For both UISNR (fig.1.5) and coil SNR (1.6) all the pure descriptive data are removed from the panel and the only values shown are the mean of the values on each voxel on the bidimensional panel and the max value. Together with those data is shown the image made by all the voxels of the plane, where the color of each voxel indicate the value of SNR in that position.

Tridimensional case is not available since not implemented because deducible from sets of bidimensional solutions and because not representable since the surface

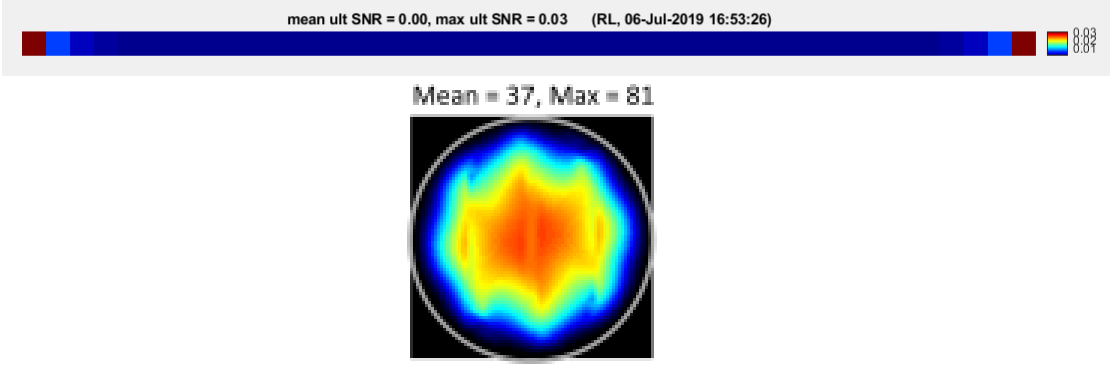

Figure 1.5: UISNR output panels for bidimensional FOV

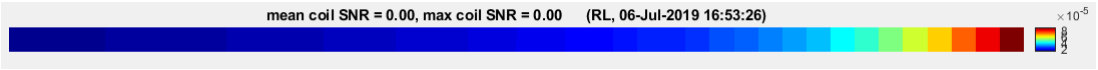

Figure 1.6: Coil SNR output panels for bimensional FOV

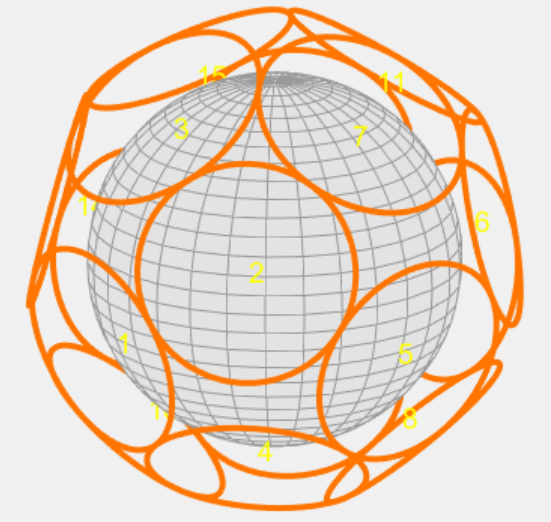

Figure 1.7: Example of a 16 coils uniform distribution around a sphere

voxels would cover those inside.

Not dependent from the FOV are the representation on the surface of the sphere.

Ideal current pattern is show both on a sphere (fig.1.8) or on an unrolled sphere resulting as a plane (fig.1.9). The image show how the ideal current pattern affect the surface of the sphere, is possible to deduce the direction of the current vectors from the arrows and the compared value from the length of the vectors (the effective value of any point is visible selecting it with the cursor).

Not dependent from the FOV are the representation on the surface of the sphere.

In output is sent the configuration of the coils described directly by the input (fig.1.7). It is useful to have idea of how the coils are distributed: since the angles have to be expressed just with couples of numbers is easy to don't have the certainty of the distribution.

Ideal current pattern is show both on a sphere (fig.1.8) or on an unrolled sphere resulting as a plane (fig.1.9). The image show how the ideal current pattern affect the surface of the sphere, is possible to deduce the direction of the current vectors from the arrows and the compared value from the length of the vectors (the effective value of any point is visible selecting it with the cursor).

All of those inputs give a complete hint on the behavior of the EM field on the sample, an example of an inspection of the effect of an MRI scans with different characteristics is shown in fig.1.10.

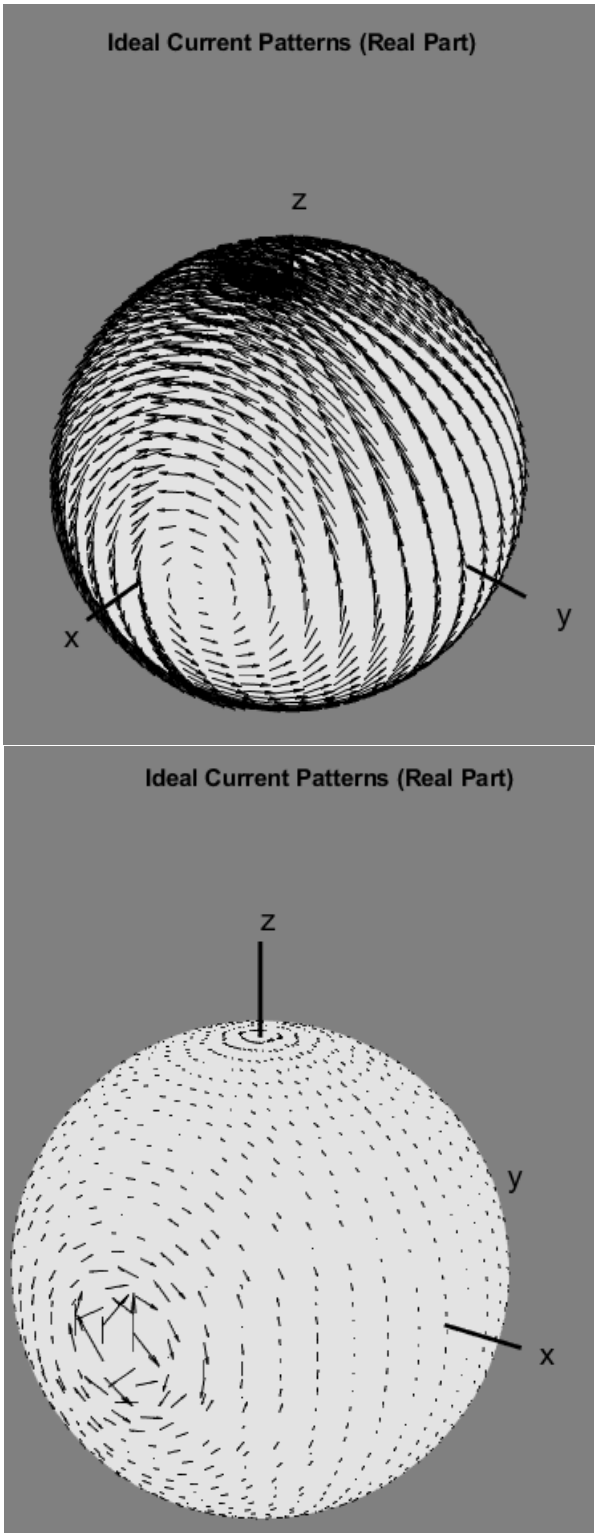

Figure 1.8: Examples of ideal current pattern on the surface of a sphere

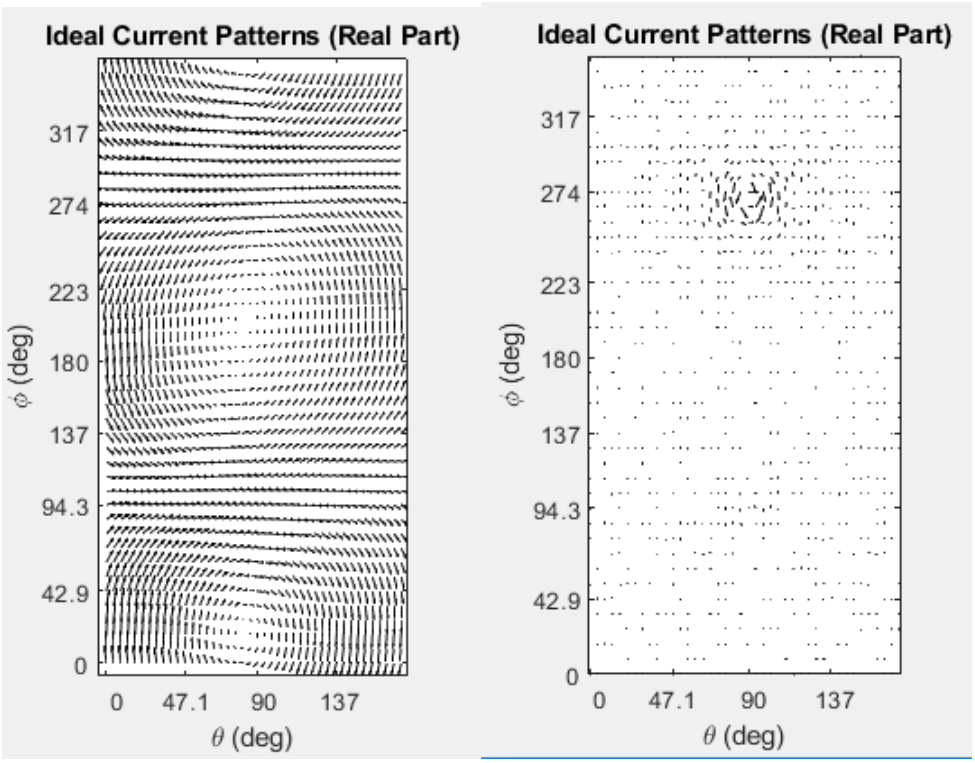

Figure 1.9: Examples of ideal current pattern on the surface of an unrolled sphere

1.6 Applicative contest and algorithms

P1 has been written in Matlab, first of all because Professor Lattanzi had more experience with this language and was also widely used in the radiology department of New York University, but mainly to make easier the validation and the comparison of the results with other existent methods having Matlab versions: CST method [2] and a Keltner study [10]. The real technical reason to use Matlab anyway was the integrated and easy tool to produce graphical objects, like those in the pictures form section 1.5. Obviously other languages might have been valid also to give a useful graphical output, like C++ and GTK, but the choice of Matlab was taken when the work begun (2005) not thinking at the possibility to integrate it in a software application, in that case maybe languages like C or java might have been a better choice, but the attention was just on the math oriented library of Mathlab where are already implemented a lot of function for EM, for example besselh() for the calculation of the Henkel function.

The code is quite big, made of around three thousand lines of code in the main body of the program plus sixteen functions covering some specific calculation, so is impossible to report a detailed description of the program. Anyway the algorithms that implement the theory presented in this chapter will be described highlighting more precisely on some more relevant parts. In order to do this are skipped all the

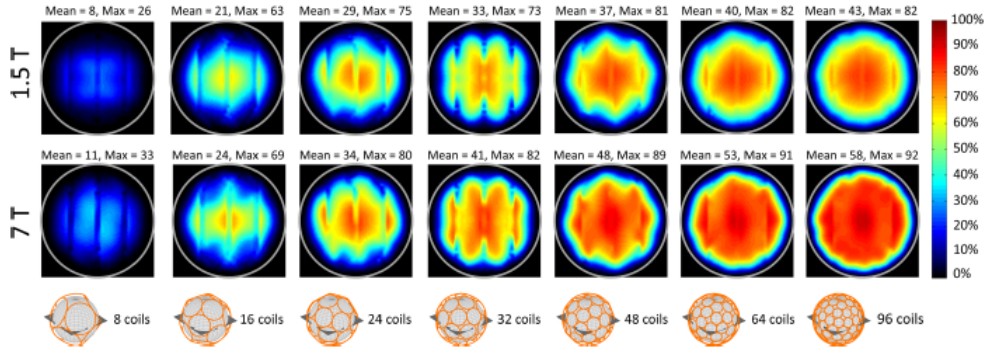

Figure 1.10: Coil performance maps for a transverse plane evaluated at different field strengths and for an increasing number of loop coils closely packed around a spherical sample with 8.4 cm radius, for the case of 4-fold accelerated parallel imaging. Each voxel shows the SNR of the array normalized by the corresponding ultimate intrinsic SNR. Coil noise was included in the calculation of the SNR of the arrays. Mean and maximum values are reported above each map. The gray circle indicates the surface of the sphere.

elementary application of the formulas on the code, leaving what is important for the definition of a path to the targets. All the following algorithms are inserted in a loop that cycles on all the voxels in the FOV.

The first calculation described is the is the calculation of the UISNR.

A section of code which is important highlighting is the calculation of the $B^{f1}_{M,N}$ and $D^{f1}_{M,N}$ coefficients. Starting from the system made by expressions 1.7, 1.8, 1.9, a possible way to find a solution is the analythical calculation of the solution expression. In the specific situation this is possible since the layers are a finite number, N=4[1], so the system can be solved analytically starting from 1.9 where is possible to find $B^{(N-1)1}_{M,N} = -R^{H,V}_{F(N-1)} D^{(N-1)1}_{M,N}$, than sobstituting progressively the coefficients the end of the calculation is an equation with just $B^{11}_{M,N}$ unknown in function of $T^{H,V}_{(P,F)f}$ and $R^{H,V}_{(P,F)f}$.

$$B^{11}_{M,N} = -\frac{\frac{R^{H,V}_{F3}}{T^{H,V}_{P2}T^{H,V}_{P1}} + \frac{R^{H,V}_{F2}}{T^{H,V}_{F2}T^{H,V}_{P1}} + \frac{R^{H,V}_{F1}}{T^{H,V}_{F2}T^{H,V}_{F1}} + \frac{R^{H,V}_{F3}R^{H,V}_{P2}R^{H,V}_{F1}}{T^{H,V}_{P2}T^{H,V}_{P1}}}{\frac{1}{T^{H,V}_{F2}T^{H,V}_{F1}} + \frac{R^{H,V}_{F3}R^{H,V}_{P2}}{T^{H,V}_{P2}T^{H,V}_{F1}} + \frac{R^{H,V}_{F3}R^{H,V}_{P1}}{T^{H,V}_{F2}T^{H,V}_{F1}} + \frac{R^{H,V}_{F2}R^{H,V}_{P1}}{T^{H,V}_{F2}T^{H,V}_{P1}}} \tag{1.32}$$

Having $B^{11}_{M,N}$, all the other coefficients can be derived from the equations 1.81.9. In P1 is present directly expression 1.32, the calculation was done analytically on paper from Professor Lattanzi following [15] directives for a three layer sphere, than all the remaining (non zero) coefficients are derived from expressions 1.81.9. In listing 1.1 is shown the code.

Listing 1.1: DGF coefficents starting expression

[1]N=4 since in the theory is considered a layer also the air surrounding the sphere.

```
B_m_11 = -(R_H_F_3/(T_H_P_2*T_H_P_1)+
          R_H_F_2/(T_H_F_2*T_H_P_1)+
          R_H_F_1/(T_H_F_2*T_H_F_1)+
          (R_H_F_3*R_H_P_2*R_H_F_1)/(T_H_P_2*T_H_F_1))/
          (1/(T_H_F_2*T_H_F_1)+
          (R_H_F_2*R_H_F_2)/(T_H_P_2*T_H_F_1)+
          (R_H_F_3*R_H_P_1)/(T_H_P_2*T_H_P_1)+
          (R_H_F_2*R_H_P_1)/(T_H_F_2*T_H_P_1));

B_n_11 = -(R_V_F_3/(T_V_P_2*T_V_P_1)+
          R_V_F_2/(T_V_F_2*T_V_P_1)+
          R_V_F_1/(T_V_F_2*T_V_F_1)+
          (R_V_F_3*R_V_P_2*R_V_F_1)/(T_V_P_2*T_V_F_1))/
          (1/(T_V_F_2*T_V_F_1)+
          (R_V_F_2*R_V_F_2)/(T_V_P_2*T_V_F_1)+
          (R_V_F_3*R_V_P_1)/(T_V_P_2*T_V_P_1)+
          (R_V_F_2*R_V_P_1)/(T_V_F_2*T_V_P_1));

B_m_21 = (1/T_H_F_1)*(B_m_11 + R_H_F_1);
B_n_21 = (1/T_V_F_1)*(B_n_11 + R_V_F_1);
D_m_21 = (1/T_H_P_1)*(1+ R_H_P_1*B_m_11);
D_n_21 = (1/T_V_P_1)*(1+ R_V_P_1*B_n_11);
B_m_31 = (1/T_H_F_2)*(B_m_21 + R_H_F_2*D_m_21);
B_n_31 = (1/T_V_F_2)*(B_n_21 + R_V_F_2*D_n_21);
D_m_31 = (1/T_H_P_2)*(R_H_P_2*B_m_21 + D_m_21);
D_n_31 = (1/T_V_P_2)*(R_V_P_2*B_n_21 + D_n_21);
D_m_41 = (1/T_H_P_3)*(R_H_P_3*B_m_31 + D_m_31);
D_n_41 = (1/T_V_P_3)*(R_V_P_3*B_n_31 + D_n_31);
```

A construct strictly dependent form the mode expansion is the spherical harmonics $X_{l,m}(\theta, \varphi)$ calculation (listing 1.2), in the code named as Y_l_m for confusing variables name problems. Even if this calculation is quite simple, it is also quite important since is repeated in all the four calculation the program perform.

Listing 1.2: Spherical harmonics calculation

```
if m > 0
Y_l_m = ((-1)^m)*legendrenorm*(1/sqrt(2))*
        legendrefunctions((m+1),:).*exp(1i*m*phiset);
elseif m == 0
Y_l_m = legendrenorm*legendrefunctions(1,:);
else
```

```
Y_l_m = ((-1)^abs(m))*conj(((-1)^abs(m))*
        legendrenorm*(1/sqrt(2))*
        legendrefunctions((abs(m)+1),:).
        *exp(1i*abs(m)*phiset));
end
```

From expression 1.20 we can notice the two important matrix: the sensitivity matrix S from expression 1.16 and the Ψ_{mode} is the modes' noise covariance matrix from expression 1.18.

The sensitivity matrix contains the complex signal sensitivities associated with each mode at the target position r_0 and at all aliased positions and is derived by applying the principle of reciprocity [6]. The code is shown in listing 1.3.

Listing 1.3: Sensitivity matrix calculation

```
for irho = 1:length(rset)

if rset(irho) < radius3
S_M(irho) = mag_scaling_r4*D_m_41*(N_x(irho) - 1i*N_y(irho));
S_E(irho) = mag_scaling_r4*D_n_41*(M_x(irho) - 1i*M_y(irho));

elseif (rset(irho) < radius2) && (rset(irho) >= radius3)
S_M(irho) = mag_scaling_r3*(B_m_31*(N3_x(irho) - 1i*N3_y(irho))
        + D_m_31*(N_x(irho) - 1i*N_y(irho)));
S_E(irho) = mag_scaling_r3*(B_n_31*(M3_x(irho) - 1i*M3_y(irho))
        + D_n_31*(M_x(irho) - 1i*M_y(irho)));

elseif (rset(irho) < radius1) && (rset(irho) >= radius2)
S_M(irho) = mag_scaling_r2*(B_m_21*(N3_x(irho) - 1i*N3_y(irho))
        + D_m_21*(N_x(irho) - 1i*N_y(irho)));
S_E(irho) = mag_scaling_r2*(B_n_21*(M3_x(irho) - 1i*M3_y(irho))
+ D_n_21*(M_x(irho) - 1i*M_y(irho)));

else
S_M(irho) = 0;
S_E(irho) = 0;
end %end if
end %end for
smallr = rset < eps;
S_E(smallr) = 0;

if l == 1,
```

```
switch m
case -1
S_M(smallr) = -1i*mu*k_4*k_0*currentradius*
        currentradius*(1/3)*sqrt(3/(2*pi));

case 0
S_M(smallr) = 0;

case 1
S_M(smallr) = 1i*mu*k_4*k_0*currentradius
        *currentradius*(1/3)*sqrt(3/(2*pi));
end % end switch
else
S_M(smallr) = 0;
end % end if

S = [S_M ; S_E];
```

The modes' noise covariance matrix Ψ_{mode} in expression 1.15 is calculated by integrating electric field products over the volume of the sphere, it is the most memory end time consuming calculation. In listing 1.4 are showed also the functions "computePsiM_allregions(myL, ko, r1, r2, type)", that computes the value of the elements of Psi M using Gaussian quadrature to solve the integrals, and the function "spherbessJ(nu, z)", that computes the spherical Bessel function. Also other routine functions are involved in the calculation of the modes' noise covariance matrix Ψ_{mode}, anyway all are quite similar to the two presented in listing 1.4, the difference is just the field of work or wich bessel funcition the calculate.

Listing 1.4: Modes' noise covariance matrix calculation

```
term1 = (abs(B_m_21)^2)
*computePsiM_allregions(l, k_2, radius2 ,radius1 , 2)
+ (abs(D_m_21)^2)
*computePsiM_allregions(l, k_2, radius2 ,radius1 , 1)
+ (B_m_21*conj(D_m_21))
*computePsiM_allregions(l, k_2, radius2 ,radius1 , 4)
+ (D_m_21*conj(B_m_21))
*computePsiM_allregions(l, k_2, radius2 ,radius1 , 4);

term2 = (abs(B_n_21)^2)
*computePsiE_allregions(l, k_2, radius2 ,radius1 , 2)
+(abs(D_n_21)^2)
```

```
*computePsiE_allregions(l, k_2, radius2 ,radius1, 1)
+(B_n_21*conj(D_n_21))
*computePsiE_allregions(l, k_2, radius2 ,radius1, 4)
+(D_n_21*conj(B_n_21))
*computePsiE_allregions(l, k_2, radius2 ,radius1, 4);

P_L_r2 = (sigma_r2/2)*P_L_scaling*
[ term1              0
0              term2];

P_L = P_L_r4 + P_L_r3 + P_L_r2;
% is shown the calculation for one layer,
%the other two are the same

P = T_hat*P_L*T_hat';
% end of main program extract


function PsiMvalue = computePsiM_allregions
        (myL, ko, r1, r2, type)
[...]
x_i=((r2 - r1)/2)*st_i + (r2 + r1)/2;
switch type
case 1
sb_0 = spherbessJ(myL,ko*x_i);
PsiMvalue = ((r2 - r1)/2)* (sum(ge_i .
        * (x_i.^2 .* abs(sb_0).^2)));

case 2
sb_0 = spherbessH(myL,ko*x_i);
PsiMvalue = ((r2 - r1)/2)* (sum(ge_i .
        * (x_i.^2 .* abs(sb_0).^2)));

case 3
sb_0_1 = spherbessJ(myL,ko*x_i);
sb_0_2 = conj(spherbessH(myL,ko*x_i));
PsiMvalue = ((r2 - r1)/2)* (sum(ge_i .
        * (x_i.^2 .*(sb_0_1.*sb_0_2))));

case 4          sb_0_1 = conj(spherbessJ(myL,ko*x_i));
```

```
sb_0_2 = spherbessH(myL,ko*x_i);
PsiMvalue = ((r2 - r1)/2)* (sum(ge_i .* (x_i.^2 .
        *(sb_0_1.*sb_0_2))));
end
% end of computePsiM_allregions


function spherbesselj = spherbessJ(nu, z)
threshold = 0.1;
i1_ind = find(abs(z) >= threshold);
i2_ind = find(abs(z) < threshold);
if length(i1_ind) > 0 % standard expression
z1 = z(i1_ind);
y1 = zeros(length(i1_ind),1);
y1 = sqrt(pi/2) .* sqrt(1./z1) .* besselj((nu + 1/2), z1);
spherbesselj(i1_ind)=y1;
end
if length(i2_ind) > 0 % series expansion
z2 = z(i2_ind);
y2 = zeros(length(i2_ind),1);
tmp_1 = gamma(3/2 + nu);
tmp_2 = sqrt(pi);
y2 = (z2.^(1/2 + nu)).*( (2.^(-1-nu)).
        *tmp_2./(tmp_1.*sqrt(z2)) -
        (2.^(-3-nu)).*tmp_2.*(z2.^(3/2)).
        /(tmp_1.*(3/2 + nu)) +
        (2.^(-6-nu)).*tmp_2.*(z2.^(7/2)).
        /(tmp_1.*(3/2 + nu).*(5/2+nu)) -
        (2.^(-8-nu)).*tmp_2.*(z2.^(11/2)).
        /(tmp_1.*(3/2 + nu).
        *(5/2 + nu).*(7/2 + nu)) );
spherbesselj(i2_ind)=y2;
end
```

With those fundamental algorithms is possible to calculate the UISNR inserting them in a double cycle for that iterate on the mode expansion coefficients l and m implementing the summations.

The calculation of the coil SNR is quite simple compared to the procedure to find the UISNR.

There are two nested for cycle which iterate both on the number of coils (i and j): the giver and the receiver of the current. For every coil (for the i coil inside the first

cycle and to the j coil inside the second) is performed the mode expansion cycling this time first on l and nested in it on m. The mode expansion produce the matrix needed in expression 1.20 for the calculation of the SNR. In listing 1.5 the example of mode expansion in one of the coil and the calculation of the coil SNR.

Listing 1.5: Extract of coil SNR calculation

```
%Description begins inside a i coil loop

for jcoil = icoil+1:ncoils,
theta_coil_j = rot_coil(jcoil,1);
phi_coil_j = rot_coil(jcoil,2);
costheta_coil_j = cos(theta_coil_j);
W_coil_j = zeros([((lmax + 1)^2 - 1) 1]);
counter_coil_j = 1;

for l = 1:lmax
rot_coil_norm_j = sqrt(4*pi/(2*l + 1));
legendrenorm_j = sqrt((2*l + 1)/(4*pi));
legendrefunctions_j =
        legendre(l,costheta_coil_j,'sch');
legendrefunctions_lminus1_j =
        legendre(l-1,costheta_coil_j,'sch');
legendrefunctions_lminus1_j =
        [legendrefunctions_lminus1_j;
        zeros(size(costheta_coil_j))];

for m = -l:l,

Listing 1.2;

W_coil_j(counter_coil_j) = ((-1)^m)*rot_coil_norm_j
        *conj(Y_l_m_j)*W_coil_z(l);
counter_coil_j = counter_coil_j + 1;
end % end m loop
end % end l loop

W_coil_j_bis = zeros(size(W_coil_j));

for ll = 1:lmax
startindex = (ll^2 - 1) + 1;
```

```
endindex = ((ll+1)^2 - 1);

for mm = -ll:ll
if mm < 0
index2 = endindex - (ll-abs(mm));
oldidx1 = startindex + abs(abs(mm)-ll);
W_coil_j_bis(index2) = W_coil_j(oldidx1);

elseif mm > 0
index1 = startindex + abs(mm-ll);
oldidx2 = endindex - (ll-abs(mm))
W_coil_j_bis(index1) = W_coil_j(oldidx2);

elseif mm == 0
index1 = startindex + ll;
W_coil_j_bis(index1) = W_coil_j(index1);
end
end
end

W_coil_j = W_coil_j_bis;
W_coil_j = W_coil_j(:);

psi_coil(icoil,jcoil) = W_coil_i.'*(P_modes.*conj(W_coil_j));
psi_coil(jcoil,icoil) = conj(psi_coil(icoil,jcoil));
end % end jcoil loop
end % end icoil loop
```

1.7 Algorithm Limitations

Results obtained from P1 are quite interesting from a scientific point of view, showing how DGFs is a quite valid alternative to traditional methods [12,14,22].

The success obtained in radiology departments of Universities and hospitals anyway raised a problem of usability. Since the program was thought just on research purpose, P1 has not an user friendly interface and the input data, even if grouped approximately at the beginning of the program, are confused in the preparation of the calculation. This chaos makes the tool hard to use since an immediate understanding is not possible for a user, consequentially the situation produce an awkward chain of requests to as help in the usage and understanding of the tool. Is also absent a clear structure of the code, making the comprehension of the code even harder.

P1 is also not optimized, there are a several redundant code parts which would have been better replace with cycles or functions, especially for what concerns the layers. More in details, all the calculation dependent from layers radii shown in sections 1.2 and 1.3, even if them are performed exactly in the same way for each radius or each layer, are replicated just changing the radius to consider.

Another sub optimal calculation example is the calculation of the coefficients $B^{f1}_{M,N}$ and $D^{f1}_{M,N}$ starting from expression 1.32 and reported (split for the TE and TM case) in listing 1.1. Even if the supposition at the head of the code are concerning a three layers sphere, the theory on the algorithm is based on is general and applicable for any number of layers. Writing expression 1.32 in this way is not an optimal solution since it cause a lack of generality in the code, limiting the possible simulations (true purpose of P1).

The high request of usage of the tool anyway, pushed Professor Lattanzi to start the development of a web application easy to use and also to elaborate P1 and other programs concerning the MR simulation thanks to DGFs in order to have a system easier to maintain, intuitive to read, more general and more performing. In the next chapter are described the modifications brought to P1 to make it more performing and clean.

Chapter 2

Proposed solution

2.1 Structure

Since, as written in section 1.7, P1 has not a really logical structure, with input and preparation to calculation merged and a lot of redundant code lines, the code and all its parts were rearranged giving it a logical division as much as possible. Fig.2.1 shows a complete graphical overview of the structure.

The idea behind the redesign of the structure is making possible and easy a black box approach, ignoring completely the logic behind calculation part and avoiding if desired the contact with the core of the code. This is a really user oriented choice: the aim of the program is producing simulations in order to find the best coil configuration and excitation to have the best SNR in the desired point. According to the purpose of the system, are identified as users researchers in the field of electromagnetism or medicine and MRI, medical doctors, coil design engineers or technicians and software engineers (to integrate the program in larger systems). For users which are quite inside the theory showed in chapter 1 (which are not the totality of the identified users) the input approach is not so different between P1 and P2, for everyone else the understanding of the program might be quite challenging. Since in P1 the input data are not completely separated from the preparation of the code and often not even indicated clearly, an inexpert user will have difficulties for sure even for searching and understanding which are inputs and what are just variables expressed in the code. This because the black box approach is possible but in a "shaded"[1] way.

In P2, to produce a complete black box approach, the input is completely detached from the rest of the code, so an inexpert user can face just the data the code need from outside with no possibility of confusion or misunderstanding. The starting function takes as input a Json with all the useful data and simply extract

[1] With shaded I mean that the a complete black box approach is not possible at the beginning but progressively some parts of the code can be ignored focusing just on the input

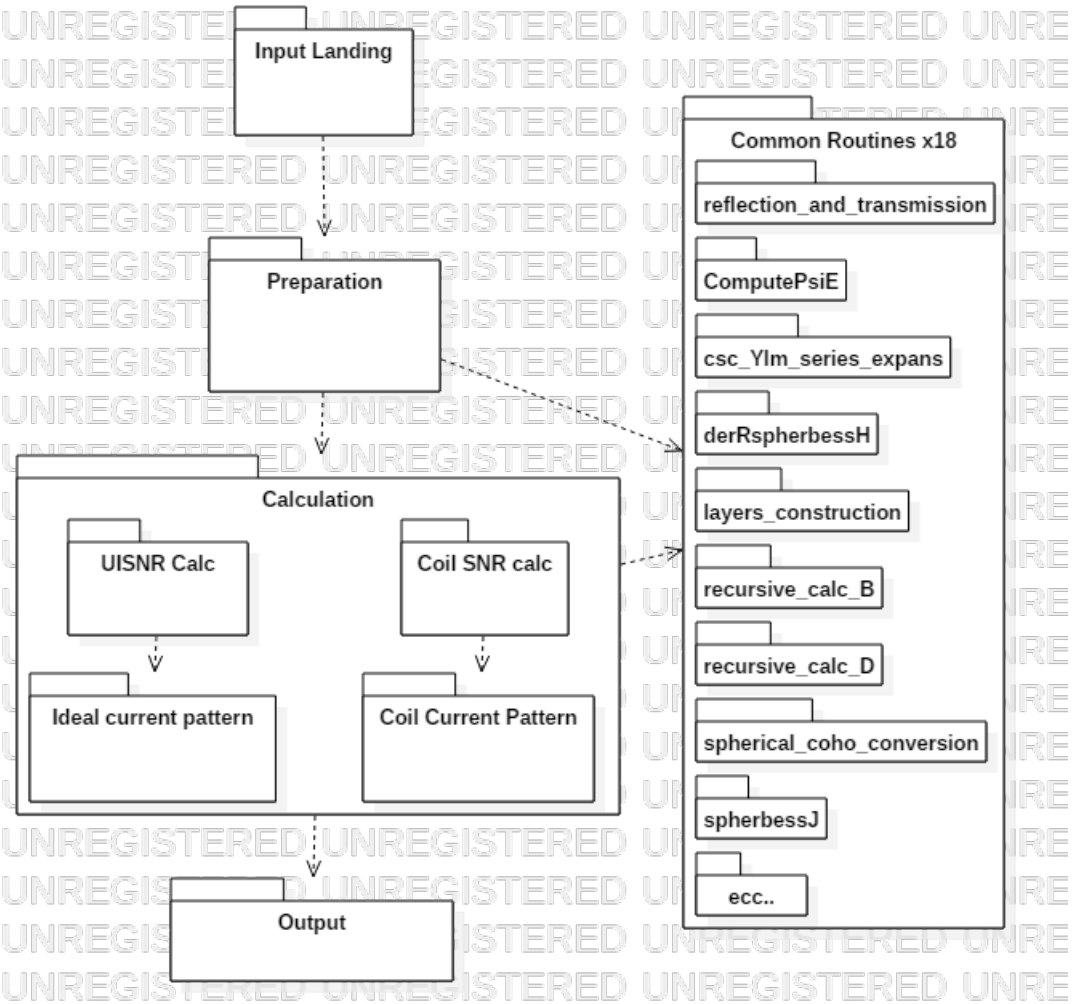

Figure 2.1: P2 structure with a distinct separation between input, preparation, calculation, output and the eighteen coroutines. This structure is thinked to allow the general user just to have a relation with the input and the output and interact with the rest of the code as a black box.

those data from it in a set of variables. This short solution require, from an understanding point of view, to know what is an input of an MRI machine since is exactly what is simulated by the input, so every medic or coil design technician can use it. From a software engineer point of view instead what is immediate to understand is the structure of the required Json in order to design the output of a communicating program.

The preparation to calculation in P1 is split. There is a part mixed with the inputs which is dedicated to the calculation of the physical quantities directly derivable from the input. The other part is related to the theory, with the necessary transformation to perform the calculation and the creation of the vectors and matrix to contain what will be calculated after. In P2 those two sections are grouped in one which manage all the preparation. Since the input is isolated, the first part of the separation become a block to small to be meaningful both from an actual meaning point of view and from the amount of relevant operations, so it is merged with the second part.

The calculation in P1 is quite chaotic, mainly for the several code repetitions and the absence of a real order without a clear separation between the four different calculations (UISNR, coil SNR and the respective current patterns). In P2, the structure of the two upper sections allows a clear separation of the calculations, using the variables previously created in the preparation section. This approach make the code easy to read since is immediate for an expert of the subject understand what the program is doing in the considered part. Also from a debugging or a future development point of view, having the four calculations separate is interesting, since an error (which might be a malfunctioning not noticed in the code or simply a set of input which produce suspicious results) in the output can be mapped exactly in a specific part of the code, excluding all other possibilities.

The last two sections are the only two where a nice work of logical separation and structuring is given also in P1, so it is not deeply transformed in P2.

The output is performed at the end of calculation. P1 is a program with research purpose and results are obviously the most important part, considering a correct calculation reproduction. According to this, the section is very well done, from any point of view: any final user identified at the beginning of the section is always able to understand what are the results, according to his needs. In P2 the purpose is the same even if there is the integration in a platform that change slightly the situation. Anyway, even if in an hypothetical platform all the matlab plot would be created separately from the program, was made the choiche to leave this part approximately the same, with some minor change depending from some different calculation approach.

The section named "common routines" comprehend all small functions used several times in the program. In P1 the purpose of creating a routine function is to

separate a specific part of the calculation, maybe related to a specific theory in a paper, from the main curse of the calculation and also the alternative code repetition, even if some other parts are repeated anyway. In P2, the logic is maintained and are also maintained all the common routines form P1. An inspection of the code anyway revealed a lot of repetitions, several of them replaceable with a function other with cycles. All the functions that was possible to extract from the code were extracted, making the code more clean. Some other common routines were generated from a different approach to some problems in the calculation. At the end of the restructuring the set of common routines of P2 is slightly bigger than the set of P1. The only part of P2 which not interact with any common routine is the input acquisition, this according to the complete black box approach.

On top of this redesign, results that P1 has a lot of lines unused and some sections not working, oddments of a research oriented usage of the code. A light cleaning of the code is performed, leaving the parts needed for the calculation and erasing what is not useful even from a future development point of view.

2.2 Layers management and generalization

As described in sections 1.2 and 1.3, all the calculation of the DGF and after the UISNR is dependent from the layers characteristics (radius, dielectric constant and conductivity). In P1, all layers are described by three variables (radius, dielectric constant and conductivity) and are considered as totally separated, so the situation is having three of all the descriptive quantities.

Listing 2.1: Layers characteristics in P1

```
%% % ** INNER MOST LAYER - REGION 4 ** % %%
radius3_set = [0.1];
nradin = length(radius3_set);
epsilon_r4 = 60;
sigma_r4 = 0.45;

%% % **  REGION 3 (radius3 < radius < radius2) ** % %%
radius2_set = radius3_set + 0.005;
epsilon_r3 = 32;
sigma_r3 = 0.1;

%% % **  REGION 2 (radius2 < radius < radius1) ** % %%
radius1_set = radius2_set + 0.002;
epsilon_r2 = 1;
sigma_r2 = 0.1;
```

```
%% % ** OUTER MOST LAYER - REGION 1 (always AIR) ** % %%
currentradius_set = [0.14];
% [m] radius where current density is defined
% (outermost radius)

outerfov_radius_set = 0.16;
% how large the FOV for fields calculations
```

Since the only relation that relate the three layers is the definition of the radius (relation useless for theory purpose), without additional constructs like arrays, or matlab structure, or classes, ecc... is not possible to avoid the repetition of the code for each layer. Those constructs are indeed missing in P1, this produce a strictly linear and inefficient calculation on this topic with situation, like the code below for the calculation of the complex wave numbers in layers k, with long section of code doing repetitively the same thing:

Listing 2.2: Example of redundant lines in P1, calculation of the complex wave number in each layer

```
k_0_squared = omega*omega*epsilon_0*mu;
k_0 = sqrt(k_0_squared);

k_4_squared = omega*mu*(omega*epsilon_r4+1i*sigma_r4);
k_4 = sqrt(k_4_squared);

k_3_squared = omega*mu*(omega*epsilon_r3+1i*sigma_r3);
k_3 = sqrt(k_3_squared);

k_2_squared = omega*mu*(omega*epsilon_r2+1i*sigma_r2);
k_2 = sqrt(k_2_squared);
```

In P2 the approach chosen is completely different. Instead asking all the characteristics of a layer separately, the input Json has to contain an array with the radii, another one with the dielectric constant and a last one with the conductivity for each layer. The radii vector contains always n+2 values, since n is the number of layers, one is the radius where is defined the current density and one is the dimension of the FOV. With those three information is built a class, completely absent in P1. The class Sphere (fig. 2.2) is a class that contain information about the depth of the layer, its lower bound as in_radius_set, the upper bound as out_radius_set and the electrical properties epsilon (dielectric constant) and sigma (conductivity). The dept is added to have a quick access to the position of the respective layer in the sphere, the inner most layer has dept equal to n (the number of layers) and the

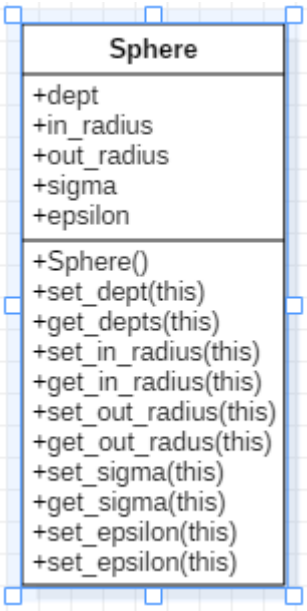

Figure 2.2: Class Sphere of P2

outer most has dept 1. For each layer are indicated two radii in order to give a more realistic imagine of the sample and to make the management of the various layers in the calculation easier, indeed a lot of operations require both the upper bound and the lower bound so having this information in just one object is better, even if the information is deductible from the previous layer. The class has not so many methods, limited to a constructor and set and get methods for each attribute. This is also reasonable since the body in fact in the simulation is just a passive object with no active interactions.

Listing 2.3: Funcion layers_construction(radii,epsilons,sigmas) for the creation of an array of objects Sphere

```
function [layers_array, n_layers] =
        layers_construction(radii,epsilons,sigmas)
for d=1: (length(radii)-1)
        obj=Sphere();
        obj.set_dept(length(radii) - d - 1);
        %outer most radius
        if d == (length(radii)-1)
                obj.in_radius = radii(d);
                obj.out_radius = radii(d + 1);
        else
                %generic radius
                obj.out_radius = radii(d);
                obj.epsilon = epsilons(d);
```

```
                obj.sigma = sigmas(d);
                if d == 1
                        %innermost radius
                        obj.in_radius = 0;
                else
                        obj.in_radius = radii(d-1);
                end
        end
        layers_array(d) = obj;
end
n_layers=d;
```

As shown in listing 2.3, a first for cycle runt on the length of the radii array generate at each iteration a Sphere object with the proper characteristics and store it in the layers_array(d)[2], at the end of the function is defined also n_layers, a shortcut to the dimension of layers_array. Note that also the one named "Region-1" in P1 is also defined as a Sphere object with 0 by default electric characteristics never used in the code. An extension class was considered during the design but was evaluated useless since not very meaningful.

In P1, thanks to this simple but efficient shrewdness, all the code dependent from the layers is managed by a for cycle that iterates on a variable (usually named i_lay) until n_layers, or n_layers-1 if the "air" layer has not to be considered. In this way the redundant code widely present in P1 is completely vanished in P2, giving a more systematic approach to the problem. In listing 2.4 is shown the P2 respective part of listing 2.2:

Listing 2.4: Example of cyclic version of listing 2.2, calculation of the complex wave number in each layer

```
k=zeros(1,n_layers);
for i_lay=1:n_layers
        if i_lay == n_layers
                kt=omega*omega*epsilon_0*mu;
                k(i_lay)=sqrt(kt);
                k_0=k(i_lay);
        else
                kt =omega*mu*
                        (omega*layers_array(i_lay).epsilon+
                        1i*layers_array(i_lay).sigma);
                k(i_lay)=sqrt(kt);
```

[2]This array is the only one not having a fixed size but dynamically built, this because not knowing the size of the array radii is impossible to dimensionate it.

```
        end
end
```

Lets add some observations.

To save memory, Matlab asks to define the dimension of every array. This is not necessary if every single variable defined dependent from a radius, epsilon or sigma is not shared in a structure or not need to be related to other similar variables dependent from different quantities. Anyway defining an array of variables instead just the variables does not involve a big difference in the memory usage, since the number of variables defined does not change between P1 and P2.

The management with layers_array spread itself "naturally" to each quantity depending on layers characteristics, indeed happen to be quite comfortable pivoting on n_layers and an array to store the results of calculation like listing 2.4. All those quantities are managed in P1 with code repetitions totally absent in P2.

Last observation is concerning the generality. P1 is limited to the study of a three layers sphere, this is done on purpose since three layers are considered enough to have a meaningful simulation of the human head. Taking as input inside the Json three arrays and not just eleven variable give to P2 a complete generality on this topic. With the only precondition that radii, epsilons and sigmas are consistent between them, according to their unbounded dimensions is possible to simulate a sphere with any number of layer. The only limit has become the memory of the machine.

2.3 DGF coefficients calculation in the rielaborated code

In P1 the solution of the DGF coefficients is performed solving analytically the linear system reported in expressions 1.7, 1.8 and 1.9, the solution produce expression 1.32 and the Matlab transposition in listing 1.1. Listing 1.1 can be easily adapted to use as variables for transmission and refection, which are dependent from layers radii and EM characteristics, those stored in some vectors with a simple indicization. Anyway this solution is still valid only for a three layer sphere, opposing to the research of generality wanted for to P2.

The first attempt to get a completely general version of the DGF coefficients solution was the run time construction and the solution of a system. Defining an array of the variables A 1xn, an array of the solution coefficients B nx1 and a matrix of the known coefficients C nxn, the system can be solved by A=B/C[3]. This specific procedure is valid only if C is squared. In P2 A would have been an array with all the DGF coefficients (unknown), B would have been an array with all 0 and a 1 as first element deriving from the first expression in 1.7, C a square matrix mostly covered by 0 with the reflection and transmission variables placed according to the

[3]This writing way is admissible just in Matlab, in algebra of matrix is not valid the commutative properti so the correct writing is: $A = C^{-1}B$

system made by 1.7, 1.8 and 1.9. All the layers could be constructed iterating the radiation and reflection coefficients arrays. This solution was totally independent from the number of layers and theoretically correct, unfortunately it is mandatory dependent from the invertibility of C, which can not be guaranteed in P2.

The second and definitive attempt is a recursive solution (listing 2.5). In the system of expressions 1.7, 1.8 and 1.9, expressions 1.8 are particular since are completely general and valid for all n that goes from 2 to N-1. With a top-down approach starting from the first expression of 1.9, was analyzed the behavior of the analytical solution by substitution noticing some recursive behavior exactly on the expressions corresponding to expressions 1.8. The recursion starts form the outer most layer begins to go deep finding the base cases which is in the innermost layer. Digging in the sphere, the recursion split the path at every step, following the recursions generated from the coefficient B and the recursions generated from D. Every recursion , both B and D, return two coefficients, named as in listing 2.5 coeffb and coeffk. Coeffb is the coefficient of B terms in the system, coeffk is the coefficient of D terms. The base case of B recursion takes its coeffb and coeffk from the first expression of 1.9. The base case of D recursion takes its coeffb and coeffk from the second expression of 1.9 instead. The rest of the bottom up recursion part is based on expressions 1.8, every step coeffb is the sum of coefficients of the first expression and coeffk is the sum of coefficients of the second expression, what is different from the base case is the presence of unknown B and D unknown variables. Those variables are the results of the precedent step of the recursion and can be substituted as follow: the B coefficient in the first equation is the coeffb of a B recursion, the D coefficient of the first equation is the coeffb of a D recursion, the B coefficient in the second equation is the coeffk of a B recursion, the D coefficient of the second equation is the coeffk of a D recursion. The step before the end is slightly different since based on the first equation of expression 1.7. At the end of the recursion the result is two coefficients, $B^{11}_{M,N}$ now can be computed as $B^{11}_{M,N} = -\frac{coeffk}{coeffb}$.

Explaining a recursive algorithm is quite difficult, I add the complete code in listing 2.5.

Listing 2.5: Recursive solution to find DGF coefficients.

```
B_m(1)= -coeffk/coeffb;

function [coeffb,coeffk] = recursive_calc_B(tf,tp,rf,rp,d)
n=length(tf);
if n==1
        disp('ERROR: not enough layers');
end
if d == n
```

```
        coeffb=1/tf(n);
        coeffk=rf(n)/tf(n);
else
        [bb,kb] = recursive_calc_B(tf,tp,rf,rp,d+1);
        [bd,kd] = recursive_calc_D(tf,tp,rf,rp,d+1);
        if d==1
                coeffb = bb + (rf(1)*bd);
                coeffk = kb + (rf(1)*kd);
        else
                coeffb = (bb/tf(d)) + ((rf(d)*bd)/tf(d));
                coeffk = (kb/tf(d)) + ((rf(d)*kd)/tf(d));
        end
end

function [coeffb,coeffk] = recursive_calc_D(tf,tp,rf,rp,d)
n=length(tf);
if n==1
        disp('ERROR: not enough layers');
end
if d == n
        coeffb=rp(n)/tp(n);
        coeffk=1/tp(n);
else
        [bb,kb] = recursive_calc_B(tf,tp,rf,rp,d+1);
        [bd,kd] = recursive_calc_D(tf,tp,rf,rp,d+1);
    coeffb = ((rp(d)*bb)/tp(d)) + (bd/tp(d));
        coeffk = ((rp(d)*kb)/tp(d)) + (kd/tp(d));
end
```

The recursive solution is not dependent from the length of the input, completely general, is totally generated and solved run time, theoretically correct and always defined, so it has been inserted in P2 to manage spheres of any number of layers.

2.4 Other choices and considerations

The previous subsections talks about just layers dependent calculations. Those calculation affect mainly the half of the code concerning the UISNR. The other half of the code is dependent from the coils number and characteristic. This section, contrarily to the one on UISNR, is totally general and can work for any number of coils and any rotation. Since also the code is not affect of bad repetitions, this section is not modified since is already perfectly working in P1. In P2 there are just

some adjustments to the structure and replication of the logic because was the most reasonable thing to do.

Same reasoning was done for the output. As told in section 2.1, it is the most important part of the program and in P1 it was already well done.

Was made the choice to leave P2 in Matlab. Even if the integration in a web application would prefer different frameworks, the Mathworks library offer some prebuilt functions which are extremely convenient for the purpose of the code. Since some parts of the structure and all the co-routines functions are also taken unchanged, maintaining the code in Matlab allows to reuse all useful parts of the original code. The last reason to use Matlab is for have the same environment to make comparisons. Having the possibility to save the workspace as math files speeds up incredibly the operation of comparisons of results. Also the integrated system of plotting is very useful, and having also the same style in the presentation of results help to understand the differences. The downside anyway is the small difficult Matlab has reading Jsons: Matlab read vectors in Jsons with the dimension swapped due to a different style of writing, so is necessary to transpose the vector once it is read in Matlab. This problem does not happen for matrix.

2.5 Validation

The validation of P2 was done by comparing the results with some configurations of the input with the result of P2 run with on the same configuration. Since P1 was validated comparing it with other studies [2, 10], it can be consider a term of comparison for validation, therefore all the experiments will involve P2 and P1. The experiments to validate the code are quite simple. Since what has changed in P2 from P1 is the part concerning the UISNR, the tests will take in account this information. The configuration of all the most meaningful tests is very basic: just one coil of radius 2,5 cm placed at 1 cm from the surface of the sphere, sphere with same radius and electromagnetic properties for both the codes, same choice of FOV for both the algorithms, the rest of the configuration is chosen as basic as possible. No significant differences where emerged during the validation as shown in fig. 2.3.

The experiment was repeated with several possible combination of data keeping fixed the coil first, than it was also performed with random but coherent values for the configuration of the input also for the coils. Layers under the fourth were performed in P1 defining needed adjacent layers with the same EM properties. Still no meaningful differences. Notice that the two results were not exactly the same, but since the difference between the two was always under a magnitude of 10^{-16}it can be ignored, this will be covered better in section 3.4.

Another test done was simulating a sphere with several layers with certain EM properties and a sphere with just one layer with the same EM properties of the multi

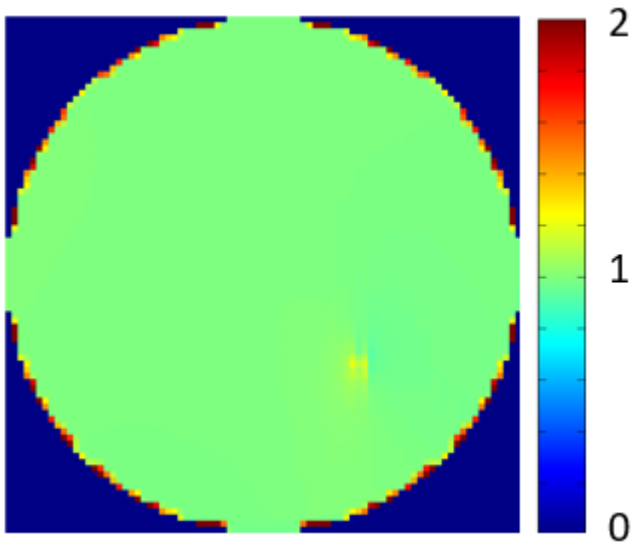

Figure 2.3: Ratio of P1 and P2 SNR, the green color correspond to value 1, which means that the two code give the same result.

layer sphere. This is a behavioral test so I checked the two configurations just on P2. Also in this situations, the two configurations produced an equivalent result, confirming an appropriate behavior of the simulation. Also this test was performed with some random settings, anyway the result was always the same.

The last test is also behavioral. I tested, with the same configurations of the first test, that the UISNR value is a coil SNR upper bound for each voxel in the FOV according to the theory. Also this time all the result were coherent with the theory, so P1 is able to simulate UISNR as ideal value and the coil SNR as effective value of the configuration.

The results of the tests are equal according to the prevision, this meas that the result is not depending from the number of layers but just from the configuration. Since P2 has passed successfully all the tests useful for the purpose, it can be considered validated an therefore it can be used to "produce" acceptable results.

2.6 New solution

P1 is designed to solve, at least partially, limitations of P2 also thinking on the purpose of the work which is the integration in a application.

P2 is "user friendly" as much as possible, allowing a full black box approach. According to the use of the code, an user can have a targeted access to the code without entering in contact with undesired part of it. The design hypothesizes also the use for different type of users according to their knowledge of electromagnetism and MRI: an user who want to just use the code can have access just on the portions of input and output, an user which can understand the code and want to investigate

it can have a targeted access to the code without need to understand the separation of the various parts. The design produced a structure (fig 2.1) well defined with a clear separation of the components of the calculation.

P2 is more optimized respect to P1. Is obviously impossible to say that the code is fully optimized, but at least several problems of optimization are solved analyzing which parts are repeated an which are useless or not working in P1. The not working paths are reasonably removed. Is left what might be useful for future developments of the code and is in an advanced state on that purpose. The management of the layers as three separated and unrelated values is completely replaced by an array approach to the problem. All the code repeated dependent from the radii or the EM properties of layers is replaced with a cycle on the length of the sphere objects and perform the considered lines of code.

P2 is completely general. In P1, even if is possible a simulation of any number of coil, the simulation of the scanned body is limited to four layers even if the theory consider a general case. Replacing listing 1.1 with listing 2.5 allow a complete generalization of the code. P2 is able to simulate every kind of MRI sphere scan, with any number of coils and any number of precision in the simulation of the human head. This is a big upside because allow to manage all possible situation with just one program. The benefit of the generalization, even if create an unlimited freedom to the setting simulation, is limited by the bond of reality, since obviously there are limitation deriving from what exist in the real world. A real human head might have around ten different layers with different tissues, coils are bounded to a sphere and even if is possible to apply a lot of them, far or close one from each other or even one over another, there is a limit also to them. Anyway having the opportunity of a completely free simulation give potential to the code, even if it might be meaningless under certain conditions.

The only controversial choice is maintaining Matlab as framework but, as presented in section 2.4, it have an amazing library and integrated graphic tool for the purpose of the code. Obviously another choice would have been better for an integration in other systems, but this opportunity is still possible, so no limitations rise from this choice.

Chapter 3

Performance Analysis

3.1 Analysis introduction

Even if P2 is more structured, optimized and general, it does not mean that P1 is necessarily worst. To consider one of the two codes better than the other is necessary to make an analysis of the performance, comparing the two algorithm on the memory usage, execution time and result precision. An answer to those questions is immediate if we consider just the versions described of P1 and P2. P2 indeed can manage also less than four layers, so the code in those cases cycle less times, using less registers to perform operations, terminating faster and with more precision due to a less number of operation performed which bring an approximation each. Under those condition P1 and P2 are not properly comparable, since technically they do a different thing, even if the purpose is always the simulations for coil design.

An objection can be raised to P2 saying that, still maintaining the analytical approach to the solution, a version of P1 (P1') with an ad hoc solution for each number of layers, may have the same performances of P2. In order to face this problem is chosen to ampliate P1, creating different versions which compute the SNR for a specific number of layer each, and perform some testing on memory, execution time and precision between P1 and each of them. Every version need some adaptation, essentially the same for each. Are added to the code the new layers features (radius, dielectric constant and conductivity) and all the lines of code depending, directly or not directly, from the layers has to be repeated again until all added layers are covered. Has also to be modified the expression 1.32 for the calculation of $B^{11}_{M,N}$ coefficient of the DGF since it is valid just for four layers. Adapting 1.32 can be made analytically by writing the system of equations 1.7, 1.8, 1.9: the difference layer by layer is the number of expression 1.8 is repeated. All the repetition contribute in the same way to the growth of the final expression, the trend

of grow can be summarized as follows:

$$\begin{aligned} num_i &= T_{P1}^{H,V} num_{i-1} + \frac{R_{F1}^{H,V}}{T_{F1}^{H,V}} den_{i-1} \\ den_i &= T_{f1}^{H,V} den_{i-1} + \frac{R_{P1}^{H,V}}{T_{P1}^{H,V}} num_{i-1} \end{aligned} \tag{3.1}$$

where num_{i-1} and den_{i-1} are numerator and denominator of the expression to find $B_{M,N}^{11}$ for a sphere with one layer less, but with the layer of application of the field incremented by one. Note that 3.1 follows exactly the same reasoning of the recursive solution in listing 2.5. This solution allow to calculate faster than normal calculation on paper the various expressions for each modification of P1, the problem is the grown of the terms. As can be evicted from expression 3.1, adding a layers means having an expression with a numerator and a denominator that are the sum of terms of denominator and numerator of the previous layer. This means the increment of terms in the expression is exponential (2^{n-1}), so it becomes immediately extremely big, generating problem even to the transcription. The stop to the iteration happened at nine layers, where the expression to calculate $B_{M,N}^{11}$ has 256 terms at numerator and same number at denominator.

3.2 Memory usage performances evaluation

The memory approach is the same in all the versions with some small differences. In P1, and alternative versions for more than four layers, the variables of the input are directly defined in the code. Than all the new variables defined by the input variables or other variables are allocated in the memory. Variables are not all of the same type but are present also vectors and matrices, even if the majority of them is just a single value (often imaginary). The only particularity is the usage of structures to storage variables in order to make shorter the list of arguments passed to some functions. As suggest from Matlab, all the arrays and matrix are defined with a fixed dimension before the calculation of them, in the entire code of P1 and modifications are no vectors with dimension updated at run time. This description is valid for every modification of P1. The variables which impact on the memory usage are the number of layers, the number of coils, the value of mode expansions and the dimension of the FOV. Some variables are derived by them or by quantities them related, in particular there are some matrix that derive from three of them like efield_ult_set = zeros(numbasis,2*tempindex_r,2*tempindex_c,3) where numbasis is derived from the mode expansion and the temp indexes are derived from the dimension of the FOV. Since there are structure dependent from three high impact, the complexity of every modification of P1 is respectively for the best case

and for the worst case:

$$\begin{matrix} \Omega(n) \\ \mathcal{O}(an^3 + bn^2 + cn + d) \end{matrix} \tag{3.2}$$

The best case is assumed to be n since all quantities are considered at their dimensional minimum and a n is intended as the number of layers considered depending on the modification of P1. The worst case is not totally defined, I didn't calculated rigorously all the contribution to the memory computational complexity, what I can say is that the memory computational complexity is function of structures of one, two and three dimensions. The asymptotically complexity is based on the worst case and is $\mathcal{O}(n^3)$.

In P2 the input variables are defined reading values from a Json, passed as function argument. Also in P2 are present variable of various dimension, essentially the configuration is the same of P1. What is missing in P2 is the building of the structures to reduce the number of arguments to some functions. Since the function involved, according to the different structure in P1, is just one I chose to avoid this redundancy saving some memory space, even if the saving is relatively small. P2 anyway has a construction that consume more memory respect to P1: the class sphere has impact in the memory consumption since storage the information of the layers in an object inside an array. This particularity has another small impact on the memory, since the number of stored variables is the same of P1 if the layers considered are the same and the memory in excess is just to store the pointer relations. To populate the class sphere is needed an array which dimension is defined at run time since it depends on the number of layers unknown before the execution. Another aspect that might affect the memory is the calculation of the $B^{11}_{M,N}$ coefficient. P2 and the various modification of P1 calculate the coefficient in two different way. Modification load in memory a quite big expression all at once, in particular the nine layers version has a total of 513 divisions and 510 addictions to keep track in the memory. P2, with the recursive solution presented in listing 2.5, should result lighter since the operation are performed separately so the usage of registers is different. Matlab, in particular, clear the workspace of an ended function, so at every recursion the previous one is reallocated. The variables which impact on the memory usage are, again, the number of layers, the number of coils, the value of mode expansions and the dimension of the FOV and the derived variables are the same of P1. All the reasoning and supposition expressed for P2 anyway can affect more or less the coefficient of the memory computational complexity, but the situation can't change deeply. The complexity of P2 indeed will be:

$$\begin{matrix} \Omega(1) \\ \mathcal{O}(an^3 + bn^2 + cn + d) \end{matrix} . \tag{3.3}$$

This time the best case i 1 since the lower number of possible layers is one. The asymptotically complexity is based on the worst case and is $\mathcal{O}(n^3)$ also for P2.

Comparing the two sets of memory computational complexity is possible to realize that in the worst case P2 and the versions of P1 have the same complexity since it does not depend just on the layers. For what concern the best case instead, P1 modifications has a lower bound dependent on the number of layers considered, so P2 is more performing in all situation different from the lower bound of P1 version (average case).

Considering P1' instead, since it group all the modifications of P1, can have the best case of the one layer version:

$$\begin{matrix} \Omega(1) \\ \mathcal{O}(an^3 + bn^2 + cn + d) \end{matrix} . \tag{3.4}$$

So P1' has orientatively the same memory computational complexity of P2 and also the same asymptotic computational complexity $\mathcal{O}(n^3)$ of P2.

3.3 Execution time performances evaluation

The comparisons concerning the time execution performances are made reproducing tests similar to the test executed for the validation of the code. Since P1 and P2 are not different from what concerns the code execution non dependent from the number of layers and layers features, is reasonable to suppose that their execution times are equal for that part of code, supposition confirmed by the tests. According to this assumption, I chose a setting with a single coil, a standard FOV showing one single voxel in the middle of the sphere, EM field at 5T and a fixed number of mode expansion (lmax=50 exactly to guarantee the convergence). For what concern the sphere instead, is chosen a sphere with inner radius of 1cm and every other layer with a radius increment of 0,02cm, dielectric constant and conductivity are simulated for a generic tissue ($\varepsilon \neq 1, \sigma \neq 0$) and for the empty space ($\varepsilon = 1, \sigma = 0$). On P1 and its modifications, lower number of layers is performed theoretically merging layers choosing for all the merged layers the same EM features. Tests for every version of P1 and P2 are repeated several time in order to produce an average value of the execution time. Tests are repeated 20 times each.

According to the expectation, every version of P1 should have constant average time execution since, even if some layers are not significant, the calculation has to be performed entirely also on non significant layers. P2 instead, since can skip completely the calculation of meaningless layers just not inserting them in the input, should terminate quicker. The only situation where P2 should have the same execution time of a version of P1 is when they are processing the same number of layers. Obviously when P2 simulate a number of layer greater than the possibility of a version of P1 the execution time will be higher than the execution time of the P1 modification, but this result is not meaningful for the test since. Theoretically, can

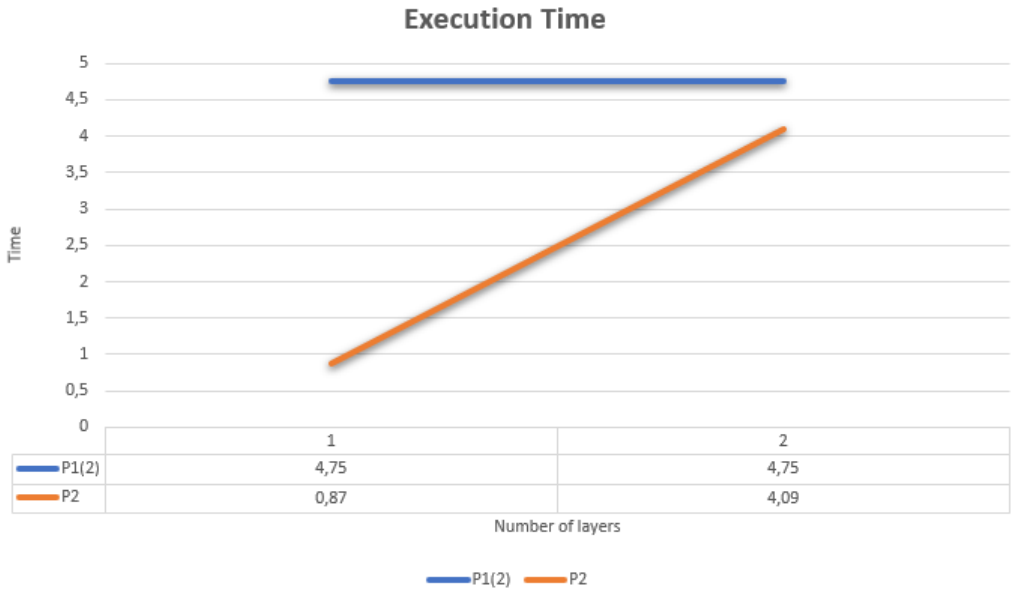

Figure 3.1: Average execution times on a sphere of two layers of up to two generic tissues with a single coil, single voxel in the FOV and mode expansion lmax=50

be supposed a version of P1 which perform the calculation for that number of layers then, comparing the execution time with the one from P2, notice that the execution times would be equal. Those supposition are based on the operation which consume the most time of execution. Algorithm in listing 1.4, is the leech of the program. P2 can avoid it not specifying the meaningless layers, action impossible to do for P1 modifications. Following graphics show the suppositions are verified comparing the execution time.

As shown in fig.s 3.1,3.2,3.3,3.4,3.5,3.6,3.7,3.8, is clear that P2 has better performances since can avoid the modes' noise covariance matrix if possible. Anyway the worst cases are equal for all the versions of P1 and P2. P2 result slightly faster than P1's modifications due to some code cleaning and optimization which reduce the time execution of a small amount.

P1' has average execution times for each layer equal to the execution time of the respective modification of P1 so, again, it has the same best and worst time of P2 (fig. 3.9).

The asymptotic time computational complexity can be analyzed in general or considering just the limitation to a generalization of layers, putting apart the rest of the variables. In both cases anyway, what influence the execution time the most are the number of layers, the number of coils, the value of mode expansions and the dimension of the FOV. If we consider the number of layers as only factor of work, the asymptotic time computational complexity is $\mathcal{O}(n)$. From a general point of view instead all the other factors have an important influence on the execution, so the

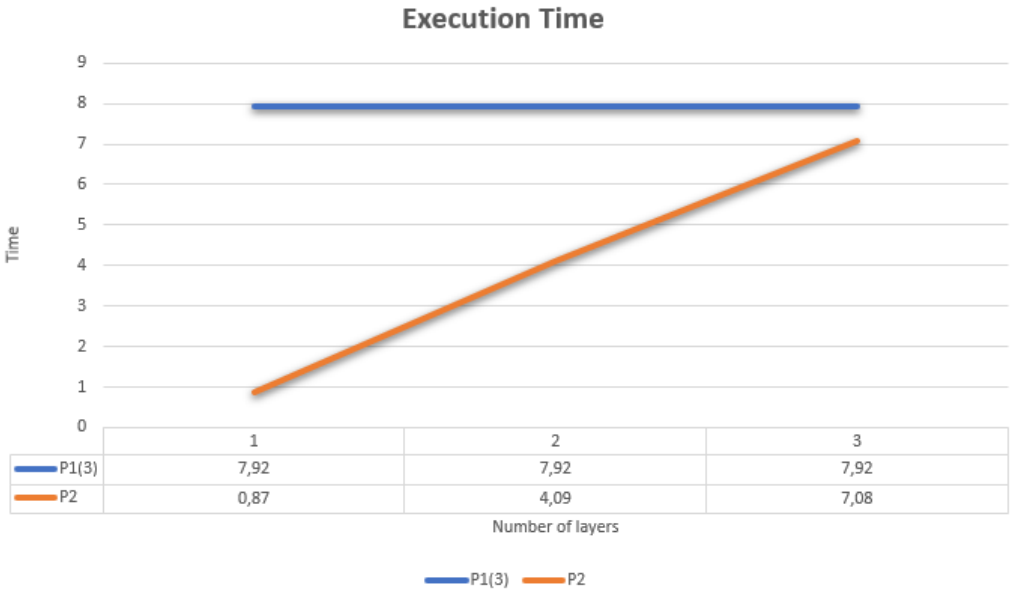

Figure 3.2: Average execution times on a sphere of three layers of up to three generic tissues with a single coil, single voxel in the FOV and mode expansion lmax=50

Since

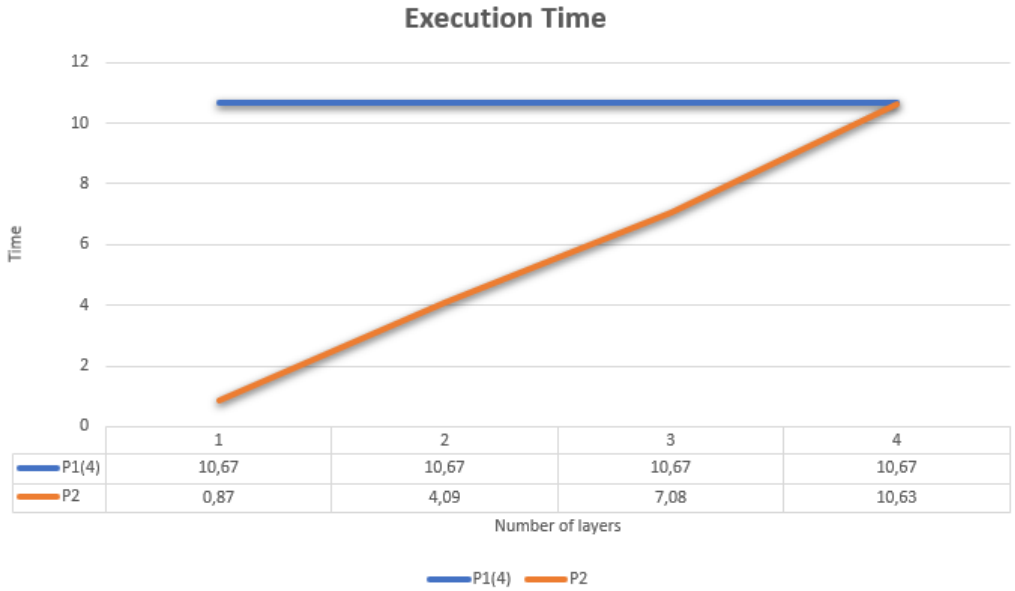

Figure 3.3: Average execution times on a sphere of four layers of up to four generic tissues with a single coil, single voxel in the FOV and mode expansion lmax=50

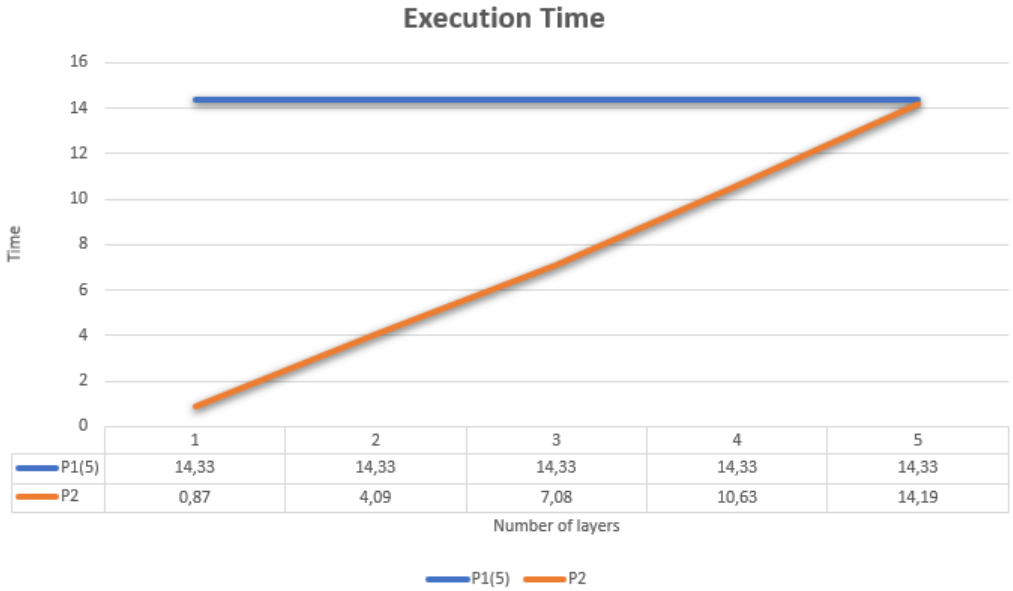

Figure 3.4: Average execution times on a sphere of five layers of up to five generic tissues with a single coil, single voxel in the FOV and mode expansion lmax=50

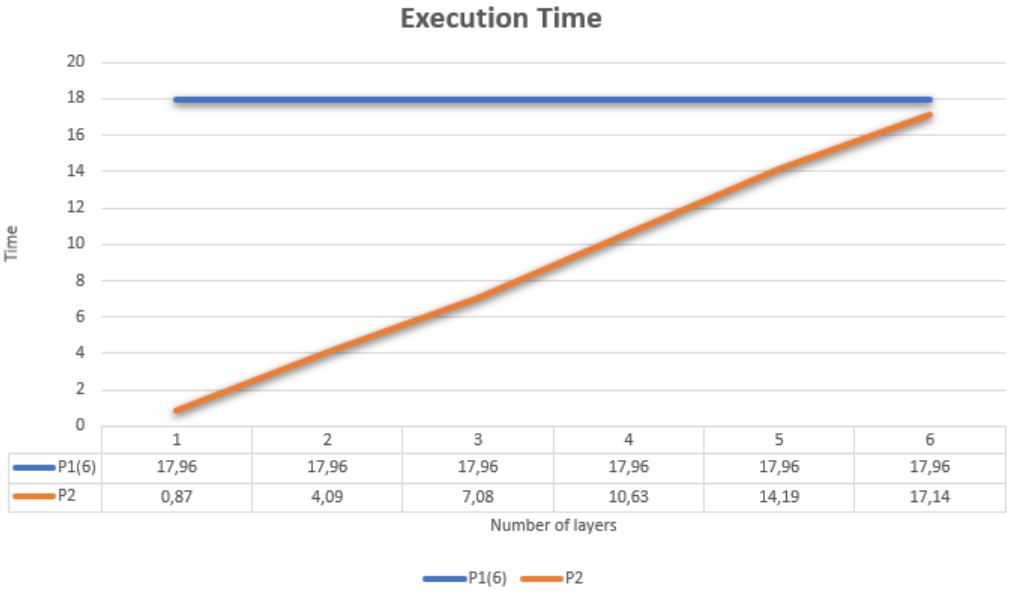

Figure 3.5: Average execution times on a sphere of six layers of up to six generic tissues with a single coil, single voxel in the FOV and mode expansion lmax=50

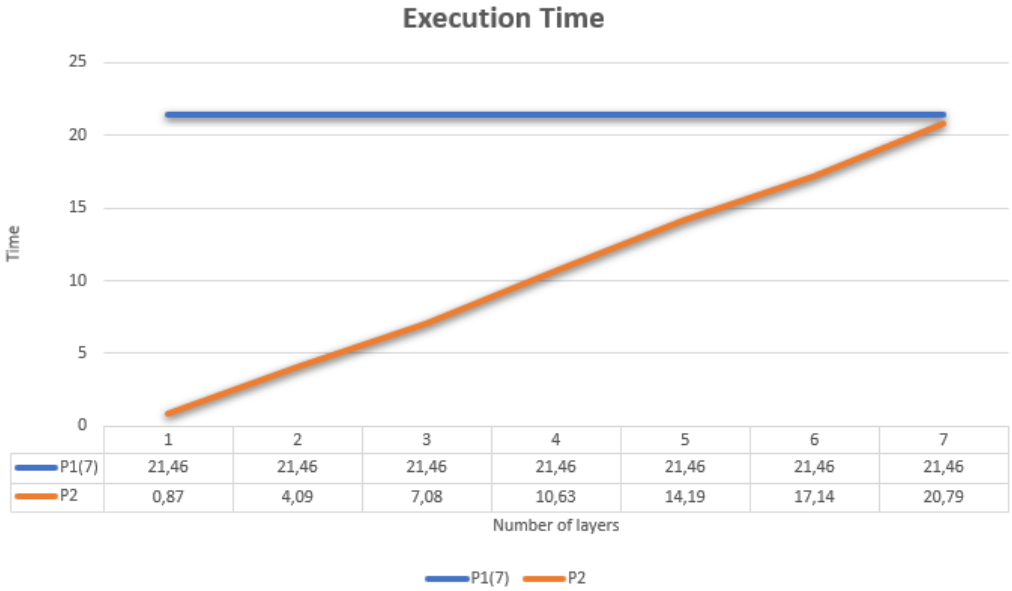

Figure 3.6: Average execution times on a sphere of seven layers of up to seven generic tissues with a single coil, single voxel in the FOV and mode expansion lmax=50

Figure 3.7: Average execution times on a sphere of eight layers of up to eight generic tissues with a single coil, single voxel in the FOV and mode expansion lmax=50

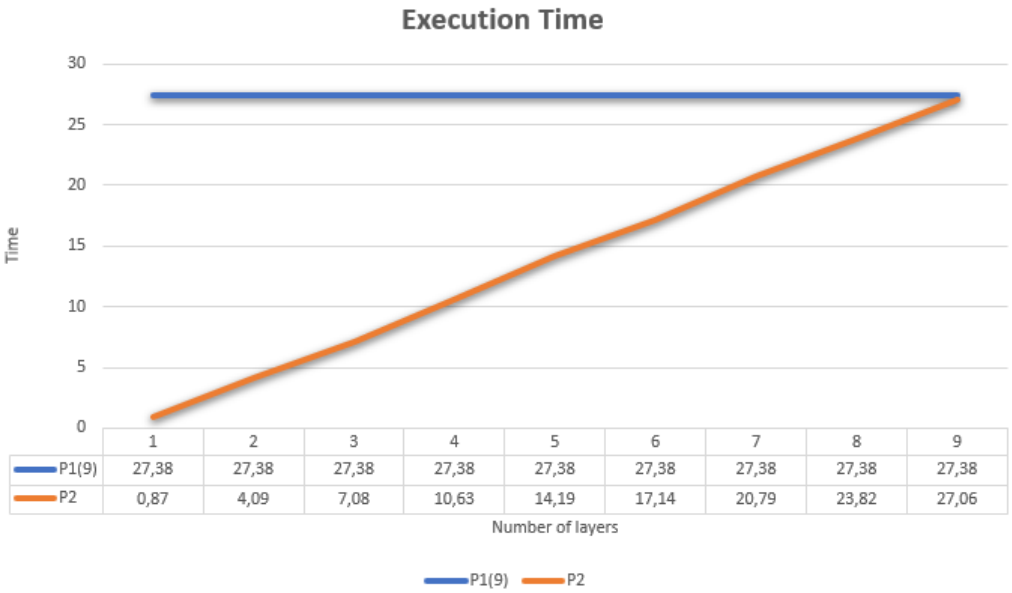

Figure 3.8: Average execution times on a sphere of nine layers of up to nine generic tissues with a single coil, single voxel in the FOV and mode expansion lmax=50

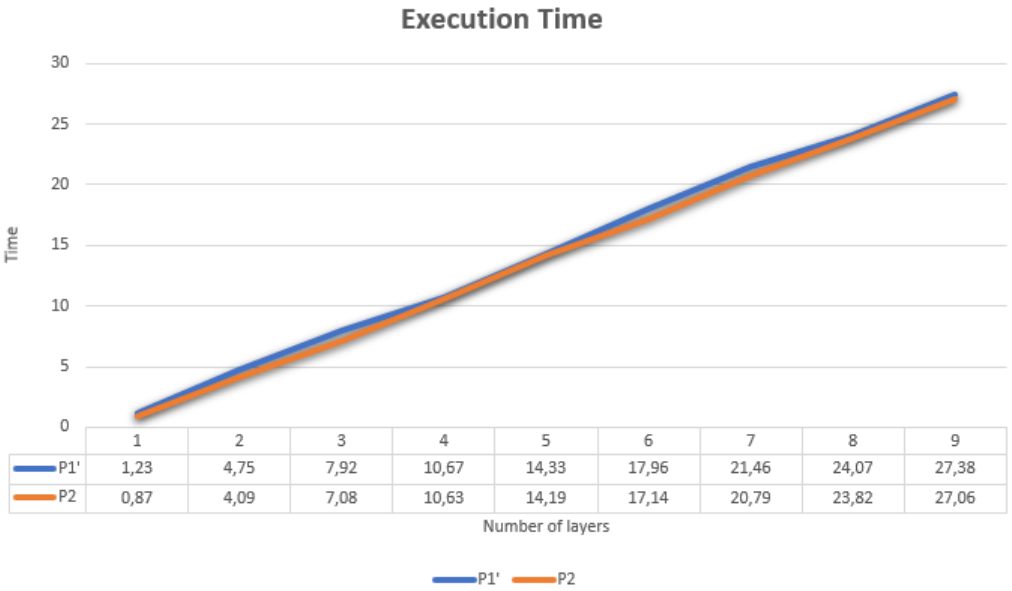

Figure 3.9: Average execution times on a sphere from one to nine layers of up to nine generic tissues with a single coil, single voxel in the FOV and mode expansion lmax=50 using P1' and P2.

asymptotic complexity become $\mathcal{O}(n^4)$. The reasoning on the asymptotic complexity are valid for both P1 and P2.

3.4 Precision performances evaluation

Analyzing P1 and P2 precisions might be an excess of zeal. Both codes are validated, P1 with previous valid works and P2 with P1, so both provide an accurate result of the SNR with an acceptable approximation for research usage. Also does not exists experimental or theoretical standard or tables to compare the calculation result with a "considered perfect" value of SNR. Therefore, is impossible to make an evaluation of the absolute precision of P1 and P2. What can be done is analyze a relative precision: the difference in values according to the effect the calculation have on them. The configurations of the experiments are the same of the ones used for the execution time. P1 and P2 are not different from what concerns the code execution non dependent from the number of layers and layers features also in the case of the precision. According to this assumption, is chosen again a setting with a single coil, a standard FOV showing one single voxel in the middle of the sphere, EM field at 5T and a fixed number of mode expansion (lmax=50 exactly to guarantee the convergence). For what concern the sphere instead, is chosen a sphere with inner radius of 1cm and every other layer with a radius increment of 0,02cm, dielectric constant and conductivity are simulated for several generic tissues ($\varepsilon \neq 1, \sigma \neq 0$) and for the empty space ($\varepsilon = 1, \sigma = 0$). On P1 and its modifications, lower number of layers is performed theoretically merging layers choosing for all the merged layers the same EM features. Contrary to execution time, the tests are performed one time per configuration since the approximation, even if depending from the machine, is fixed once the environment is determined. The tests are grouped by configuration and each configuration is performed for all the versions of P1 and for P2. Then every result of P1 is subtracted to the one of P2 respective to the same configuration. The analysis will be based on the differences in the results.

From the results of the test emerges that the values produced by versions of P1 and by P2 have some extremely small differencies. Since the calculator can't handle a perfect zero, it is approximated if the computation is dealing with really small values. The quantities the calculation of SNR bight be around the order of 10^{-5}, so the approximation is present for sure. Those approximation affect the calculation according to the number of operation executed.

The first difference emerged by the approximation is that results of P2 minus the version of P1, on a configuration where the number of layers is the best the version of P1 can manage, result to be approximately constant and equal to an unit of $2,6389 \cdot 10^{-23}$ per layer considered. The unit number is calculated in the average of the differences, keeping fixed the number of layers and changing radii and EM

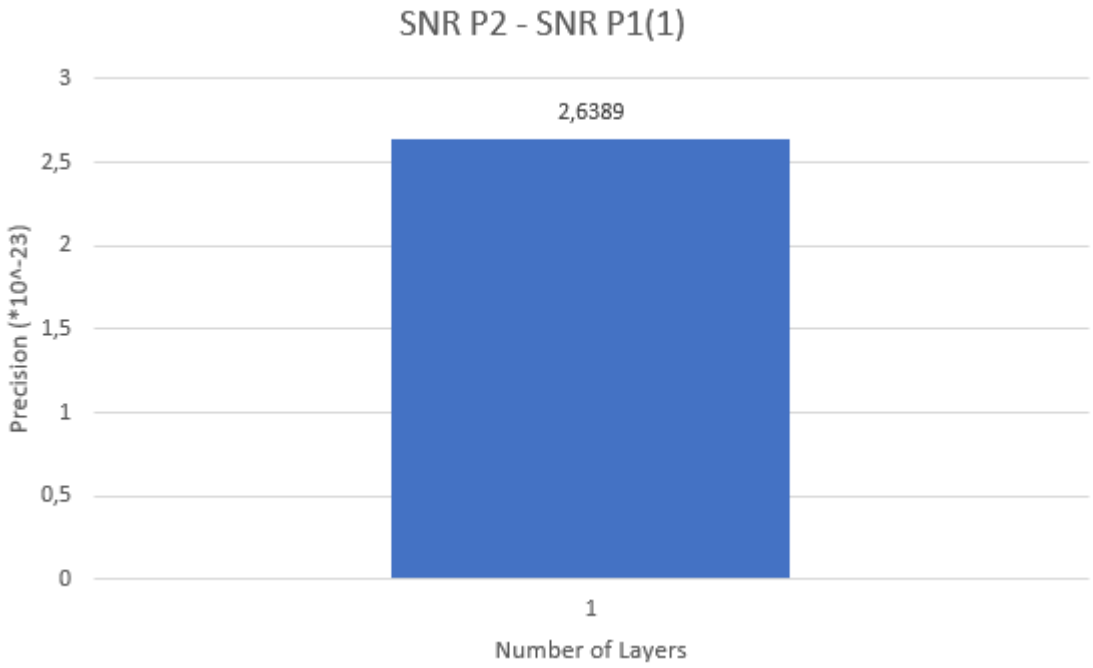

Figure 3.10: Average difference SNR(P2)-SNR(P1(1)) on a sphere of up to one layers of up to one generic tissues with a single coil, single voxel in the FOV and mode expansion lmax=50 for the contribution of the DGF coefficient calculation.

features. This behavior is present because P1 perform several "useless" operations, a trivial example is deriving the radii from the sum of inner radius plus an increment or some division performed as $a \times (1/b)$ instead than a/b.

Also computing the coefficient $B^{11}_{M,N}$in a different way produce some differences in the results. The computation of a one big expression made by several operation from an approximation point of view is different from performing a sequence of small and simple operations. According with this, can be noticed the differences of values from P2 and P1 deriving from this issue is, on average, is $7,9623 \cdot 10^{-23}$, multiplied by the number of layers considered minus one. Make an exception the version for one layer of P1, since it practically perform the same operations of P2, so the difference in the approximation is zero. Fig.s 3.10,3.11,3.12,3.13,3.14,3.15,3.16,3.17,3.18 show the trend of growth of the average difference of the results. Notice P1 is always more affected by approximation. The last factor that affect the results is the modes' noise covariance matrix, which is the most onerous calculation in the algorithm. Anyway here the trending is directed in an opposite direction. Indeed the difference caused by the approximation is high when a modification of P1 is performing a calculation on a sphere with low number of layers respect to its limit, but, incrementing the number of layers, P2 start to perform the modes' noise covariance matrix calculation as much as P1 reducing the difference. This time the average unitary approximation is bigger than the contribution given from the other two previous analysis. This time

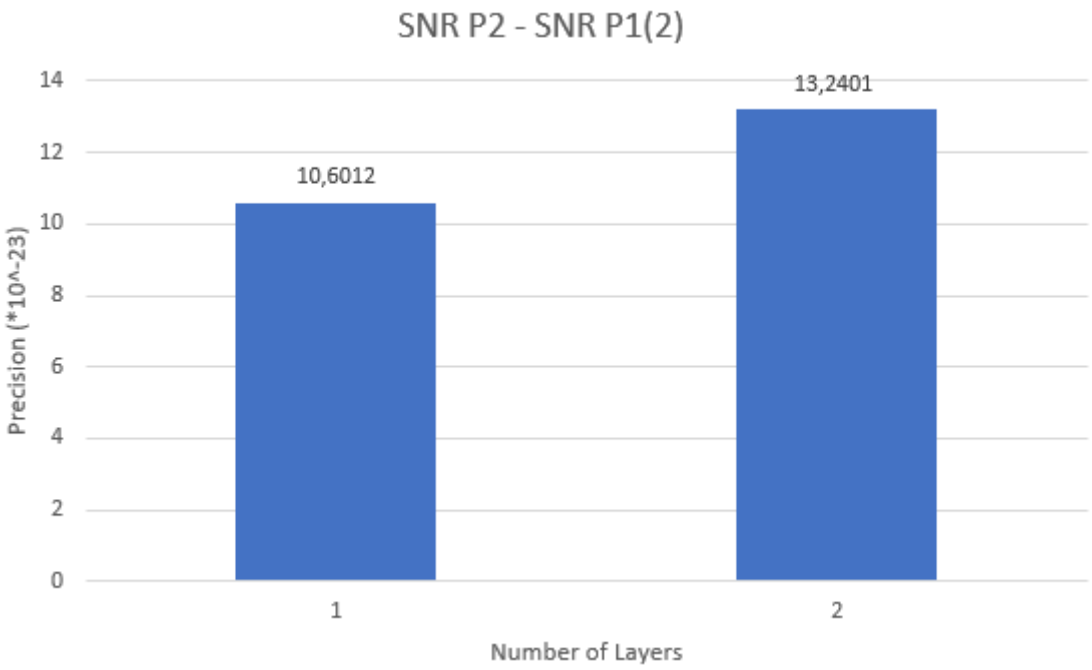

Figure 3.11: Average difference SNR(P2)-SNR(P1(2)) on a sphere of up to two layers of up to two generic tissues with a single coil, single voxel in the FOV and mode expansion lmax=50 for the contribution of the DGF coefficient calculation.

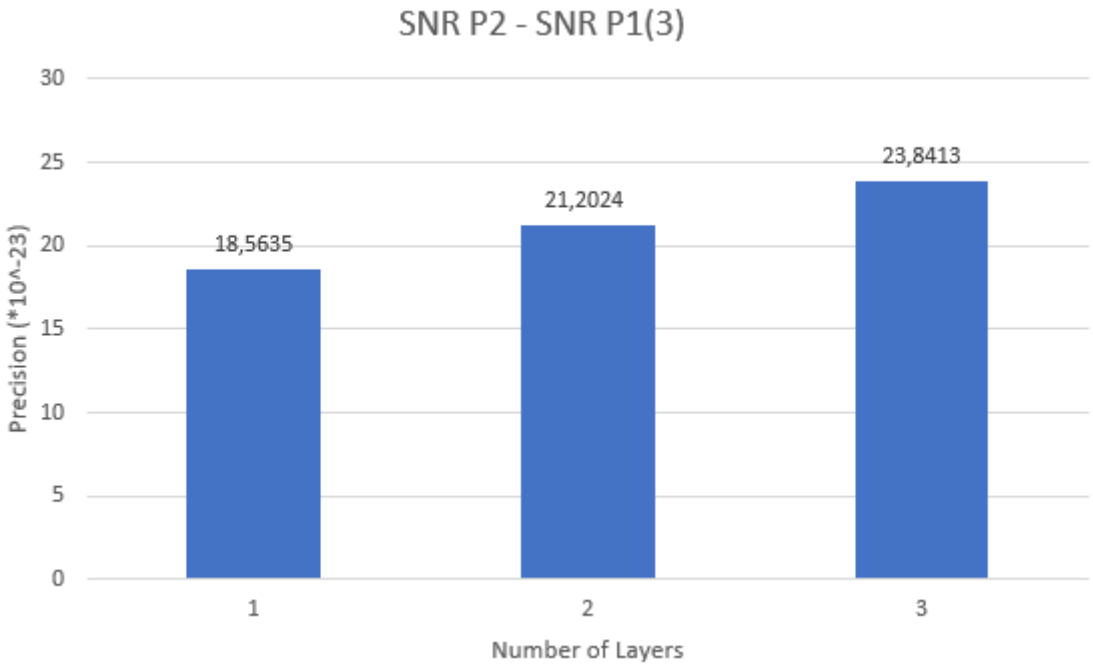

Figure 3.12: Average difference SNR(P2)-SNR(P1(3)) on a sphere of up to three layers of up to three generic tissues with a single coil, single voxel in the FOV and mode expansion lmax=50 for the contribution of the DGF coefficient calculation.

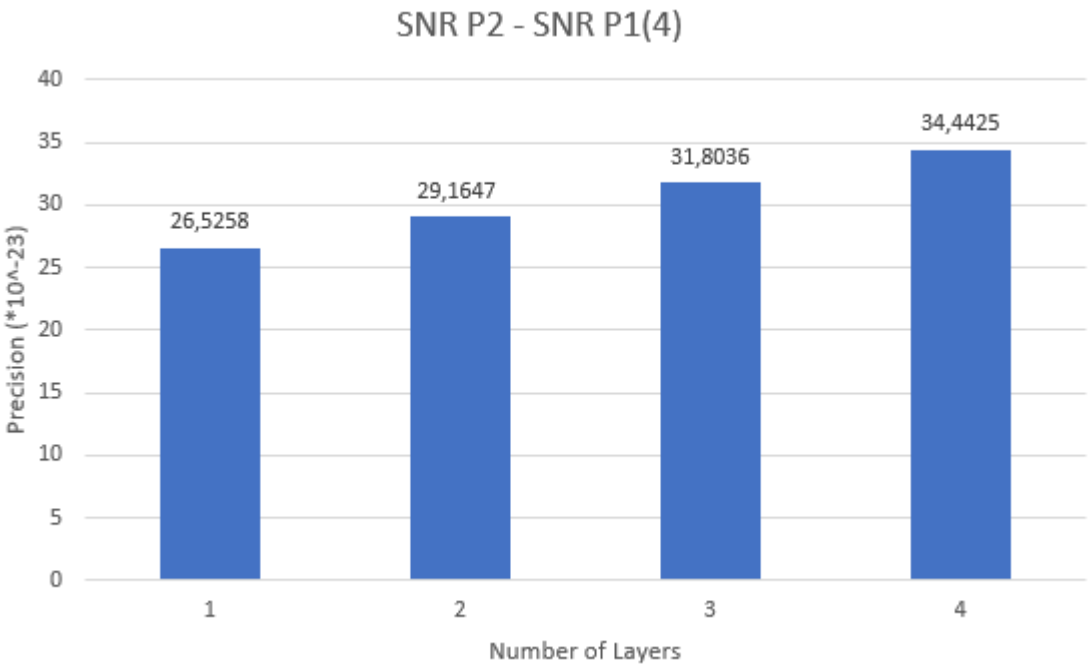

Figure 3.13: Average difference SNR(P2)-SNR(P1(4)) on a sphere of up to four layers of up to four generic tissues with a single coil, single voxel in the FOV and mode expansion lmax=50 for the contribution of the DGF coefficient calculation.

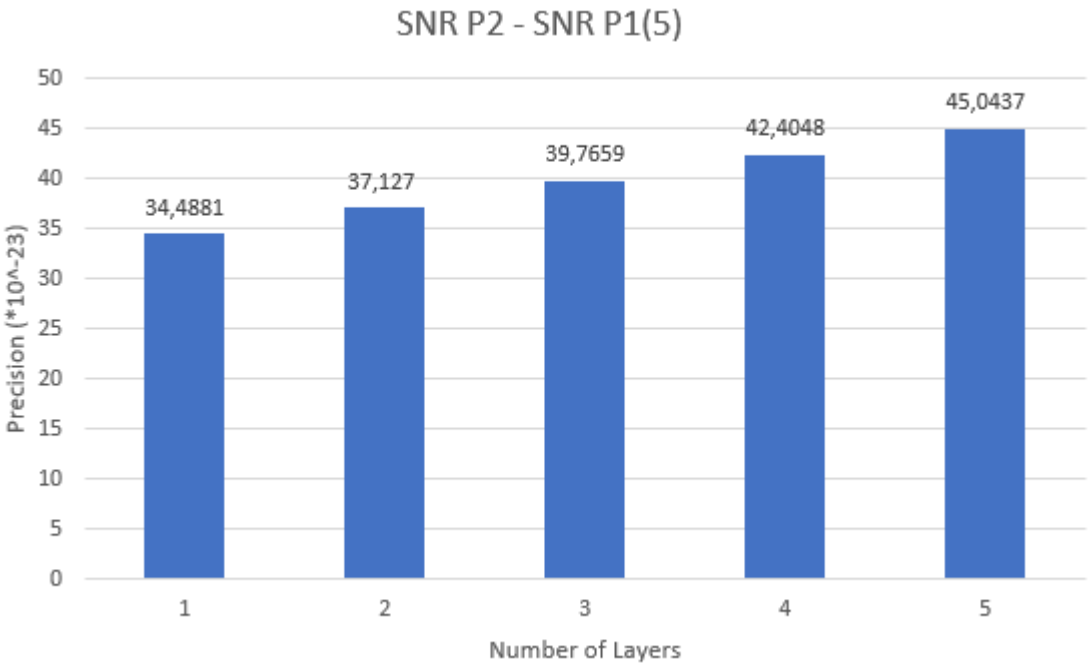

Figure 3.14: Average difference SNR(P2)-SNR(P1(5)) on a sphere of up to five layers of up to five generic tissues with a single coil, single voxel in the FOV and mode expansion lmax=50 for the contribution of the DGF coefficient calculation.

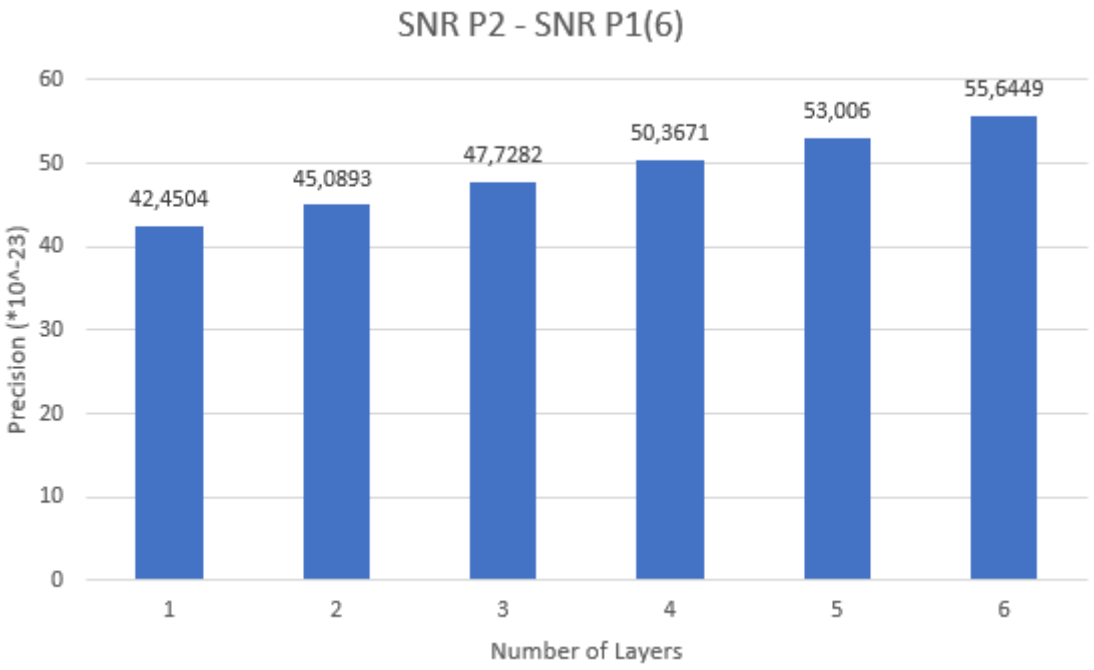

Figure 3.15: Average difference SNR(P2)-SNR(P1(6)) on a sphere of up to six layers of up to six generic tissues with a single coil, single voxel in the FOV and mode expansion lmax=50 for the contribution of the DGF coefficient calculation.

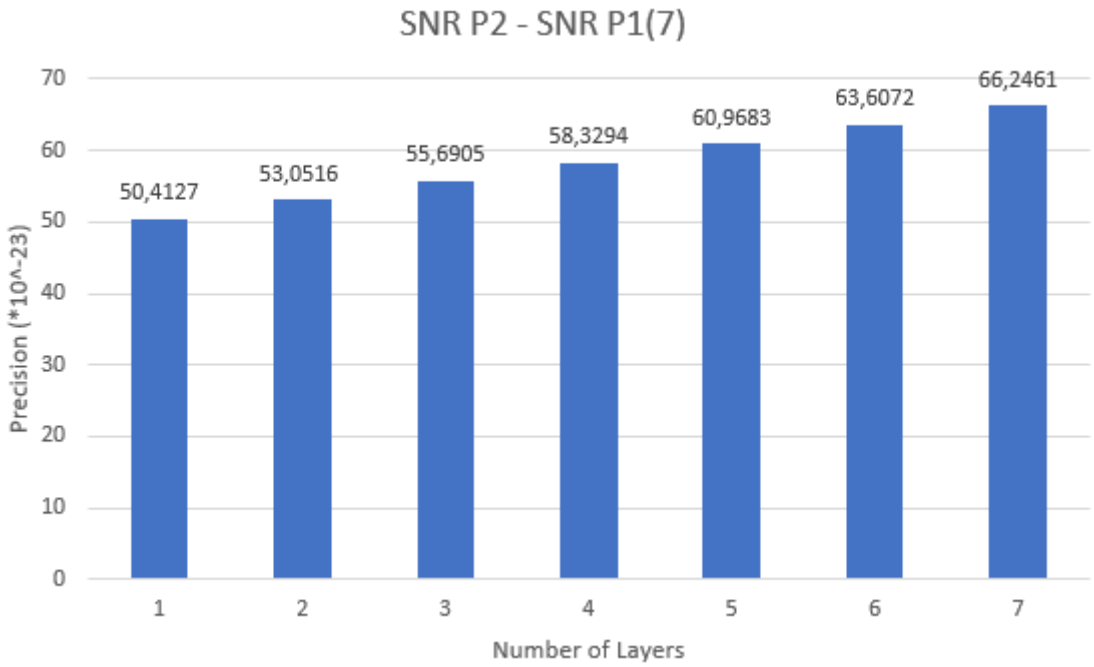

Figure 3.16: Average difference SNR(P2)-SNR(P1(7)) on a sphere of up to seven layers of up to seven generic tissues with a single coil, single voxel in the FOV and mode expansion lmax=50 for the contribution of the DGF coefficient calculation.

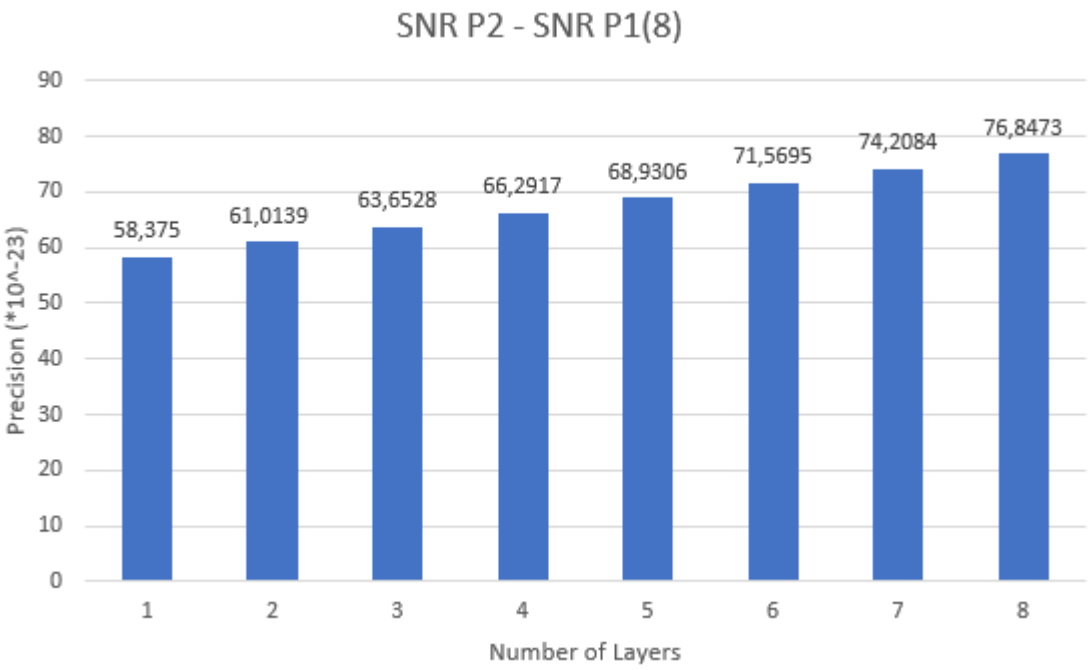

Figure 3.17: Average difference SNR(P2)-SNR(P1(8)) on a sphere of up to eight layers of up to eight generic tissues with a single coil, single voxel in the FOV and mode expansion lmax=50 for the contribution of the DGF coefficient calculation.

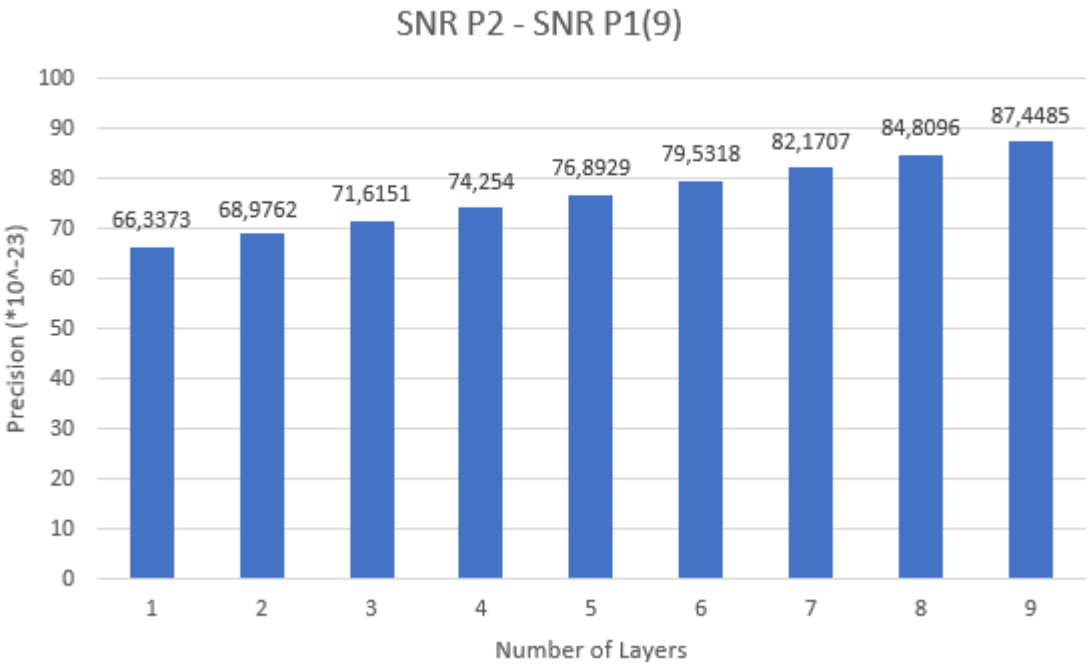

Figure 3.18: Average difference SNR(P2)-SNR(P1(9)) on a sphere of up to nine layers of up to nine generic tissues with a single coil, single voxel in the FOV and mode expansion lmax=50 for the contribution of the DGF coefficient calculation.

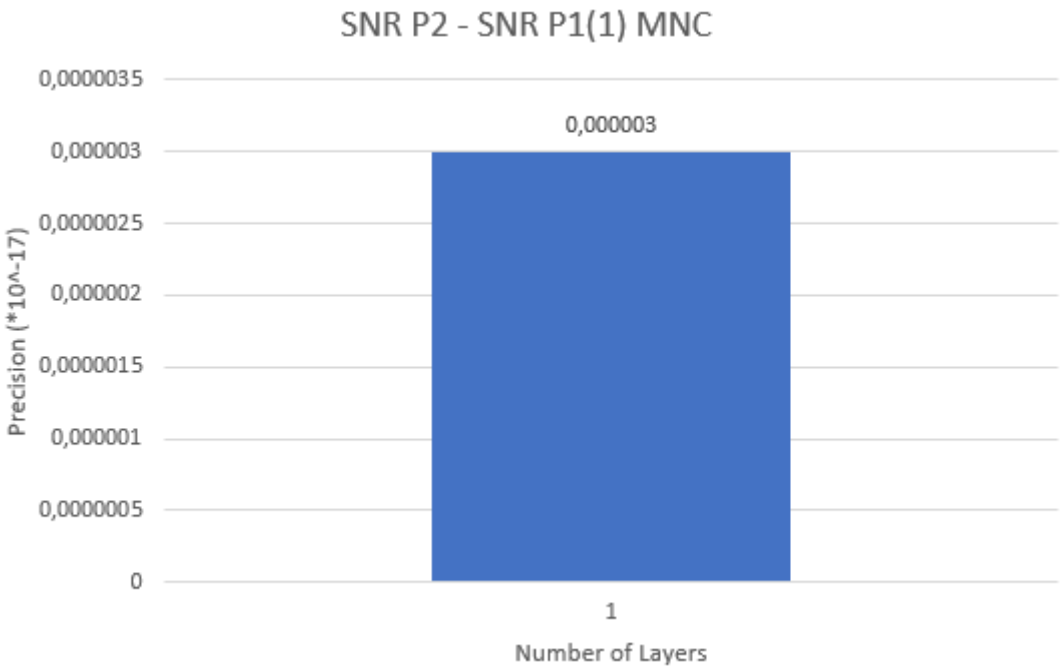

Figure 3.19: Average difference SNR(P2)-SNR(P1(1)) on a sphere of up to one layers of up to one generic tissues with a single coil, single voxel in the FOV and mode expansion lmax=50 for the contribution of the modes' noise covariance matrix.

the difference is $1,3878 \cdot 10^{-17}$. In fig.s 3.19,3.20,3.21,3.22,3.23,3.24,3.25,3.26,3.27 is shown the trend, the contribution of the first two observations is not visible since relatively quite small. Also this time P1 versions are most affected by approximations.

Taking now into account the version P1', it takes the best cases of each P1 version. The result is that the difference from the approximation by the modes' noise covariance matrix is always zero since it is calculated the same number of time for every number of layers for both the algorithms. Anyway the difference in precision of the two algorithms is still not equal to zero, because the approximation given by the writing choice and those concerning the calculation of $B^{11}_{M,N}$ coefficient are still present (fig 3.28). So there is an actual difference between P1' and P2 because P1' is more affected by the approximation performed by the machine.

Even if in the entire section talks about approximation, is important to remember and notice that the order of magnitude of the differences between the results of the code are extremely low compered to the effective result. The percentage of those differences is around the 0,000001%, therefore both algorithms can be considered equivalent.

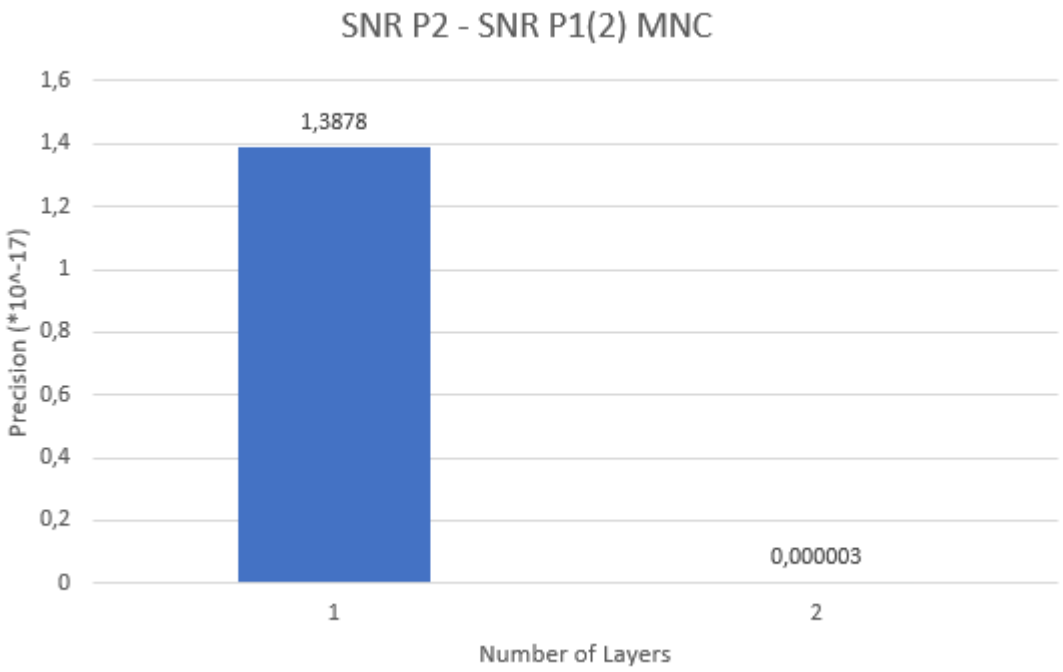

Figure 3.20: Average difference SNR(P2)-SNR(P1(2)) on a sphere of up to two layers of up to two generic tissues with a single coil, single voxel in the FOV and mode expansion lmax=50 for the contribution of the modes' noise covariance matrix.

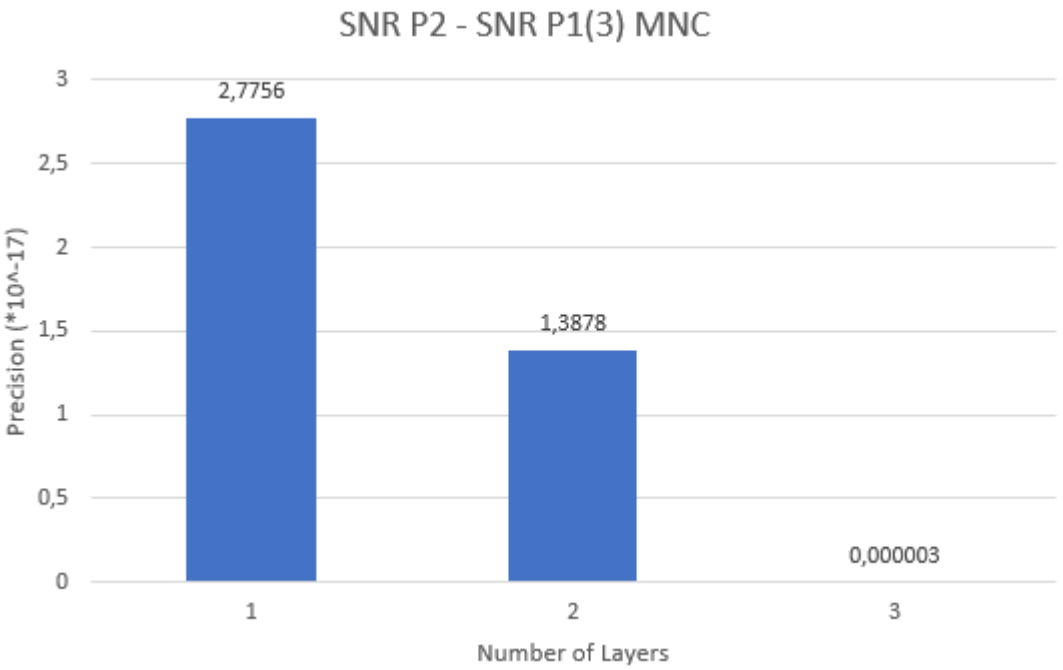

Figure 3.21: Average difference SNR(P2)-SNR(P1(3)) on a sphere of up to three layers of up to three generic tissues with a single coil, single voxel in the FOV and mode expansion lmax=50 for the contribution of the modes' noise covariance matrix.

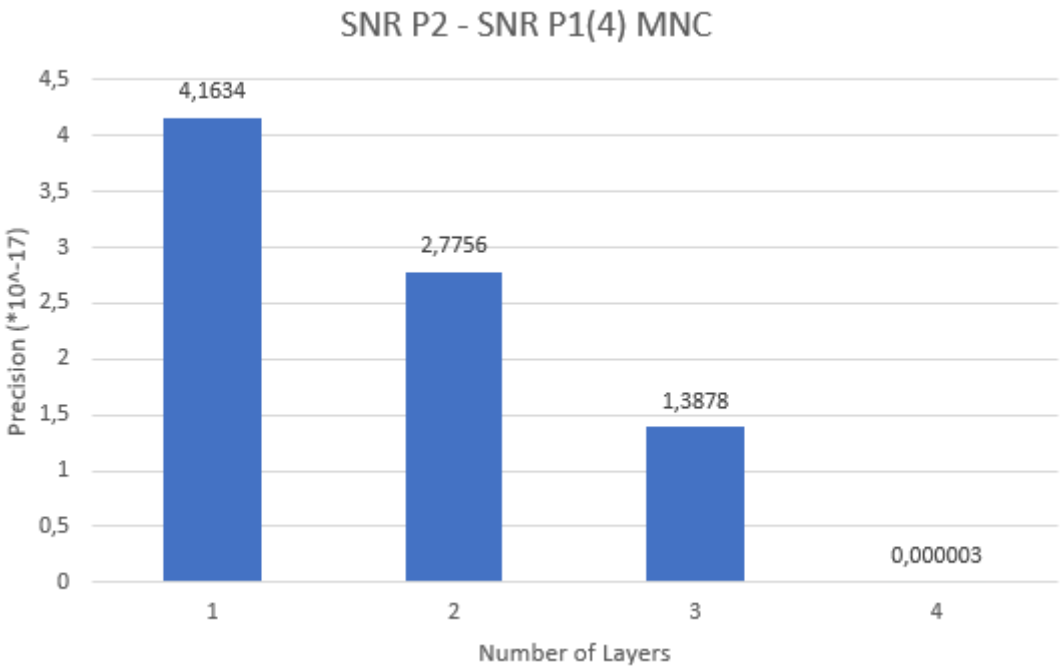

Figure 3.22: Average difference SNR(P2)-SNR(P1(4)) on a sphere of up to four layers of up to four generic tissues with a single coil, single voxel in the FOV and mode expansion lmax=50 for the contribution of the modes' noise covariance matrix.

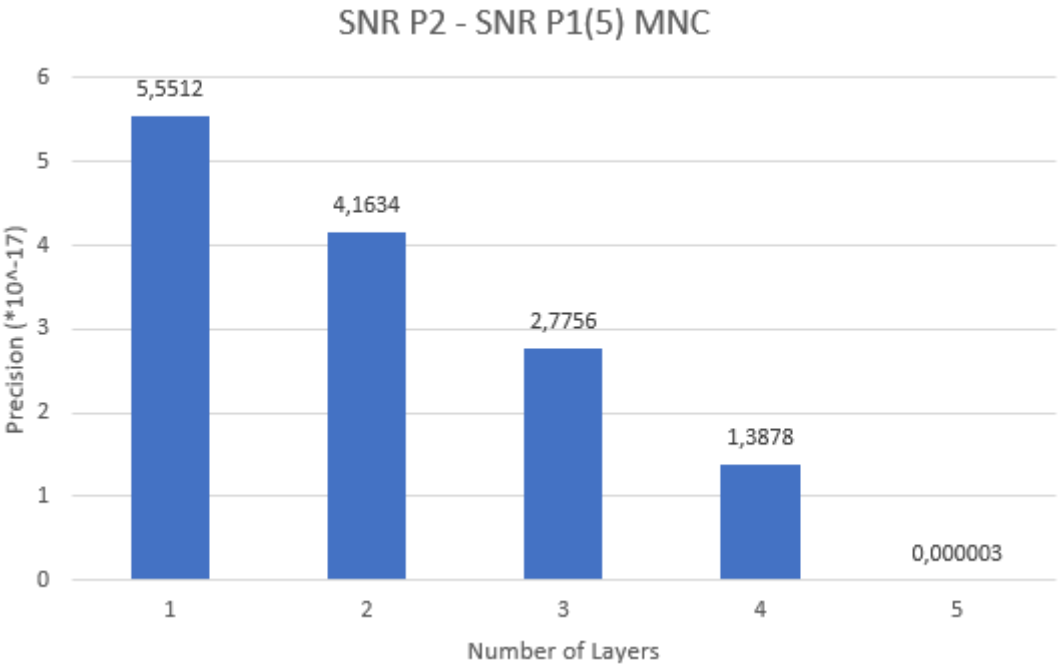

Figure 3.23: Average difference SNR(P2)-SNR(P1(5)) on a sphere of up to five layers of up to five generic tissues with a single coil, single voxel in the FOV and mode expansion lmax=50 for the contribution of the modes' noise covariance matrix.

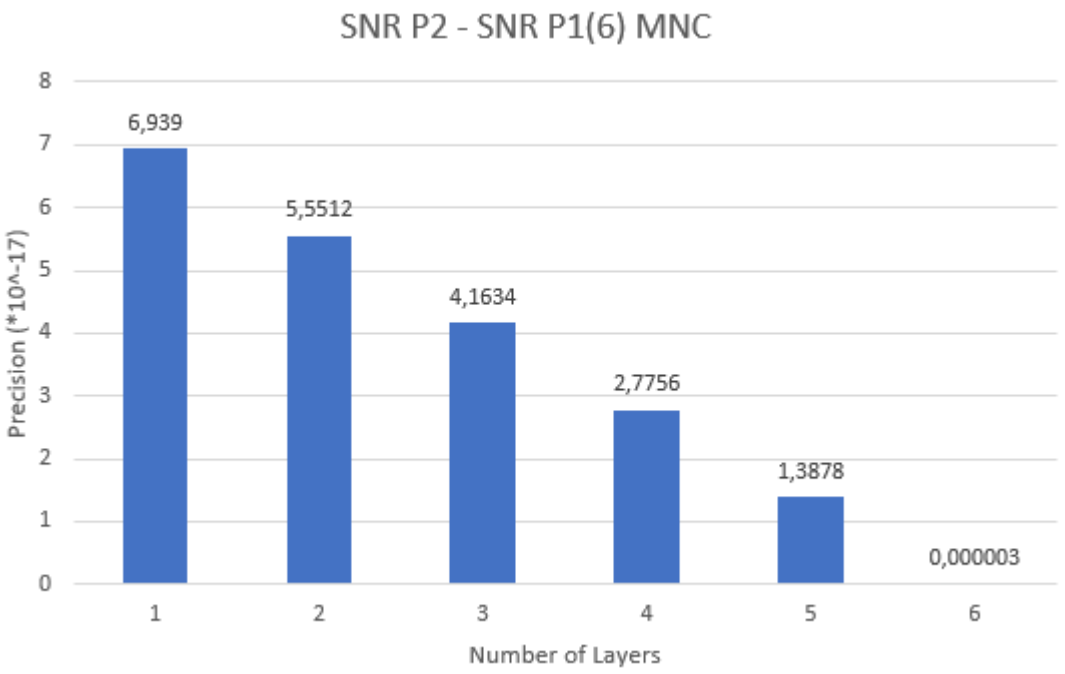

Figure 3.24: Average difference SNR(P2)-SNR(P1(6)) on a sphere of up to six layers of up to six generic tissues with a single coil, single voxel in the FOV and mode expansion lmax=50 for the contribution of the modes' noise covariance matrix.

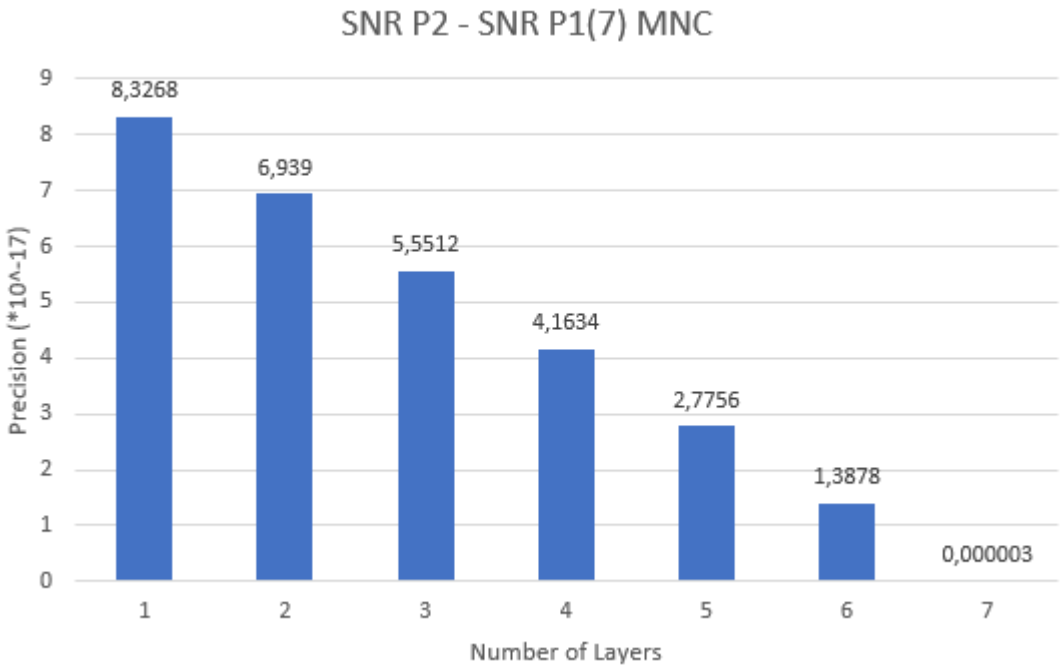

Figure 3.25: Average difference SNR(P2)-SNR(P1(7)) on a sphere of up to seven layers of up to seven generic tissues with a single coil, single voxel in the FOV and mode expansion lmax=50 for the contribution of the modes' noise covariance matrix.

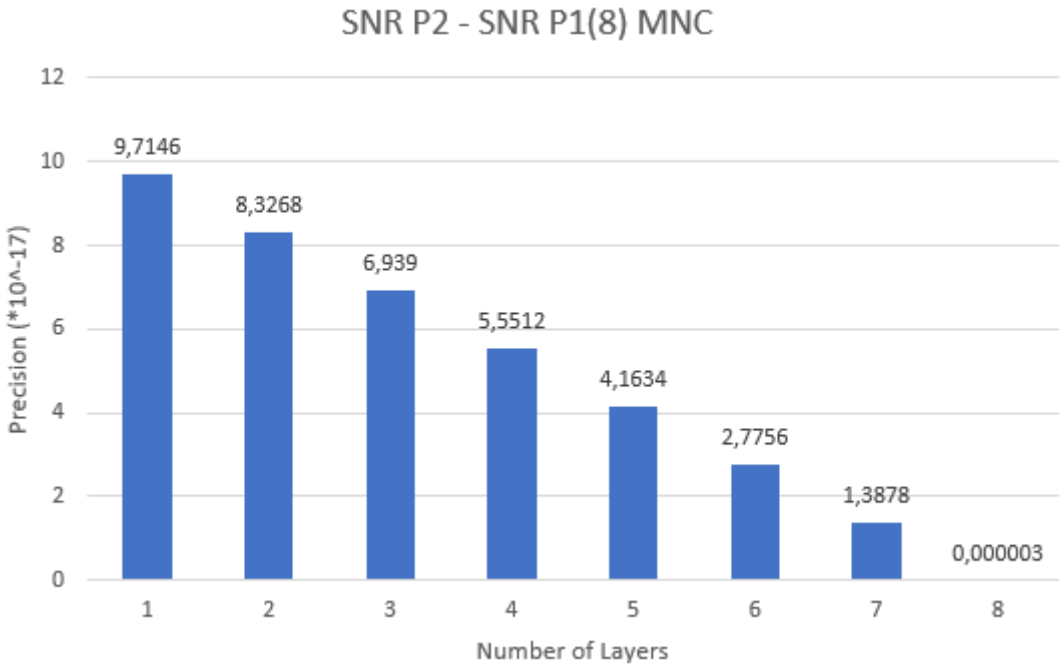

Figure 3.26: Average difference SNR(P2)-SNR(P1(8)) on a sphere of up to eight layers of up to eight generic tissues with a single coil, single voxel in the FOV and mode expansion lmax=50 for the contribution of the modes' noise covariance matrix.

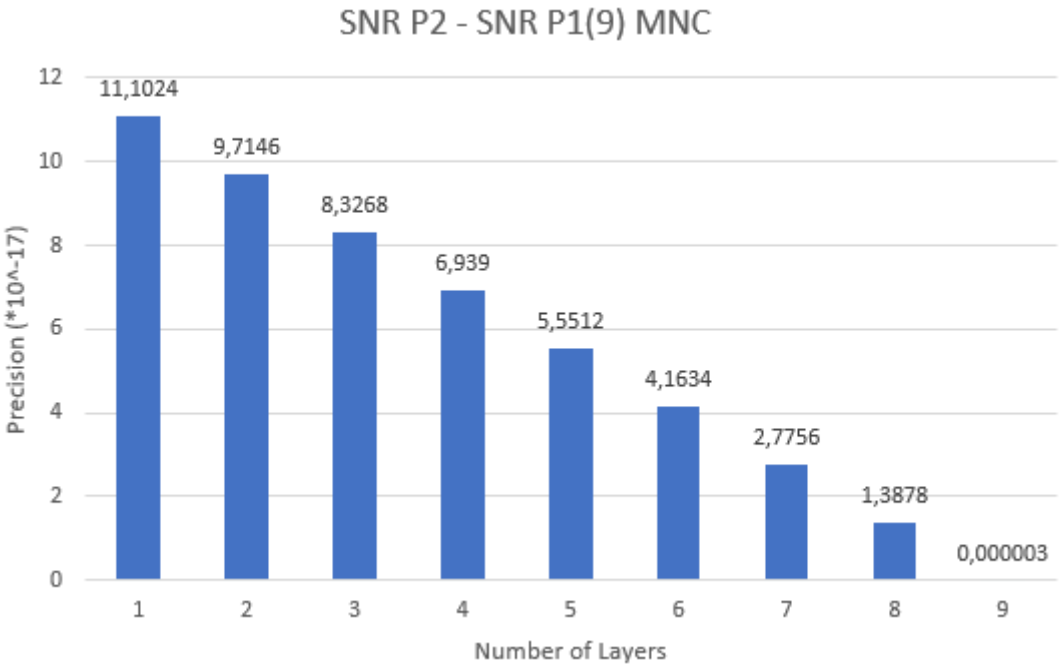

Figure 3.27: Average difference SNR(P2)-SNR(P1(9)) on a sphere of up to nine layers of up to nine generic tissues with a single coil, single voxel in the FOV and mode expansion lmax=50 for the contribution of the modes' noise covariance matrix.

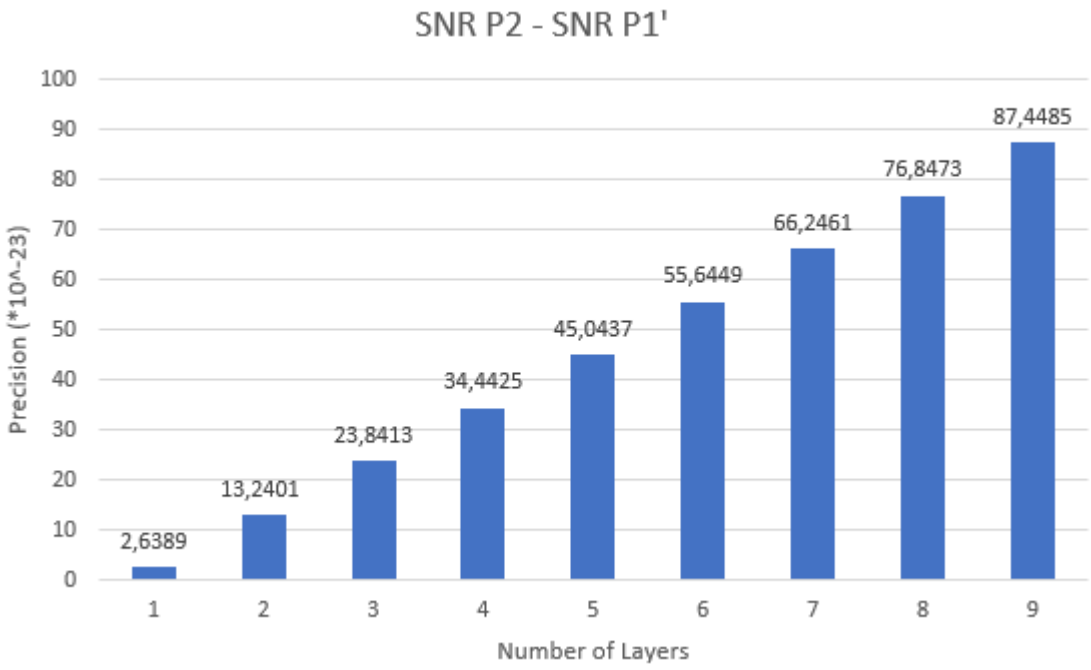

Figure 3.28: Average difference SNR(P2)-SNR(P1') on a sphere from two to nine layers of up to nine generic tissues with a single coil, single voxel in the FOV and mode expansion lmax=50 for the contribution of the DGF coefficient calculation.

3.5 Dataset for the experiments

Tissue	Radius (m)	Dielectric constant ε	Conductivity σ (1/(Ω*m))
Cerebellum	0,01	40	0,4
White Matter	0,04	32	0,2
Grey Matter	0,1	45	0,2
Water	0,11	80,2	0,005
Skull	0,115	18	0,7
Fat	0,116	3	0,1
Muscle	0,116	50	0,2
Skin	0,117	39	0
Hair	0,12	15	0
Void	/	1	0

Data of this table are from [3].

Coil Data			
Rotation in X	Rotation on Y	Radius (m)	Copper Conductivity σ (1/(Ω*m))
-180,+180	-180,+180	0,0125	58x10^6

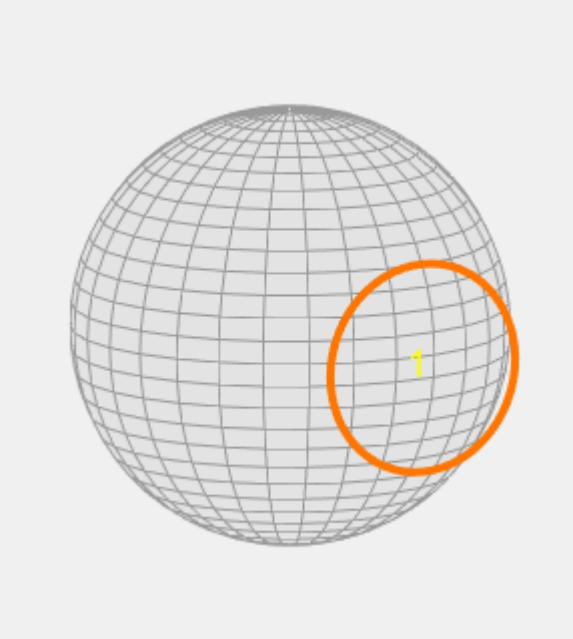

Figure 3.29: Graphical representation of the tests settings

3.6 Performances evaluation summary

The testing executed in this chapter are aimed to see if P2 would be able to bring some increment in performances respect to P1. Appeared clear from the beginning that the the general version of the algorithm would have been more performing than a specific one since it can be adapted to the configuration. From the experiments emerged that P2 is more performing in the best and in the average cases in all memory consumption, execution time and precision, especially in the first two categories. This statement is true for every version of P1 specialized for the calculation of a specific number of layers. The version to calculate the SNR for just one layer might be considered an exception, since the best case and the worst case coincide for this algorithm resulting as performing as P2.

Even if P2 outclass separate versions of P1, those versions can be grouped in a single algorithm P1'. P1', speaking about performance, is the assemblage of all the best cases of the various versions of P1. Under this assumption, the difference in performance between P2 and P1' are no more so different. Both algorithms are general so the can handle the calculation in similar way and this obviously comport also similar performances. Indeed, P1' result very similar to P2 in all memory consumption, execution time and precision. Very similar anyway is not equal since some differences are present, especially for what concern the memory, but very small so they can be ignored. Even if ignored, those differences are all in favor of P2, so we can say that is "infinitesimally" better than P1'.

The summary of the analysis is therefore that P2 does not provide a considerable

increment of performances, but at least a little one. This just from the point of view of memory consumption, execution time and precision. P2 is still better than P1' about generalization, because P1' can't be a real solution since all the expression to calculate $B^{11}_{M,N}$has to be written in the code compile time, and this is not a real possibility. Also, even if the memory of storage of the compiled algorithm is not an issue anymore, it would be a problem for a complete version of P1' because it would need of infinite memory for the storage and for the execution, instead P2 is dependent just from the execution memory consumption.

3.7 Technology used for the development and testing

All the codes are written in Mathlab, both for P1 and P2. The choice for P1 were made in 2005 by Professor Lattanzi. Matlab was at that time the most used framework for mathematical and physical simulations in the Radiology department of New York University, therefore Professor Lattanzi was quite familiar with it. Another reason is that Matlab has an integrated library ready to perform calculation for EM applications like the Bessel's or the Henkel's functions. Matlab has also an integrated way to generate and display windows and graphics, very useful for a research point of view. The last motivation is because it makes easy comparing results with other works on the same topic, still written in mathlab. The entertainment in P2 of the Matlab framework is done for the same reasons of Professor Lattanzi. Since P2 has to be validated, having the possibility to save workspaces and directly comparing them was quite useful. Another reason is the possibility to reuse the code when no significant changes were produced.

To help the code writing was used Brackets as text editor. It was useful for write the Jsons needed for validations and tests, and for input in general. The various versions of the equations to compute $B^{11}_{M,N}$ in the modifications of P1 where computed following the algorithm 3.1 logic, coping and pasting what was needed on Brackets to speed up the operations.

To storage data from the test and to make graphics to display them was used Excel. Was possible also on Mathlab but I am more familiar with Excel so I chose to speed up and simplify the work with the Excel framework.

All the tests were performed on my personal computer. The processor is an intel core i7 and the operative system is Windows 10. It is important to specify since the execution time[1] and the precision of the algorithms are dependent from the machine. The effect on the precision does not influence the result of the algorithm practically, but, since the machine has to approximate the zero, output values might be affected by some approximations which depends from the machine and operative system used.

[1]The time computational complexity is obviously not dependent from the machine, indeed here I'm referring to the effective time of execution of the code.

No data are taken from simulation on other machines.

Conclusions

In the field of modeling EM fields for the design of efficient coils, able to produce MR images with high quality, having the possibility to perform fast but rigorous simulation is, as shown, fundamental.

In this work is presented a software that use mode expansions with DGFs to characterize the fullwave EM field in a dielectric sphere. Thanks to this program is possible to perform quick simulations for scans of three layers sphere with any number of coils. Given the lack of generality of the program, is presented also a solution based on the same theory and algorithm that, thanks to an object based approach and a recursive solution, allows simulation also with any number of layers. The two algorithms are then compared from a performance point of view, analyzing memory consumption, execution time and precision.

Form a research point of view, since the SNR is strictly dependent from the geometry and EM features of coils and body, the result is quite useful since the solution presented allows fully general simulation. Results from the performance point of view are also good, the modification of the original software is more performing under any point of view. Anyway, supposing a version of the system made by grouping several versions of P1 specialized in different number of layers, the performances become quite similar, with P2 presenting an really small advantage. The benefits proposed by P2 anyway are relevant even comparing it with P1'.

Future developments on this subject are essentially infinite. From a software engineering point of view, it can be uploaded in a web application and start collecting data useful for the coil design. From a biomedical engineering or medical point of view instead an interesting development can be expand the calculation also to cylinders based on existing theory [19, 23, 24] in order to make possible the simulation of an entire body.

Bibliography

[1] D. Amati. M. e. rose -elementary theory of angular momentum. *Wiley*, 16:1160–1160, 1960.

[2] Giuseppe Carluccio, Gillian Haemer, and Christopher Collins. Snr improvement when a high permittivity material helmet-shaped former is used with a close-fitting head array, 2018.

[3] Danilo De Rossi Carpi Federico. ProprietÃ elettriche dei tessuti non eccitabili, 2009.

[4] Christopher M. Collins and Michael B. Smith. Calculations ofb1 distribution, snr, and sar for a surface coil adjacent to an anatomically-accurate human body model. *Magnetic resonance in medicine*, 45:692–699, 2001.

[5] Mark A Griswold, Peter M Jakob, Robin M Heidemann, Mathias Nittka, Vladimir Jellus, Jianmin Wang, Berthold Kiefer, and Axel Haase. Generalized autocalibrating partially parallel acquisitions (grappa). *Magnetic Resonance in Medicine: An Official Journal of the International Society for Magnetic Resonance in Medicine*, 47(6):1202–1210, 2002.

[6] D. I. Hoult. The principle of reciprocity in signal strength calculations?a mathematical guide. *Concepts in Magenetic Resonance*, 12:173–187, 2000.

[7] David I. Hoult. Sensitivity and power deposition in a high-field imaging experiment. *Magnetic Resonance Imaging*, 12:46–67, 2000.

[8] Tamer S. Ibrahim, Chad Mitchell, Petra Schmalbrock, Robert Lee, and Donald W. Chakeres. Electromagnetic perspective on the operation of rf coils at 1.5-11.7 tesla. *Magnetic resonance in medicine*, 54:683–690, 2005.

[9] Alayar Kangarlu, Tamer S. Ibrahim, and Frank G. Shellock. Effects of coil dimensions and field polarization on rf heating inside a head phantom. *Magnetic resonance in medicine*, 23:53–60, 2005.

[10] J. R. Keltner, J. W. Carlson, M. S. Roos, S. T. S. Wong, T. L. Wong, and T. F. Budinger. Electromagnetic fields of surface coilin vivo nmr at high frequencies. *Magnetic resonance in medicine*, 22:467–480, 1991.

[11] Riccardo Lattanzi, Aaron K. Grant, Jonathan R. Polimeni, Michael A. Ohliger, Graham C. Wiggins, Lawrence L. Wald, and Daniel K. Sodickson. Performance evaluation of a 32-element head array with respect to the ultimate intrinsic snr. *NMR in Biomedicine*, pages n/a–n/a, 2009.

[12] Riccardo Lattanzi and Daniel K. Sodickson. Ideal current patterns yielding optimal signal-to-noise ratio and specific absorption rate in magnetic resonance imaging: Computational methods and physical insights. *Magnetic resonance in medicine*, 68:286–304, 2012.

[13] Riccardo Lattanzi, Daniel K. Sodickson, Aaron K. Grant, and Yudong Zhu. Electrodynamic constraints on homogeneity and radiofrequency power deposition in multiple coil excitations. *Magnetic resonance in medicine*, 61:315–334, 2009.

[14] Riccardo Lattanzi, Graham C. Wiggins, Bei Zhang, Qi Duan, Ryan Brown, and Daniel K. Sodickson. Approaching ultimate intrinsic signal-to-noise ratio with loop and dipole antennas. *Magnetic resonance in medicine*, 79:1789–1803, 2018.

[15] Le-Wei Li, Pang-Shyan Kooi, Mook-Seng Leong, and Tat-Soon Yee. Electromagnetic dyadic green's function in spherically multilayered media. *IEEE Transactions on Microwave Theory and Techniques*, 42:2302–2310, 1994.

[16] Ogan Ocali and Ergin Atalar. Ultimate intrinsic signal-to-noise ratio in mri. *Magnetic resonance in medicine*, 39:462–473, 1998.

[17] Michael A. Ohliger, Aaron K. Grant, and Daniel K. Sodickson. Ultimate intrinsic signal-to-noise ratio for parallel mri: Electromagnetic field considerations. *Magnetic resonance in medicine*, 50:1018–1030, 2003.

[18] Klaas P Pruessmann, Markus Weiger, Markus B Scheidegger, and Peter Boesiger. Sense: sensitivity encoding for fast mri. *Magnetic resonance in medicine*, 42(5):952–962, 1999.

[19] W. Schnell, W. Renz, M. Vester, and H. Ermert. Ultimate signal-to-noise-ratio of surface and body antennas for magnetic resonance imaging. *IEEE Antennas and Propagation Group Newsletter*, 48:418–428, 2000.

[20] Daniel K Sodickson and Warren J Manning. Simultaneous acquisition of spatial harmonics (smash): fast imaging with radiofrequency coil arrays. *Magnetic resonance in medicine*, 38(4):591–603, 1997.

[21] Chen-To Tai. *Dyadic Green functions in electromagnetic theory.* Institute of Electrical & Electronics Engineers (IEEE), 1994.

[22] Manushka V. Vaidya, Daniel K. Sodickson, and Riccardo Lattanzi. Approaching ultimate intrinsic snr in a uniform spherical sample with finite arrays of loop coils. *Magnetic resonance in medicine*, 44:53–65, 2014.

[23] H. Vesselle and R. E. Collin. The signal-to-noise ratio of nuclear magnetic resonance surface coils and application to a lossy dielectric cylinder model. i. theory. *IEEE Antennas and Propagation Group Newsletter*, 42:497–506, 1995.

[24] H. Vesselle and R. E. Collin. The signal-to-noise ratio of nuclear magnetic resonance surface coils and application to a lossy dielectric cylinder model. ii. the case of cylindrical window coils. *IEEE Antennas and Propagation Group Newsletter*, 42:507–520, 1995.

[25] Florian Wiesinger, Peter Boesiger, and Klaas P. Pruessmann. Electrodynamics and ultimate snr in parallel mr imaging. *Magnetic resonance in medicine*, 52:376–390, 2004.